TR/10/01 August 2001

Software tools for Stochastic Programming: A Stochastic Programming Integrated Environment (SPInE).

P Valente, G Mitra, C Poojari, T Kyriakis.

Department of Mathematical Sciences
Brunel University, Uxbridge, UB8 3PH

Software tools for Stochastic Programming: A Stochastic Programming Integrated Environment (SPInE)

Patrick Valente, Gautam Mitra, Chandra A. Poojari, Triphonas Kyriakis

Department of Mathematical Sciences, Brunel University, West London, UK.

Abstract

SP models combine the paradigm of dynamic linear programming with modelling of random parameters, providing optimal decisions which hedge against future uncertainties. Advances in hardware as well as software techniques and solution methods have made SP a viable optimisation tool. We identify a growing need for modelling systems which support the creation and investigation of SP problems. Our SPInE system integrates a number of components which include a flexible modelling tool (based on stochastic extensions of the algebraic modelling languages AMPL and MPL), stochastic solvers, as well as special purpose scenario generators and database tools. We introduce an asset/liability management model and illustrate how SPInE can be used to create and process this model as a multistage SP application.

1 Introduction and background

Stochastic Programming (SP) is an established approach to optimum decision making under uncertainty. SP models have proven to be more suitable than their deterministic counterparts in many applications, ranging from portfolio allocation and asset/liability management (see Mulvey and Ziemba eds. 1998) to the planning of power systems (see Pereira and Pinto 1985), from transportation (see Powell 1988) and telecommunication (see Gaivoronski 1995) to supply chain networks planning (see Eppen et al. 1989, Lucas et al. 1996, Escudero et al. 1999, Shapiro 2001). SP models combine the paradigm of dynamic linear programming with modelling of random parameters (scenario generation). The solution of such SP models leads to optimal decisions which hedge against future uncertainties.
Advances in hardware as well as software techniques and solution methods have made SP a viable optimisation tool. Unfortunately, there are not many modelling systems which support the creation and investigation of SP models. In fact, the practical exploitation of SP models runs into various difficulties. In order to understand these difficulties, it is important to first identify the key steps in the process of conceptualisation, formulation, solution and analysis of the results of mathematical programming (MP) models (Dominguez-Ballesteros et al. 1999).

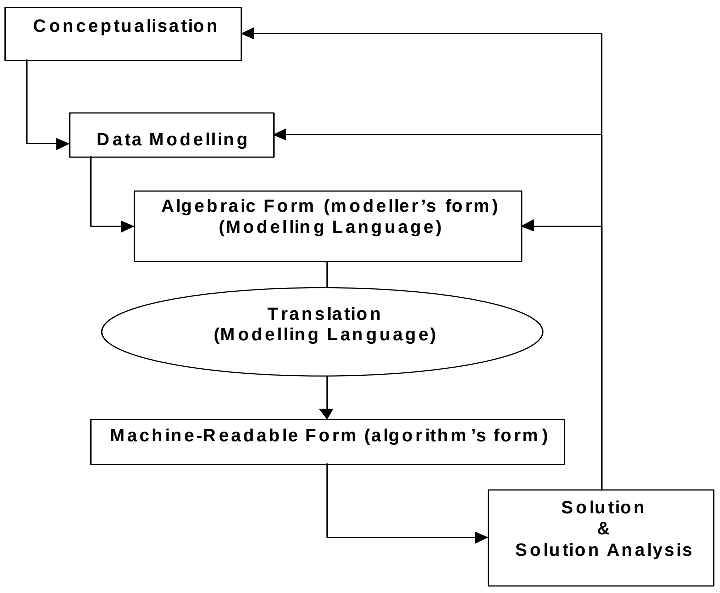

Conceptualisation
Data Modelling
Algebraic Form (modeller's form) (Modelling Language)
Translation (Modelling Language)
Machine-Readable Form (algorithm's form)
Solution & Solution Analysis

Figure 1. The modelling process.

The model conceptualisation stage involves an analysis of the real world decision problem. The available information is collected and used to understand the data and the decision-making requirement of the application under investigation. The data modelling stage involves the extraction and classification of the available information.
The categories obtained by this analysis provide the sets and parameters which are used in the algebraic formulation of the model. The algebraic formulation of the model is generally implemented using an Algebraic Modelling Language (AML), which translates the abstract modeller's form into a machine-readable form. The translation can be considered as *model instantiation*. After supplying appropriate data, the combined model and data instance can be subsequently processed by a suitable solver, which in turn produces the model's results. Modern algebraic modelling systems are

capable of supporting this entire process for Linear Programming (LP), Mixed Integer Programming (MIP), Quadratic Programming (QP), and to some extent Non-Linear Programming (NLP) models. Moreover, these systems are able to interact with corporate data warehouses and data marts stored in relational, object oriented or in other emerging standards.
The availability of algebraic modelling languages such as *AMPL* (Fourer et al. 1993), *AIMMS* (Bisschop et al. 1993), *GAMS* (Brooke et al. 1998) and *MPL* (Maximal 2000) and the corresponding model development environments have contributed in the following ways:

(i) Model development and prototyping has become a high productivity process. This has lead to widespread acceptance of optimisation by the end user community based on the proof of concept applications rapidly developed by OR/MS analysts.

(ii) There have been many examples of deployment of optimisation as an embedded inference engine in Decision Support Systems (DSS).

In many cases, moving on to SP formulation of these decision problems is a natural step forward. Unfortunately, this progress is hampered due to the lack of commensurate model development tools in the domain of Stochastic Programming. A number of investigators have reported development of SP modelling systems. These are in different stages of completion and use, and are summarised in Table 1.

Name	Affiliation	System Name
JJ Bisshop, et al.	Paragon Decision Technology	AIMMS
A Meeraus, et al.	GAMS	GAMS
B Kristjansson	Maximal Software	MPL
R Fourer, et al.	Northwestern University	AMPL
MAH Dempster, et al.	Cambridge University	STOCHGEN
E Fragniere, et al.	University of Geneva	SETSTOCH
A King, et al.	IBM	OSL/SE
HI Gassmann, et al.	Dalhousie University	MSLiP
G Infanger et. al.	Stanford University	DECIS
P Kall, et al.	University of Zürich	SLP-IOR
G Mitra, et al.	Brunel University	SPInE

Table 1. Current development of software tools for SP.

The designers of AIMMS, GAMS and AMPL have put forward conference presentations and examples of use of their systems to develop SP models. No major implementation, however, have been reported. SP language extensions for AMPL have also been proposed in (Gassmann and Ireland 1996). The MPL modelling system is being extended to include SPInE's functionalities.
Other practitioners have focused on the automatic generation of Stochastic Programming model instances. For instance, Dempster' s group has developed and used the Stochastic Programming tool *STOCHGEN* in a number of industrial settings (Consigli and Dempster 1998). *SETSTOCH* (reported by Condevaux-Lanloy and Fragnière 2000) is another system which retrieves the dynamic structure of an SP model by reordering the generated matrix using external information on the temporal relations between the decision variables and the parameters.
Several specialised solvers for SP have also been produced. IBM' s *OSL Stochastic Extension* (King 1994) is an optimisation library which provides a set of routines for the manipulation and solution of multistage Stochastic

Programming problems presented in SMPS standard format (Birge et al. 1987). *MSLiP* (Gassmann 1990) is another established solver for multistage stochastic linear programs (SLP) based on the nested Benders decomposition method. *DECIS* (Infanger 1997) is a system for solving large-scale stochastic programs, which uses Benders decomposition and Monte Carlo sampling techniques. Finally, *SLP-IOR* (Kall and Mayer 1996) is an integrated model management system for stochastic linear programming problems and is designed to support the life cycle of an SP problem including model formulation, solution and analysis of the model instance and the results. In investigating SP models with the existing software tools, however, several difficulties are encountered:

(a) In the conceptualisation and subsequent formulation phase of a stochastic program, it is necessary to develop models of randomness and models of optimum resource allocation. Data modelling similarly splits into two main activities: (i) specification and collection of deterministic data; (ii) identification and categorisation of random parameters. The data definition facilities of the AMLs do not support this.

(b) Special purpose application programs are required to model the uncertainty. For instance, geometric Brownian motion is used to forecast asset prices in many financial applications, whereas sampling from a lognormal probability distribution may be used to create scenarios for products demand in a supply chain problem. These specialised applications (scenario generators) need to be integrated with the algebraic modelling systems.

(c) The algebraic modelling systems do not provide appropriate syntax for the implementation of SP models. Information related to the random parameters cannot be specified as such using the existing constructs and many specialists agree that these AMLs need to be extended. Some suggestions have been made (Gassmann and Ireland 1996, Fourer and Gay 1997, Entriken 1997, also see Table 1) but practitioners do not seem to have found a solution which fully addresses the requirements of SP modelling.

(d) The special structure of SP models is represented compactly and efficiently by following the SMPS standard (Birge et al. 1987). All the AML systems are capable of providing models in MPSX format (IBM 1976) or equivalent, but do not have facilities to produce SP model instances in the SMPS format.

(e) Solving SP models is a challenging task, as the sizes of these models are usually very large; thereby they are unmanageable by classical algorithms based on sparse simplex or interior point methods. Decomposition algorithms such as nested Benders decomposition (Benders 1962; Van Slyke and Wets 1969), stochastic decomposition methods (Higle and Sen 1996), Monte Carlo and Importance Sampling (Infanger 1994), Lagrangean decomposition (Mulvey and Ruszczyński 1995) require both the implementation of specialised solvers and a close coupling with the algebraic modelling system. The analysis of the results, together with the investigation of the stochastic measures associated with the SP models, are difficult to carry out without the use of database systems and customised viewers.

In this paper, we address some of these challenges and introduce SPInE (Stochastic Programming Integrated Environment). The original design of the software system (Messina and Mitra 1997) has been revised and updated. Our revisions take into account many of the issues discussed above and our design objective is to create a flexible software tool for Stochastic Programming practitioners. The rest of this paper is organised as follows: section 2 introduces the classes of SP models which are supported by our system. In this section we also introduce an example, which is used throughout the paper in order to illustrate the features of SPInE. Section 3 focuses on SAMPL and SMPL, which are extensions of the AMPL and MPL modelling languages respectively. In section 4 we discuss the rationale underlying the parameter passing interface which connects the special purpose scenario generators to the SPInE system. In section 5, we give an overview of the solution algorithms implemented in SPInE and consider the performance and scale up properties of these algorithms. In section 6 we describe the software architecture of SPInE: the illustrative example given in section 2 is used to explain the investigation of SP models within the SPInE system. Our aim is to make SPInE widely available to the industrial and academic research community. We have therefore prepared a library of SP models in SMPL and SAMPL and also collected test problems in SMPS: These two model libraries are summarised in Appendix A and Appendix B respectively.

2 Problem statement

In this section we consider the family of models which are now well established and come under the broad heading of *optimum decision making under uncertainty*. We first introduce these concepts and terminology which we subsequently use to illustrate the features and capabilities of the SPInE environment.

2.1 Classes of Stochastic Programming problems

We follow the classification of Stochastic Programming problems introduced by (Gassmann and Ireland 1996). We make a small extension of their categorisation by adding the Expected Value models as a subclass of the distribution problems leading to a working taxonomy shown in Figure 2.

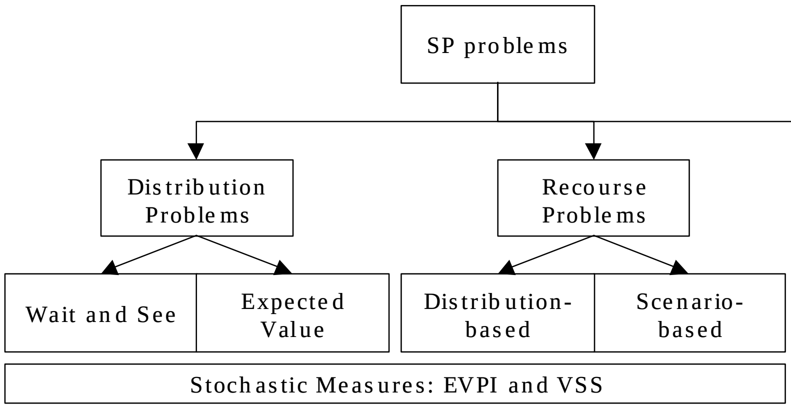


Figure 2 Taxonomy of SP problems.

We illustrate these classes by first considering the linear programming problem:

$$\begin{aligned} Z \quad &= \quad \min cx \\ \textit{subject to} \quad & Ax = b \\ & x \geq 0 \\ \textit{where} \quad & A \in R^{m \times n}; c, x \in R^n; b \in R^m \end{aligned} \tag{1}$$

Let *(Ω, ℑ, P)* denote a (discrete) probability space where $\xi(\omega), \omega \in \Omega$ denote the realizations of the uncertain parameters. Let us denote the realizations of *A, b, c* for a given event *ω* as:

$$\xi(\omega) \ or \ \xi_\omega = (A, b, c)_\omega \tag{2}$$

The associated probabilities of these realizations are often denoted as $p(\xi(\omega))$ *or* $p_{\xi(\omega)}$. For notational convenience we denote these probabilities as $p(\omega)$.

The classes of stochastic models illustrated in Figure 2 are defined below.

Distribution problems

The optimisation problems which provide the distribution of the objective function value for different realisations of the random parameters and also for the expected value of such parameters are broadly known as the distribution problems.

The Expected Value Problem

The Expected Value (EV) model is constructed by replacing the random parameters by their expected values. Such an EV model is thus a linear program, as the uncertainty is dealt with before it is introduced into the underlying linear optimisation model. It is common practice to formulate and solve the EV problem in order to gain some insight into the decision problem. Let the feasible regions corresponding to the problem stated in (1) and (2) be defined as:

$$F^{\omega} = \{x \mid Ax = b, x \geq 0\} \quad \textit{for} \quad (A,b,c)_{\omega} \quad \textit{or} \quad \xi(\omega) \tag{3}$$

We can reconsider (1) as an expected value or an average value problem where:

$$(\bar{A},\bar{b},\bar{c}) = \bar{\xi} = E[\xi(\omega)] = \sum_{\omega\in\Omega} p(\omega)\xi(\omega)$$

and the optimisation problem is defined as:

$$\begin{aligned} &Z_{EV} = \min \bar{c}x \\ &\textit{where} \\ &x \in \bar{F} \equiv \{x \mid \bar{A}x = \bar{b}\} \end{aligned} \tag{4}$$

Let x^*_{EV} denote an optimal solution to the above problem.
This solution can be evaluated for all possible realisations $\xi(\omega) \mid \omega \in \Omega$. We can thus determine the corresponding objective function values and compute what is called the expectation of the expected value solution:

$$Z_{EEV} = E\left[c(\omega)x^*_{EV}\right] \tag{5}$$

If for some $\omega \in \Omega$, $x^*_{EV} \notin F^{\omega}$, that is x^*_{EV} is not feasible for some realisation $\xi(\omega)$ of the random parameters, we set:

$$Z_{EEV} \to +\infty \tag{6}$$

Wait and See Problems

Wait and See (WS) problems assume that the decision-maker is somehow able to wait until the uncertainty is resolved before implementing the optimal decisions. This approach therefore relies upon perfect information about the future. Because of its very assumptions such a solution cannot be implemented and is known as the “passive approach”. Wait and see models are often used to analyse the probability distribution of the objective value, and consist of a family of LP models, each associated with an individual scenario. The corresponding problem is stated as:

$$\begin{aligned} &Z(\omega) = \min c(\omega)x \\ &\text{subject to } \; x \in F^{\omega} \end{aligned} \tag{7}$$

The expected value of the wait and see solutions is defined as:

$$Z_{WS} = E[Z(\omega)] = \sum_{\omega \in \Omega} Z(\omega)\, p(\omega) \tag{8}$$

Stochastic Programming Problems with Recourse

Stochastic Programming Problems with recourse are dynamic LP models characterised by uncertain future outcomes for some parameters. In general, Stochastic Programming problems can be formulated as follows:

$$\begin{aligned} &Z_{HN} = \min E[c(\omega)x] \\ &\text{where} \quad x \in F \end{aligned} \tag{9}$$

$$\text{and} \qquad F = \bigcap_{\omega \in \Omega} F^{\omega} \tag{10}$$

The optimal objective function value Z_{HN} denotes the minimum expected costs of the stochastic optimisation problem. The optimal solution $x^* \in F$ hedges against all possible events $\omega \in \Omega$ that may occur in the future.

