FortSP: A Stochastic Programming Solver

Francis Ellison Gautam Mitra Chandra Poojari
Victor Zverovich

August 17, 2009

http://www.optirisk-systems.com

http://www.carisma.brunel.ac.uk

Preface

FortSP is a large scale stochastic programming (SP) solver, which processes linear scenario-based SP problems with recourse. It also supports scenario-based problems with chance constraints and integrated chance constraints. Implementation of stochastic mixed integer programming algorithms is available to a limited extent - enhancements are planned for a future release. Several different SP algorithms are available for the solution, statistical measures such as expected value of perfect information (EVPI) and value of the stochastic solution (VSS) may be calculated, and it can use FortMP, CPLEX or CLP as its embedded, underlying solver engine.

Contents

1 Introduction to FortSP **4**
- 1.1 The Problem . . . 4
- 1.2 Data Provision . . . 4
- 1.3 Solution Methods . . . 5
- 1.4 External Solvers . . . 6
- 1.5 Options and Controls . . . 6
- 1.6 System Summary . . . 7

2 Mathematical Description of the Problem **10**
- 2.1 Scenario Tree . . . 10
- 2.2 Two-stage Recourse Models . . . 10
- 2.3 Multi-stage Recourse Models . . . 12
- 2.4 Chance and Integrated Chance Constraints . . . 13

3 Data Provision in SMPS **15**
- 3.1 SMPS Input Format . . . 15
- 3.2 Core File and Random Parameter Values . . . 15
- 3.3 Time File . . . 16
- 3.4 Stoch File . . . 17
- 3.5 Chance and Integrated Chance Constraint Data . . . 22
- 3.6 Options and Controls for SMPS Data . . . 23

4 Data Provision in SAMPL **24**
- 4.1 SAMPL Input Format . . . 24
- 4.2 SMPS Import . . . 24
- 4.3 The `solve` Statement . . . 24
- 4.4 The `print` Statement . . . 25
- 4.5 The `write` Statement . . . 25
- 4.6 The `model` and `data` Statements . . . 25
- 4.7 The `option` Statement . . . 25
- 4.8 Built-in Names . . . 26
- 4.9 Example . . . 27

5 SP Solution Methods **31**
- 5.1 Deterministic Equivalent . . . 31
- 5.2 Cutting Plane (Benders) . . . 31

5.3 Stochastic Decomposition . . . 31
5.4 Ancillary Algorithms - EV and WS . . . 31
5.5 Statistical Measures - EVPI and VSS . . . 32
5.6 Algorithm Controls and Options . . . 32

6 Solver Options **38**
6.1 Solvers Available . . . 38
6.2 Solver Options and Controls . . . 38

7 Output Files and Logging **41**
7.1 Output and Log Filenames . . . 41
7.2 Output Controls and Options . . . 41

References **43**

A Option and Control Summary **44**
A.1 Principle Options and Controls . . . 44
A.2 Miscellaneous commands . . . 53

B Known Weaknesses **54**

C Examples of Use **55**
C.1 An Example Using the Option File . . . 55
C.2 Another Example Using the Option File . . . 61
C.3 An Example in SAMPL Using SMPS Input . . . 64

D Performance on Test Models **66**
D.1 Experimental Setup . . . 66
D.2 Data Sets . . . 66
D.3 Computational Results . . . 66

1 Introduction to FortSP

1.1 The Problem

FortSP is a solver for stochastic linear, and stochastic linear mixed integer programs. In such a problem the decision variables are governed by linear constraints, with a linear expression for the objective. To a limited extent some decision variables may be of binary or integer type. Certain data values will be unknown precisely, and only represented by a discrete range with a probability given for each set of uncertain values. Knowledge of the random values will become known in a stage-by-stage progression, and in recourse problems there will be decision variables reserved for each stage which adapt the solution to unfolding events. User may add a rider to some constraints that they need only hold with a certain probability.

Recourse Problems

A recourse problem is one in which only some decision variables must be fixed immediately. Other variables are fixed in stages - those of one period taking into account the scenario values that have become known in current and in previous stages, but with future stages still unknown. These postponed decisions are known as recourse variables.

Single Stage

When there are no recourse variables, and all decision variables must be fixed without knowing the random values, this is then a single-stage model.

Chance and Integrated Chance Constraints

These appear as normal constraints, but whether satisfied or not is subject to the uncertainty. Chance constraints need only hold with a certain probability in the final solution. Integrated chance constraints limit the expected violation of the underlying inequalities. The expected violation can be either expected shorfall or surplus depending on the type of the underlying constraint. FortSP allows these types of constraints in single and two-stage problems only.

1.2 Data Provision

Information on the model and controls on the execution may be presented on data files either in SMPS (Stochastic MPS) form, or in SAMPL (Stochastic AMPL) form. These methods involve use of the stand-alone version of FortSP. It is also possible to use FortSP as a callable library with entries at which the data is passed internally using SIR (Stochastic Internal Representation).

SMPS Format

FortSP uses a subset of SMPS (Birge et al., 1987), which is the original standard for stochastic data provision. In this method three separate data-files are provided:

- Core file: a generic presentation of the variables and constraints in MPS (MPSX) layout. Data values need not feature any particular scenario, but every random data element must be represented.
- Time file: Specifying the subdivisions of the core file that belong to each stage.

- Stoch file: specifying every scenario, the values of random data, and probabilities. The layout of stoch-file data lines corresponds to that of the core file. Only discrete forms of random distributions are considered.

SAMPL Format

AMPL is an algebraic modelling language for mathematical programming problems without any random component. SAMPL (Valente et al., 2009) is an extension of AMPL able to specify stochastic components in the appropriate staging. It allows to represent SP problems using syntax similar to the algebraic notation which is more economical and intelligible compared to SMPS. Unlike SMPS the whole problem can be specified in one file or divided into several files according to user requirements. Solver options can be also specified in the input files, as well as commands to generate an output report from the solution.

FortSP Library Input

When FortSP is used as a subroutine library, input data is entered via subroutine calls over the argument interface. This interface closely resembles the form of internal data storage, and may be termed 'Stochastic Internal Representation'. SIR as a file format does not exist at present, but may be added in future as a medium for saving problem data.

SIR is employed by the SPInE (SP Integrated Environment) system (Valente et al., 2008) with SAMPL modelling. In SPInE all features of SAMPL are present while FortSP currently supports a limited subset.

1.3 Solution Methods

A variety of stochastic algorithms are available and these can obtain solutions in one of the following forms:

- 'Here and Now' (HN) solution
- 'Wait and See' (WS) solution
- 'Expected Value' (EV) solution

HN solution provides the most exact answer to the original SP problem (also the most difficult) and to solve it there are the following SP algorithms:

- Deterministic Equivalent (DEQ)
- Cutting Plane (CP): either Benders Decomposition for recourse problems, or a special cutting plane algorithm for single-stage problems with integrated chance constraints
- Stochastic Decomposition (SD), using a random sampling technique. This is limited to 2-stage recourse problems, with other restrictions to be described later

The following statistical measures may be derived from the solutions of HN, WS and EV problems:

- The expected value of perfect information (EVPI)
- The value of the stochastic solution (VSS).

1.4 External Solvers

Most of the algorithms prepare sub-problems for solution in the form required by the solver FortMP, which has all the necessary capabilities, but other solvers may also be employed through the Open Solver Interface (OSI) of the COIN-OR (Lougee-Heimer, 2003) system. So far the following solvers have been made available:

- FortMP (Ellison et al., 2008), which is the 'natural' solver
- CPLEX
- CLP

Currently CLP and CPLEX can only be used for solving deterministic equivalent problems.

1.5 Options and Controls

Controls to steer the system by selecting the desired features are to be provided with the data input. With SAMPL input the `option` statement can be used to set any option. In the case of SMPS input either SAMPL script that imports SMPS and set the necessary options or a separate option file must be provided. The decision between SAMPL script or option file is made with the command for execution - for example:

```
fortsp mysp.opt
```

names the option file to be used by having the extension `opt`. Any other extension, such `mod` in the command

```
fortsp mysp.mod
```

invokes the SAMPL translator.

The default input name is `fortsp.opt` (originally `spinesol.opt`, but now revised), and any other name can be specified on the command line that invokes stand-alone FortSP. For example the command `fortsp D:\SPfolder\Myoptions.opt` would invoke execution with data input specified in the named file together with all other controls and options that are detailed later in this manual.

On this file one control is entered on each line. A list of all options is given in Appendix A.

Each option comprises a keyword followed by a value that is not necessarily numeric. Values are of the following types:

Text	For example, a filename
Switch	This is either `ON` or `OFF`
Option	A keyword
Value	An integer or real numeric value.

Keywords may be given in uppercase or in lowercase, and with or without the underlines used below purely for spaces in compound keywords.

The format of an option specified by OPT-file is as follows:

```
<option-keyword> <value>
```

where the values may be text-strings, switch-settings (ON or OFF), or numeric according to the specific option. Blank lines may appear anywhere in the opt-file, and comment lines are indicated by an asterisk (*) in position 1.

In SAMPL it is possible to specify one, two or more options separated by commas in the same statement. The keyword `option` is used - the layout being as follows:

```
option <name> <value>, <name> <value>, ...  ;
```

Option names in SAMPL are closely equivalent to option keywords in an opt-file, but some differences will appear. Values are either text-strings or numeric, all switches being indicated with 1 - ON, or 0 - OFF.

An example on the opt-file is as follows:

```
*   Opt-file option example
    MODEL_HN ON
    HN_ALGORITHM DETEQI
```

The same example in SAMPL would be:

```
# SAMPL option example
option SolveHN 1, SPAlg DetEq;
```

1.6 System Summary

Figure 1 gives a view of the FortSP system from the user perspective. According to the choice of the inner solver - that is FortMP, CPLEX of CLP - user must have that DLL and also the DLLs above it in the hierarchy, together with the prime executable `fortsp.exe`.

Figure 2 is a simplified summary of the SP algorithm structure. The actual path taken by execution depends on a variety of indications - for example:

- The model-types chosen for solution (one or more of HN, EV, WS)
- The model-types needed for statistical measures (EVPI and VSS)
- The algorithm to solve the HN model

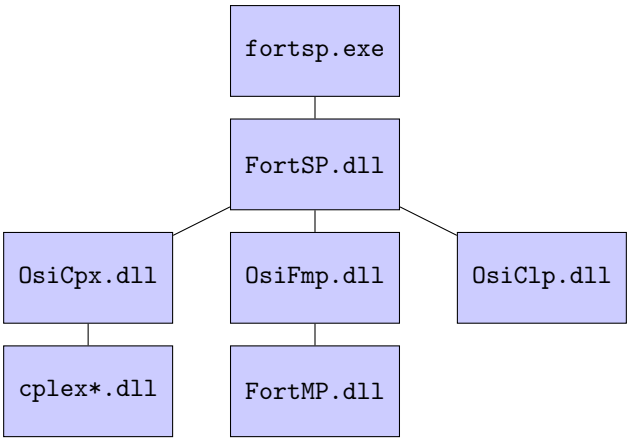


Figure 1: FortSP system

However there are various limitations to bear in mind

- The special cutting plane algorithm applies only to single-stage problems with ICC. Other problems with chance and integrated chance constraints must be solved using deterministic equivalent approach.
- Stochastic decomposition applies only to two-stage recourse problems with no random objective values, and with no random elements in the second-stage columns of the core matrix (that is vector c_2 and matrix A_{22} named below in section 2.2, model 2).
- It is not as yet possible to combine SD for the HN model with the EV model or the WS model (and therefore with calculating EVPI or VSS). Hence the SD box in figure 2 leads out only to the end of execution.

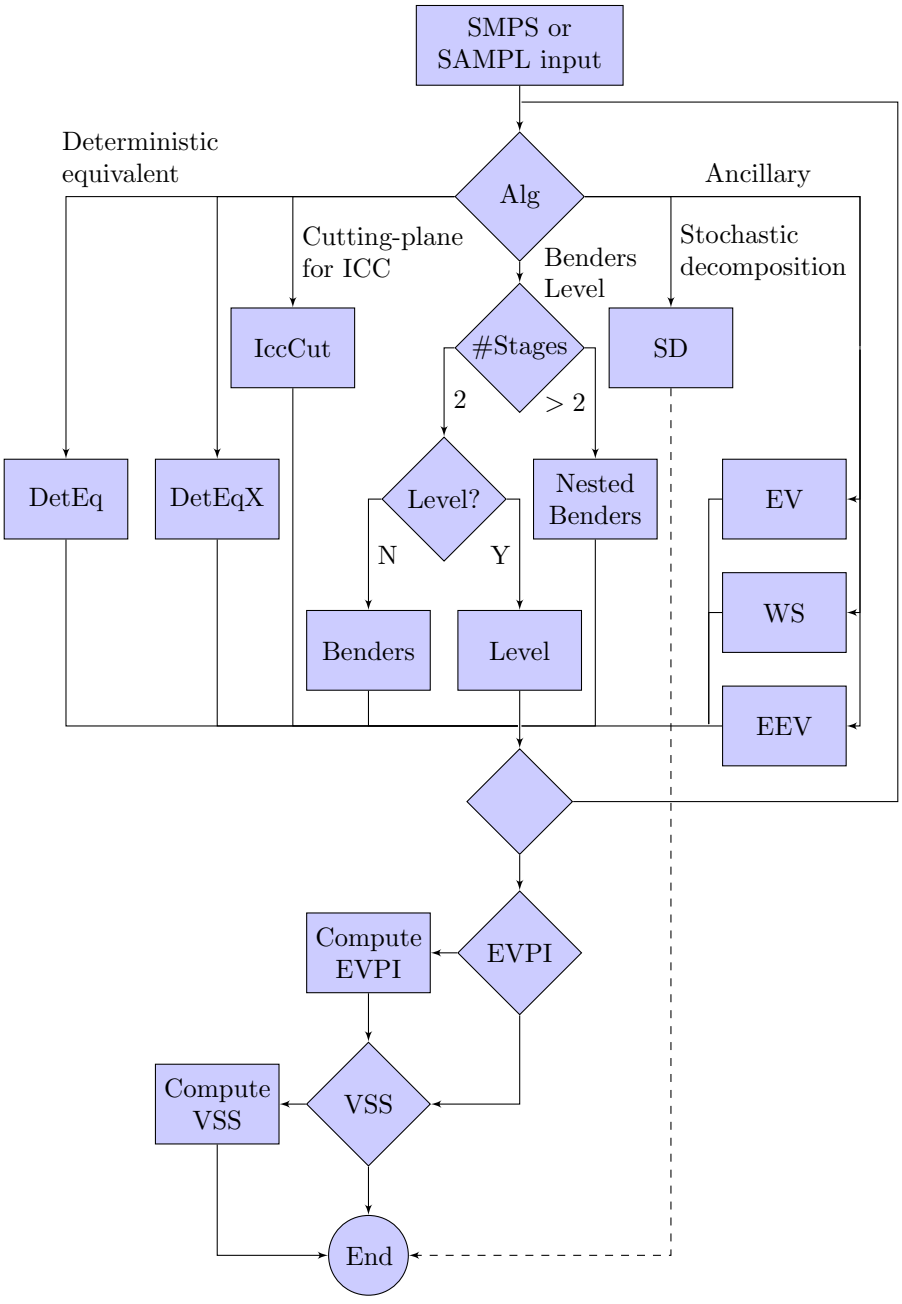


Figure 2: FortSP flowchart

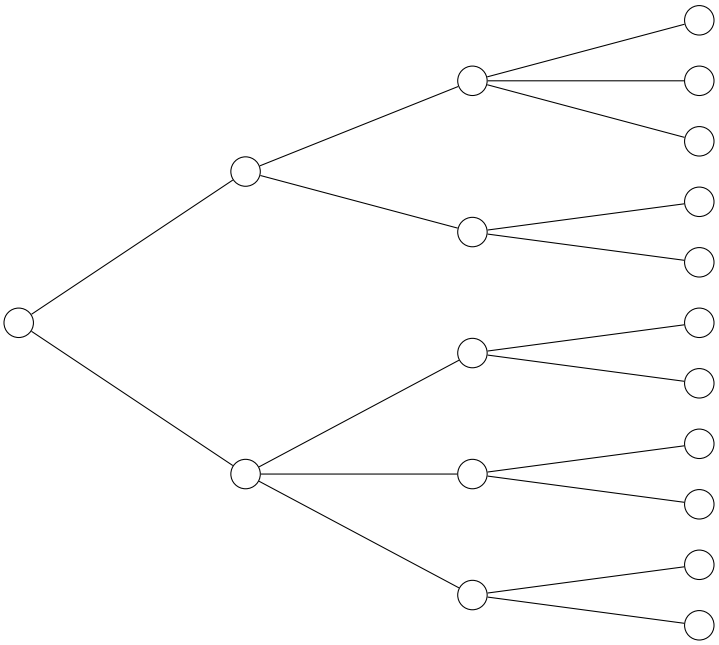


Figure 3: Scenario tree example

2 Mathematical Description of the Problem

Refer to *Introduction to Stochastic Programming* (Birge and Louveaux, 1997) for a detailed problem statement and mathematical background.

2.1 Scenario Tree

Stochastic programming can handle large numbers of decision variables and capture their complex interrelationships stated as constraints in algebraic form. The essence of SP is the confluence of optimum decision-taking model with the modelling of the random parameter behaviour. In order to model the behaviour of random parameter values we consider a limited, discrete sample of the events that may occur between any two stages of the decision-taking process, and this gives rise to a tree-like branching illustrated in figure 3.

Figure 3 illustrates the scenario tree of a 4-stage decision problem. Each node represents an optimisation problem for the decisions to be taken there, and the bundle of arcs leading from the node represents the sampled behaviour in that situation. By scenario is meant the unfolding of all events (arcs) connecting the current (first) decision (root node) to some decision in the final stage (leaf node). In general, decision tree analysis can handle only small data sets, so for realistic problem sizes there is a need for multi-stage SP.

2.2 Two-stage Recourse Models

A two-stage planning horizon is one where immediate (Here and Now) decisions (x_1) have to be taken before all the problem elements have become known. Once this happens there are further, second-stage decisions (x_2) to be taken according to the newly discovered events. After splitting the problem into known and unknown (uncertain) elements we have a first-stage problem as follows:

$$\begin{array}{lccccc} \text{Minimize} & & & c_1^T x_1 + \theta(x_1) & & \\ \text{Subject to} & g_1 & \leq & A_{11}x_1 & \leq & h_1 \\ & l_1 & \leq & x_1 & \leq & u_1 \end{array} \tag{1}$$

where the function θ is the expectation of second-stage utility, given the decisions x_1 in the first stage.

If we select a particular scenario, this utility function will be expressed as follows:

$$\begin{array}{lccccc} \text{Minimize} & & & c_2^T x_2 & & \\ \text{Subject to} & g_2 & \leq & A_{21}x_1 + A_{22}x_2 & \leq & h_2 \\ & l_2 & \leq & x_2 & \leq & u_2 \end{array} \tag{2}$$

without any further θ-function. Now since x_1 is known, the second-stage problem for all scenarios can be solved, from which we can derive the expectation of utility (probability-weighted average of the minima), and this defines the θ-function in the first-stage objective.

To express this more exactly we assume the (hypothetical) existence of separate second-stage decision variables x_{2s} for each scenario $s = 1, 2, \ldots, S$. Couple these with corresponding values for the uncertain data, and the second-stage model for each scenario becomes:

$$\begin{array}{lccccc} \text{Minimize} & & & c_{2s}^T x_{2s} & & \\ \text{Subject to} & g_{2s} & \leq & A_{21s}x_1 + A_{22s}x_{2s} & \leq & h_{2s} \\ & l_{2s} & \leq & x_{2s} & \leq & u_{2s} \end{array} \tag{3}$$

So for the expectation we combine the probability-weighted minima of all the second-stage models, and the entire problem becomes:

$$\begin{array}{lccccl} \text{Minimize} & & & c_1^T x_1 + \sum_{s=1}^{S} p_s (c_{2s}^T x_{2s}) & & \\ \text{Subject to} & g_1 & \leq & A_{11}x_1 & \leq & h_1 \\ & g_{2s} & \leq & A_{21s}x_1 + A_{22s}x_{2s} & \leq & h_{2s}, \forall s = 1, \ldots, S \\ & l_1 & \leq & x_1 & \leq & u_1 \\ & l_{2s} & \leq & x_{2s} & \leq & u_{2s}, \forall s = 1, \ldots, S \end{array} \tag{4}$$

where p_s is the probability of scenario s. This formulation of the problem is known as the Deterministic Equivalent (DEQ).

It was observed in the 50's that the form (4) is precisely the form solvable with the dual of Danzig-Wolfe decomposition, also known as Benders' decomposition or the L-shaped method. In this method a solution x_1 to the model (1) allows dual-solutions of model (3) to be calculated and applied to form an aggregated 'cut', which is a constraint added to model (1) - thus giving a new solution x_1, and so an iterative process is developed. Theory shows that the iterations converge to precisely the solution of the deterministic equivalent model (4).

It would seem simpler just to solve the DEQ model (4) were it not for the greatly increased size of the problem. However, the DEQ is useful if the number of scenarios is fairly small.

There is a second form of the DEQ that is obtained by postulating separate stage 1 decision variables for each scenario, and equating them by adding explicit constraints

$$x_{1s_1} = x_{1s_2}$$

for all pairs of scenarios s_1 and s_2 (enough pairs to make all scenario-values the same). This is known as DEQ with Explicit Non-Anticipativity (NA). The original form (4) is known as DEQ with Implicit NA. For these large problems Interior Point Method (IPM) is usually chosen in the solution. However, Implicit NA formulation may give difficulty with IPM owing to column density, and this can be overcome in some cases by using Explicit NA.

2.3 Multi-stage Recourse Models

In a multi-stage (multi-period) planning horizon with more than two stages decisions must be taken at each stage with knowledge only of the uncertainty in that stage and in previous stages. We can easily extend the modelling shown in (1) and (3) by considering a θ-function in all stages except the last. Thus stage 1 is:

$$\begin{array}{lccccc} \text{Minimize} & & & c_1^T x_1 + \theta_1(x_1) & & \\ \text{Subject to} & g_1 & \leq & A_{11}x_1 & \leq & h_1 \\ & l_1 & \leq & x_1 & \leq & u_1 \end{array} \tag{5}$$

The stages are now given by subscript t where $t = 1, \ldots, T$, and so for an intermediate stage $t < T$ we can say:

$$\begin{array}{lccccc} \text{Minimize} & & & c_t^T x_t + \theta_t(x1, x2, \ldots, x_t) & & \\ \text{Subject to} & g_t & \leq & A_{t1}x_1 + A_{t2}x_2 + \ldots + A_{tt}x_t & \leq & h_t \\ & l_t & \leq & x_t & \leq & u_t \end{array} \tag{6}$$

with variables $x_1, x_2, \ldots, x_{t-1}$ known already. For the last stage we have:

$$\begin{array}{lccccc} \text{Minimize} & & & c_T^T x_T & & \\ \text{Subject to} & g_T & \leq & A_{T1}x_1 + A_{T2}x_2 + \ldots + A_{TT}x_T & \leq & h_T \\ & l_T & \leq & x_T & \leq & u_T \end{array} \tag{7}$$

with variables $x_1, x2, \ldots, x_{T-1}$ known already. Note that each θ-function depends on the decisions in that stage and in previous stages, up until the last stage, which has no θ-function. The constraints of the t-th stage involve $x_t, x_{t-1}, \ldots, x_1$ (see [1]).

The solution of such a model requires 'nesting'. A specific model corresponds to a specific node of the scenario tree (exampled in figure 3). Hence to solve the sub-model for a given node we need the values of decisions along the path leading up to that node, and all the solutions to the sub-tree of nodes leading from it. Given a proposed solution for everything up to stage $T-1$, we can adjust the solution of stage $T-1$ by applying 2-stage Benders to each bundle of paths leading from stage $T-1$. The same process applies to stage $T-2$ by nesting the last-stage 2-stage Benders inside to form a 3-stage Benders solver. And so on for

[1] In a 'Markovian' situation the t-th stage involves only x_t and x_{t-1}. FortSP handles non-Markovian as well as Markovian situations

the whole tree. Actually the multi-stage Benders algorithm is much faster, using the 'Fast Forward, Fast Back' algorithm described in (Birge and Louveaux, 1997) chapter 7.

Solving multi-stage problems with Deterministic Equivalent also involves nesting, and here the nesting is in the formulation. Consider the (hypothetical) existence of scenario decision variables for every sub-path on the scenario tree connecting one node to its parent, and remember that the probability of that path is the sum of the probabilities of all paths leading through it. Assemble the constraints for all the scenarios and combine in the objective the sub-path-probability-weighted sum of all scenario objectives (too complicated to express here in mathematical terminology). This gives the implicit NA version of the DEQ. For the explicit NA version we consider separate variables for every scenario in every stage, and add all the constraints needed to equate all the variables for different scenarios that lie along the same sub-path everywhere in the tree.

2.4 Chance and Integrated Chance Constraints

In addition to multistage recourse problems described above, FortSP supports single and two-stage problems with individual chance constraints and integrated chance constraints (ICC). By a singe-stage SP problem we mean the one in which all decisions take place in the first stage and then the random parameters realise. The difference from a two-stage problem is that the latter has also a recourse action.

A probabilistic or chance constraint is a constraint that must hold with some minimum probability level. In the framework of model (2) an individual chance constraint corresponding to i-th row ($1 \leq i \leq m_2$) can be formulated as:

$$\mathrm{P}\{g_2^i \leq A_{21}^i \boldsymbol{x_1} + A_{22}^i \boldsymbol{x_2} \leq h_2^i\} \geq \alpha, \tag{8}$$

where $0 < \alpha < 1$ is a reliability level, g_2^i and h_2^i, denote i-th elements of vectors $\boldsymbol{g_2}$ and $\boldsymbol{h_2}$; A_{21}^i and A_{22}^i denote i-th rows of matrices A_{21} and A_{22}.

The chance constraint constraint (8) has the following deterministic equivalent form:

$$\begin{array}{rcll} g_{2s}^i & \leq & A_{21s}^i \boldsymbol{x_1} + A_{22s}^i \boldsymbol{x_{2s}} + M v_s & (s = 1, \ldots, S) \\ & & A_{21s}^i \boldsymbol{x_1} + A_{22s}^i \boldsymbol{x_{2s}} - M w_s \leq h_{2s}^i & (s = 1, \ldots, S) \\ v_s & \leq & z_s & (s = 1, \ldots, S) \\ w_s & \leq & z_s & (s = 1, \ldots, S) \\ \multicolumn{4}{l}{\sum_{s=1}^{S} p_s z_s \leq 1 - \alpha,} \end{array}$$

where M is a suitably chosen large constant, v_s, w_s and z_s are additional binary variables.

Similarly, below is the formulation of an individual ICC if g_2^i is infinite for all realisations of random parameters:

$$\mathbb{E}[(h_2^i - A_{21}^i \boldsymbol{x_1} - A_{22}^i \boldsymbol{x_2})^-] \leq \beta, \tag{9}$$

where $\beta \geq 0$ and $(a)^- := \max\{-a, 0\}$ is a negative part of $a \in \mathbb{R}$.

The ICC (9) has the following deterministic equivalent form:

$$A^i_{21s}\boldsymbol{x_1} + A^i_{22s}\boldsymbol{x_{2s}} - w_s \leq h^i_{2s} \quad (s = 1, \ldots, S)$$
$$\sum_{s=1}^{S} p_s w_s \leq \beta,$$

where w_s are additional variables. Note that in the case of integrated chance constraints introduced variables are continuous which makes deterministic equivalents of ICCs computationally more tractable than those of chance constraints.

If h^i_2 is infinite for all realisations of random parameters the constraint will look like:

$$\mathbb{E}[(A^i_{21}\boldsymbol{x_1} + A^i_{22}\boldsymbol{x_2} - g^i_2)^-] \leq \beta,$$

The case of both g^i_2 and h^i_2 finite results in a special case of a joint ICC and is not currently supported.

3 Data Provision in SMPS

3.1 SMPS Input Format

Whereas MPSX defines the format for LP data input, Stochastic MPS (SMPS) defines the input format for stochastic programming problems. FortSP implements the most important provisions of SMPS which is described by Birge et al. (1987). Since the solver is initially designed for use within the SPInE system its stochastic input fulfils the requirements of models generated by SPInE, and other provisions of SMPS are not supported in full. SPInE generates solver stochastic data in the form of discrete scenarios, and FortSP supports this form and also discrete blocks form and discrete independent form.

The stochastic items time-stage, block and scenario are all to be identified by an index with optional prefix for readability, although SMPS standard calls for identification by the full name (prefix plus index). FortSP ignores every prefix, but nevertheless user should adopt the same prefix for the same item throughout, in conformity with the SMPS standard. A future version of FortSP may be upgraded to identify by name rather than by index.

Three input files are required in order to specify a stochastic problem:

- Core file which is the fundamental problem template in the form of an LP problem using the MPSX standard
- Time file specifying which rows and columns of the core-file belong to which time stage
- Stoch file specifying the alternative values taken by each random parameter value in the core file

The user may specify precise names for the input files or may give the basename - or 'Generic' name - of these files so that extensions are appended automatically. Denoting a base name by `<problem>` the resulting filenames are:

`<problem>.cor` for core file
`<problem>.tim` for time file
`<problem>.sto` for stoch file

3.2 Core File and Random Parameter Values

The core file expresses a linear programming problem or linear mixed integer problem in the format known as MPSX, familiar to the users of LP solvers and described in the manuals of many of such system, for example, FortMP (Ellison et al., 2008). In this format the file is divided into `ROWS`, `COLUMNS`, `RHS`, `RANGES` and `BOUNDS` sections, and data records have a fixed format as follows:

Field 1. Positions 2-3 (code)
Field 2. Positions 5-12 (1st name field)
Field 3. Positions 15-26 (2nd name field)
Field 4. Positions 30-37 (1st numeric field)
Field 5. Positions 40-47 (3rd name field)
Field 6. Positions 50-61 (2nd numeric field)

This layout is also used for data in the time and stoch files described below.