The classical stochastic linear program with recourse makes the dynamic nature of SP explicit, by separating the model' s decision variables into first stage strategic decisions which are taken facing future uncertainties and second stage recourse (corrective) actions, taken once the uncertainty is revealed. The formulation of the classical two-stage SP model with recourse is as follows:

$$\begin{aligned} Z = \; &\min \qquad cx + E_{\omega} Q(x,\omega) \\ &\text{subject to} \quad Ax \qquad\qquad = b \\ &\qquad\qquad x \geq 0, \end{aligned} \tag{11}$$

where:

$$
\begin{aligned}
Q(x,\omega) = \quad & \min\ f(\omega)y(\omega) \\
\textit{subject to} \quad & D(\omega)y(\omega) = d(\omega) + B(\omega)x \\
& y(\omega) \geq 0. \\
& \omega \in \Omega
\end{aligned}
\tag{12}
$$

The matrix A and the vector b are known with certainty. The function *Q(x, ω)*, referred to as the recourse function, is in turn defined by the linear program (12). The technology matrix *D(ω)*, also known as the recourse matrix, the right-hand side *d(ω)*, the inter-stage linking matrix *B(ω)*, and the objective function coefficients *f(ω)* of this linear program are random. For a given realisation *ω*, the corresponding recourse action *y(ω)* is obtained by solving the problem set out in (12).
As the future unfolds in several sequential steps and subsequent recourse actions are taken, we deal with the generalisation of the two-stage recourse problem, known as multistage Stochastic Programming problem with recourse. A decision made in stage t should take into account all future realisations of the random parameters and such decisions only affect the remaining decisions in stages t+1 ... T. In Stochastic Programming this concept is known as non-anticipativity. The general formulation of a multistage recourse problem is set out in equations (13) - (16) below:

$$
Z_{HN} = \min_{x_1} \left\{ c_1x_1 + E_{\xi_2}\left[\min_{x_2} c_2x_2 + E_{\xi_3|\xi_2}\left[\min_{x_3} c_3x_3 + \ldots + E_{\xi_T|\xi_{T-1}|\ldots|\xi_2} \min_{x_T} c_Tx_T\right]\right]\right\} \tag{13}
$$

subject to:

$$
\begin{array}{lllllll}
A_{11}x_1 & & & & & = b_1 \\
A_{21}x_1 + & A_{22}x_2 & & & & = b_2 \\
A_{31}x_1 + & A_{32}x_2 + & A_{33}x_3 & & & = b_3 \\
\vdots & & & \ddots & & \vdots \\
A_{T1}x_1 + & A_{T2}x_2 + & A_{T3}x_3 + & \ldots & + A_{TT}x_T & = b_T
\end{array}
\tag{14}
$$

$$
\ell_t \leq x_t \leq u_t; \tag{15}
$$

where: $t = 1,\ldots,T$ represents the planning horizon and the vectors:

$$
\xi_t = (b_t, c_t, A_{t1}, \ldots A_{tT}) \quad \forall \mathrm{t} \in [2,\ldots,T] \tag{16}
$$

are random vectors on a probability space *(Ω, ℑ, P)*.
It is important to stress the difference between decision stages and model time periods. Although these coincide in many applications, a stage can be regarded in general as a time period where new information about the state of nature is provided, that is the realisation of the random vectors can be observed. The term *"Here and Now"* is often used to refer to recourse problem, reflecting the fact that decisions are taken before perfect information on the states of nature is revealed.

Scenario based recourse problems

Let us reconsider the random parameter vector $\xi(\omega)$ introduced in (2) and used in the definition of the given class of models. In the discrete statement of the problem the event parameter takes the range of values $\omega = 1,\ldots,|\Omega|$; there are associated random vector realisations $\xi(\omega)$ and probabilities $p(\omega)$ such that:

$$\sum_{\omega\in\Omega} p(\omega) = 1 \quad \text{and} \quad \Xi = \bigcup_{\omega\in\Omega} \xi(\omega) \tag{17}$$

In (17), Ξ is the set of all random vectors and is often called the *set of scenarios*.

For the multistage recourse problem (13) - (16), if the probability distribution of the random parameter vectors is discrete, the uncertainty defines a random structure in the form of an event tree, which represents the possible sequence of realisations (scenarios) over the time horizon (see Figure 3). When the event tree is explicitly given, we refer to the model as a *scenario based recourse problem*.

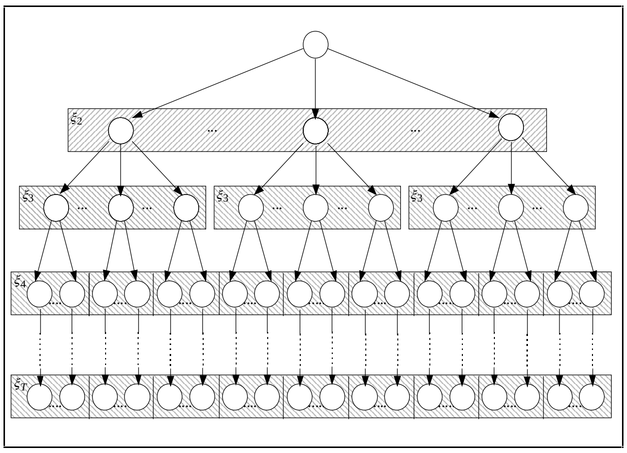


Figure 3. Event tree and scenarios

In the multistage problem (*scenario based*), the event tree serves two purposes:

(i) specify the model of randomness (the scenario generation) and

(ii) define the algebraic model structure including hierarchal (or recursive) non anticipativity restrictions.

In general, individual scenarios are interpreted as leaf enumeration of the event tree (Messina and Mitra 1997).

Distribution based recourse problems

An event tree can be also implied by defining the probability distributions of the random parameters, in which case the model is called *distribution based recourse problem*. Gassmann and Ireland (Gassmann and Ireland 1996) expand this concept in their work. This second class of problems, however, introduces various difficulties in the model specification using algebraic modelling languages and in terms of the solution process, in particular when some of the random parameters are continuously distributed. An approximation can be achieved by adopting appropriate sampling procedures, whereby the distributions may be replaced by an event tree.

Chance-Constrained Problems

Another important class of Stochastic Programming models are the chance-constrained problems (CCP) (Charnes and Cooper 1959). These can be dynamic or static models where one or more constraints are probabilistic. The general formulation of a chance-constrained problem is:

$$\begin{aligned} &Z_{CCP} = \min cx \\ &s.t. \\ &A_0 x = b_0 \\ &P\{A_i x \geq h_i\} \geq \beta_i \quad i = 1..I, \end{aligned} \tag{18}$$

where:

$$\beta_i \in [0,1]$$

is a reliability level and:

$$\xi_i = (A_i, h_i) \quad \forall i = 1..I$$

is a random vector on the probability space *(Ω, ℱ, P)*.
If the A_i is a row vector, the i-th constraint is called individual chance constraint. If A_i is a $r \times c$ matrix with $r > 1$, then the i-th constraint is referred to as joint chance constraint.

2.2 Stochastic measures in SP recourse problems :EVPI and VSS

It can be shown that the three objective function values Z_{EEV}, Z_{HN}, Z_{WS} are connected by the following ordered relationship:

$$Z_{WS} \leq Z_{HN} \leq Z_{EEV} \tag{19}$$

The inequality:

$$Z_{HN} \leq Z_{EEV} \tag{20}$$

can be argued in the following way: any feasible solution of the average value approximation is already considered in the Here and Now model, therefore the optimal Here and Now objective must be better. The difference between these two solutions defines the *Value of the Stochastic Solution* (VSS):

$$VSS = Z_{EEV} - Z_{HN} \tag{21}$$

This is a measure of how much can be saved by implementing the (computational expensive) Here and Now solution as opposed to the deterministic expected value solution.
Another important index is represented by the *Expected Value of Perfect Information* (EVPI):

$$EVPI = Z_{HN} - Z_{WS} \tag{22}$$

This property of the stochastic optimisation problems is interpreted as the expected value of the amount the decision maker is willing to pay to have perfect information (i.e. knowledge) about the future scenarios. A relatively small EVPI indicates that better forecasts will not lead to much improvement; a relatively large EVPI means that incomplete information about the future may prove costly. In (Birge and Louveaux 1997) some useful bounds on the EVPI and VSS are discussed:

$$0 \le EVPI \le Z_{HN} - Z_{EV} \le Z_{EEV} - Z_{EV} \tag{23}$$

$$0 \le VSS \le Z_{EEV} - Z_{EV} \tag{24}$$

These can help in estimating the relative benefit of implementing the costly Stochastic Programming solution, as opposed to approximate solutions obtained by processing the Expected Value LP problem.

2.3 Illustrative example: an asset/liability management model

The finance industry, rather belatedly, has embraced the Markowitz mean-variance model (Markowitz 1952; see also Jobst et al. 2001) for portfolio planning and asset/liability management applications. A major criticism of this approach (Luenberger 1997) is that it is a single period, static model. Carino and Turner (Carino and Turner 1997) illustrate the superiority of Stochastic Programming dynamic asset allocation models, over the mean-variance approach. In fact, active portfolio management breaks away from myopic static decisions and implies revisiting the strategy, and re-balancing of the portfolio positions as financial conditions change. We use an ALM multistage Stochastic Programming model with downside risk constraints (Kyriakis 2001) to illustrate the capabilities of the SPInE system.

Algebraic formulation

The ALM problem:

An investor faces the problem of creating a portfolio allocating assets out of a universe of I assets. Each asset is characterised by a price, which is (the only) random variable. The possible future prices are represented by an event tree. The goal of the investor is to maximise the portfolio wealth at the end of the time horizon T. He needs to take into account future obligations (liabilities). Asset buying and selling decisions are made, and each trade has an associated transaction cost. The deviation of the portfolio value from a predefined target is taken as measure of the risk. In each time stage the investor can decide the amount of assets to buy, sell and hold in the portfolio.

We implement this problem as a multistage stochastic program with recourse, using a split-variable deterministic equivalent representation (Dempster 1988; Messina and Mitra 1997):

Sets and indices:

T	denotes the number of time period in the time horizon
Assets	is the set of assets in our universe, where $\lvert Assets \rvert = I$
Scenarios	is the set of scenarios, where $\lvert Scenarios \rvert = Sc$
t = 1..T	denote time periods,
i = 1..I	denote an asset,
s = 1 ..Sc	indicates a scenario.

Parameters:

$price_{its}$	$i \in$ *Assets*, t=*1..T*, $s \in$ *Scenarios* is the price of asset i in period t, for scenario s
p_s	$s \in$ *Scenarios* is the weight (probability) associated to scenario s
$L_t \geq 0$	t=*1..T* is the expected liability at time period t
$F_t \geq 0$	t=*1..T* is the funding available in time period t
$A_t > 0$	t=*1..T* is the predefined target for time period t
$H0_i \geq 0$	$i \in$ *Assets* is the initial composition of the portfolio
$R \geq 0$	is the maximum deviation from the target accepted by the investor (in fraction)
$g \geq 0$	is the transaction cost rate

Decision variables:

$H_{its} \geq 0$	$i\in$ *Assets* t=*1..T*, $s\in$ *Scenarios* is the amount of assets of type i held in time period t under scenario s
$B_{its} \geq 0$	$i\in$ *Assets* t=*1..T*, $s\in$ *Scenarios* is the amount of assets of type i bought in time period t under scenario s
$S_{its} \geq 0$	$i\in$ *Assets*, t=*1..T*, $s\in$ *Scenarios* is the amount of assets of type i sold in time period t under scenario s

Objective function:

Maximise the expected value of the final portfolio wealth:

$$\max \sum_{s=1}^{Sc} p_s \sum_{\mathrm{i}=1}^{\mathrm{I}} price_{\mathrm{iT}} H_{iT}$$

Subject to:

Asset holding constraints:

$$H_{its} = HO_i + B_{its} - S_{its} \qquad t = 1, i = 1..I, s = 1..Sc$$

$$H_{its} = H_{it-1s} + B_{its} - S_{its} \qquad t = 2..T, i = 1..I, s = 1..Sc$$

Fund Balance constraints:

$$(1-g)\sum_{i=1}^{I} price_{its} S_{its} - L_t + F_t = (1+g)\sum_{i=1}^{I} price_{its} B_{its} \qquad t = 1..T, s = 1..Sc$$

Downside risk constraints:

$$A_{\mathrm{t}} - \sum_{\mathrm{i}=1}^{\mathrm{I}} price_{\mathrm{its}} H_{its} \leq A_t \mathrm{R} \qquad t = 2..T, s = 1..Sc$$

To complete the formulation we add to this a set of non-anticipativity constraints, which depend on the event tree structure.

Event tree representation

Let T represent a time horizon of one year, which is divided into four quarters. A decision is made at the beginning of each quarter, leading to a four-stage stochastic program. The event tree used in this example is shown in Figure 4. The leftmost node represents the strategic decision (to be made today) while the subsequent nodes represent conditional "recourse" decisions at later stages, when the portfolio is re-balanced.