In the stochastic model a number of scalars will have uncertain values - they are denoted here as *random parameter values*. They may be anywhere in the core file other than in the `ROWS` section. Each random parameter value must have a representative, finite value assigned to it in the core model, and this value must be recognisable by the input. It may not be zero in the `COLUMNS` or the `RANGES` section, or left as infinite in the `BOUNDS` section. However it does not have to be a value corresponding to any particular scenario.

Both constraints and variables must be grouped according to the stage at which they apply. These separate groups are to be in the order of time stage in the core file (constraints in the `ROWS` section and variables in the `COLUMNS` section). As a result of this ordering the constraint matrix should have a lower block triangular form, with blocks for each stage.

MIP in the form of binary or integer descriptions may be applied to any decision variable (SOS and semi-continuous are not supported). However if it applies to variables other than the first here-and-now stage then the HN model must be solved using Deterministic Equivalent methods - Benders' decomposition would not be suitable.

In the stochastic data (stoch file) the above data-line format is employed without the headers such as `COLUMNS`, `RHS` etc that are used in the core file. For this reason certain names are reserved as keywords for stochastic data and should not be used either as row or column names, or as the leading characters of row or column names. These are:

```
RHS    BOUND  OBJ
LHS    BND    COST
RIGHT  RANGE
LEFT   RNG
```

Included are any variants of these keywords with lower case lettering. The following are examples of illegal names:

```
rhs     BOUNDSET  objective
Rhside  bnd1      costrow
LeftHS  RANGEABC
```

An exception is made for `OBJ` and `COST`, which may be used in the name of the actual objective row (but not in any other row).

3.3 Time File

The time file specifies the first member of each stage grouping in the constraints and variables of the core model (hence the need to group these items by stage). The first line is as follows:

Positions 1-4	Keyword `TIME`
Field 3 (15-22)	Problem name

This is followed by the period header line as follows:

Positions 1-7	Keyword `PERIODS`
Field 3 (15-22)	Keyword `IMPLICIT` (optional)

After this one line is included for each stage as follows:

Field 2 (5-12) Starting variable name
Field 3 (15-22) Starting constraint name
Field 5 (40-47) Stage name

Time stages are indexed $1, 2, \ldots$ with stage number 1 being the first, here-and-now stage.

Finally the data ends with the following line:

Positions 1-6 Keyword `ENDATA`

The following is an example:

```
TIME          EXAMPLE
PERIODS       IMPLICIT
    C1        R1                       STAGE1
    C6        R3                       STAGE2
    C8        R19                      STAGE3
ENDATA
```

Note that in place of `R1` it is possible to use the objective row name. The objective row is moved by the input into the first row position, wherever it is found in the data.

3.4 Stoch File

All random data and the discrete distributions are specified in the stoch file. The first line is as follows:

Positions 1-5 Keyword `STOCH`
Field 3 (15-22) Problem name

This is followed by a header line that specifies the form of data input as follows:

Positions 1 onwards Keyword defining the data form as one of
`INDEP BLOCKS SCENARIOS`
`CHANCE ICC`
Field 3 (15-22) Keyword `DISCRETE`

After this header line there are data-lines as described below and the file is terminated as before:

Positions 1-6 Keyword `ENDATA`

Sample values for random parameter values are presented in the stoch file in the same form as they appear in the core file, that is:

	Random parameter value in section:			
Field	`COLUMNS`	`RHS`	`RANGES`	`BOUNDS`
1: 2-3				Bound type
2: 5-12	Column name	Vector name	Vector name	Vector name
3: 15-22	Row name	Row name	Row name	Column name
4: 25-36	Sample value	Sample value	Sample value	Sample value
5: 40-47	2nd row name	2nd row name	2nd row name	
6: 50-61	Sample value	Sample value	Sample value	

Fields 5 and 6 have a different use for INDEP-form data (see below), and otherwise can only be used for random parameter values with the same description in fields 1 and 2.

Any value greater or equal to 1.7×10^{38} (less or equal to -1.7×10^{38}) used as a row or column bound is treated as a positive (negative) infinity.

Stochastic data form INDEP

INDEP is used when each separate random parameter value has an independent distribution. Scenarios are built by selecting one possibility for each random parameter value, the set of all scenarios is then formed by taking all combinations of possible selections.

Each INDEP data line can describe only one random parameter value as follows:

Fields 1-4	As described above
Field 5 (40-47)	Stage name
Field 6 (50-61)	Probability value of the sample (must sum to 1 for each random parameter value)

Sequence should be according to time stage, and with separate samples of the same random parameter value collected into consecutive lines. The following is an example:

```
STOCH         EXAMPLE
INDEP         DISCRETE
    C6        OBJ         2.5           STAGE2    0.5
    C6        OBJ         3.0           STAGE2    0.5
    C6        R3          5.0           STAGE2    0.33
    C6        R3          5.5           STAGE2    0.67
    C8        R19         1.0           STAGE3    0.25
    C8        R19         2.0           STAGE3    0.25
    C8        R19         3.0           STAGE3    0.5
ENDATA
```

The above example has 12 ($2 \times 2 \times 3$) scenarios, illustrated in the following table:

	Base		Value of			
Scen	Scen	Stage	(C6,OBJ)	(C6,R3)	(C8,R19)	Probability
1	core	2	2.5	5.0	1.0	0.04125
2	1	3	-	-	2.0	0.04125
3	1	3	-	-	3.0	0.08250
4	1	2	-	5.5	1.0	0.08375
5	4	3	-	-	2.0	0.08375
6	4	3	-	-	3.0	0.16750
7	1	2	3.0	5.0	1.0	0.04125
8	7	3	-	-	2.0	0.04125
9	7	3	-	-	3.0	0.08250
10	7	2	-	5.5	1.0	0.08375
11	10	3	-	-	2.0	0.08375
12	10	3	-	-	3.0	0.16750

Here the Base Scen column is the scenario containing the default values for any unstated random parameter values. The first scenario is always based on the core problem and specifies values for all random parameter values. The Stage column states the stage at which a difference appears from the base.

Stochastic data form `BLOCKS`

The nature of this form is very similar to `INDEP`, except that individual independent random parameter values give place to independent blocks (or sets) of random parameter values. The stage number and probability distribution become properties of the block rather than the individual random parameter value. Each new block and each new set of block values is introduced with a header line as follows:

Field 1 (2-3)	Keyword `BL`
Field 2 (5-12)	Block name
Field 3 (15-22)	Stage name
Field 4 (25-36)	Probability value of the sample (must sum to 1 over the samples of each block)

Blocks with the same name should be grouped together.

Values for the members of each block are entered in a way similar to `INDEP` data, except that fields 5 and 6, not being required for stage and probability, may contain a second data entry for fields 3 and 4 as tabled above in the general stoch file description.

The following is an example:

```
STOCH         EXAMPLE
BLOCKS        DISCRETE
 BL BLOCK1    STAGE2     0.5
    C6        OBJ        2.5           R3        5.0
 BL BLOCK1    STAGE2     0.5
    C6        OBJ        3.0           R3        5.5
 BL BLOCK2    STAGE3     0.25
    C8        R19        1.0
    RHS       R19        100.0
 BL BLOCK2    STAGE3     0.25
    C8        R19        2.0
    RHS       R19        200.0
 BL BLOCK2    STAGE3     0.5
    C8        R19        3.0
ENDATA
```

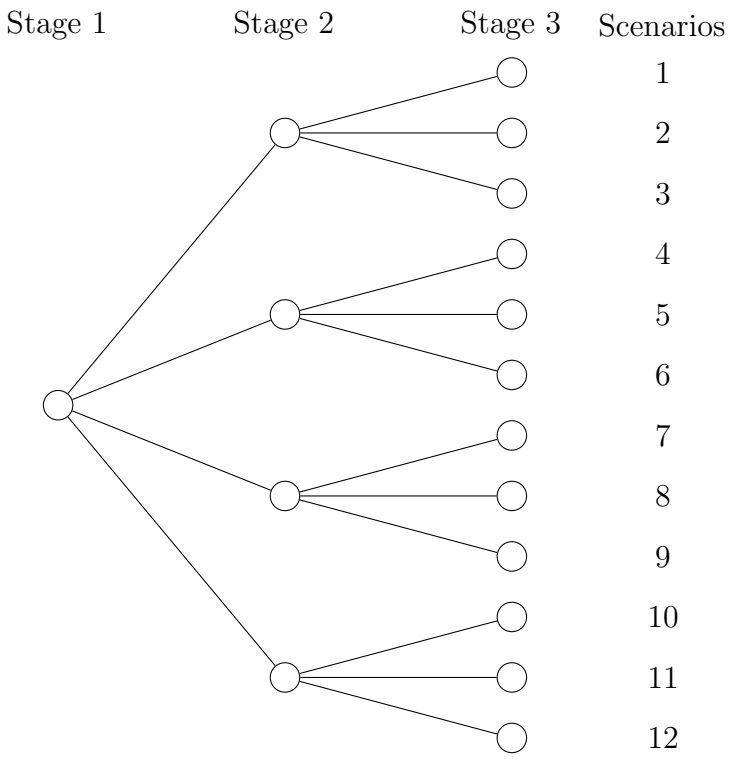
This example gives rise to scenarios in the same manner as before, illustrated as follows:

Scen	Base Scen	Stage	Value of (C6,OBJ) (C6,R3)	Value of (C8,R19) (RHS,R19)	Probability
1	Core	2	2.5	1.0	0.125
			5.0	100.0	
2	1	3	-	2.0	0.125
				200.0	
3	2	3	-	3.0	0.250
				-	
4	1	2	3.0	1.0	0.125
			5.5	100.0	
5	4	3	-	2.0	0.125
				200.0	
6	5	3	-	3.0	0.250
				-	

Note that block samples do not have to restate values in the block if they duplicate the previous sample (SMPS standard). Hence the base for samples other than the first of a block is the previous sample. So in the above example scenarios 3 and 6 assign value 200.0 to `(RHS,R19)`.

Stochastic data form SCENARIOS

Scenarios have been introduced in the examples above. It may be observed that the branching of scenarios from each other (i.e. the base scenario connection) forms an event tree in which decisions may be taken at the nodes. The tree for the `INDEP` example above looks as follows:



In this diagram each scenario is represented by a full path through the nodes from left to right.

Scenario-form data is prepared from such a tree, which should be known directly or implicitly. For each scenario it is only necessary to enter the information that differs from its base scenario - that is the earlier scenario from which it branches. Where several branches issue from one node (to the right) the later scenarios may be considered as branching from any earlier scenario in that bundle. Thus for example:

Scenario 10 could branch from 1, 4 or 7

Scenario 6 could branch from 4 or 5

Scenario 1 must always branch from the core model (and provide values for all the random parameter values).

Each scenario is preceded in the stoch file by a scenario header line as follows:

Field 1 (2-3)	Keyword `SC`
Field 2 (5-12)	Scenario name
Field 3 (15-22)	Should contain `ROOT` for scenario 1. For other scenarios enter the name of the base scenario
Field 4 (25-36)	Probability value of the scenario (must sum to 1 over all scenarios)
Field 5 (40-47)	Stage index with optional prefix

Data lines for scenarios follow the layout tabled in the general stoch file description.

Here is how the `BLOCKS` example can be presented in `SCENARIOS` form:

```
STOCH         EXAMPLE
SCENARIOS     DISCRETE
 SC SCEN1     ROOT       0.125         STAGE1
    C6        OBJ        2.5           R3        5.0
    C8        R19        1.0
    RHS       R19        100.0
 SC SCEN2     SCEN1      0.125         STAGE3
    C8        R19        2.0
    RHS       R19        200.0
 SC SCEN3     SCEN2      0.250         STAGE3
    C8        R19        3.0
 SC SCEN4     SCEN1      0.125         STAGE2
    C6        OBJ        3.0           R3        5.5
    C8        R19        1.0
    RHS       R19        100.0
 SC SCEN5     SCEN4      0.125         STAGE3
    C8        R19        2.0
    RHS       R19        200.0
 SC SCEN6     SCEN5      0.250         STAGE3
    C8        R19        3.0
ENDATA
```

3.5 Chance and Integrated Chance Constraint Data

Data for chance constraints and ICC are are presented on the STOCH file in additional sections preceding the random data.

`CHANCE` section

Chance constraints can be represented in the stoch file using the `CHANCE` section where the reliability parameters α are supplied (see section 2.4). The constraints themselves are defined in the core file and the distributions of their stochastic elements are defined in extra sections of the stoch file.

After a section header consisting of a single keyword `CHANCE` in position 1 each line describes a single chance constraint and has the following structure:

Field 1 (2-3)	`L` or `G` denoting constraint sense as in the `ROWS` section
Field 2 (5-12)	Name of a group of constraint
Field 3 (15-22)	Row name
Field 4 (25-36)	Reliability parameter α, see section 2.4

Example:

```
CHANCE
 G  CC1       R1          0.95
 L  CC1       R2          0.10
```

The `CHANCE` section allows one or more groups of chance constraints to be defined. In the above example, the name of the group is `CC1`. FortSP uses the first group and ignores all others.

`ICC` section

The `ICC` section is very similar to the `CHANCE` section. It starts with the keyword `ICC` followed by the lines in the form described below:

Field 1 (2-3)	`L` or `G` denoting constraint sense as in the `ROWS` section
Field 2 (5-12)	Name of a group of constraint
Field 3 (15-22)	Row name
Field 4 (25-36)	Parameter β for ICC, see section 2.4

Example:

```
ICC
 L  ICC1      R8          10.0
```

The `ICC` section allows one or more groups of ICCs to be defined. In the above example, the name of the group is `ICC1`. FortSP uses the first group and ignores all others.

3.6 Options and Controls for SMPS Data

FortSP gives two possibilities of controling solver execution together with SMPS input. One is through a separate option file which is described in Section 1.5. Another is through SAMPL and its SMPS import feature described in Section 4.2.

The following is a table of the options relevant to SMPS input.

Opt-file Name	`GENERIC_FILENAME`
Description	Specifies a stub or generic name for input and output files (i.e. file-name without any extension). A standard extension is added for each actual filename.
Value	String
Default	`SPmodel`
Opt-file Name	`CORE_FILE`
Description	Actual name of the core file
Value	String
Default	
Opt-file Name	`TIME_FILE`
Description	Actual name of the time file
Value	String
Default	
Opt-file Name	`STOCH_FILE`
Description	Actual name of the stoch file
Value	String
Default	
Opt-file Name	`OPT_DIR`
SAMPL Name	`SmpsObjSense`
Description	The sense of optimisation for SMPS problems
Value	`MIN` or `MAX`
Default	`MIN`
Opt-file Name	`SPS_WORKING_DIR`
Description	Name of the folder to which the current working directory is transferred immediately after input of the option-file has completed, and before any other input. All I/O files are located in the local working directory except where a different path is given with a specific file-name command. Files not so named take the generic name followed by a standard extension, and so are located in this directory. In the option-file (not in SAMPL) the local working directory can be changed by setting this option before opening any other I/O file.
Value	String
Default	

4 Data Provision in SAMPL

The primary input format accepted by FortSP when used in the SPInE system is Stochastic AMPL, or SAMPL, which is described in details in *SAMPL/SPInE User Manual* (Valente et al., 2008). SAMPL is an extension of the AMPL modelling language for stochastic programming. It has the advantage of being easier to understand and more compact than SMPS. As an experimental feature this release of a standalone FortSP solver provides a limited support for SAMPL input.

4.1 SAMPL Input Format

Current version of the SAMPL translator used in FortSP accepts only two-stage SP problems expressed in a subset of the language. The syntax can be inferred from the example in Section 4.9. Details of modelling with SAMPL can be found elsewhere (Valente et al., 2008), this section only describes the scripting features that can be used to control FortSP and present the results.

4.2 SMPS Import

FortSP provides a special form of the `read` statement for importing SMPS problems into the SAMPL environment. It allows to work with problems in both formats in a uniform way.

Syntax

read-stmt:
read smps (*basename*) **;**
read smps (*core-filename, stoch-filename, time-filename*) **;**

The first form can be used if the names of the SMPS input files differ only in extension which is `cor` for the core file, `sto` for the stoch file and `tim` for the time file. The basename is then their common filename without extension. The second form allows to specify all three filenames.

SMPS doesn't specify the direction of optimisation (objective sense) and FortSP assumes minimisation by default. It can be changed by setting the `SmpsObjSense` option before importing the problem. Possible values for this options are `Minimize` and `Maximize`.

During the import the name of the SAMPL problem is derived from the SMPS problem name with all spaces and characters not allowed in SAMPL identifiers replaced with underscores, e.g. `problem-1` is changed to `problem_1`. The objective name is transformed in the same way. Two sets called `_SMPS_ROWS` and `_SMPS_COLS` are introduced which contain the names of the first-stage rows and columns. SMPS variables and constraints can be accessed using names `_smps_var` and `_smps_con` respectively. The variable `_smps_var` is indexed over the set `_SMPS_COLS` and the constraint `_smps_con` is indexed over `_SMPS_ROWS`.

4.3 The `solve` Statement

The `solve` statement instantiates the current problem and solves it.

Syntax

solve-stmt:

solve ;

4.4 The `print` Statement

The `print` statement evaluates each expression in the list and prints the result to the standard output.

Syntax

print-stmt:
print *[indexing :] expr-list* **;**

expr-list:
expr
expr-list **,** *expr*

Example: `print {c in _SMPS_COLS}: _smps_var[c];`

4.5 The `write` Statement

The `write` statement writes the current problem in the SMPS form.

Syntax

write-stmt:
write m*filename* **;**

Example: `write mout;`

4.6 The `model` and `data` Statements

The `model` and `data` statements have two forms. The one without arguments switches the current mode. For example the statement `data;` enters the data mode. The second form which takes a filename argument translates the specified file.

Syntax

model-stmt:
model *[filename]* **;**

data-stmt:
data *[filename]* **;**

4.7 The `option` Statement

The `option` statement sets and/or prints the option values.

Syntax

option-stmt:
option *option-list* **;**

option-list:

option
option-list , *option*

option:
name [expr]

Here *name* is an option name and *expr* is an optional expression of compatible type. If the expression is not specified the option value is printed. Otherwise the expression is evaluated and the result is assigned to the option. Example: `option Solver OsiClp, LPAlg Dual;`

4.8 Built-in Names

SAMPL recognizes the following built-in names.

Name	Description
`Infinity`	The value for representing an infinite bound
`evpi`	The expected value of perfect information
`vss`	The value of the stochastic solution

Both `evpi` and `vss` apply to the current problem.

In addition to the builtin names the following suffixes are supported.

Suffix	Description
`.rc`	Reduced cost of a variable
`.body`	Current value of constraint body
`.ev`	Current value in the EV problem
`.ev_rc`	Reduced cost of a variable in the EV problem
`.ev_body`	Current value of constraint body in the EV problem

4.9 Example

As an example let's consider the farmer's problem from *Introduction to Stochastic Programming* (Birge and Louveaux, 1997).

A European farmer has 500 acres of land where he plans to grow wheat, corn, and sugar beets. He wants to decide how much land to devote to each crop in order to maximize profit and produce enough grain to feed his cattle. The farmer knows that at least 200 tons (T) of wheat and 240 T of corn are needed for cattle feed. All that remains after satisfying the feeding requirements is sold. Selling and purchase prices as well as planting costs are given in the following table.

	Wheat	Corn	Sugar Beets
Planting cost ($/acre)	150	230	260
Selling price ($/T)	170	150	36 under 6000 T
			10 above 6000 T
Purchase price ($/T)	238	210	-
Min. requirement (T)	200	240	-

Note that sugar beet has two selling prices because the European Commission imposes a quota on its production. Any amount above the quota is sold at a lower price.

The uncertainty in the problem comes from the weather conditions that affect yields. In this problem three possible scenarios are considered. The yields in tons per acre are given below for each crop and scenario.

	Wheat	Corn	Sugar Beets
Above	2.0	2.4	16.0
Average	2.5	3.0	200.0
Below	3.0	6.0	24.0

Here is a SAMPL formulation of the model:

```
set Crops;

scenarioset Scenarios;
probability P{Scenarios};
tree Tree := twostage;

param TotalArea;                # acre
param Yield{Crops, Scenarios}; # T/acre
param PlantingCost{Crops};     # $/acre
param SellingPrice{Crops};     # $/T
param ExcessSellingPrice;      # $/T
param PurchasePrice{Crops};    # $/T
param MinRequirement{Crops};   # T
```

```
param BeetsQuota;              # T

# Area in acres devoted to crop c
var area{c in Crops} >= 0;

# Tons of crop c sold (at favourable price in case of beets)
# under scenario s
var sell{c in Crops, s in Scenarios} >= 0, suffix stage 2;

# Tons of sugar beets sold in excess of the quota under
# scenario s
var sellExcess{s in Scenarios} >= 0, suffix stage 2;

# Tons of crop c bought under scenario s
var buy{c in Crops, s in Scenarios} >= 0, suffix stage 2;

maximize profit: sum{s in Scenarios} P[s] * (
    ExcessSellingPrice * sellExcess[s] +
    sum{c in Crops} (SellingPrice[c] * sell[c, s] -
                     PurchasePrice[c] * buy[c, s])) -
    sum{c in Crops} PlantingCost[c] * area[c];

s.t. totalArea: sum {c in Crops} area[c] <= TotalArea;

s.t. requirement{c in Crops, s in Scenarios}:
    Yield[c, s] * area[c] - sell[c, s] + buy[c, s]
        >= MinRequirement[c];

s.t. quota{s in Scenarios}: sell['beets', s] <= BeetsQuota;

s.t. beetsBalance{s in Scenarios}:
    sell['beets', s] + sellExcess[s]
        <= Yield['beets', s] * area['beets'];
```

The data for the farmer's problem are as follows:

```
data;

set Crops := wheat corn beets;
set Scenarios := below average above;

param TotalArea := 500;

param P :=
    below   0.333333
    average 0.333333
    above   0.333333;
```

```
param Yield:
           below average above :=
    wheat    2.0     2.5   3.0
    corn     2.4     3.0   3.6
    beets   16.0    20.0  24.0;

param PlantingCost :=
    wheat 150
    corn  230
    beets 260;

param SellingPrice :=
    wheat 170
    corn  150
    beets  36;

param ExcessSellingPrice := 10;

param PurchasePrice :=
    wheat 238
    corn  210
    beets 100; # Set to a high value to simplify the objective

param MinRequirement :=
    wheat 200
    corn  240
    beets   0;

param BeetsQuota := 6000;
```

Finally the script file needs to be provided which loads model and data files, solves the problem and retrieves the optimal value and solution. Due to the flexibility of the AMPL language SAMPL is based on, it is possible to combine model, data and script in one file which can be convenient in some cases. However, in general it is not recommended since it makes more difficult to use the same model with different data sets.

Let's assume that the model and data are stored in the files `farmer.mod` and `farmer.dat`. The following script loads these files, solves the problem, computes EVPI and VSS and prints the results:

```
# Read the model and data.
model farmer.mod;
data farmer.dat;

# Set the options.
option ComputeEvpi 1, ComputeVss 1, VssFStage 1;

# Instantiate and solve the problem.
solve;
```

```
# Print the results.
print 'Optimal value =', profit;
print 'EVPI =', evpi;
print 'VSS =', vss;
print;
print 'First-stage solution:';
print {c in Crops}: 'area[', c, '] =', area[c], '\
';
print 'totalArea =', totalArea.body;
```

Running FortSP with the command `fortsp <script filename>` will produce the following output:

```
Optimal solution found
Optimal value = 108389.5654
EVPI = 7015.64583
VSS = 1149.8943

First-stage solution:
area[ wheat ] = 170
area[ corn ] = 80
area[ beets ] = 250

totalArea = 500
```

5 SP Solution Methods

5.1 Deterministic Equivalent

The simplest solution approach is to formulate a deterministic equivalent of the SP problem and use a linear programming (LP) solver to optimise it. FortSP fully supports automatic formulation of deterministic equivalents either with implicit or with explicit non-anticipativity. This method is feasible and sometimes advantageous especially if the number of scenarios is relatively small.

FortSP can also formulate deterministic equivalents of two-stage problems with individual chance constraints and integrated chance constraints.

5.2 Cutting Plane (Benders)

The following decomposition algorithms are available in FortSP for solving the Here-and-Now (HN) problem:

- Benders' decomposition - L-shaped method
- Level decomposition variant
- Nested Benders' decomposition

The first two methods are applicable for two-stage problems and the last allows solving multi-stage problems. These algorithms take advantage of a specific structure of stochastic programming problems and make it possible to solve problems with large number of scenarios. The level decomposition applies a regularisation that is particularly effective for larger numbers of scenarios.

In addition to finding here-and-now values for decision variables in the first stage the system may extend this to recourse values for the various scenarios in future stages.

For integrated chance constraints an efficient cutting-plane algorithm (Klein Haneveld and Vlerk, 2002) is provided.

In certain cases the addition of optimality cuts (refer to Birge and Louveaux (1997)) creates an unbounded situation as θ is a free variable. As an ad-hoc fix for this a large negative lower bound is applied to θ, which is retained until no longer needed. If not large enough then the algorithm may halt prematurely with a cycling status. It may then be possible to obtain the correct solution by specifying a lower value for θ, for example -100000.

5.3 Stochastic Decomposition

Description of stochastic decomposition is deferred for now.

5.4 Ancillary Algorithms - EV and WS

The system may also evaluate the following special problems:

- Expected value (EV) problem, which assumes that all data will take their expected values.
- Wait-and-see (WS) problem, which is obtained by solving a separate sub-problem for each scenario assuming that all random data is already known. The final WS solution is the probability-weighted average of these solutions for each scenario.

Each special problem can be evaluated in addition to the main HN problem. When statistical measures are invoked the EV and/or the WS algorithms will be called as required, whether or no a corresponding option has been set. see next section 5.5.

5.5 Statistical Measures - EVPI and VSS

The expected value of perfect information (EVPI) is computed as the difference between the optimal values of the wait-and-see (WS) and the here-and-now (HN) problems. Therefore the EVPI option implies both options for HN and WS.

In order to calculate the value of the stochastic solution (VSS), we need to know the expectation of the expected value solution (EEV). EEV is calculated by solving the EV problem, fixing the obtained solution in the WS sub-problems, and computing the probability weighted objective value. Hence the VSS option implies all three solver options: HN, EV and WS.

5.6 Algorithm Controls and Options

Options for SP algorithms are as follows:

Opt-file Name	`MODEL_HN`
SAMPL Name	`SolveHN`
Description	Flag specifying whether to solve the here-and-now problem
Value	Boolean
Default	`ON`
Opt-file Name	`MODEL_EV`
SAMPL Name	`SolveEV`
Description	Flag specifying whether to solve the expected value problem
Value	Boolean
Default	`OFF`
Opt-file Name	`MODEL_WS`
SAMPL Name	`SolveWS`
Description	Flag specifying whether to solve the wait-and-see problem
Value	Boolean
Default	`OFF`

Opt-file Name	OUTPUT_EVPI
SAMPL Name	ComputeEvpi
Description	Flag specifying whether to compute the expected value of perfect information (EVPI) The expected value of perfect information requires the solution of both HN and WS models. Setting this control ON forces both the HN switch and the WS switch to be ON. EVPI is the absolute difference between the HN and WS solution objectives.
Value	Boolean
Default	OFF

Opt-file Name	OUTPUT_VSS
SAMPL Name	ComputeVss
Description	Flag specifying whether to compute the value of the stochastic solution (VSS)[2]
Value	Boolean
Default	OFF

Opt-file Name	VSS_FIX_FSTAGE
SAMPL Name	VssFStage
Description	Flag specifying whether to fix only the first stage when computing the value of the stochastic solution[2]
Value	Boolean
Default	OFF

Opt-file Name	HN_ALGORITHM
SAMPL Name	SPAlg
Description	Stochastic programming algorithm to be used
Value	The possible values for this option are listed in the table below.

Opt-file Name	SAMPL Name	Description
	Auto	The algorithm is chosen automatically (default)
DETEQI	DetEq	The deterministic equivalent problem with implicit non-anticipativity is constructed and solved
DETEQE	DetEqX	The deterministic equivalent problem with explicit non-anticipativity constraints is constructed and solved
BENDERS	Benders	Benders' decomposition
	Level	Variant of level decomposition
STDCMP		Stochastic decomposition

The default algorithm - Auto - chosen by the system - is Benders' decomposition for all recourse problems and deterministic equivalent with implicit nonanticipativity for problems containing chance constraints and integrated chance constraints. An exception to this is single-stage ICC problems, which by default are solved with the special cutting plane algorithm. All other CC and ICC problems are solved only with deterministic equivalent, and any specification for Benders' or stochastic decomposition is ignored.

Default	Auto

Opt-file Name	MAX_TIME
SAMPL Name	MaxTime
Description	Time limit in seconds
Value	Nonnegative number
Default	3600

Options for Benders' decomposition:

[2]VSS - Value of Stochastic Solution - requires the solution of both HN and EV models. Setting this control ON forces both the HN switch and the EV switch to be ON. In order to calculate VSS we need to know the EEV - Expected value of the Expected Value solution. EEV is calculated by solving the EV model, fixing the result so obtained in all the WS models (all stages but the last), which are then solved to give a probability-weighted average value for the objective - which is the VSS. Option VSS_FIX_FSTAGE can be used to restrict the fix that is performed to first stage variables only (although in theory this is not correct, the theoretical result is often meaningless as a complete fix may be infeasible).