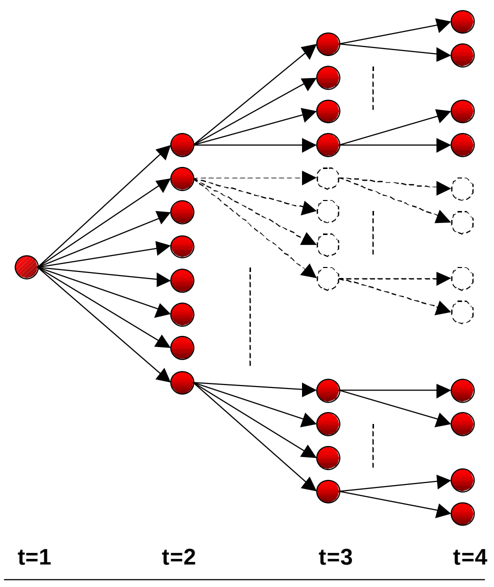


Figure 4: Event tree structure

The further we look into the future, the less accurate is our knowledge about the state of nature. This justifies our choice of a tree structure in which the number of branches (alternative asset prices) decreases in later stages. We consider 8 possible outcomes in the second stages, 4 conditional outcomes in the third stage and 2 conditional outcomes in the last stage leading to a total of 8 x 4 x 2 = 64 scenarios.
The asset prices are generated using a GARCH based model (Kyriakis 2001). The discussion of the scenario generation is postponed until section 4. The conditional decisions induced by the scenario tree are modelled via a set of non-anticipativity constraints. A scenario *s* is a data path from the root of the event tree to any of the leaves. In general, the decisions at each stage have to be the same for all scenarios which are indistinguishable up to that stage (they pass through the same node).
Formally, let:

$$K_t$$

denote the number of nodes in stage t, and let:

$$n_{tk} \qquad k \in [1,..,K_t], t \in [1,..,T]$$

denote the *k-th* node of the *t-th* stage of the event tree. We define:

$$bundle_{tk}$$

as the set of scenarios passing through node n_{tk}. In our example, we have:

$$K_1 = 1, K_2 = 8, K_3 = 32, K_4 = 64$$

The composition of the bundles is therefore:
bundle$_{11}$={1,2,...,64} (t=1)
bundle$_{21}$={1,2,..,8}, bundle$_{22}$={9,10,..,16}, ..., bundle$_{28}$={57,..,64} (t=2)
and so on. We can define the bundles of this problem in a compact form as:

$$bundle_{tk} = \left\{ 1 + \frac{Sc}{K_t}(k-1), ..., \frac{Sc}{K_t}k \right\} \qquad (25)$$

where *Sc* is the total number of scenarios. This relation is valid for any scenario tree with a constant number of branches at each node of a given stage.
The non-anticipativity can be thus expressed as:

$$(S,H,B)_{its_q} = (S,H,B)_{its_r}, \forall s_r, s_q \in Bundle_{tk}, r \neq q, \forall i,t \qquad (26)$$

2.4 Illustrative model formulated in AMPL and MPL

The implementations of this model in the established modelling languages AMPL (Fourer et al. 1993) and MPL (Maximal 2000) are set out below. The values of the deterministic parameters, as well as the scenario data, are read from a database. In AMPL, the directives for the database connections are separated from the model definition. The AML systems translate these declarative formulations into the matrix of the deterministic equivalent model.

Formulation of the model in AMPL

```
set I      := 1..23;  #asset type
set T      := 1..4;   #time stages
set Sc     := 1..64;  #scenarios

param g := 0.025;      # Transactions cost ratio
param R := 0.2;        # Risk level;
param L{T};            # Liabilities;
param H0{I};           # Initial portfolio;
param F{T};            # Funding
param A{T};            # Targets
param P{Sc};           # scenario probabilities
param price{T,I,Sc};   # asset prices

var S{T,I,Sc} >=0;
var H{T,I,Sc} >=0;
var B{T,I,Sc} >=0;

maximize wealth : sum{s in Sc, i in I} P[s]* H[4,i,s]* price[4,i,s];

######subject to

###### ASSET HOLDING CONSTRAINTS #######

assetholding1{i in I, s in Sc}:         H[1,i,s]=H0[i]+B[1,i,s]-S[1,i,s];
assetholding2{i in I, t in 2..4,s in Sc}: H[t,i,s]=H[t-1,i,s]+B[t,i,s]-
S[t,i,s];

###### FUND BALANCE CONSTRAINTS #######

fundbalance{t in T,s in Sc}:  sum {i in I} B[t,i,s]*price[t,i,s]*(1+g) -
                              sum {i in I} S[t,i,s]*price[t,i,s]*(1-g) =
                              F[t]-L[t];

###### DOWNSIDE RISK CONSTRAINT ######

zeta{ t in 2..4, s in Sc}:  A[t]- sum {i in I} H[t,i,s]*price[t,i,s]<=
                            R*A[t];

###### NON ANTICIPATIVITY CONSTRAINT ######

nah11{i in I,s in 2..64}  :   H[1,i,s]= H[1,i,s-1];

nah21{i in I,s in 2..8}   :   H[2,i,s]= H[2,i,s-1];
nah22{i in I,s in 10..16}:    H[2,i,s]= H[2,i,s-1];
…
nah28{i in I,s in 58..64}:    H[2,i,s]= H[2,i,s-1];

nah31{i in I,s in 2..2}   :   H[3,i,s]= H[3,i,s-1];
nah32{i in I,s in 4..4}   :   H[3,i,s]= H[3,i,s-1];
…
nah316{i in I,s in 64..64}:   H[3,i,s]= H[3,i,s-1];

#### same for variables S and B ####
```

Table 2. Formulation of the ALM model in AMPL.

Formulation of the model in MPL

```
TITLE   ALM_46_MS;

INDEX
i          = 1..23; ! I=23
t          = 1..4;  ! T=4
s          = 1..64; ! Sc=64

DATA
g              = 0.025;
R              = 0.2;
L[t]           = DATABASE(tbl_liabs,liability);
H0[i]          = DATABASE(tbl_portfolio,quantity);
F[t]           = DATABASE(tbl_incomes,Income);
A[t]           = DATABASE(tbl_targets,Target);
P[s]           = DATABASE(tbl_Probabilities,Prob);
price[t,i,s]   = DATABASE(tbl_PricesSP,return);

DECISION VARIABLES
S[t,i,s];
H[t,i,s];
B[t,i,s];

MODEL
MAX wealth = SUM(s,i,t=4: P[s]*H[t,i,s]*price[t,i,s]);

SUBJECT TO
!********* ASSET HOLDING CONSTRAINTS *******
assetholding[i,t=1,s]:         H[t,i,s]=H0[i]+B[t,i,s]-S[t,i,s];
assetholding[i,t>1,s]:         H[t,i,s]=H[t-1,i,s]+B[t,i,s]-S[t,I,s];

!********* FUND BALANCE CONSTRAINT *********
fundbalance[t,s]:               SUM(i:B[t,i,s]*price[t,i,s]*(1+g)) -
                                SUM(i:S[t,i,s]*price[t,i,s]*(1-g)) =
                                F[t]-L[t];

!********* DOWNSIDE RISK CONSTRAINT *******
zeta[t>1,s]:             A[t]-SUM(i:H[t,i,s]*price[t,i,s])<=R*A[t];

!********* NON ANTICIPATIVITY CONSTRAINT *********
nah11[t=1,i,s>1]: H[t,i,s]= H[t,i,s-1];

nah21[t=2,i,s=2..8]: H[t,i,s]= H[t,i,s-1];
nah22[t=2,i,s=10..16]: H[t,i,s]= H[t,i,s-1];
…
nah28[t=2,i,s=58..64]: H[t,i,s]= H[t,i,s-1];

nah31[t=3,i,s=2]: H[t,i,s]= H[t,i,s-1];
nah32[t=3,i,s=4]: H[t,i,s]= H[t,i,s-1];
…
nah316[t=3,i,s=64]: H[t,i,s]= H[t,i,s-1];

!*** same for variables S and B***********
…
END
```

Table 3. Formulation of the ALM model in MPL

An examination of the non-anticipativity constraints indicates that this explicit form of representation is not immediately natural: it can be laborious and error-prone. Also, the important requirement of separating the data from the model (Geoffrion 1992) is lost in the above representation. This can be avoided using alternative but equally laborious model formulations (see for instance Gassman and Ireland 1995). Finally, the matrices generated for these deterministic equivalent models contain redundancies and are often too big to be handled by both solvers and modelling systems. This strongly makes the case for extending the algebraic modelling languages in terms of syntax and in generation capabilities in order to support the SP modelling steps.

3 Extending Algebraic Modelling Languages for SP

The use of the MPS input format as a representation standard and the extensive use of algebraic modelling languages have become well established and have substantially facilitated the investigation of deterministic optimisation problems. SMPS was introduced as a standard for representing stochastic programs in a more compact and efficient way, exploiting the inherent SP model structure. However, SMPS is a machine-readable format and has no high level representational power. Our illustrative example introduced in section 2.3 and our discussions in section 2.4 highlight the difficulties of using existing algebraic modelling languages to formulate SP. This is mainly due to the lack of constructs for the definition of the randomness of the model coefficients and for the declaration of the scenario tree structure. An examination of other investigators' works (Table 1) reveals that some extensions have been proposed to overcome these limitations, but have not yet been deployed.
We have designed and adopted a direct approach, whereby we provide extensions to AMLs to formulate SP recourse problems and Chance Constrained Problems with natural and concise constructs (Valente et al. 2001). This approach allows us to extend the syntax of AMPL and MPL into what we call SAMPL and SMPL respectively.

3.1 SMPL and SAMPL: an introduction

A Stochastic Programming model can be considered as a linear programming model extended and refined by the introduction of uncertainty (see Figure 5). More precisely, the underlying LP optimisation model is extended by taking into account the probability distribution of the LP coefficients which are random variables. Such distributions are provided by models of randomness (implemented in *scenario generators*), which are specific to the particular optimisation problems under investigation.

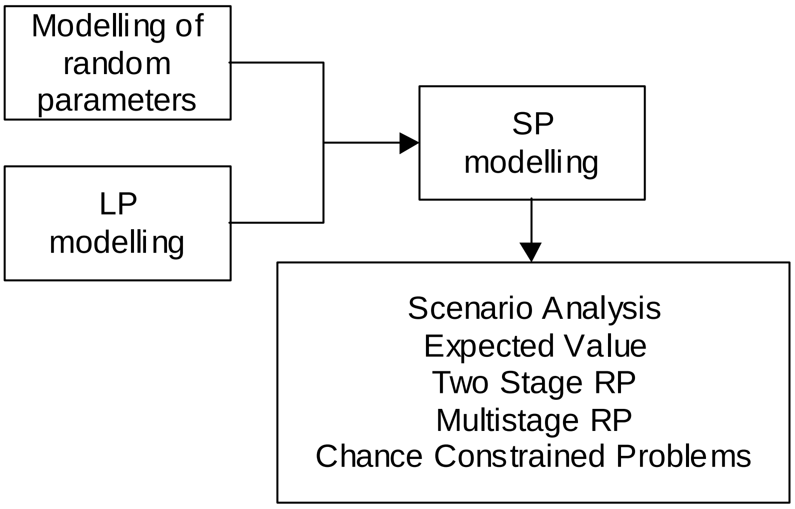


Figure 5. The combined paradigm

As we consider the taxonomy and the classes of stochastic models introduced in Figure 2, it becomes immediately obvious that AMLs are neither specifically designed nor well suited to construct these classes of models. In fact, the strong coupling between the model structure and the data structure which arises in the models of randomness makes it very difficult separate model definition from data definition (see section 4.1).

If the probability distributions of the random parameters are discrete, it is always possible to define a deterministic equivalent model for Stochastic Programming problems with recourse. As we discussed in section 2.4, this approach suffers from a number of drawbacks, and often leads to unmanageable models when the size of the problems increases. The difficulties of working with the deterministic equivalent model are summarised below:

a) The unnatural *non-anticipativity* constraints have to be declared to reflect the model structure induced by the scenario tree.

b) The replication of decision variables leads to a high level of redundancy.

c) The size of the deterministic equivalent models grows exponentially with the number of decision stages considered.

d) The staircase structure of the matrix is lost due to the internal processing of the modelling systems, precluding the possibility of exploiting structure information by the solution algorithm.

Ideally, a modelling language for Stochastic Programming problems should include a set of constructs which allow the modeller to capture the effects induced by the uncertainty on the underlying model structure, as well as provide a compact representation of the model instance. We present a generic approach of modifying AMLs, which is based on the concepts of underlying deterministic model and stochastic information in respect of the random parameters.

The underlying deterministic model

In a Stochastic Programming problem, it is always possible to identify an underlying deterministic model (also called the *core* model). This model captures the logical structure of the problem as well as the dynamical relations within decision variables, their bounds and the objective function. In a scenario-based recourse problem, for instance, the core represents the model associated with a particular sequence of realisations of the random parameters (scenario). The definition of the underlying deterministic model makes use of the standard constructs provided by the existing modelling languages (AMPL or MPL). The core model could be linked to the model of randomness in two ways:

a) Making variables, parameters and constraints explicitly parametric in the scenario index
b) Marking the appropriate coefficients as random parameters in such a way that they can be treated implicitly.

The first approach requires that a scenario dimension must be introduced a priori and precludes the possibility of describing models with continuous distributions; it also implies the replications of variables and constraints. We adopt the second approach, whereby we write a pure deterministic model and we use new language constructs to identify the random parameters of the problem. Such constructs also define the effects of the uncertainty on the underlying model structure.

Declaration of the random structure

Once the underlying deterministic problem has been implemented, it is necessary to merge it with the information related to the model of randomness which characterises the problem. We expand the language syntax in order to capture such *stochastic information*. The items of information can be summarised as follows:

Stochastic Class: identifies the category to which the SP problem belong (distribution-based recourse problem, scenario-based recourse problem, chance-constrained Problem)

Time dimension: the index used to describe the temporal horizon in the underlying model needs to be uniquely identified.

Stages: Decision stages are defined in terms of a partition of the time horizon

Scenario dimension: the index used to identify the scenarios needs to be uniquely identified, because the realisations of the random parameters in scenario-based problems are defined using this index.

Scenario Tree: for scenario-based problems, it represents the structure of the event tree.

Scenarios probability distribution: the (discrete) probabilities associated with the scenarios.

Random parameters probability distributions: only required for distribution-based problems

Random data: defines and marks the random parameters of the problem in scenario-based problems.

Chance constraints: Probabilistic constraints in chance-constrained problems need to be explicitly declared.

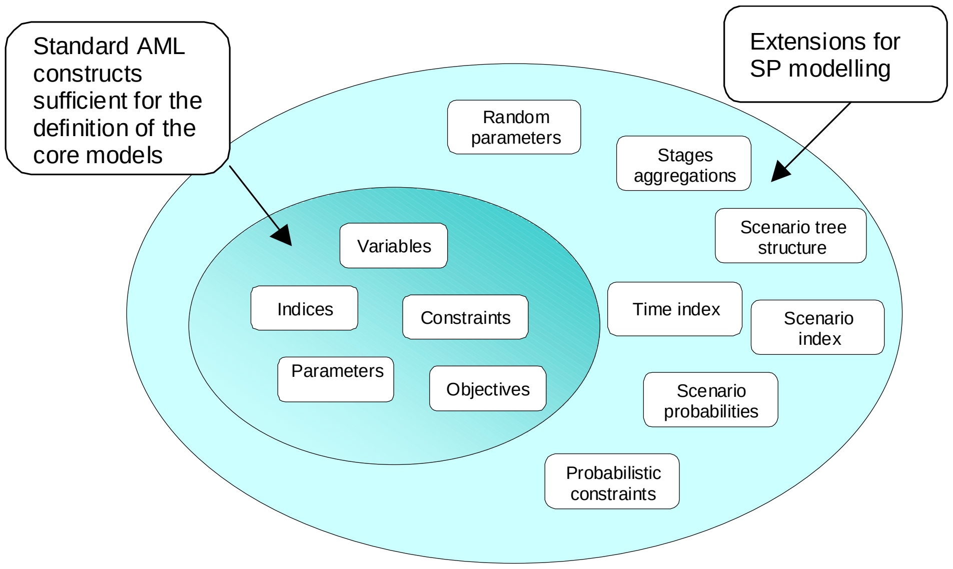


Figure 6. Extended language constructs

Figure 6 shows how the basic constructs of a modelling language for linear programming are extended to capture the stochastic information. The design of the new constructs is adapted to be consistent with the grammar of the underlying modelling language. We have successfully applied this approach to the MPL and AMPL languages. We refer to (Valente et al. 2001) for a detailed description of the syntax. The combined SMPL and SAMPL parser is the core of the modelling system embedded into SPInE. This system also generates data model instances in SMPS format and in a Stochastic Intermediate Representation (SIR).