Opt-file Name	BEN_LEVEL_DECOMP
Description	Flag specifying whether to use level decomposition. In SAMPL level decomposition is enabled by setting the SPAlg option to Level.
Value	0 or 1
Default	1

Opt-file Name	BEN_PREPROC_EXP
SAMPL Name	BenPPExpVal
Description	Flag specifying whether to obtain the initial first stage solution by solving the EV problem
Value	Boolean
Default	ON

Opt-file Name	BEN_FFFB
SAMPL Name	BenFfffb
Description	Flag specifying whether to use fast forward, fast back method for multi-stage
Value	Boolean
Default	OFF

Opt-file Name	BEN_THLOB
SAMPL Name	BenThetaLB
Description	Lower bound for θ used when necessary to avoid unbounded situations. In certain cases the addition of optimality cuts creates an unbounded situation as θ is a free variable. As an ad-hoc fix for this, a large negative lower bound is applied to θ, which is retained until no longer needed. If not large enough then the Benders algorithm may halt prematurely with a final condition Cycling (status 6 or larger). It may then be possible to obtain a correct solution by specifying a lower value for this option, for example -100000.
Value	Number
Default	-10000

Opt-file Name	BEN_CUT_FACTOR
SAMPL Name	BenCutFactor
Description	Maximum cuts per child scenario
Value	Integer
Default	20

Opt-file Name	BEN_MAX_ITER
SAMPL Name	BenMaxIter
Description	Iteration limit for Benders' decomposition
Value	Nonnegative integer
Default	10000

Options for stochastic decomposition:

Opt-file Name	SD_MAX_ITER
SAMPL Name	SDMaxIter
Description	Maximum iterations
Value	Integer
Default	10000

Opt-file Name	SD_MAX_SCEN
SAMPL Name	SDMaxScen
Description	Maximum scenarios
Value	Integer
Default	1000

Opt-file Name	SD_MAX_DVD
SAMPL Name	SDMaxDvd
Description	Maximum dual vertex - deterministic
Value	Integer
Default	1000

Opt-file Name	SD_MAX_DVS
SAMPL Name	SDMaxDvs
Description	Maximum dual vertex - stochastic
Value	Integer
Default	10000

Opt-file Name	SD_MAX_INFEZ
SAMPL Name	SDMaxInf
Description	Maximum infeasibility cuts
Value	Integer
Default	5

Opt-file Name	SD_EXP_VAL
SAMPL Name	SDExpVal
Description	Flag specifying whether to obtain the initial first stage solution by solving the EV problem
Value	Boolean
Default	ON

Opt-file Name	SD_INPUT_LOBND
SAMPL Name	SDInputLo
Description	Flag specifying whether to input θ lower bound (if not then auto-calculated)
Value	Boolean
Default	OFF

Opt-file Name	SD_LOBND
SAMPL Name	SDLoBnd
Description	Lower bound for θ In the SD algorithm a probable lower bound is calculated when SD_INPUT_LOBND is OFF. If SD_INPUT_LOBND is ON, or if the calculation fails, then SD_THLOB may supply the missing value.
Value	Number
Default	

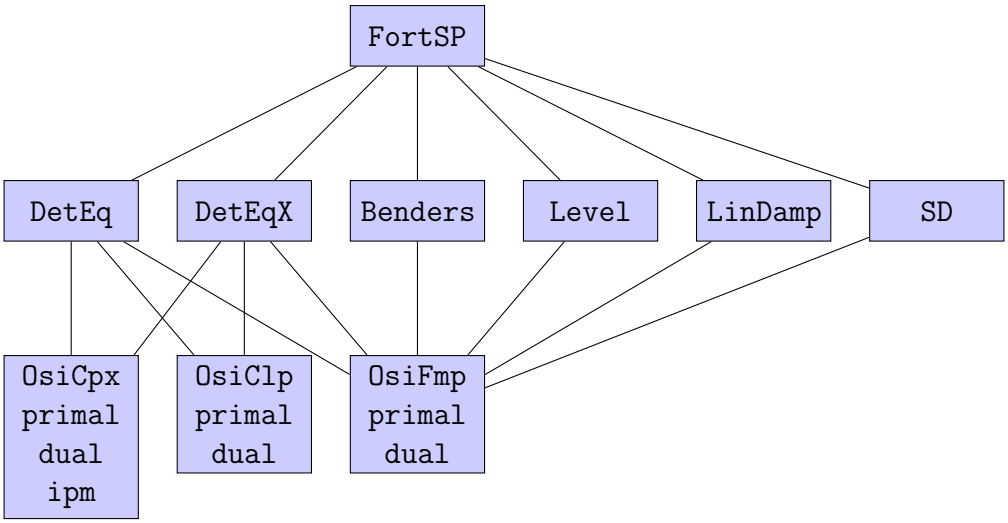


Figure 4: FortSP Algorithms and Solvers

6 Solver Options

6.1 Solvers Available

FortSP has a powerful plug-in system that allows to connect it to different LP solvers through the COIN-OR (Lougee-Heimer, 2003) Open Solver Interface. On the Windows platform a plug-in is a dynamically linked library (DLL) that provides access to a single solver. Currently there are 3 plug-in DLLs: `OsiClp.dll` for CLP, `OsiCpx.dll` for CPLEX and `OsiFmp.dll` for FortMP. The current solver plug-in can be selected using the `Solver` option which takes on the plug-in filename with optional extension as a value.

Figure 4 illustrates which combinations of algorithms and plug-ins are supported in FortSP. Compatible modules are connected by arcs so, for example, it is possible to solve deterministic equivalent problems with any solver and LP algorithm while for Benders' decomposition only FortMP is currently supported.

6.2 Solver Options and Controls

Options for LP or QP solver execution are as follows:

Opt-file Name	
SAMPL Name	`Solver`
Description	Solver plug-in filename
Value	String
Default	`OsiFmp`

Opt-file Name	DEQ_ALGORITHM
SAMPL Name	LPAlg
Description	This option specifies which LP algorithm should be used to solve a deterministic equivalent problem and all linear programming subproblems that are constructed in the course of solving the SP problem. When using the option-file, option DEQ_ALGORITHM applies only to deterministic equivalent, while USE_IPM applies more generally.
Value	The possible values for this option are listed in the table below.

Opt-file Name	SAMPL Name	Description
	Auto	The algorithm is chosen automatically (default)
SSX	Primal	Primal simplex method
	Dual	Dual simplex method
IPM	Ipm	Interior point method

Default	Auto

Opt-file Name	USE_IPM
Description	Flag specifying whether to use interior-point method
Value	Boolean
Default	OFF

Opt-file Name	BASIS_RESTART
SAMPL Name	WarmStart
Description	Flag specifying whether to use warm start
Value	Boolean
Default	ON

Opt-file Name	SOLVER_CPLEX
Description	Flag specifying whether to use CPLEX
Value	Boolean
Default	OFF

Opt-file Name	`USE_FORTMP_SPECS`
SAMPL Name	`UseFortMPSpecs`
Description	Flag specifying whether to use extra SPECS-command file (only with the FortMP solver.) A SPECS command file with the name `fortmp.spc` may be used to refine the options when FortMP is the solver in use. See the FortMP manual (Ellison et al., 2008). Commands are to be provided in sections corresponding to the type of sub-problem that is being solved, according to the following table:

Section ID	Description
ALL	Section that applies to every call to the solver. Must appear first in the SPECS file.
DeqImna	Section to handle Deterministic Equivalent - Implicit NA
DeqExna	Section to handle Deterministic Equivalent - Explicit NA
ExpVal	Section to handle Expected Value solutions
Wsprob	Section to handle Wait and See scenario sub-problems
BendRoot	Section to handle Benders root-node sub-problem solutions (multi-stage)
BendNode	Section to handle Benders node sub-problem solutions other that root or leaf (multi-stage)
BendLeaf	Section to handle Benders leaf sub-problem solutions with no warm restart (multi-stage)
BenRLeaf	Section to handle Benders leaf sub-problem solutions with warm restart (multi-stage)
Ben2Mast	Section to handle Benders master-problem solutions (two-stage)
Ben2Sprb	Section to handle Benders sub-problem solutions (two-stage)
LevelQP	Section to handle Benders Level-method QP solutions (two-stage)

	The section ID is named in a `BEGIN` line - e.g. `BEGIN (DeqImna)` - which is followed by the SPECS commands for that section. Each section is terminated with a line `END`.
Value	Boolean
Default	`OFF`

On the option-file the solver is named in the keyword - to be `ON` or `OFF` - while in SAMPL the corresponding name of the plug-in DLL is to be named. Default in either case is FortMP, whose plug-in name is `OsiFmp`. Other possibilities are `OsiCpx` for CPLEX and `OsiClp` for CLP. In the former case CPLEX DLL has to be available.

7 Output Files and Logging

Filename for the output file makes use of the generic name (or basename) unless specifically given by the OUTPUT_FILE option. The form of default output filename is as follows

<basename>.sol

7.1 Output and Log Filenames

Outputs from the system comprise two files as follows

- Solution-file:- Giving the model-type solutions that are requested with status and values for both primal and dual solutions. This is limited by default to values for the first stage only, extendable to further stages by option.
- Log-file:- Giving the outline of processing carried out and diagnostics of any unusual events or errors occurring.

Filename for these files make use of the 'generic' name (or basename), as in the case of default input files. With this name referred to as <model> the output filenames are:

- <model>.sol
- <model>.log

In SAMPL the logging is turned off by default. It can be turned on by setting the LogFile option.

7.2 Output Controls and Options

Options for solution output and logging are as follows:

Opt-file Name	OUTPUT_FILE
Description	Actual name of the output file
Value	String
Default	

Opt-file Name	LOG_FILE
SAMPL Name	LogFile
Description	Actual name of the log file
Value	String
Default	

Opt-file Name	LAST_STAGE_OUTPUT
SAMPL Name	LastStageOutput
Description	Last stage for which scenario values are required
Value	Integer
Default	1

Opt-file Name	BEN_LOG_PRINT
SAMPL Name	BenLogPrint
Description	Code for items to be logged (Benders multi-stage only) Additional logged output can be generated with the option BEN_LOG_PRINT. This should be used with caution as the log-file can easily be swamped. Certain values of use are: • 3 - for solution status of each node (plus the default) • 19 - for details of the node tree (plus the above) • 95 - for description of every cut applied (plus the above) with option 3 the output volume may be reduced by specifying BEN_LOG_FREQUENCY - that is the interval to leave between node solution-status logs.
Value	Integer
Default	1

Opt-file Name	BEN_LOG_FREQUENCY
SAMPL Name	BenLogFreq
Description	Logged every this number of passes (Benders multi-stage only)
Value	Integer
Default	1

References

Ariyawansa, K. A. and Felt, A. J. (2004). On a new collection of stochastic linear programming test problems. *INFORMS Journal on Computing*, 16(3):291–299.

Birge, J. R., Dempster, M. A. H., Gassmann, H. I., Gunn, E. A., King, A. J., and Wallace, S. W. (1987). A standard input format for multiperiod stochastic linear programs. *COAL Newsletter*, 17:1–19.

Birge, J. R. and Louveaux, F. V. (1997). *Introduction to Stochastic Programming*. Springer-Verlag, New York.

Consigli, G. and Dempster, M. A. H. (1998). Dynamic stochastic programming for asset-liability management. *Annals of Operations Research*, 81:131–162.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213.

Ellison, E. F. D., Hajian, M., Jones, H., Levkovitz, R., Maros, I., Mitra, G., and Sayers, D. (2008). *FortMP Manual*. Brunel University: London, Numerical Algorithms Group: Oxford. http://www.optirisk-systems.com/manuals/FortmpManual.pdf.

Holmes, D. (1995). A (PO)rtable (S)tochastic programming (T)est (S)et (POSTS). http://users.iems.northwestern.edu/~jrbirge/html/dholmes/post.html.

Kall, P. and Mayer, J. (1998). On testing SLP codes with SLP-IOR. In Giannessi, F., Rapcsák, T., and Komlósi, S., editors, *New Trends in Mathematical Programming: Homage to Steven Vajda*, pages 115–135. Kluwer Academic Publishers.

Klein Haneveld, W. K. and Vlerk, M. H. (2002). Integrated chance constraints: reduced forms and an algorithm. Technical report, University of Groningen, Research Institute SOM (Systems, Organisations and Management).

König, D., Suhl, L., and Koberstein, A. (2007). Optimierung des Gasbezugs im liberalisierten Gasmarkt unter Berücksichtigung von Röhren- und Untertagespeichern. In *Sammelband zur VDI Tagung "Optimierung in der Energiewirtschaft" in Leverkusen*.

Lougee-Heimer, R. (2003). The Common Optimization INterface for Operations Research. *IBM Journal of Research and Development*, 47(1):57–66.

Valente, C., Mitra, G., Sadki, M., and Fourer, R. (2009). Extending algebraic modelling languages for stochastic programming. *Informs Journal on Computing*, 21(1):107–122.

Valente, P., Mitra, G., Poojari, C., Ellison, E. F., Di Domenica, N., Mendi, M., and Valente, C. (2008). *SAMPL/SPInE User Manual*. OptiRisk Systems. http://www.optirisk-systems.com/manuals/SpineAmplManual.pdf.

Zverovich, V., Fábián, C. I., Ellison, F., and Mitra, G. (2009). A computational study of a solver system for processing two-stage stochastic linear programming problems. Working paper, CARISMA, Brunel University.

APPENDICES

A Option and Control Summary

A.1 Principle Options and Controls

The following is a complete table of controls and options. References to the notes that follow the table are given in parenthesis.

Smps Input Options

Opt-file Name	`GENERIC_FILENAME`
Description	Specifies a stub or generic name for input and output files (i.e. file-name without any extension). A standard extension is added for each actual filename.
Value	String
Default	`SPmodel`
Opt-file Name	`CORE_FILE`
Description	Actual name of the core file
Value	String
Default	
Opt-file Name	`TIME_FILE`
Description	Actual name of the time file
Value	String
Default	
Opt-file Name	`STOCH_FILE`
Description	Actual name of the stoch file
Value	String
Default	
Opt-file Name	`OPT_DIR`
SAMPL Name	`SmpsObjSense`
Description	The sense of optimisation for SMPS problems
Value	`MIN` or `MAX`
Default	`MIN`
Opt-file Name	`SPS_WORKING_DIR`
Description	Name of the folder to which the current working directory is transferred immediately after input of the option-file has completed, and before any other input. All I/O files are located in the local working directory except where a different path is given with a specific file-name command. Files not so named take the generic name followed by a standard extension, and so are located in this directory. In the option-file (not in SAMPL) the local working directory can be changed by setting this option before opening any other I/O file.
Value	String
Default	

Algorithm Options

Opt-file Name	MODEL_HN
SAMPL Name	SolveHN
Description	Flag specifying whether to solve the here-and-now problem
Value	Boolean
Default	ON
Opt-file Name	MODEL_EV
SAMPL Name	SolveEV
Description	Flag specifying whether to solve the expected value problem
Value	Boolean
Default	OFF
Opt-file Name	MODEL_WS
SAMPL Name	SolveWS
Description	Flag specifying whether to solve the wait-and-see problem
Value	Boolean
Default	OFF
Opt-file Name	OUTPUT_EVPI
SAMPL Name	ComputeEvpi
Description	Flag specifying whether to compute the expected value of perfect information (EVPI) The expected value of perfect information requires the solution of both HN and WS models. Setting this control ON forces both the HN switch and the WS switch to be ON. EVPI is the absolute difference between the HN and WS solution objectives.
Value	Boolean
Default	OFF
Opt-file Name	OUTPUT_VSS
SAMPL Name	ComputeVss
Description	Flag specifying whether to compute the value of the stochastic solution (VSS)[3]
Value	Boolean
Default	OFF
Opt-file Name	VSS_FIX_FSTAGE
SAMPL Name	VssFStage
Description	Flag specifying whether to fix only the first stage when computing the value of the stochastic solution[3]
Value	Boolean
Default	OFF

Opt-file Name	HN_ALGORITHM
SAMPL Name	SPAlg
Description	Stochastic programming algorithm to be used
Value	The possible values for this option are listed in the table below.

Opt-file Name	SAMPL Name	Description
	Auto	The algorithm is chosen automatically (default)
DETEQI	DetEq	The deterministic equivalent problem with implicit non-anticipativity is constructed and solved
DETEQE	DetEqX	The deterministic equivalent problem with explicit non-anticipativity constraints is constructed and solved
BENDERS	Benders	Benders' decomposition
	Level	Variant of level decomposition
STDCMP		Stochastic decomposition

The default algorithm - Auto - chosen by the system - is Benders' decomposition for all recourse problems and deterministic equivalent with implicit nonanticipativity for problems containing chance constraints and integrated chance constraints. An exception to this is single-stage ICC problems, which by default are solved with the special cutting plane algorithm. All other CC and ICC problems are solved only with deterministic equivalent, and any specification for Benders' or stochastic decomposition is ignored.

Default	Auto

Opt-file Name	MAX_TIME
SAMPL Name	MaxTime
Description	Time limit in seconds
Value	Nonnegative number
Default	3600

[3]VSS - Value of Stochastic Solution - requires the solution of both HN and EV models. Setting this control ON forces both the HN switch and the EV switch to be ON. In order to calculate VSS we need to know the EEV - Expected value of the Expected Value solution. EEV is calculated by solving the EV model, fixing the result so obtained in all the WS models (all stages but the last), which are then solved to give a probability-weighted average value for the objective - which is the VSS. Option VSS_FIX_FSTAGE can be used to restrict the fix that is performed to first stage variables only (although in theory this is not correct, the theoretical result is often meaningless as a complete fix may be infeasible).

Options for Benders' Decomposition

Opt-file Name	BEN_LEVEL_DECOMP
Description	Flag specifying whether to use level decomposition. In SAMPL level decomposition is enabled by setting the SPAlg option to Level.
Value	0 or 1
Default	1

Opt-file Name	BEN_PREPROC_EXP
SAMPL Name	BenPPExpVal
Description	Flag specifying whether to obtain the initial first stage solution by solving the EV problem
Value	Boolean
Default	ON

Opt-file Name	BEN_FFFB
SAMPL Name	BenFfb
Description	Flag specifying whether to use fast forward, fast back method for multi-stage
Value	Boolean
Default	OFF

Opt-file Name	BEN_THLOB
SAMPL Name	BenThetaLB
Description	Lower bound for θ used when necessary to avoid unbounded situations. In certain cases the addition of optimality cuts creates an unbounded situation as θ is a free variable. As an ad-hoc fix for this, a large negative lower bound is applied to θ, which is retained until no longer needed. If not large enough then the Benders algorithm may halt prematurely with a final condition Cycling (status 6 or larger). It may then be possible to obtain a correct solution by specifying a lower value for this option, for example -100000.
Value	Number
Default	-10000

Opt-file Name	BEN_CUT_FACTOR
SAMPL Name	BenCutFactor
Description	Maximum cuts per child scenario
Value	Integer
Default	20

Opt-file Name	BEN_MAX_ITER
SAMPL Name	BenMaxIter
Description	Iteration limit for Benders' decomposition
Value	Nonnegative integer
Default	10000

Options for Stochastic Decomposition

Opt-file Name	SD_MAX_ITER
SAMPL Name	SDMaxIter
Description	Maximum iterations
Value	Integer
Default	10000
Opt-file Name	SD_MAX_SCEN
SAMPL Name	SDMaxScen
Description	Maximum scenarios
Value	Integer
Default	1000
Opt-file Name	SD_MAX_DVD
SAMPL Name	SDMaxDvd
Description	Maximum dual vertex - deterministic
Value	Integer
Default	1000
Opt-file Name	SD_MAX_DVS
SAMPL Name	SDMaxDvs
Description	Maximum dual vertex - stochastic
Value	Integer
Default	10000
Opt-file Name	SD_MAX_INFEZ
SAMPL Name	SDMaxInf
Description	Maximum infeasibility cuts
Value	Integer
Default	5
Opt-file Name	SD_EXP_VAL
SAMPL Name	SDExpVal
Description	Flag specifying whether to obtain the initial first stage solution by solving the EV problem
Value	Boolean
Default	ON
Opt-file Name	SD_INPUT_LOBND
SAMPL Name	SDInputLo
Description	Flag specifying whether to input θ lower bound (if not then auto-calculated)
Value	Boolean
Default	OFF

Opt-file Name	SD_LOBND
SAMPL Name	SDLoBnd
Description	Lower bound for θ In the SD algorithm a probable lower bound is calculated when SD_INPUT_LOBND is OFF. If SD_INPUT_LOBND is ON, or if the calculation fails, then SD_THLOB may supply the missing value.
Value	Number
Default	

Solver Options

Opt-file Name	
SAMPL Name	`Solver`
Description	Solver plug-in filename
Value	String
Default	`OsiFmp`

Opt-file Name	`DEQ_ALGORITHM`
SAMPL Name	`LPAlg`
Description	This option specifies which LP algorithm should be used to solve a deterministic equivalent problem and all linear programming subproblems that are constructed in the course of solving the SP problem. When using the option-file, option `DEQ_ALGORITHM` applies only to deterministic equivalent, while `USE_IPM` applies more generally.
Value	The possible values for this option are listed in the table below.

Opt-file Name	SAMPL Name	Description
	`Auto`	The algorithm is chosen automatically (default)
`SSX`	`Primal`	Primal simplex method
	`Dual`	Dual simplex method
`IPM`	`Ipm`	Interior point method

Default	`Auto`

Opt-file Name	`USE_IPM`
Description	Flag specifying whether to use interior-point method
Value	Boolean
Default	`OFF`

Opt-file Name	`BASIS_RESTART`
SAMPL Name	`WarmStart`
Description	Flag specifying whether to use warm start
Value	Boolean
Default	`ON`

Opt-file Name	`SOLVER_CPLEX`
Description	Flag specifying whether to use CPLEX
Value	Boolean
Default	`OFF`

Opt-file Name `USE_FORTMP_SPECS`
SAMPL Name `UseFortMPSpecs`
Description Flag specifying whether to use extra SPECS-command file (only with the FortMP solver.)
A SPECS command file with the name `fortmp.spc` may be used to refine the options when FortMP is the solver in use. See the FortMP manual (Ellison et al., 2008). Commands are to be provided in sections corresponding to the type of sub-problem that is being solved, according to the following table:

Section ID	Description
ALL	Section that applies to every call to the solver. Must appear first in the SPECS file.
DeqImna	Section to handle Deterministic Equivalent - Implicit NA
DeqExna	Section to handle Deterministic Equivalent - Explicit NA
ExpVal	Section to handle Expected Value solutions
Wsprob	Section to handle Wait and See scenario sub-problems
BendRoot	Section to handle Benders root-node sub-problem solutions (multi-stage)
BendNode	Section to handle Benders node sub-problem solutions other that root or leaf (multi-stage)
BendLeaf	Section to handle Benders leaf sub-problem solutions with no warm restart (multi-stage)
BenRLeaf	Section to handle Benders leaf sub-problem solutions with warm restart (multi-stage)
Ben2Mast	Section to handle Benders master-problem solutions (two-stage)
Ben2Sprb	Section to handle Benders sub-problem solutions (two-stage)
LevelQP	Section to handle Benders Level-method QP solutions (two-stage)

The section ID is named in a `BEGIN` line - e.g. `BEGIN (DeqImna)` - which is followed by the SPECS commands for that section. Each section is terminated with a line `END`.
Value Boolean
Default `OFF`

Output Options

Opt-file Name	OUTPUT_FILE
Description	Actual name of the output file
Value	String
Default	

Opt-file Name	LOG_FILE
SAMPL Name	LogFile
Description	Actual name of the log file
Value	String
Default	

Opt-file Name	LAST_STAGE_OUTPUT
SAMPL Name	LastStageOutput
Description	Last stage for which scenario values are required
Value	Integer
Default	1

Opt-file Name	BEN_LOG_PRINT
SAMPL Name	BenLogPrint
Description	Code for items to be logged (Benders multi-stage only) Additional logged output can be generated with the option BEN_LOG_PRINT. This should be used with caution as the log-file can easily be swamped. Certain values of use are: • 3 - for solution status of each node (plus the default) • 19 - for details of the node tree (plus the above) • 95 - for description of every cut applied (plus the above) with option 3 the output volume may be reduced by specifying BEN_LOG_FREQUENCY - that is the interval to leave between node solution-status logs.
Value	Integer
Default	1

Opt-file Name	BEN_LOG_FREQUENCY
SAMPL Name	BenLogFreq
Description	Logged every this number of passes (Benders multi-stage only)
Value	Integer
Default	1

Execution command

The execution command may have an argument naming the option file or SAMPL script file - fully qualified by the path if different from the current path. By default this file is `fortsp.opt` in the current working directory.

Keyword and Name Format

On the option file both keywords and text-type values are not case-sensitive. Underline separators may be omitted, and in three-part keywords only the first letter of the third part is significant. In SAMPL the option names and text-type values are case-sensitive and must be given exactly.

Many controls are simple switches that have values `ON` or `OFF` in the option file. In SAMPL these controls have values 0 signalling `OFF` and 1 signalling `ON`.

A.2 Miscellaneous commands

A number of additional commands are available that are designed mainly for research purposes. There are also aliases provided to allow for existing usages of the system. A description will be given here in forthcoming versions of this manual.

B Known Weaknesses

1. SMPS fortmat as used by FortSP relies on indices for scenario- block- and stage-names. An alphabetic prefix is allowed, which makes the great majority of existing SMPS data-files intelligible, but the prefix is ignored when identifying a name. Indices must be consecutive, beginning at 1.

2. On the SMPS Stoch-file random values for the objective, RHS, bound-set and range-set are to be specified with keywords [4]. Consequently these keywords may not be used in the Core-file, and the actual core-file names for these objects are unknown to the Stoch-file input routine.

3. A free-form capability for Core files that allows 15-character numeric fields is available, but this does not extend to Time and Stoch files.

4. SAMPL support is experimental and many features are not supported in this release.

5. There is no option in SAMPL for Stochastic Decomposition. This method must be specified with an option file, and does not so far permit ancillary algorithms or statistical measures to be calculated.

6. Execution time is heavily dependent on the total number of scenarios. If this is very large the deterministic equivalent solvers become impossible to use and Benders' solver may become too lengthy for the user's satisfaction. Stochastic decomposition can handle many more scenarios, especially with the INDEP and BLOCKS forms for the STOCH file, but the accuracy of solution is subject to uncertainty, and the stopping criteria still need to be verified for many problems.

7. A procedure to select a useful subset of scenarios by importance or by Monte-Carlo sampling has been programmed but is not yet tested to any extent.

8. The Benders' solver is designed for Markovian stochastic data in which the interaction between stages in the constraint matrix forms a stair pattern. This means that any one stage is constrained directly by the previous stage decisions only and not by decisions earlier than the previous stage. Two-stage models are Markovian. Non-Markovian multi-stage models should also be solvable with Benders, but, to our knowledge, no mathematical proof has been published.

9. In rare cases Benders' solver may be halted by a cycling status with the true optimum solution not yet reached (see the note on the `BenThetaLB` option). Cycling may also result from attempt to solve a non-Markovian model, or from degeneracy if a feasible solution is difficult to find.

[4] The core-file objective name can also be used for random objective values.

C Examples of Use

C.1 An Example Using the Option File

The following is a simple example of a 4-stage model:

Core File (MYSP.cor)

```
NAME          MYSP      (MIN)
ROWS
  N Z
  G R1
  G R2
  G R3
  L R4
  G R5
  G R6
  L R7
  G R8
  G R9
  L R10
COLUMNS
    X1        Z                3.0   R1               1.0
    X1        R2               1.0
    X2        Z                2.0   R1               1.0
    Y1        Z              -15.0   R2              -3.0
    Y1        R3               1.0   R4               1.0
    Y1        R5               2.0
    Y2        Z              -12.0   R5              -1.0
    Y2        R6               1.0   R7               1.0
    Y2        R8               3.0
    Y3        Z               -4.0   R8              -1.0
    Y3        R9               1.0   R10              1.0
RHS
    RHS1      R3               3.2   R4               4.0
    RHS1      R6               3.2   R7               7.0
    RHS1      R9               1.0   R10              1.0
ENDATA
```

Time File (MYSP.tim)

```
TIME          MYSP
PERIODS
    X1        R1                    STAGE001
    Y1        R2                    STAGE002
    Y2        R5                    STAGE003
    Y3        R8                    STAGE004
ENDATA
```

Stoch File (MYSP.sto)

```
STOCH         MYSP
SCENARIOS     DISCRETE
 SC SCEN0001  ROOT      0.125          STAGE001
    RHS       R3             3.2
    RHS       R4             4.0
    RHS       R6             3.2
    RHS       R7             7.0
    RHS       R9             4.0
    RHS       R10            8.0
 SC SCEN0002  SCEN0001  0.125          STAGE004
    RHS       R9             3.0
    RHS       R10            6.0
 SC SCEN0003  SCEN0001  0.125          STAGE003
    RHS       R6             4.8
    RHS       R7             9.0
    RHS       R9             9
    RHS       R10            12.0
 SC SCEN0004  SCEN0003  0.125          STAGE004
    RHS       R9             6.4
    RHS       R10            18.0
 SC SCEN0005  SCEN0001  0.125          STAGE002
    RHS       R3             1.2
    RHS       R4             4.0
    RHS       R6             2.2
    RHS       R7             7.5
    RHS       R9             3.0
    RHS       R10            6.0
 SC SCEN0006  SCEN0005  0.125          STAGE004
    RHS       R9             4.0
    RHS       R10            9.0
 SC SCEN0007  SCEN0005  0.125          STAGE003
    RHS       R6             2.8
    RHS       R7             6.0
    RHS       R9             5.2
    RHS       R10            12.0
 SC SCEN0008  SCEN0007  0.125          STAGE004
    RHS       R9             4.4
    RHS       R10            8.0
ENDATA
```

Option File (MYSP.opt)

The following option file causes all forms of output to be generated:

```
INPUT_TYPE SMPS
OPT_DIR MIN
SPS_WORKING_DIR E:\spine\QA
GENERIC_FILENAME MYSP
```

```
MODEL_EV ON
MODEL_HN ON
MODEL_WS ON
OUTPUT_EVPI ON
OUTPUT_VSS ON
HN_algorithm BENDERS
BEN_FFFB ON
BEN_PREPROC_EXPVAL ON
VSS_FIX_FSTAGE ON
```

The run may be started with MYSP.opt named as the argument, or with the above file actually named as `fortsp.opt`.