3.2 Illustrative model formulated in SAMPL and SMPL

Following the approach outlined in section 3.1, we need to specify the stochastic information relating to the ALM problem. Table 4 contains the declarations in SMPL and SAMPL of the components which together comprise the required stochastic information.

Description	SMPL definition	SAMPL definition
Stochastic class The model is a scenario-based recourse problem (SBRP)	**STOCHASTIC CLASS** `SBRP;`	**class SBRP;**
Time The time index is of this problem is represented by *t*.	**TIME** `t;`	**timeset** `T;`
Scenarios The asset prices outcomes are given in the form of an event tree. We use the index *s* to identify the scenarios.	**SCENARIO** `s;`	**scenarioset** `Sc;`
Probabilities We consider scenarios with equal probability.	**PROBABILITIES** **ALL_EQUAL**;	**probability** **all_equal**;
Stages The time horizon is divided into 4 quarters, which also identify the decisional stages. There is therefore a 1 to 1 relation between time periods and stages, which we indicate as *ONE_TO_ONE*.	**STAGES PARTITION** `ONE_TO_ONE;`	**stages partition:** **one_to_one;**
Tree The event tree used in this model is a symmetric tree. The number of branches at each node varies in different time stages (8,4,2), but is constant for nodes within a given stage.	**TREE** **MULTIBRANCH(8,4,2)**;	**tree** multibranch**{8,4,2}**;
Random data The only random parameter is the price of the assets over time. This parameter is scenario-dependent, and is therefore indexed over the scenario index.	**RANDOM DATA** `price = DATABASE(tbl_PricesSP,return);`	**random param** `price;`

Table 4. Definition of the stochastic information of the ALM model

The complete model formulations in SAMPL and SMPL are set out in Table 5 and Table 6 respectively.

An SP presented in SAMPL or SMPL can be separated into two parts:

(a) Part 1 which contains the declaration of the underlying core LP using "standard" AML statements.

(b) Part 2 which which contains some structural details covering the stochastic aspects of the model. This includes the definition of the scenario tree structure, the partitioning of variables and constraints into stages and an implicit reference to a scenario generator which provides random data parameter values to instantiate the SP model.

```
# asset/liability management model ALM48ms.mod

set T   := 1..4;    #time horizon
set I   := 1..23;   #asset type

param g := 0.025;  # Transactions cost rate
param R := 0.2;    # Risk level;
param L{T};        # Liabilities;
param H0{I};       # Initial portfolio;
param F{T};        # Funding
param A{T};        # Targets

var S{T,I} >=0;
var H{T,I} >=0;
var B{T,I} >=0;

maximize wealth : sum{i in I} H[4,i]*price[4,i];

subject to

###### ASSET HOLDING CONSTRAINTS #######

assetholding1{i in I}:               H[1,i]=H0[i]+B[1,i]-S[1,i];
assetholding2{i in I, t in 2..4]:    H[t,i]=H[t-1,i]+B[t,i]-S[t,i];

###### FUND BALANCE CONSTRAINTS #######

fundbalance{t in T}:          sum {i in I} B[t,i]*price[t,i]*(1+g) -
                              sum {i in I} S[t,i]*price[t,i]*(1-g) =
                              F[t]-L[t];

###### DOWNSIDE RISK CONSTRAINT ######

zeta{ t in 2..4}:             A[t]-sum {i in I} H[t,i]*price[t,i] <=R[t]*A[t];

# stochastic framework

class SBRP;

timeset  T;

scenarioset  Sc;

probability all_equal;

random param price;

stages partition one_to_one;

tree multibranch{8,4,2};

# end
```

Table 5. Formulation of the ALM model in SAMPL

```
TITLE   ALM_64_MS;

INDEX
i              = 1..23;
t              = 1..4;
DATA
g              = 0.025;
R              = 0.2;
L[t]           = DATABASE(tbl_liabs,liability);
H0[i]          = DATABASE(tbl_portfolio,quantity);
F[t]           = DATABASE(tbl_incomes,Income);
A[t]           = DATABASE(tbl_targets,Target);
price[t,i];

DECISION VARIABLES
S[t,i];
H[t,i];
B[t,i];


MODEL
MAX wealth = SUM(t=4,i:H[t,i]*price[t,i]);

SUBJECT TO
!********* ASSET HOLDING CONSTRAINTS *******

assetholding[i,t=1]:                  H[t,i]=H0[i]+B[t,i]-S[t,i];
assetholding[i,t>1]:                  H[t,i]=H[t-1,i]+B[t,i]-S[t,i];

!********* FUND BALANCE CONSTRAINT *********

fundbalance[t]:                       SUM(i:B[t,i]*price[t,i]*(1+g))-
                                      SUM(i:S[t,i]*price[t,i]*(1-g))=
                                      F[t]-L[t];

!********* DOWNSIDE RISK CONSTRAINT *******
zeta[t>1]  :                       A[t]-SUM(i:H[t,i]*price[t,i])<=R*A[t];


STOCHASTIC CLASS
SBRP;

TIME
t;

SCENARIO
s;

PROBABILITIES
ALL_EQUAL;

STAGES
PARTITION: ONE_TO_ONE;

TREE
MULTIBRANCH(8,4,2);

RANDOM DATA
price  = DATABASE(tbl_PricesSP,return);
```

Table 6. Formulation of the ALM model using the SMPL extended language.

We observe that:

(i) the explicit definition of the non-anticipativity constraints has been eliminated,

(ii) the separation of data definition from model definition, which is one of the main advantages of the use of AMLs, is preserved.

4 Scenario generation

If we revisit section 2.1 where we define the scenario based recourse problem and the Figure 5 in section 3.1 where we describe the combined paradigm of models of randomness and the optimum allocation of LP models, we gain some insight into the nature and structure of the SP models.

The algebraic (LP) model captures the logic of the application's domain. Similarly, the scenario generators are also special purpose applications developed to capture the randomness properties of a particular application's domain. Typically, a consumer product supply chain model and an energy distribution model both require scenarios of forecast demand, but the factors which influence the demands and the forecasting models may be very different. Again in finance applications the asset prices under consideration may be generated using different models of credit risk, interest rate risk or other considerations. In designing an SP modelling support system our goal is therefore to develop an appropriate parameter passing interface which enables us to connect diverse special purpose scenario generators (which capture very valuable domain knowledge) to our modelling system.

In section 4.1 we outline the design principles, in section 4.2 we consider some well established approaches to scenario generation and in section 4.3 we provide an example of connecting a GARCH based asset price generator for our illustrative ALM model.

4.1 Integrating scenario generators in SPInE

In section 2.1 we introduced the concept of scenarios and noted that the event tree within SP serves two different purposes:

(i) define the model of randomness (scenario generation) and

(ii) specify the algebraic structure of the decision variables and constraints.

A scenario generator φ captures in a procedural form a domain-specific model of randomness. In particular it uses historical information, an event tree structure and some other specification parameters. We can thus separate the main groups of parameters as

$\boldsymbol{H}$	: History,
$\boldsymbol{\tau}$:	: Event Tree,
$\boldsymbol{\theta}$	: Remaining Parameters.

The set of scenarios Ξ is then seen as the collection of scenarios which are output by the generation procedure:

$$\varphi(H,\tau,\theta) \Rightarrow \Xi \qquad (27)$$

In the algebraic form of the SP model we also need to specify the 'variable and constraint' tree structure, which we label as $\boldsymbol{\tau'}$. Thus using the extended AML we provide a specification of $\boldsymbol{\tau'}$ in the SP model through the *tree* declaration.
For consistency, of course, we need the two trees to be congruent. In other words we need to ensure that the event tree structure τ used by the special purpose scenario generator is 'compatible' with the τ' specified in the SP model (see Valente et al. 2001). The requirement for scenario generator parameter passing and tree consistency conditions are illustrated in Figure 9. When a special purpose scenario generator is connected to SPInE, the two trees τ and τ' are compared for

consistency. The scenario generator then creates the set of scenarios and the associated probabilities $p(\omega)$.

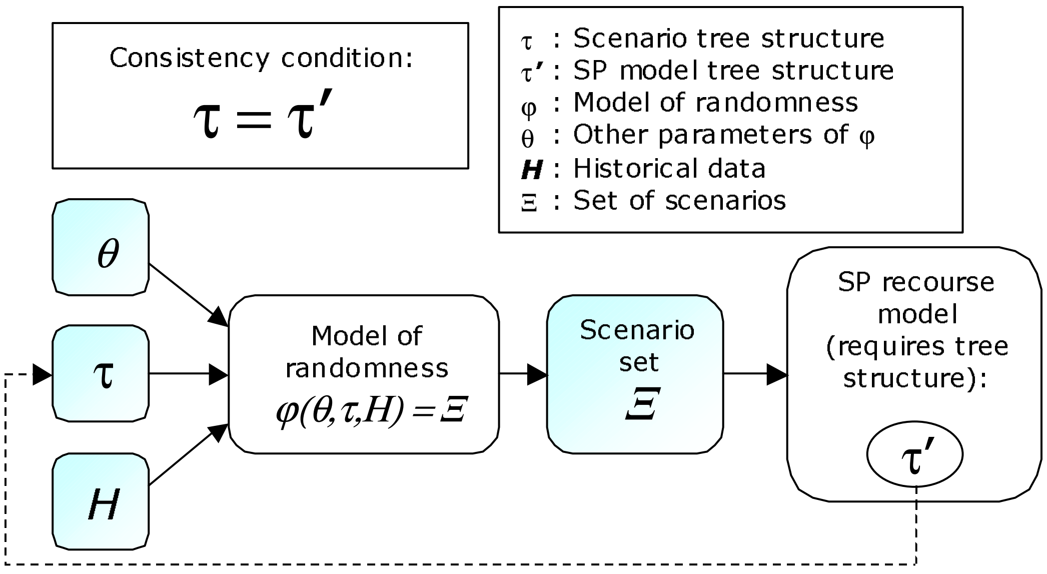

Consistency condition:
$\tau = \tau'$
τ : Scenario tree structure
τ' : SP model tree structure
φ : Model of randomness
θ : Other parameters of φ
H : Historical data
Ξ : Set of scenarios
θ
τ
H
Model of randomness
$\varphi(\theta,\tau,H) = \Xi$
Scenario set
Ξ
SP recourse model (requires tree structure):
τ'

Figure 7. Scenario Generation Parameter Paring and Tree Consistency

The data interface for the presentation of the scenarios to the modelling system is based on ODBC connections. This allows the scenario generator to store the output in virtually any type of database (including text files). The flexible interface with scenario generators and the ability to create in-sample scenarios for SP model optimisation and out-of-sample scenarios for simulation (see section 6.3) make the connection to external generators a valuable feature of SPInE.

4.2 Models of randomness

There are some well-established approaches to scenario generation; having examined a number of reported applications we have identified three modelling methodologies which are likely to encompass a wide range of SP models. These are moment-based methods, diffusion processes and time series based methods. Our design goal is to provide templates for connecting these types of generators.

Moment based scenario generation

One of the major problems in the scenario generation is to make sure that the outcomes sampled from the random model are consistent with the underlying probability distribution, or with the decision-maker beliefs about the future. Høyland and Wallace (Høyland and Wallace 2001), present an algorithm whereby the decision-maker specifies a distribution in terms of marginal moments, correlation matrix and higher co-moments and the generated scenarios are consistent with the specification.

Diffusion processes

Diffusion processes are widely used in finance to model the paths of the future prices of stocks, interest rate term structure and the value derivatives. These models of randomness are continuous time models.

Time series

Time series models are commonly used to estimate parameters which explain the behaviour of a random variable based on past observations.

4.3 An example: Connecting SPInE to an asset prices generator

In this section, we give a broad overview of the scenario generator for stock prices and liabilities developed in (Kyriakis 2001). We have integrated this generator in SPInE and used it for our asset/liability management example. This generator uses some ideas put forward by Mulvey (Mulvey 1996) who developed the CAP:Link model for Tower Perrins. Also see (Zenios 1993) for a description of optimisation as used in the finance industry. Figure 8 provides a compact outline description of the relationship diagram of the scenario generator.

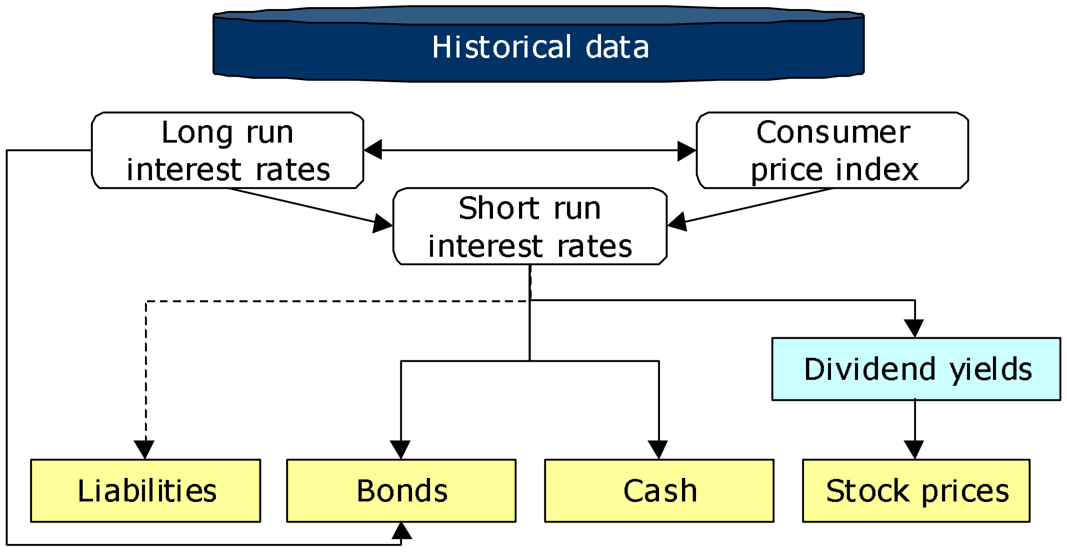


Figure 8. Asset prices generator: architecture

The macroeconomic environment is modelled using a Generalised Autoregressive model with Conditional Heteroscedasticity, GARCH(1,1).
GARCH processes are used to model random variables characterised by non-constant variance over time (heteroscedasticity), including a disturbance term which is function of past realisations and other exogenous factors. GARCH(1,1) is hence used to model the short run interest rates, long run interest rates and the consumer price index, and to capture the interrelationships between these variables through time. The coefficients for the model are estimated using a 19 years period from March 1978 to March 1996. These represent the *H* input parameter of the scenario generator.
The scenarios produced by these processes are next used to simulate paths for the dividend yield of either an index or individual assets. A combination of a stochastic process and an autoregressive model simulate the behaviour of the dividend yields. Brownian motion is used to capture the dependence of the dividends from their past values.
The scenario generator hereby illustrated uses autoregressive models AR(1) to relate dividend yields to the short run interest rate. Note that different models are estimated for each asset and financial index.
Finally, a multivariate autoregressive model, specific to each asset or index, generates future projections for the returns of the index or the assets. In this case, past returns and the current dividend yield explain the movements of the expected returns.

5 Stochastic Solver in SPInE

The proposal and adoption of the stochastic mathematical programming input data standard, SMPS (Birge et. al. 1987), as well as the library of models maintained by researchers (Ariyawansa and Felt 2001, Birge 2001) prepared to this specification, has made it easier to develop solvers and evaluate their performance. The SMPS input format extends the MPSX standard and is designed to achieve efficient conversion to SP models by representing the deterministic and the stochastic information in separate information streams presented as data files. The solver developed and integrated in SPInE has a clearly set out coupling with the modelling system and can accept model data instances either in SIR or in the SMPS format. As a result the interface is extremely flexible and the solver, which implements a number of established and a few innovative algorithms may be used as a stand-alone tool.

5.1 Solution techniques

Solution methods for SP problems have been widely studied and these can be broadly classified into four categories:

Universe problem

In this approach, which involves solving the deterministic equivalent, we consider all possible outcomes $\omega \in \Omega$ and solve the corresponding problem exactly (Birge and Louveaux 1997). This is not always possible, because there may be too many possible realisations $\xi(\omega) \in \Xi$. The universe problem can be solved in two ways depending on how the non-anticipativity constraints are considered; these non-anticipativity constraints can be considered either implicitly or explicitly.

Decomposition

Mathematical decomposition of the deterministic equivalent program is primarily based on using the L-shaped decomposition (see Van Slyke and Wets 1969), in multistage setting, for example, (Ho and Loute 1974; Glassey 1973), and Ruszczyński (Ruszczyński 1993) have implemented parallel versions. A variant of the nested decomposition algorithm, called the regularized decomposition method (Ruszczyński 1986), has also been proposed to solve large problems. The progressive hedging (or scenario aggregation) method (Mulvey and Ruszczyński 1995), is an augmented Lagrangean based technique with mathematical decomposition. For its implementation for problems with network structure see (Mulvey and Vladimirou 1991).

Sampling-Based

In sampling-based methods, one iteratively draws random samples from the underlying probability distribution for computing stochastic quasi-gradient (Dupac 1965), (Kushner 1971) or for stochastic L-shaped decomposition (Higle and Sen 1996) procedures that enjoy asymptotic convergence properties. Noting that the variance of the samples is the key to the convergence process, various variance reduction schemes, such as importance sampling within the L-shaped method (Bowermann and Koehler 1978), (Infanger 1994) have been employed for better algorithm performance.

Successive approximation

This approach is based on computation of bounds on the optimal objective value. See (Andradottoir 1995 and 1996), (Birge and Qui 1995), (Birge and Wets 1986) and (Frantzeskakis and Powell 1990), for general details on such schemes. These methods employ lower and upper bounding functions on the expected value function at each decision stage; these bounds are then used in the outer approximation.

5.2 The solver system in SPInE

The solver system which we have embedded into SPInE is designed to variously process the family of SP models which include:

(i) Expected Value Problem

(ii) Wait and See Problems

(iii) Here and Now Problem (Recourse Models).

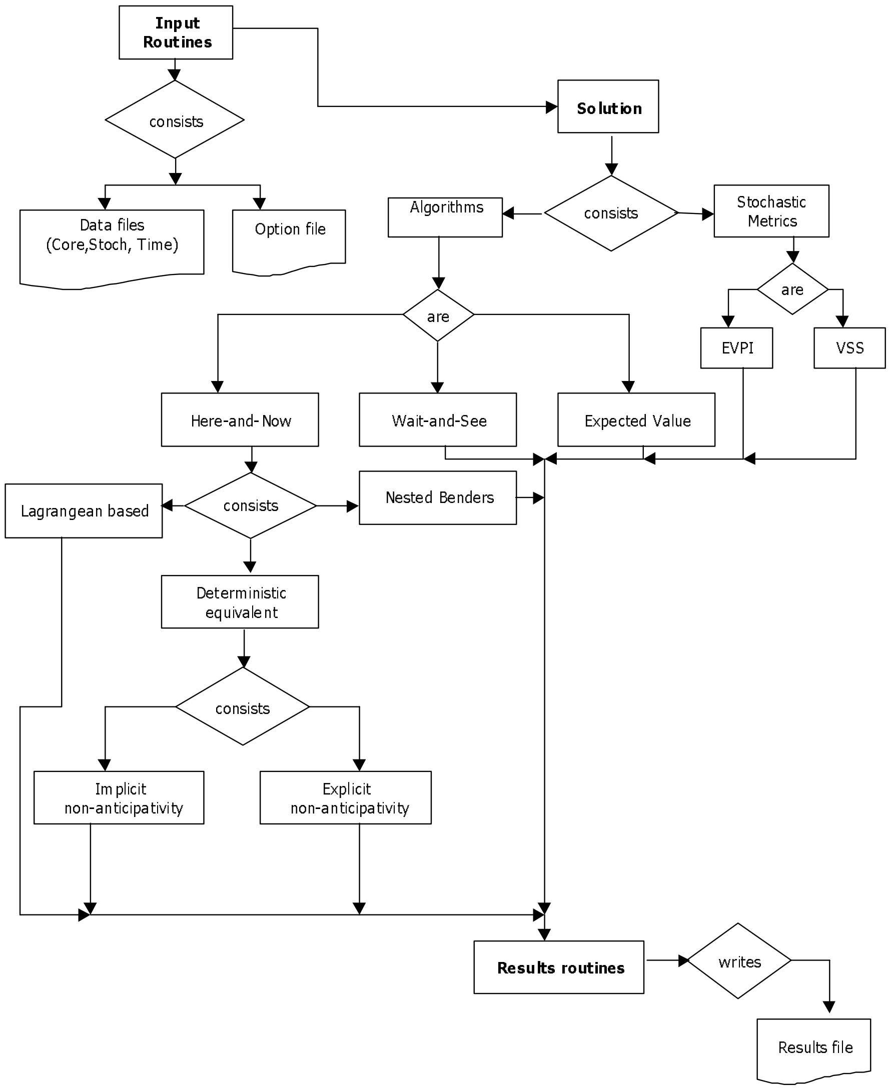


Figure 9. Entity relationship diagram of SPInE's solver.

Figure 9 shows the entity relationship diagram of the solver system. SP models capture the two important aspects of decision-making, namely time and uncertainty, but their computational realizations suffer from the curse of dimensionality. This in turn requires that the data structure of the solver must be efficient to (i) capture the dynamic evolution of uncertain parameters, and (ii) scale up (of the model size through scenarios and stages) to process real world models.

Algorithms

A number of solution algorithms have been implemented and tested within SPInE. Currently, we use FortMP (Ellison et. al 1999) as the main LP/MIP solver engine but the design allows us to replace it by any other powerful solver engine such as CPLEX. The Table 7 sets out the SP model and solver algorithm combinations which have been tested or planned for inclusion within SPInE.

SP Model	**Algorithm**	**Comments**
Two-stage linear SP	Universe (Deterministic equivalent). DEQ implicit, DEQ explicit	The user controls the specification of the non-anticipativity constraints, thus resulting in two possible representations, namely *implicit* and *explicit*. We have implemented both approaches in the solver system.
	Benders Decomposition	Implemented and tested.
	Stochastic Decomposition	Currently being implemented.
	Benders and Importance sampling.	Currently being implemented.
Integer two-stage SP	Lagrangean relaxation	This algorithm is described in our working paper (Poojari and Mitra 2001) and also in our Supply chain paper (Poojari et. al. 2000)
	Lagrangean relaxation and importance sampling.	To be implemented.
Multi-stage SP	Deterministic equivalent	Implemented and tested.
	Nested Benders	Implemented and tested.
	Nested Benders and Importance sampling	To be implemented.
	EVPI based importance sampling.	To be implemented. This is based in the work done by (Dempster and Thompson 1999).

Table 7. Solution algorithms of the solver system

Control

The solution algorithms to be deployed are chosen by control parameters specified via a parameter file. The full details of these parameters and the parameter file can be found in (Poojari and Mitra 2001); examples of setting these controls are shown in section 6.2. The solver can be also deployed to process the Expected Value problem, and the Wait and See problems for all scenario instances. Furthermore there exist control switches to calculate the stochastic measures EVPI and VSS.

5.3 Quality Assurance/Benchmark and Scale Up Properties

The embedded SP solver module has been extensively tested for a wide range of quality assurance (QA) test problems which have been collected from a number of sources. A summary of these QA models is given in Appendix B; a paper describing the algorithm and computational performance of the SP solver engine is under preparation (Poojari and Mitra 2001). The following five algorithms make up the most important part of the SP solver engine.

(i) Deterministic equivalent split-variable (DEQ explicit): model processed by IPM,
(ii) Deterministic equivalent compact variable (DEQ implicit): model processed by SSX,
(iii) (Nested) Benders decomposition (NBD): the master and subproblems processed by SSX,
(iv) Wait and See (WS): The individual problems processed by SSX.
(v) Expected Value (EV): The expected value LP problem processed by SSX.

The computational platform is set out in Table 8 in a summary form:

Processor	**Memory**	**Software**
Pentium III 500 MHz	128 Mbyte RAM	FortMP solver compiled using Digital Fortran and running under Win NT

Table 8. Computational Platform

Table 9 displays the summary of a subset of test models taken from this QA set.

Model	**Rows**	**Columns**	**Non-Zeroes**
Pgp2	9	20	40
FXM	330	457	2589
Pltexp	270	732	1491
Storm	713	1380	4037
Phone	24	93	207

Table 9. Model Summary.

The relative performances of the three Here-and-Now algorithms (Benders decomposition, DEQ implicit, and DEQ explicit), the Wait-and-See algorithm (WS), and the Expected Value (EV) are shown in Table 10.

Description		**SP models**				**Stochastic Metric**	
Name	**Stage/ Scen.**	**HN (time)**		**WS (time)**	**EV (time)**	**EVPI**	**VSS**
Pgp2	2/8	DEQ Implicit	508.975 (1s)	449.844 (1s)	431.406 (1s)	59.130	∞
		DEQ Explicit	508.975 (1s)				
		Benders Decomposition	508.975 (8s)				
Fxm	2/16	DEQ Implicit	18417.065 (24s)	18416.75 (4s)	18416.75 (1s)	0.38	0.38
		DEQ Explicit	18417.127 (47s)				
		Benders Decomposition	18417.13 (46s)				
Pltexp	3/36	DEQ Implicit	-13.965 (32s)	-13.968 (5s)	-14.304 (1s)	0	∞
		DEQ Explicit	-13.968 (399s)				
		Benders Decomposition	-13.968 (19s)				
Storm	2/8	DEQ Implicit	15535210 (26s)	15488580 (5s)	15459240 (1s)	46637.65	∞
		DEQ Explicit	15535210 (23s)				
		Benders Decomposition	15535217 (94s)				
	2/27	DEQ Implicit	15508969 (323s)	15459253 (17s)	15476546 (1s)	20389	∞
		DEQ Explicit	15508969 (890s)				
		Benders Decomposition	15508969 (381s)				
	2/125	DEQ Implicit	15512047 (8138s)	15476584 (80s)	15459219 (1s)	52829	∞
		DEQ Explicit	Not enough memory				
		Benders Decomposition	15512048 (2521s)				
	2/1000	DEQ Implicit	Not enough memory	15766791 (649s)	15750516 (1s)	35714	∞
		DEQ Explicit	Not enough memory				
		Benders Decomposition	15802505 (18882s)				
Phone	2/32768	DEQ Implicit	Not enough Memory	36.9 (935s)	36.9 (1s)	0	∞
		DEQ Explicit	Not enough Memory				
		Benders Decomposition	36.9 (15943s)				

Table 10. Computational Results

In order to study the scale up properties of these algorithms we investigated the two models STORM and PHONE.
STORM is a two-period freight-scheduling problem described in (Mulvey and Ruszczynski 1995), the problem is held at the University of Michigan and provided by Adam Berger. PHONE is a two-period telecommunication network problem described in (Sen, Doverspike and Cosares 1994).

These models were investigated for progressively larger sizes: for STORM 8 to 1000 scenarios, and for PHONE 100 to 32000 scenarios. The corresponding processing times are set out in Table 11 and Table 12, and are also plotted separately in Figure 10 and Figure 12 to a log (time) vs. linear (number of scenarios) scale. We have experimented the scale-up property of the nested Benders algorithm for different number of scenarios. In Figure 11 and Figure 13 we plot the performance of Benders algorithm for different instances (number of scenarios) of the Storm and Phone models.

Benders decomposition also extends well for parallel implementation (see Ariyawansa and Hudson 1991, and MirHassani et. al 2000, Zenios and Censor 1997, Wright 2001). Our solver is available on NEOS over a client server platform. Our earlier experience of solving large SP models on a parallel platform using client-server architecture is described in (Poojari et al 2001). In the following tables and figures, we use the following legend:

NB ≡ Nested Benders decomposition,
DI ≡ DEQ Implicit,
DE ≡ DEQ Explicit,
WS ≡ Wait-and-See.

Scenarios	NB	DE	DI	WS
1	1s	1s	1s	1s
8	198s	24s	18s	5s
10	310s	74s	30s	7s
25	781s	1293s	200s	17s
27	839s	913s	255s	18s
50	1342s	5533s	930s	32s
70	1601s	29719s	1893s	51s
100	2912s	∞	3375s	65s
200	5559s		15742s	145s
550	14410s		159446s	345s
700	15202s		∞	441s
1000	18222s			628s

Table 11. Performance time (in seconds) for algorithms on the STORM model.

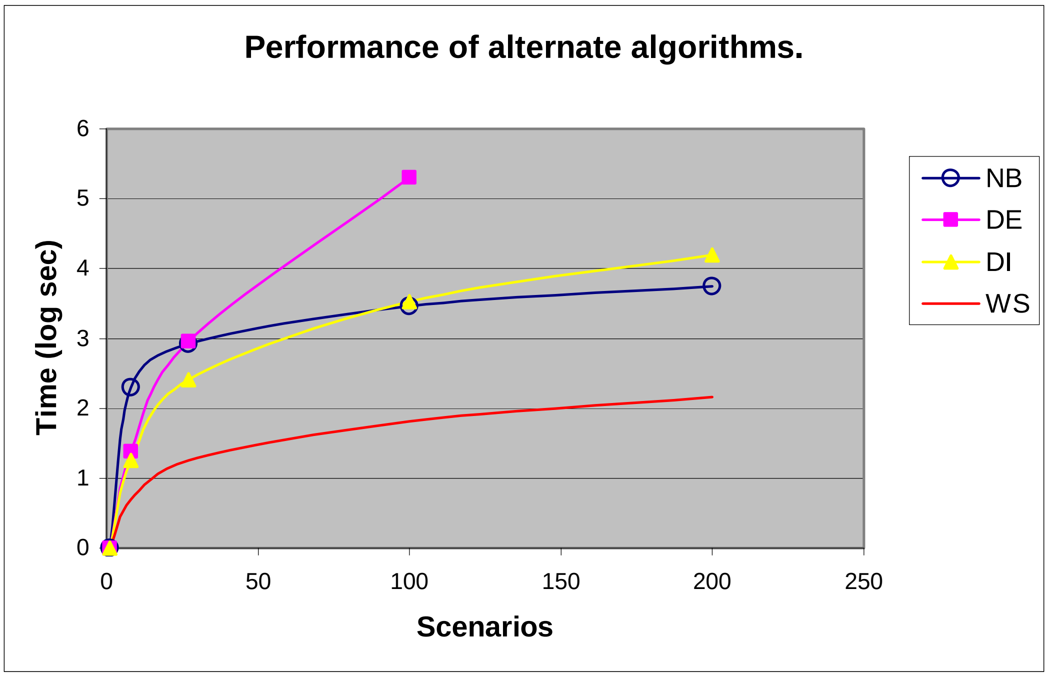


Figure 10. Performance of the algorithms on the STORM model.