Output Solution File (MYSP.sol)

Outputs from the run are as follows:

```
WS
WS Scenario = 1
Obj         -140, Prob = 0.125
STATUS = 3: Optimal LP solution
Variables
  Name Index Stage     Value         D.val           Lob        Upb
  X1       1   1            12             0             0       1e+035
  X2       2   1             0             2             0       1e+035
Constraints
  Name Index Stage    SPrice        RowAct           Lhs        Rhs
  R1       2   1             0            12             0       1e+035
END
```

```
WS Scenario = 2
Obj         -132, Prob = 0.125
STATUS = 3: Optimal LP solution
Variables
  Name Index Stage     Value         D.val           Lob        Upb
  X1       1   1            12             0             0       1e+035
  X2       2   1             0             2             0       1e+035
Constraints
  Name Index Stage    SPrice        RowAct           Lhs        Rhs
  R1       2   1             0            12             0       1e+035
END

. . . .
Etc, repeated as above for scenarios 3, 4, 8
. . . .

WS Summary
Obj         -149
STATUS = 3: Optimal LP solution
END
```

```
ExpVal
Obj         -152
STATUS = 3: Optimal LP solution
Variables
  Name Index Stage     Value         D.val           Lob        Upb
  X1       1   1           12             0             0       1e+035
  X2       2   1            0             2             0       1e+035
Constraints
  Name Index Stage    SPrice        RowAct           Lhs        Rhs
  R1       2   1            0            12             0       1e+035
END

HN = BENDERS
Obj          -149
STATUS = 3: Optimal LP solution
Variables
 Name Index  Stage  Scen      Value      D.val         Lob      Upb
  X1    1      1      1         12          0           0       1e+035
  X2    2      1      1          0          2           0       1e+035
Constraints
 Name Index  Stage  Scen      Value      D.val         Lhs      Rhs
  R1    2      1      1         12          0           0       1e+035
END

EVPI =            0
VSS =            0
```

The output shown above includes only 1st stage values and comprises:

- The WS solution for each scenario
- The WS summary result
- The EV solution
- The HN solution
- Values of the statistical measures EVPI and VSS

The statistical measures EVPI and VSS are both zero in this simple example since the values of EEV and WS are both equal to the HN objective. Note that if the option VSS_FIX_FSTAGE ON had been omitted, then the EEV would in fact have been infeasible and VSS would have been infinite. Theoretically $VSS = 0$ is not correct but this value is probably more important to a user than $VSS = infinity$.

Output Log File (MYSP.log)

```
INPUT_TYPE SMPS
OPT_DIR MIN
SPS_WORKING_DIR E:\spine\QA
GENERIC_FILENAME MYSP
MODEL_EV ON
MODEL_HN ON
MODEL_WS ON
OUTPUT_EVPI ON
OUTPUT_VSS ON
HN_algorithm BENDERS
BEN_FFFB ON
BEN_PREPROC_EXPVAL ON
VSS_FIX_FSTAGE ON

Specification file read
Current Working directory: E:\spine\QA
FORTMP release version 3.02g, Aug 2002
License expires on (y)2010:(m)01:(d)01.
License expires on (y)2010:(m)01:(d)01.
FORTMP release version 3.02g, Aug 2002
License expires on (y)2010:(m)01:(d)01.
TIME TAKEN FOR INPUT MPS   =     0.00 SECS,  TOTAL SO FAR =     0.00 SECS

**** PROBLEM NAME IS: MYSP    , GENERIC FILE: MYSP ****

CORE DIMENSIONS: NR= 11, NC = 5, NNZ = 19
STOCH DIMENSIONS:- ANT = 4, ANS = 8, AND = 28
Stoch representation code :1
Number of Scenarios to process = 8
====================================
Time period 1 has 1 rows 2 columns and 2 nonzeroes
Time period 2 has 3 rows 3 columns and 4 nonzeroes
Time period 3 has 3 rows 2 columns and 4 nonzeroes
Time period 4 has 3 rows 2 columns and 4 nonzeroes
============================================

ALGORITHM:- WAIT & SEE
**********************
WS Scenario 1, OBJ = -140, STATUS = 3: Optimal LP solution
                Contributes -17.5 to final objective
WS Scenario 2, OBJ = -132, STATUS = 3: Optimal LP solution
                Contributes -16.5 to final objective
Etc, repeated as above for scenarios 3, 4, 8
. . . .
FINAL WS OBJECTIVE = -149, STATUS = 3: Optimal LP solution
Total time in WS:0

ALGORITHM:- EXPVAL
```

```
**********************
EXPVAL OBJECTIVE = -152, STATUS = 3: Optimal LP solution
Total time in ExpVal:0

ALGORITHM:- WS FOR EEV
**********************
WS Scenario 1, OBJ = -140, STATUS = 3: Optimal LP solution
                Contributes -17.5 to final objective
WS Scenario 2, OBJ = -132, STATUS = 3: Optimal LP solution
                Contributes -16.5 to final objective
Etc, repeated as above for scenarios 3, 4, 8
. . . .
FINAL EEV OBJECTIVE = -149, STATUS = 3: Optimal LP solution
Total time in WS:0
```

```
ALGORITHM:- BENDERS DECOMP
**************************
------------------ NODE DIMENSIONS ------------------
Time-stage   1, Sub-model:   NR=    1, NC=    2, NAIJ=     2
                Sub-problem: NR=   44, NC=    3, NAIJ=   125
Time-stage   2, Sub-model:   NR=    3, NC=    3, NAIJ=     4
                Sub-problem: NR=   45, NC=    2, NAIJ=    85
Time-stage   3, Sub-model:   NR=    3, NC=    4, NAIJ=     4
                Sub-problem: NR=   45, NC=    2, NAIJ=    85
Time-stage   4, Sub-model:   NR=    3, NC=    5, NAIJ=     4
                Sub-problem: NR=    5, NC=    2, NAIJ=     5
Maximum -       Sub-model:   NR=    3, NC=    5, NAIJ=     4
                Sub-problem: NR=   45, NC=    3, NAIJ=   125
Total node Col-vector size = 31(*2)
Total node Row-vector size = 354(*3)
Total node basis size      = 36
=====================================================

ALGORITHM:- EXPVAL
**********************
EXPVAL OBJECTIVE = -152, STATUS = 3: Optimal LP solution
Total time in ExpVal:0

END OF ITER# 1, STAGES 1 TO 4, OBJECTIVE = -112.5
END OF ITER# 2, STAGES 3 TO 3, OBJECTIVE = -112.5
END OF ITER# 3, STAGES 2 TO 2, OBJECTIVE = -112.5
Lower Bound -10000 applied to THETA
END OF ITER# 4, STAGES 1 TO 4, OBJECTIVE = 2526.59
END OF ITER# 5, STAGES 2 TO 2, OBJECTIVE = 2526.59
END OF ITER# 6, STAGES 1 TO 4, OBJECTIVE = -147.364
END OF ITER# 7, STAGES 1 TO 4, OBJECTIVE = -149
NEW BENDERS OPTION: FFSB
END OF ITER# 8, STAGES 1 TO 4, OBJECTIVE = -149
```

```
FINAL BENDECOMP OBJECTIVE = -149, STATUS = 3: Optimal LP solution
FINAL EVPI =            0
FINAL VSS =            0
===============
Total time =     0
===============
```

This file records the original options, the events of the run and important dimensions of the models to be solved. Final results are recorded as on the solution file, together with run-times.

C.2 Another Example Using the Option File

In the following option file only the HN model is requested, using the alternative Deterministic Equivalent (implicit) algorithm. Data is the same as before. Output is requested for all stages.

Option File (MYSP.opt)

```
INPUT_TYPE SMPS
OPT_DIR MIN
SPS_WORKING_DIR E:\spine\QA
GENERIC_FILENAME MYSP
MODEL_EV OFF
MODEL_HN ON
MODEL_WS OFF
OUTPUT_EVPI OFF
OUTPUT_VSS OFF
HN_algorithm DETEQI
LAST_STAGE_OUTPUT 4
```

Output Solution File (MYSP.sol)

```
HN = DETEQI
Obj          -149
STATUS = 3: Optimal LP solution
Variables
Name Index  Stage  Scen      Value      D.val        Lob      Upb
  X1   1      1      1         12          0          0       1e+035
  X2   2      1      1          0          2          0       1e+035
  Y1   3      2      1          4          0          0       1e+035
  Y1   3      2      5          4          0          0       1e+035
  Y2   4      3      1          7          0          0       1e+035
  Y2   4      3      3          8          0          0       1e+035
  Y2   4      3      5        7.5          0          0       1e+035
  Y2   4      3      7          6          0          0       1e+035
  Y3   5      4      1          8          0          0       1e+035
  Y3   5      4      2          6          0          0       1e+035
  Y3   5      4      3         12          0          0       1e+035
  Y3   5      4      4         18          0          0       1e+035
```

Y3	5	4	5	6	0	0	1e+035
Y3	5	4	6	9	0	0	1e+035
Y3	5	4	7	12	0	0	1e+035
Y3	5	4	8	8	0	0	1e+035

Constraints

Name	Index	Stage	Scen	Value	D.val	Lhs	Rhs
R1	2	1	1	12	0	0	1e+035
R2	3	2	1	0	-3	0	1e+035
R3	4	2	1	4	0	3.2	1e+035
R4	5	2	1	4	4.5	-1e+035	4
R2	3	2	5	0	0	0	1e+035
R3	4	2	5	4	0	1.2	1e+035
R4	5	2	5	4	7.5	-1e+035	4
R5	6	3	1	1	0	0	1e+035
R6	7	3	1	7	0	3.2	1e+035
R7	8	3	1	7	3	-1e+035	7
R5	6	3	3	0	-3	0	1e+035
R6	7	3	3	8	0	4.8	1e+035
R7	8	3	3	8	0	-1e+035	9
R5	6	3	5	0.5	0	0	1e+035
R6	7	3	5	7.5	0	2.2	1e+035
R7	8	3	5	7.5	3	-1e+035	7.5
R5	6	3	7	2	0	0	1e+035
R6	7	3	7	6	0	2.8	1e+035
R7	8	3	7	6	3	-1e+035	6
R8	9	4	1	13	0	0	1e+035
R9	10	4	1	8	0	4	1e+035
R10	11	4	1	8	0.5	-1e+035	8
R8	9	4	2	15	0	0	1e+035
R9	10	4	2	6	0	3	1e+035
R10	11	4	2	6	0.5	-1e+035	6
R8	9	4	3	12	0	0	1e+035
R9	10	4	3	12	0	9	1e+035
R10	11	4	3	12	0.5	-1e+035	12
R8	9	4	4	6	0	0	1e+035
R9	10	4	4	18	0	6.4	1e+035
R10	11	4	4	18	0.5	-1e+035	18
R8	9	4	5	16.5	0	0	1e+035
R9	10	4	5	6	0	3	1e+035
R10	11	4	5	6	0.5	-1e+035	6
R8	9	4	6	13.5	0	0	1e+035
R9	10	4	6	9	0	4	1e+035
R10	11	4	6	9	0.5	-1e+035	9
R8	9	4	7	6	0	0	1e+035
R9	10	4	7	12	0	5.2	1e+035
R10	11	4	7	12	0.5	-1e+035	12
R8	9	4	8	10	0	0	1e+035

```
  R9   10     4      8         8          0          4.4       1e+035
  R10  11     4      8         8        0.5       -1e+035            8
END
```

Output Log File (MYSP.log)

```
INPUT_TYPE SMPS
OPT_DIR MIN
SPS_WORKING_DIR E:\spine\QA
GENERIC_FILENAME MYSP
MODEL_EV OFF
MODEL_HN ON
MODEL_WS OFF
OUTPUT_EVPI OFF
OUTPUT_VSS OFF
HN_algorithm DETEQI
LAST_STAGE_OUTPUT 4

Specification file read
Current Working directory: E:\spine\QA
FORTMP release version 3.02g, Aug 2002
License expires on (y)2010:(m)01:(d)01.
License expires on (y)2010:(m)01:(d)01.
FORTMP release version 3.02g, Aug 2002
License expires on (y)2010:(m)01:(d)01.
TIME TAKEN FOR INPUT MPS   =     0.00 SECS,  TOTAL SO FAR =     0.00 SECS

**** PROBLEM NAME IS: MYSP    , GENERIC FILE: MYSP ****

CORE DIMENSIONS: NR= 11, NC = 5, NNZ = 19
STOCH DIMENSIONS:- ANT = 4, ANS = 8, AND = 28
Stoch representation code :1
Number of Scenarios to process = 8
====================================
Time period 1 has 1 rows 2 columns and 2 nonzeroes
Time period 2 has 3 rows 3 columns and 4 nonzeroes
Time period 3 has 3 rows 2 columns and 4 nonzeroes
Time period 4 has 3 rows 2 columns and 4 nonzeroes
========================================



ALGORITHM:- DETEQ with IMPLICIT NA
**********************************
DETEQI DIMENSIONS: NR=43, NC=16, NAIJ=58
FORTMP release version 3.02g, Aug 2002
License expires on (y)2010:(m)01:(d)01.
 ASGNFM: MREQ=    220952(Byte), OFFset=   10223672(Byte)
TIME TAKEN FOR INPUT/SETUP =   0.00 SECS,  TOTAL SO FAR =   0.00 SECS
```

```
SCALING IN PROGRESS ...
SCALING COMPLETE
TIME TAKEN FOR SCALE/PRSLVE=   0.00 SECS,  TOTAL SO FAR =    0.00 SECS
 CRASH(LTSF) ENDED.  VARIABLE TYPES:-  PLUS    BNDD     FIX    FREE
    LOGICALS REMOVED FROM BASIS:-        16       0       0       0
    STRUCTURALS ENTERED IN BASIS:-       16       0       0       0
 CRASH(ART) ENDED:   1 PASSES:      0 ARTIFICIALS,       0 PIVOTED OUT
TIME TAKEN FOR CRASHING    =   0.00 SECS,  TOTAL SO FAR =   0.00 SECS
 FEASIBLE BASIS REACHED AFTER ITERATION     10
 Invert demand:  Obj =-149.000      Suminf = 0.00000      ITER#     22
 STATUS = 3  -- OPTIMUM SOLUTION FOUND.    -149.000      ITER#     22
TIME TAKEN FOR PRIMAL      =   0.00 SECS,  TOTAL SO FAR =   0.00 SECS
TIME TAKEN FOR OUTPUT      =   0.00 SECS,  TOTAL SO FAR =   0.00 SECS
FINAL DETEQI OBJECTIVE = -149          , STATUS = 3: Optimal LP solution
===============
Total time =     0
===============
```

The above log includes logged output from the internal LP solver for the HN model, which is suppressed in the earlier example owing to the large volume that would be shown otherwise.