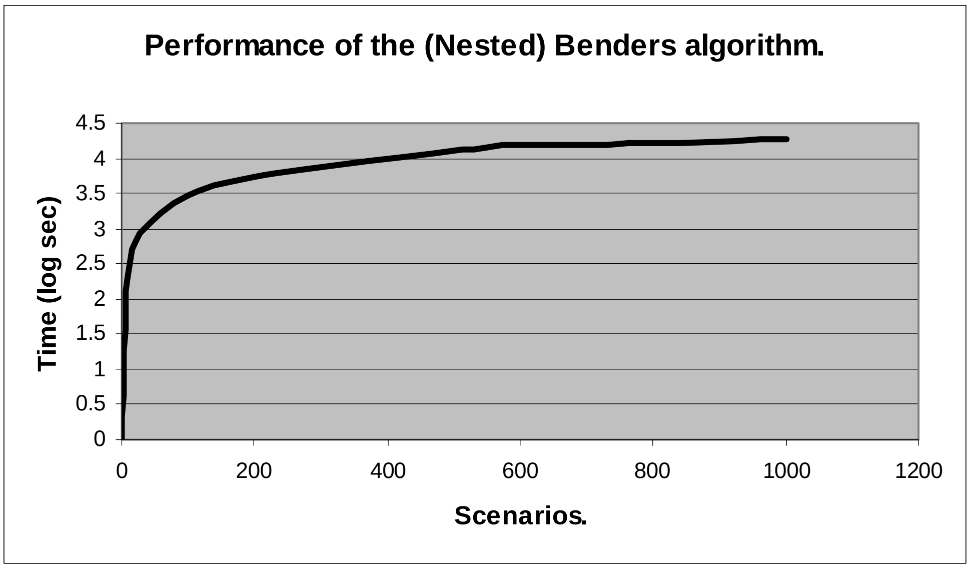


Figure 11. Performance of the (Nested) Benders on the STORM model.

Scenarios	NB	DI	DE	WS
1	1s	1s	1s	1s
100	23s	21	80s	3s
200	60s	73s	696s	6s
300	95s	206s	2376s	9s
400	139s	457s	5660s	12s
600	145s	828s	19237s	18s
1000	309s	2216s	∞	31s
2000	512s	12485s		63s
5000	1255s	∞		159s
15000	4727s			470s

25000	9651s			800s
32000	14654s			1020s

Table 12.Performance time (in seconds) for algorithms on the PHONE model.

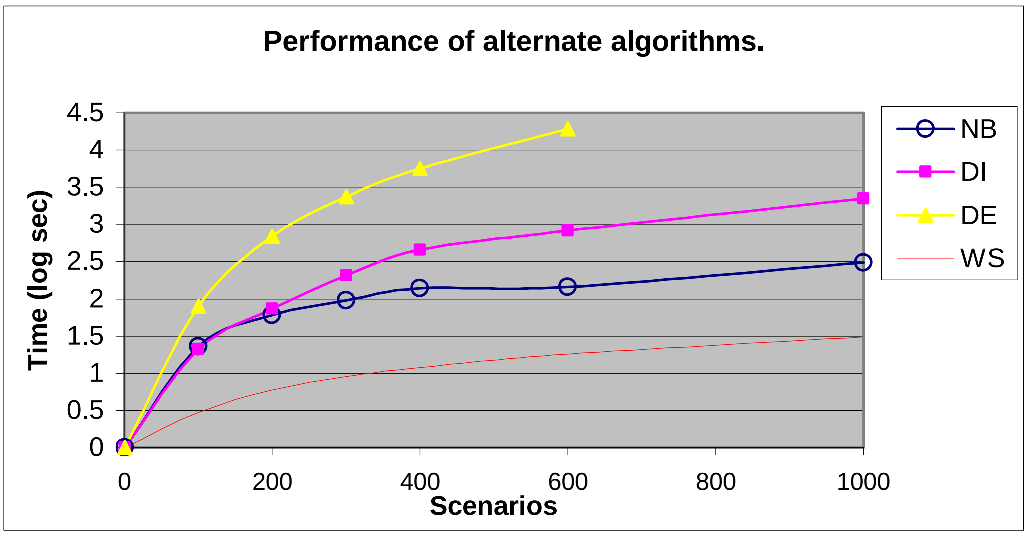


Figure 12. Performance of alternate algorithms on the PHONE model.

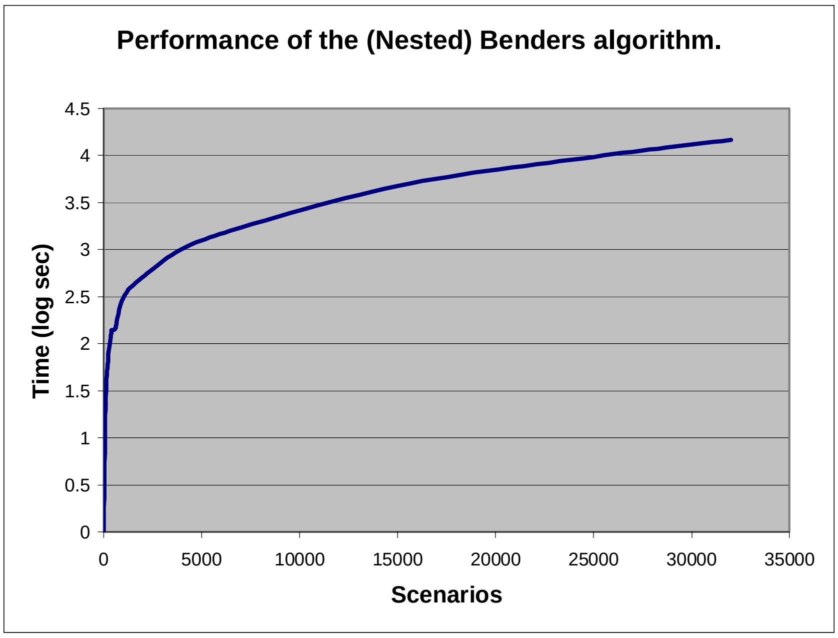


Figure 13. Performance of the (Nested) Benders on the PHONE model.

The IPM and the SSX solvers process these two deterministic equivalent representations (DEQ implicit) with varying efficiency and different scale up properties. We have not implemented any structure exploitation facility in our algorithms; in table 10 we simply set out our computational experience for these two alternate representations.

Model (Stages/Scenario)				
	Compact		Split-variable	
	IPM	SSX	IPM	SSX
Pgp2 (2/8)	1s	1s	1s	1s
Fxm (2/16)	18s	24s	47s	37s
Pltexp (3/36)	351s	32s	399s	65s
Storm (2/8)	16s	26s	23s	29s
Storm (2/27)	448s	323s	890s	329s
Storm (2/125)	NEM	8138s	NEM	13498s
Storm (2/1000)	NEM	NEM	NEM	NEM
Phone(2/32768)	NEM	NEM	NEM	NEM

NEM: Not enough memory.

Table 13. Performance of IPM and SSX on the two representations of the deterministic equivalent.

6 SPInE

A prototype of the SPInE environment was first designed in (Messina and Mitra 1997). The focus in that work was on the integration of modelling, solution and analysis tools to aid in the development and investigation of multistage recourse problems. The system was based on the MPL modelling system for the definition of the underlying core model, which was parametric in the scenario dimension. Several external files were needed to define the stochastic structure of the problem, leading to a laboriously and unnatural modelling process. However, the use of multidimensional database systems represented an innovative approach for the analysis of the results. The solver was based on the combination of MSLiP (Gassmann 1990) with the FortMP solver (Ellison et al. 1999).
We have revised the design of the first SPInE prototype adding several features, such as the support of the SMPL and SAMPL extended languages, the development of a new solver for SP multistage recourse problems, and the integration with scenario generators.
In subsection 6.1 we provide an overview of the software architecture. In 6.2 we describe the SPInE menu options and controls by means of a few example screen shots taken from the investigation of the ALM model illustrated earlier in the paper. In 6.3 we show how the system can be used, through the spreadsheet data exchange feature, to compute Value at Risk for the first stage decisions.

6.1 Software architecture: an overview

The new SPInE environment integrates a number of subsystems which are managed by a Control system. The subsystems as such are software components which may be used to create embedded applications.

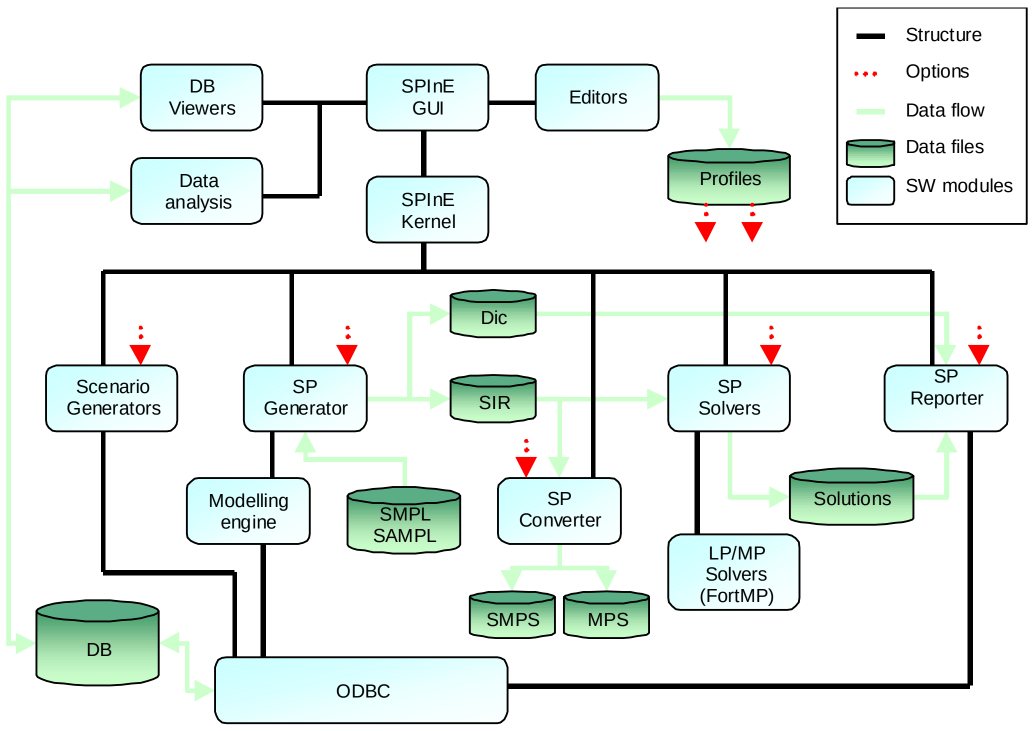


Figure 14. Software architecture of SPInE.

The diagram in Figure 14 illustrates the architecture of SPInE, and the interaction of the modules which together comprise the software environment. SPInE is divided into four main subsystems, namely Scenario Generation, Modelling, Solver, Results Analysis and the overarching Control module.

Scenario Generation

SPInE is designed to interface with scenario generators which supply the scenario data in ODBC databases or text files. An important aspect of the scenario generation interface is to establish the consistency between the SP model tree τ' and the data path tree τ underlying the scenario generation (see Section 4)

Modelling subsystem

The modelling subsystem is designed to support the language extensions SAMPL and SMPL introduced in section 3. The software module called *Stochastic Program Generator* (SPG) combines two separate parsers and a matrix generator. SPG processes together the algebraic models and the scenario data set to create an instance of the model in either SMPS format or in the Stochastic Intermediate Representation (SIR). SPG exchanges information externally using a set of files with a predefined format, and can be called as a command line application providing an options file, which contains directives for the execution. It also generates a *dictionary,* which maps the original names of the variables and constraints in the algebraic formulation to column and row numbers. The SPG module makes use of an underlying *modelling engine,* specifically OptiMax for the support of SMPL models and a comparable AMPL-based COM object, developed by our research group, for models prepared in SAMPL. The modelling system interacts directly with the scenario generators for the stochastic data and connects to the database systems which maintain the deterministic data relating to the core model of the SP problem.

Solver subsystem

Given a Stochastic Programming problem with recourse, the stochastic solver embedded in SPInE (see section 5) provides the solution to three related classes of models:

(i) Here and Now

(ii) Scenario Analysis (Wait and See)

(iii) Expected Value

For each of these, there is more than one possible *solution algorithm*. All underlying LPs may be solved using the Sparse Simplex algorithm (SSX) or the Interior Point Method (IPM). The Here and Now problem may be solved using Benders Decomposition, Lagrangean Relaxation or via the Deterministic Equivalent problem. The solver is also able to report the stochastic measures EVPI and VSS.

Results analysis

A critical phase in the development of stochastic models is the analysis of the solutions. The integration with database systems enables the exploitation of the Data Manipulation Languages (DML) which usually accompany the DBMS for the development of customised viewers and advanced data analysis tools. The SP Reporter (SPR) module of SPInE allows the user to export solution vectors using standard ODBC or using text files. The volume of the solution results produced by the stochastic solver can be very large. In fact, each decision variable has an associated optimal activity and reduced cost, for each stage and for each scenario. The investigator might be interested only in a subset of the solutions (e.g. the first stage strategic decisions). SPR provides *filtering* functionality which is used to transfer only the relevant decision data to the DBMS.

Control Module and Graphical User Interface

Each module in SPInE can be run as an independent application through script files. A control module including a Graphical User Interface (GUI) has been developed and can be used to investigate SP problems.

The GUI makes use of standard Windows objects to display and control hierarchical structures. It also provides with model management functionality, which is based on the concept of *project* (Figure 15). A project is a collection of stochastic models related to a specific problem. Each model, in turn, comprises a set of scenario data instances.

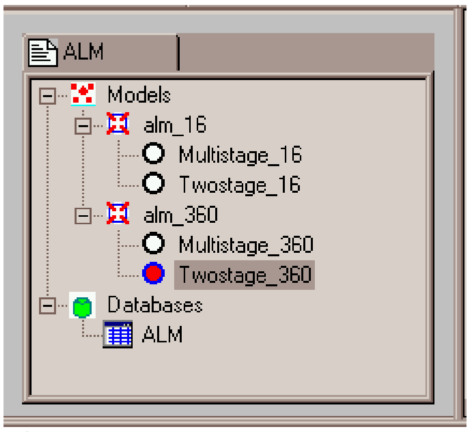


Figure 15. Project management in SPInE

The main subsystems of SPInE, namely the SP instances generator SPG, the stochastic solver SPS and the solution Reporter SPR have also been wrapped in a dynamic link library, which enables the rapid development of embedded applications.

6.2 Using SPInE: commands and controls

A sequence of control commands and dialog boxes are annotated below to illustrate a simple use of SPInE. After activating SPInE, the main window appears as shown in Figure 16. The menu bar and the alternative control features are also shown in this figure.