C.3 An Example in SAMPL Using SMPS Input

In this section we will solve the STORM problem from the POSTS collection (Holmes, 1995). It is a two-stage problem with stochasticity in the right-hand side. Since SMPS doesn't specify how to control the solver and present the output you will need to provide a script file in another format supported by FortSP which is SAMPL. Below is an example of such script:

```
# Set the options.
option SPAlg DetEq, Solver OsiClp;
option ComputeEvpi 1, ComputeVss 1, VssFStage 1;

# Import the problem in the SMPS format.
read smps('stormG2.cor', 'stormG2_27.sto', 'stormG2.tim');

# Solve the problem.
solve;

# Print the results.
print 'Optimal value =', OBJ;
print 'EVPI =', evpi;
print 'VSS =', vss;
print;
print 'First-stage solution:';
print {c in _SMPS_COLS}: c, '=', _smps_var[c], '\
';
print {r in _SMPS_ROWS}: r, '=', _smps_con[r].body, '\
';
```

Note that the SP algorithm is set to `DetEq` which means that the deterministic equivalent will be constructed and solved using the current solver. In this example we set the solver to `OsiClp`. The deterministic equivalent approach doesn't scale well with increasing problem dimensions and the number of scenarios as shown in Appendix D. Therefore, for solving larger instances of the STORM problem as well as other problems level or Benders' decomposition is recommended.

Running FortSP with the command `fortsp <script filename>` will produce the following output:

```
Optimal solution found
Optimal value = 15508982.31
EVPI = 32422.97538
VSS = 6004.125768

First-stage solution:
C0011901 = 0
C0012001 = 0
C0012101 = 0
C0012201 = 0
C0012301 = 0
C0012401 = 0
C0012501 = 0
C0012601 = 0
C0012701 = 0.66
C0012801 = 3.34
and 300 more lines...
```

D Performance on Test Models

The results presented in this section are taken from the computational study by Zverovich et al. (2009).

D.1 Experimental Setup

The computational experiments were performed on a Windows XP machine with Intel CORE2 2.4 GHz CPU and 3 GB of RAM. Deterministic equivalents were solved with CPLEX 11.0 dual simplex and barrier optimisers. Crossover to a basic solution was disabled for the barrier optimiser, for other CPLEX options the default values were used.

The times are reported in seconds with times of reading input files included. For simplex and IPM the times of constructing deterministic equivalent problems are also included though it should be noted that they only amount to small fractions of the total. FortMP linear and quadratic programming solver described by Ellison et al. (2008) was used to solve master problem and subproblems in the implementations of Benders decomposition and level method.

D.2 Data Sets

We considered test problems which were drawn from four different sources described in Table 13. Tables 14 – 18 give the parameters of these problems. Columns A and W of these tables give the dimensions of corresponding matrices in the following formulation of a two-stage SP problem:

$$\begin{aligned} \min \quad & \boldsymbol{c}^{\boldsymbol{T}}\boldsymbol{x} + \mathbb{E}Q(\boldsymbol{x}, \boldsymbol{\xi}) \\ \text{subject to} \quad & A\boldsymbol{x} = \boldsymbol{b}, \\ & \boldsymbol{x} \geq \boldsymbol{0}, \end{aligned}$$

where

$$\begin{aligned} Q(\boldsymbol{x}, \boldsymbol{\xi}) = \min \quad & \boldsymbol{q}^{\boldsymbol{T}}\boldsymbol{y} \\ \text{subject to} \quad & W\boldsymbol{y} = \boldsymbol{h} - T\boldsymbol{x}, \\ & \boldsymbol{y} \geq \boldsymbol{0} \end{aligned}$$

and vector $\boldsymbol{\xi}$ is composed of the random components of $\boldsymbol{h}, T, W$ and $\boldsymbol{q}$.

NNZ denotes the number of nonzero matrix elements. Optimal values reported in column Opt were obtained using level method. For the WATSON problems the optimal values of their two-stage approximations are specified in this column.

It should be noted that the problems generated with GENSLP do not possess any internal structure inherent in real-world problems. However they are still useful for the purposes of comparing scale-up properties of algorithms.

D.3 Computational Results

The computational results are presented in Tables 19 – 23. Iter denotes the number of iterations. For Benders decomposition and level method these are the numbers of master iterations.

Finally we present the results in the form of performance profiles. The performance profile for a solver is defined by Dolan and Moré (2002) as the cumulative distribution function

	Source	Reference	Comments
1.	POSTS collection	Holmes (1995)	Two-stage problems from the test set called POSTS
2.	Slptestset collection	Ariyawansa and Felt (2004)	Two-stage problems from the collection of stochastic LP test problems
3.	Random problems	Kall and Mayer (1998)	Artificial test problems generated with pseudo random stochastic LP problem generator GENSLP
4.	SAMPL problems	König et al. (2007), Valente et al. (2008)	Instances of the SAPHIR gas portfolio planning model formulated in SAMPL
5.	WATSON problems	Consigli and Dempster (1998)	WATSON pension fund management test problems

Table 13: Sources of test problems

Name	A	W	Scen	Deterministic Equivalent Matrix	NNZ	Opt
pltexpA2	62×188	104×272	6	686×1820	3703	-9.47935
			16	1726×4540	9233	-9.66234
fxm2	92×114	238×343	6	1520×2172	12139	18416.8
			16	3900×5602	31239	18416.8
stormG2	185×121	528×1259	8	4409×10193	27424	15535236
			27	14441×34114	90903	15508982
			125	66185×157496	418321	15512091
			1000	528185×1259121	3341696	15802590

Table 14: Parameters of test problems from POSTS collection

Name	A	W	Scen	Deterministic Equivalent Matrix	NNZ	Opt
AIRL2	2×4	6×8	25	152×204	604	269665
LandS	2×4	7×12	3	23×40	92	381.853
4node	14×52	74×186	16	1198×3028	7743	423.012
			32	2382×6004	15231	423.013
			64	4750×11956	30207	423.012
			128	9486×23860	60159	423.012
			256	18958×47668	120063	425.375
			512	37902×95284	239871	429.962
			1024	75790×190516	479487	434.113
			2048	151566×380980	958719	441.738
			4096	303118×761908	1917183	446.856
			8192	606222×1523764	3834111	446.856
			16384	1212430×3047476	7667967	446.856
			32768	2424846×6094900	15335679	446.856

Table 15: Parameters of test problems from Slptestset collection

Name	A	W	Scen	Deterministic Equivalent Matrix	NNZ	Opt
rand0	50×100	25×50	2000	50050×100100	754501	162.146
			4000	100050×200100	1508501	199.032
			6000	150050×300100	2262501	140.274
			8000	200050×400100	3016501	170.318
			10000	250050×500100	3770501	139.129
rand1	100×200	50×100	2000	100100×200200	3006001	244.159
			4000	200100×400200	6010001	259.346
			6000	300100×600200	9014001	297.562
			8000	400100×800200	12018001	262.451
			10000	500100×1000200	15022001	298.638
rand2	150×300	75×150	2000	150150×300300	6758501	209.151
			4000	300150×600300	13512501	218.247
			6000	450150×900300	20266501	239.720
			8000	600150×1200300	27020501	239.158
			10000	750150×1500300	33774501	231.706

Table 16: Parameters of test problems generated with GENSLP

Name	A	W	Scen	Deterministic Equivalent Matrix	NNZ	Opt
saphir	32×53	8678×3924	50	433932×196253	1136753	129506233
			100	867832×392453	2273403	129059362
			200	1735632×784853	4546703	141473266
			500	4339032×1962053	11366603	137871740
			1000	8678032×3924053	22733103	133036857

Table 17: Parameters of SAMPL problems

Name	A	W	Scen	Deterministic Equivalent Matrix	NNZ	Opt
WATSON.I	11×15	324×587	128	41483×75151	188828	-2271.17866
			256	82955×150287	377628	-2733.63695
			512	165899×300559	755228	-2810.75153
			1024	331787×601103	1510428	-2750.48955

Table 18: Parameters of two-stage approximations of WATSON problems

Name	Scen	CPLEX				FortSP			
		IPM		Simplex		Benders		Level	
		Time	Iter	Time	Iter	Time	Iter	Time	Iter
pltexpA2	6	0.06	14	0.15	329	0.04	1	0.03	1
	16	0.13	16	0.17	810	0.08	4	0.10	4
fxm2	6	0.09	17	0.24	1281	0.29	23	0.35	15
	16	0.20	23	0.47	3374	0.39	22	0.53	18
stormG2	8	0.38	28	0.32	3675	0.60	23	0.83	22
	27	3.33	27	0.87	13128	1.93	30	1.65	22
	125	12.33	57	7.00	71611	8.38	32	4.99	19
	1000	189.53	109	305.81	758078	80.20	41	34.46	18

Table 19: Solution times and iteration counts for POSTS problems

for a performance metric. We use the ratio of the solving time versus the best time as the performance metric. Let P and M be the set of problems and the set of solution methods respectively. We define by $t_{p,m}$ the time of solving problem $p \in P$ with method $m \in M$. For every pair (p, m) we compute performance ratio

$$r_{p,m} = \frac{t_{p,m}}{\min\{t_{p,m} | m \in M\}},$$

If method m failed to solve problem p the formula above is not defined. In this case we set $r_{p,m} := \infty$.

The cumulative distribution function for the performance ratio is defined as follows:

$$\rho_m(\tau) = \frac{|\{p \in P | r_{p,m} \leq \tau\}|}{|P|}$$

We calculated performance profile of each considered method on the whole set of test problems. These profiles are shown in Figure 5. The value of $\rho_m(\tau)$ gives the probability that method m solves a problem within a ratio τ of the best solver. For example according to Figure 5 level method was the first in more than 30% of cases and solved all the problems within a ratio 6 of the best time.

The notable advantages of performance profiles over other approaches to performance comparison are as follows. Firstly, they minimize the influence of a small subset of problems on the benchmarking process. Secondly, there is no need to discard solver failures. Thirdly, performance profiles provide a visualisation of large sets of test results as we have in our case.

Figure 5 illustrates that, while in most cases the performance of CPLEX barrier optimiser is better it was not able to solve some of the problems. Several large instances were not solved due to high memory requirements of constructing and solving deterministic equivalent. Other failures were caused by numerical difficulties. The performance profiles of pure Benders decomposition and linear damping are very similar to each other and the level method profile dominates both of them.

		CPLEX				FortSP			
		IPM		Simplex		Benders		Level	
Name	Scen	Time	Iter	Time	Iter	Time	Iter	Time	Iter
AIRL2	25	0.04	11	0.14	145	0.08	10	0.16	16
LandS	3	0.04	9	0.11	21	0.01	8	0.04	9
4node	16	0.19	17	0.20	1461	1.44	102	1.18	47
	32	0.65	15	0.37	3244	3.60	130	1.87	66
	64	0.70	17	0.88	6847	6.79	135	2.36	54
	128	0.71	26	2.48	13498	10.25	115	3.37	45
	256	1.53	30	9.88	27743	16.17	101	8.75	60
	512	3.38	30	41.74	54861	34.04	109	18.08	67
	1024	7.51	32	457.53	130701	69.13	110	36.34	68
	2048	17.93	36	1262.75	239159	240.25	184	63.28	59
	4096	44.95	45	11733.86	475971	538.26	215	129.57	63
	8192	79.73	45	*	*	1474.48	286	229.72	56
	16384	†	†	†	†	1850.52	194	459.27	58
	32768	†	†	†	†	5785.07	279	1029.74	65

Table 20: Solution times and iteration counts for Slptestset problems
* Failed to solve due to timeout
† Failed to solve due to insufficient memory
‡ Failed to solve due to numerical difficulties

		CPLEX				FortSP			
		IPM		Simplex		Benders		Level	
Name	Scen	Time	Iter	Time	Iter	Time	Iter	Time	Iter
rand0	2000	16.71	44	541.78	84571	62.68	80	33.73	42
	4000	30.11	40	2632.72	155926	112.91	72	59.41	37
	6000	56.90	52	8688.20	257614	287.09	124	137.20	58
	8000	83.35	57	*	*	341.97	110	171.42	55
	10000	142.82	79	*	*	831.67	219	293.24	76
rand1	2000	66.92	24	*	*	760.06	388	161.81	76
	4000	162.20	29	*	*	1786.76	496	258.74	69
	6000	252.81	30	*	*	2010.50	368	316.25	54
	8000	386.67	35	*	*	3063.53	435	544.61	72
	10000	†	†	*	*	4012.57	451	694.83	70
rand2	2000	164.18	22	*	*	5821.79	889	427.70	67
	4000	†	†	*	*	3881.73	397	451.03	43
	6000	†	†	†	†	7555.65	522	901.72	51
	8000	†	†	†	†	8678.47	478	883.14	41
	10000	†	†	†	†	15984.27	698	1536.32	60

Table 21: Solution times and iteration counts for generated problems
* Failed to solve due to timeout
† Failed to solve due to insufficient memory

Name	Scen	CPLEX				FortSP			
		IPM		Simplex		Benders		Level	
		Time	Iter	Time	Iter	Time	Iter	Time	Iter
saphir	50	‡	‡	255.03	73918	465.18	113	396.59	76
	100	‡	‡	916.04	143194	701.14	120	533.06	60
	200	‡	‡	7579.14	385231	‡	‡	2555.47	206
	500	†	†	†	†	2556.06	115	2339.76	59
	1000	†	†	†	†	4294.47	109	4650.19	78

Table 22: Solution times and iteration counts for SAMPL problems
† Failed to solve due to insufficient memory
‡ Failed to solve due to numerical difficulties

Name	Scen	CPLEX				FortSP			
		IPM		Simplex		Benders		Level	
		Time	Iter	Time	Iter	Time	Iter	Time	Iter
WATSON.I	128	1.71	33	1.44	7985	0.92	1	0.92	1
	256	3.89	38	3.66	15818	1.58	1	1.59	1
	512	8.91	47	5.88	30237	2.62	1	2.61	1
	1024	20.27	54	14.44	60941	5.12	1	5.31	1

Table 23: Solution times and iteration counts for two-stage approximations of WATSON problems

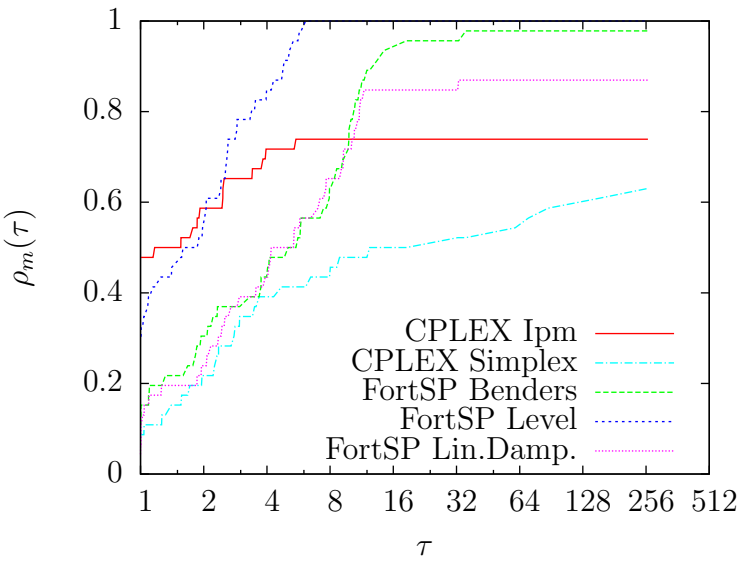


Figure 5: Performance profile in a $\log_2$ scale