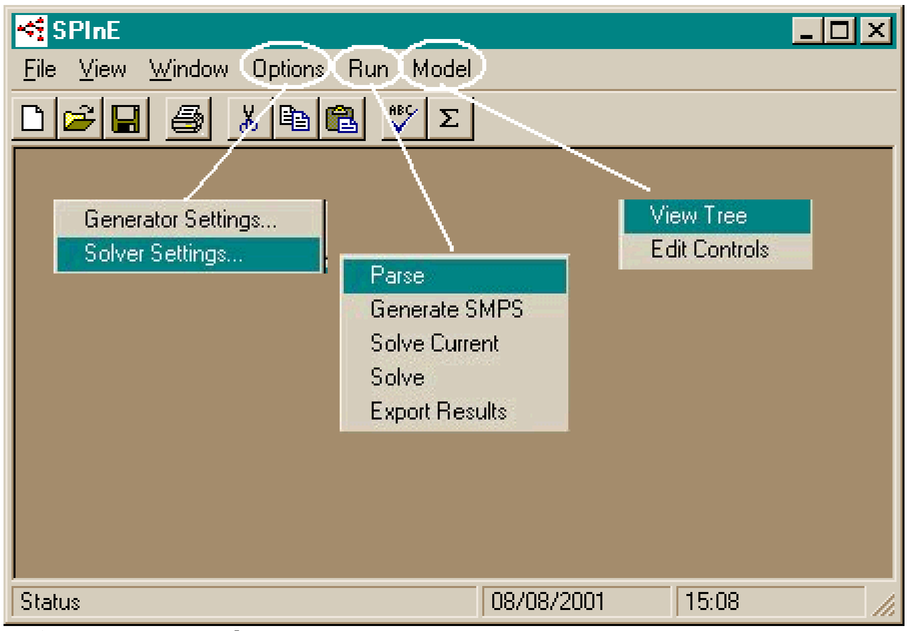
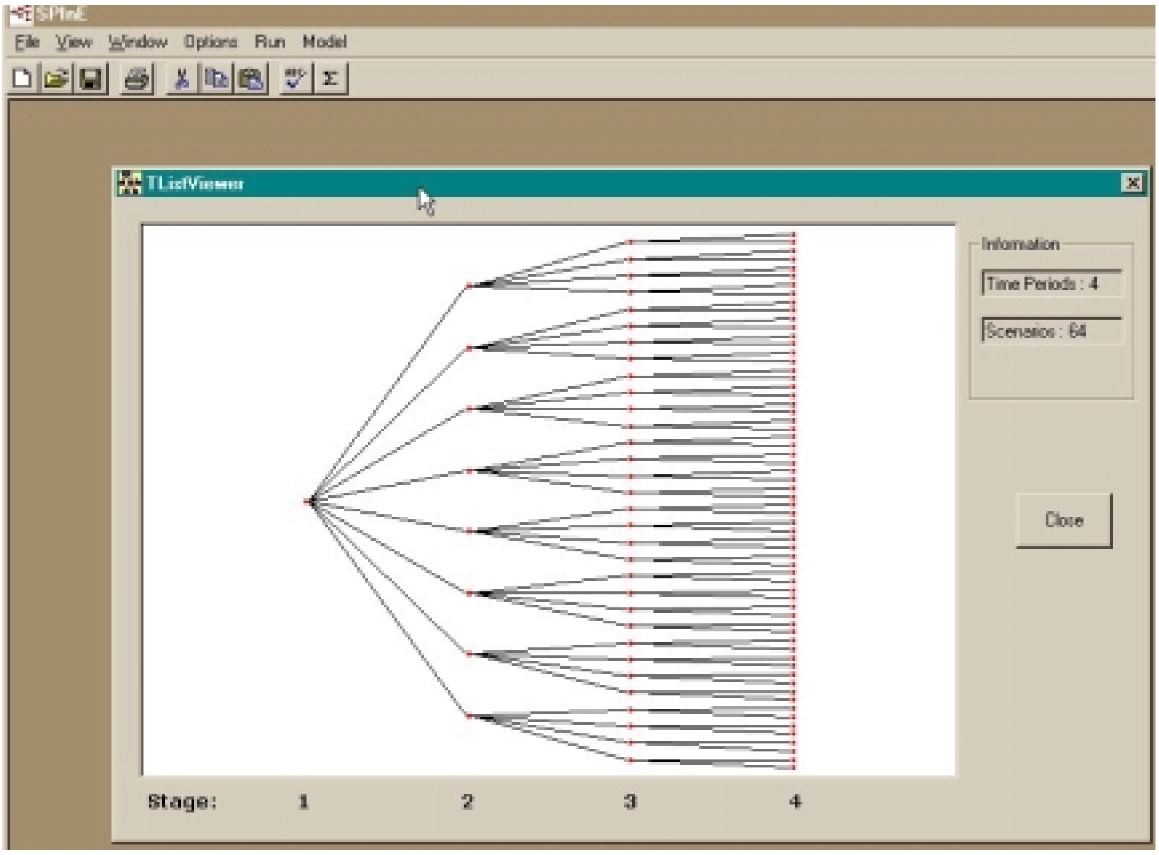


Figure 16. SPInE's menu commands

The main menu items are described as follows:

Options:	this menu item is used to specify generator and solver control settings.
Run:	this item enables the user to first parse the model, generate the SP instance, solve the model and finally export the results.
Model:	this menu item allows the user to view the scenario tree and to edit controls manually.

For the example ALM model, the *View Tree* command after *Parse* leads to the display shown in Figure 17.

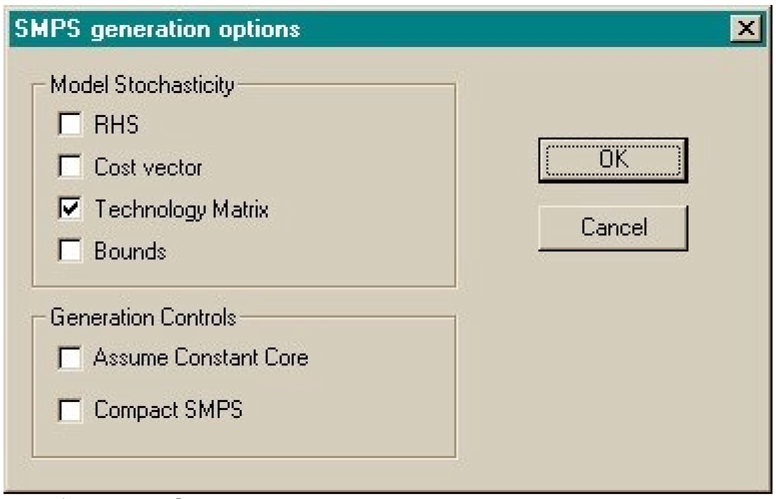


Time Periods : 4

Scenarios : 64

Figure 17. Scenario tree view

After the model has been parsed, the SMPS generation can be controlled using the dialog box shown in Figure 18.

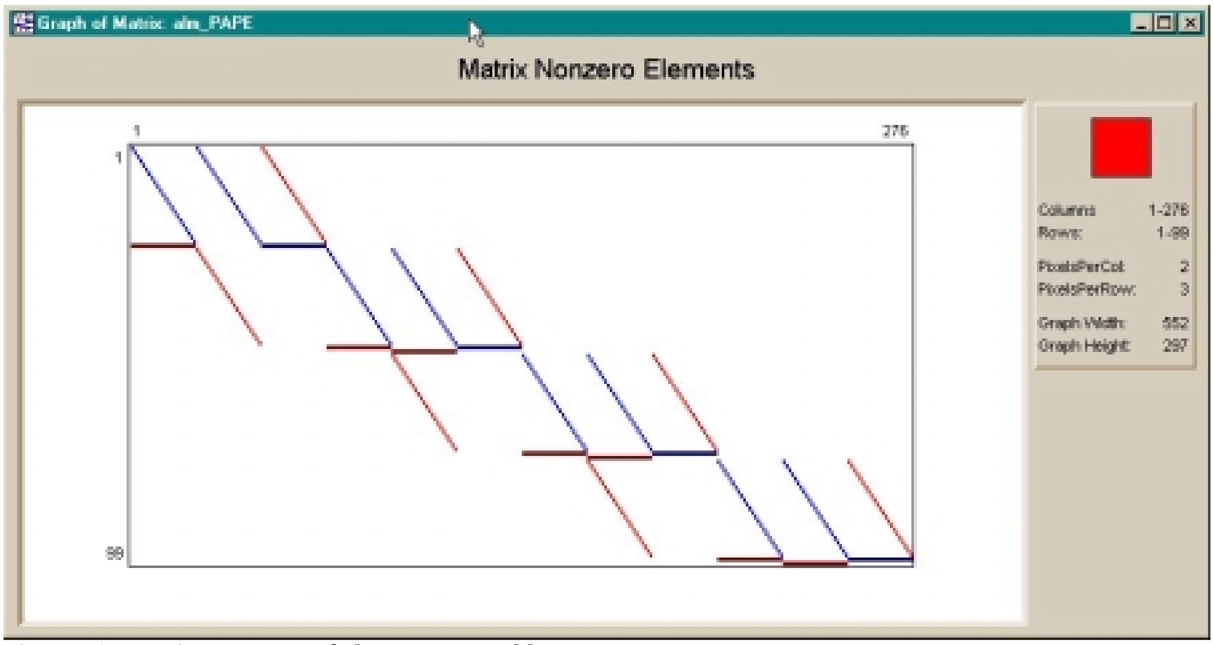

SMPS generation options

Model Stochasticity
- [ ] RHS
- [ ] Cost vector
- [✓] Technology Matrix
- [ ] Bounds

Generation Controls
- [ ] Assume Constant Core
- [ ] Compact SMPS

Figure 18. SMPS generation controls

The generated core matrix of the model can be also viewed as shown in Figure 19.

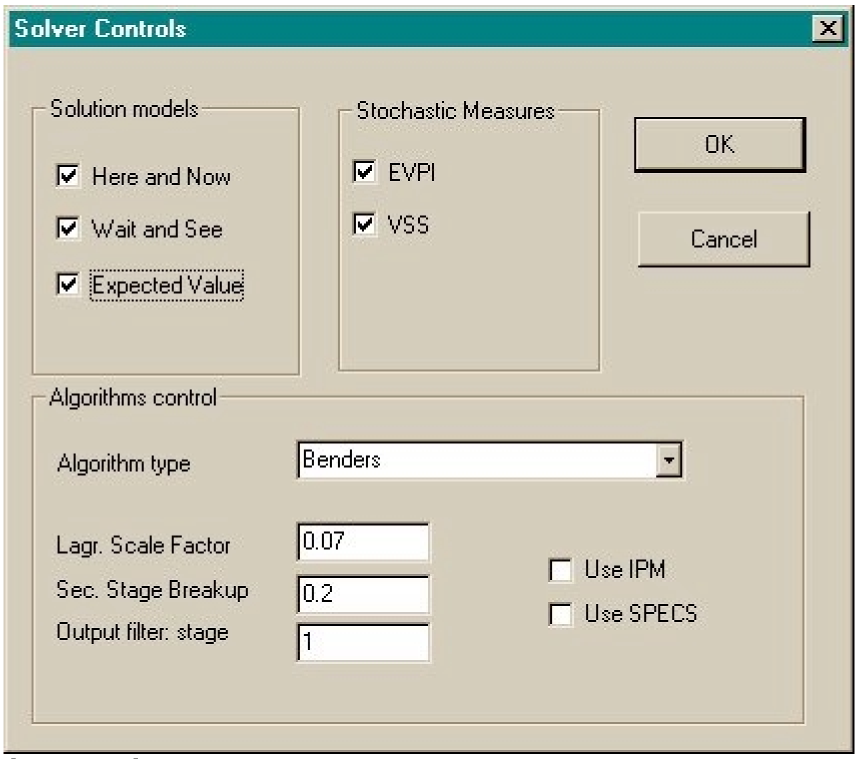

Graph of Matrix: alm_PAPE

Matrix Nonzero Elements

Columns	1-276
Rows:	1-99
PixelsPerCol:	2
PixelsPerRow:	3
Graph Width:	552
Graph Height:	297

Figure 19. Matrix structure of the core LP problem

The solver execution on the different related models (Here and Now, Expected Value, Wait and See) is controlled by the dialog box shown in Figure 20.


Solver Controls

Solution models

[✓] Here and Now

[✓] Wait and See

[✓] Expected Value

Stochastic Measures

[✓] EVPI

[✓] VSS

Algorithms control

Algorithm type: Benders

Lagr. Scale Factor: 0.07

Sec. Stage Breakup: 0.2

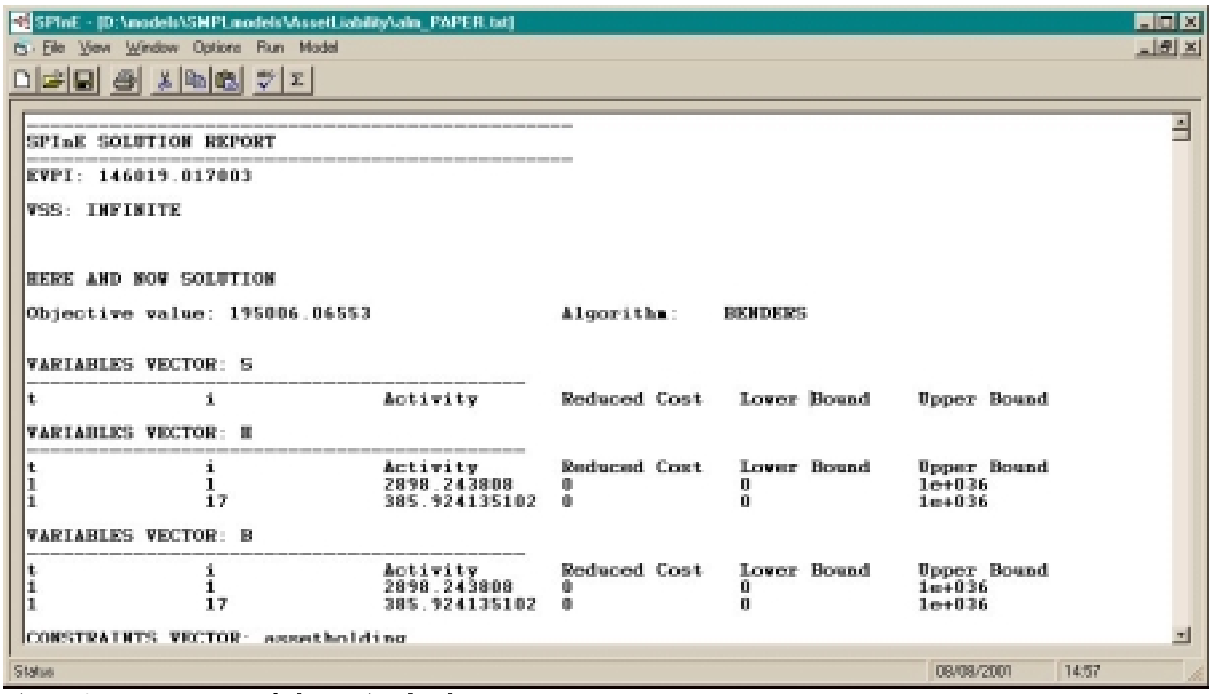
Output filter: stage: 1

[ ] Use IPM

[ ] Use SPECS

Figure 20. SP Solver controls

After solving the model, a compact report is produced if the *filter* control is used for the solver output. The results report for the ALM model is shown in Figure 21.


SPInE - [D:\models\SMPLmodels\AssetLiability\alm_PAPER.txt]
File View Window Options Run Model
Status
08/08/2001
14:57

```
---------------------------------------------------------------
SPInE SOLUTION REPORT
---------------------------------------------------------------
EVPI:  146019.017003

VSS:  INFINITE


HERE AND NOW SOLUTION

Objective value:  195006.06553                Algorithm:    BENDERS

VARIABLES VECTOR:  S
--------------------------------------------------------
t              i              Activity        Reduced Cost    Lower Bound    Upper Bound

VARIABLES VECTOR:  H
--------------------------------------------------------
t              i              Activity        Reduced Cost    Lower Bound    Upper Bound
1              1              2898.243808     0               0              1e+036
1              17             385.924135102   0               0              1e+036

VARIABLES VECTOR:  B
--------------------------------------------------------
t              i              Activity        Reduced Cost    Lower Bound    Upper Bound
1              1              2898.243808     0               0              1e+036
1              17             385.924135102   0               0              1e+036

CONSTRAINTS VECTOR:  assetholding
```

Figure 21. Text report of the optimal values

In this example, we solve the Expected Value (EV), the Wait and See (WS) and the Here and Now (HN, solved using Nested Benders Decomposition) models, and we compute the stochastic measures EVPI and VSS.

Table 14 reports the optimum values of objective functions of these models and the stochastic measures, where the risk level is set to R = 20%. The solution of the EV problem was not feasible for some of the scenarios; hence the VSS is infinite by definition.

EV	WS	HN	VSS	EVPI
151072.86	341025.08	195006.06	$+\infty$	146019.02

Table 14. Optimum values and stochastic measures for R=20%

6.3 VaR computation

SPInE can be also used to undertake more advanced investigation of an SP model. In many applications, quantification of the risk associated with a decision is becoming an important modelling issue. In general, Value at Risk (VaR) as a metric for computing risk has become widely accepted, particularly by the finance community (Rockafellar and Uryasev 2000). SPInE can be used to interact with Excel spreadsheet and can produce VaR metric for any given first stage decision using either in-sample or out-of sample scenarios.
For the given ALM model we are interested in computing and compare the VaR for the optimum first stage decisions x_{HN} given by the Here and Now model and the optimum first stage decisions x_{EV} given by the Expected Value LP model. These solution vectors are imported into Excel (see Figure 22) and are supplied as fixed values to a scenario analysis model which is used to simulate their performance.

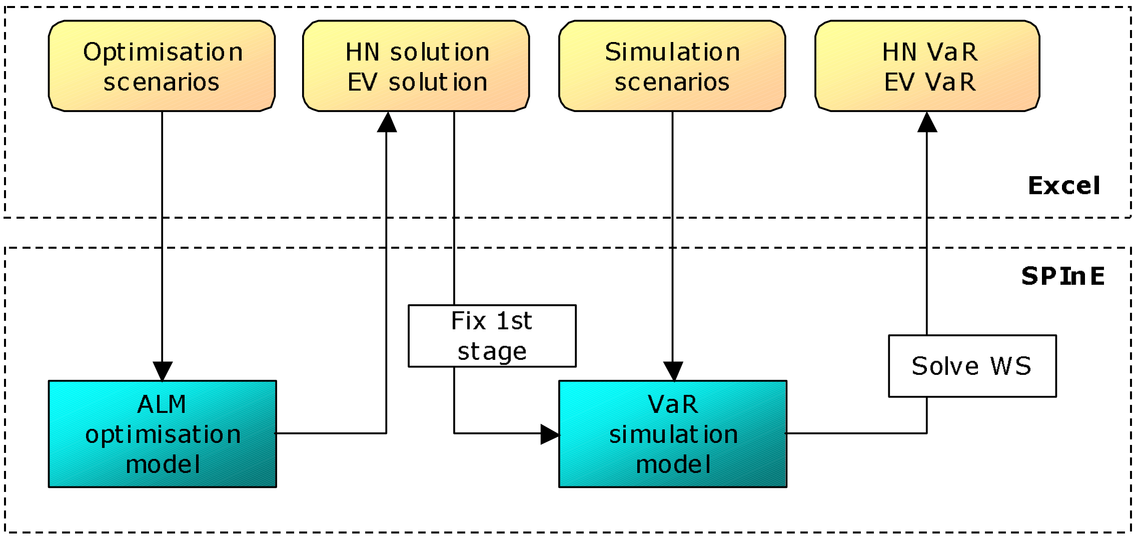


Figure 22. Computation of VaR using SPInE and spreadsheets

Two sets of frequency diagrams are computed (for x_{HN} and x_{EV}) taken over all the scenario realisations. This is equivalent to carrying out two simulation experiments, with 360 out-of-sample scenarios, again from the same generator. The optimum solution values are again exported back to Excel to display the results (see Figure 23).

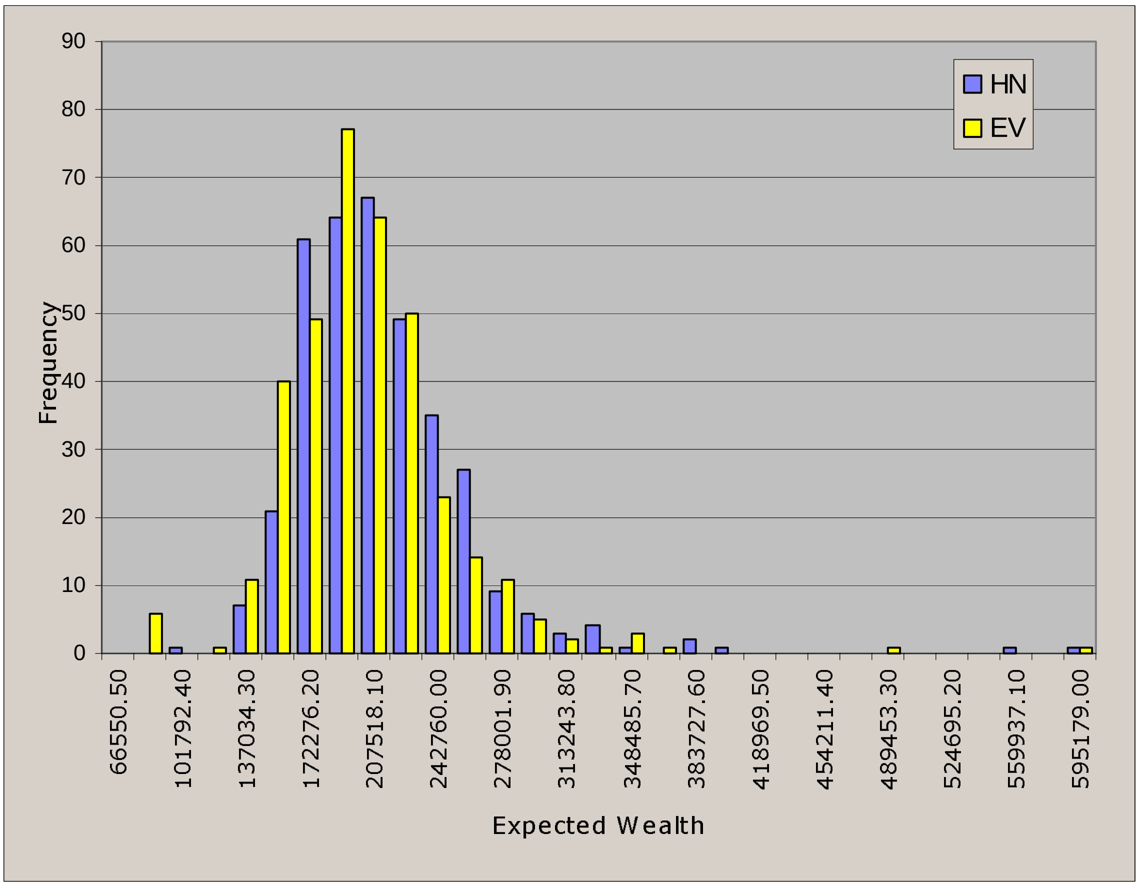


Figure 23. Distribution of the objectives for EV and HN

Thus for a β fractile of 0.99, we obtain the VaR profit levels which are shown in Table 15.

Implemented Solution	**VaR**
HN	131638
EV	82565

Table 15. VaR results.

It is clearly seen that for this model, the hedged Here and Now solution provides a better risk level than the Expected Value solution.

7 Future work and conclusions

Modelling and solving Optimum decision problems under uncertainty is a challenging task. We have highlighted the need for an integrated environment for modelling and solving SP models. In this paper, we have introduced our Stochastic Programming Integrated Environment (SPInE). This system provides practitioners with a powerful modelling system, based on two language extensions (SAMPL and SMPL) specifically designed for the definition of SP models. We have illustrated an asset/liability management model and we have used SMPL and SAMPL to formulate the corresponding multistage SP model. The interface designed for the connection to the scenario generators allows us to bring together the models of randomness with algebraic optimisation models. The variety of solution algorithms embedded in our SP solver engine and the ability to connect to databases for the analysis of the results make SPInE a complete and flexible tool for the implementation and investigation of Stochastic Programming problems.
We identify, however, a number of research issues of conceptualisation and software development which need further investigation.

(i) Connecting specific scenario generators to the SP models remains a thorny issue. Indeed in the generated model we work with a snap shot of the dynamic model. In a closely coupled modelling and solver system it is possible to investigate EVPI based sampling algorithm or importance sampling algorithms to create an iterative solution procedure.

(ii) The scenario generators can be extended to connect with historical data. Thus a framework of validation of the first stage decisions may be undertaken through simulation as well as back testing.

(iii) We aim to develop a solver which can process quadratic inequalities. Thus a range of chance-constrained models can be also processed by the system; this will enhance the scope of applying SP models.

(iv) We have earlier experience of parallelizing particular instances of SP models (MirHassani et al 2000). We wish to extend this work to include parallel implementations of Benders as well as the Lagrangean relaxation based Integer SP solver.

Appendix A

SAMPL and SMPL Libraries of SP models

We have prepared two libraries of SP models, one in SAMPL and the other in SMPL. The context of the models, the model structures and the sizes of the SMPS files which are generated, are summarised in the Table A.1 and A.2.

Model	Description	Stochasticity	Rows	Cols	Nz
ALM	Asset/Liability Management model with downside risk constraints	Technology	99	276	621
CLO	Distribution model	RHS	64	160	410
FINA	Asset/Liability Management model with diversification constraints	Technology	42	81	196
INTCLO	Mixed Integer distribution model	RHS, Bounds	251	473	1504
POWER	Simple power expansion model	RHS	9	12	33
MP	Small two stage IP model	RHS, Cost	2	2	3
MP2	Minimal two stage model	RHS	2	2	3

Table A.1. Models summary

Instance	Scenarios	Tree structure	DEQ Rows	DEQ Cols	DEQ Nz
ALM16ts	16	2 stage	1115	1998	4466
ALM48ts	48	2 stage	7080	13461	30211
ALM48ms	48	8 x 3 x 2	11560	13461	39171
ALM360ms	360	15 x 8 x 3	93895	100509	308095
CLO25ts	25	2 stage	2216	4040	9922
CLO125ms	125	5 x 5 x 5	19095	19539	70092
INTCLO10ts	10	2 stage	3675	4895	15292
FINA8ts	8	2 stage	614	393	1499
FINA8ms	8	2 x 2 x 2	676	677	2065
POWER	4	2 stage	39	51	131
MP	10	2 stage	21	21	41
MP2	5000	2 stage	10001	10001	20001

Table A.2. Structure and size of the SMPS instances

Appendix B

Quality Assurance and benchmark test problems

We have collected together a number of test problems which are taken mainly from

(a) Library supplied by Birge (Birge, 2001)

(b) Library prepared by Ariyawansa and Felt, (Ariyawansa and Felt, 2001)

(c) Industrial models developed by our research group (Mitra et al. 2001)

We use a subset of these models for the quality assurance of the solver; the content of these models, their structure and their size are summarised in Table B.1 and B.2. The full set of models is processed to gather benchmark performance data.

Model	Description	Stochasticity	Rows	Cols	Nz
SGPF5Y5[a]	Mixed Integer distribution model	RHS, Bounds	314	455	1058
AIRL[b]	Airlift operations scheduling, 2 stage mixed integer linear.	RHS	8	12	24
ASSET[b]	Network model for asset or liability management, 2 stage linear.	Technology, RHS	10	26	47
4NODE[b]	Cargo network scheduling	RHS	90	238	772
CHEM[b]	Design of batch chemical plants, 2 stage mixed integer linear.	RHS,Cost	84	80	186
ELECTRIC[b]	Electric investment planning, 2 stage linear.	RHS, Cost	9	16	36
ENV[b]	Energy and Environment planning	RHS	96	98	276
TRADE[c]	Supply chain planning	RHS	574	456	1342

Table B.1. Models summary

Instance	Scenarios	Tree structure	DEQ Rows	DEQ Cols	DEQ Nz
SGPF5Y5	3125	5 stage	157562	176151	530210
AIRL	25	2 stage	152	204	504
ASSET	37500	2 stage	187505	487513	975021
4NODE	256	2 stage	18960	47668	120112
CHEM	2	2 stage	130	121	289
ELECTRIC	2	2 stage	23	40	92
ENV	2	2 stage	288	294	852
TRADE	43	2 stage	12250	10536	37546

Table B.2. Structure and size of the SMPS instances

References

Andradottir S. (1995): A stochastic approximation algorithm with varying bounds, *Operations Research 1037-1048.*

Andradottir S. (1996): A scaled stochastic approximation algorithm, *Management Science, 475-498.*

Ariyawansa, K.A and Hudson, D.D. (1991): Performance of a benchmark parallel implementation of the Van-Slyke and Wets algorithm. *Concurrency: Practice and Experience 3, 109-128.*

Ariyawansa K.A. and Felt J.A. (2001): On a new collection of stochastic linear programming test problems, *Preprint at Optimization Online (www.optimization-online.org/).*

Benders J.F. (1962): Partitioning Procedures for Solving Mixed-Variable Programming Problems, *Numerische Mathematik 4,803-812.*

Birge J., Dempster M.A.H., Gassmann H.I., Gunn E., King A. and Wallace S.W. (1987): A standard input format for multiperiod stochastic linear programs, *Mathematical Programming Society Committee on Algorithms Newsletter 17, pp 1-19.*

Birge J.R., Louveaux F. (1997): Introduction to Stochastic Programming, *Springer-Verlag NY.*

Birge J.R. and Qi L. (1995): Continuous approximation schemes for stochastic programs, *Annals of Operations Research 56, 15-38.*

Birge J.R. and Wets R.J-B (1986): Designing approximation schemes for stochastic optimisation problems, *Mathematical Programming Study 27, 54-102.*

Birge J.R. Holmes D (2001): Computational results web page *(http://users.iems.nwu.edu/~jrbirge//html/dholmes/POSTresults.html)*

Bisschop J.J. and R. Entriken (1993): AIMMS User Guide, Version 2.03, *Paragon Decision Technology, The Netherlands.*

Bowermann B.L. and Koehler A.B. (1978): An optimal policy for sampling from uncertain distribution, *Communication Statistics - A theory methods 7,1041-1051.*

Brooke A., Kendrick D., Meeraus A., Raman R. (1998): GAMS: A User' s Guide, *Gams Development Corporation.*

Cariño D. R. and Turner A. L. (1997): Multistage Planning for Asset allocation, *World Wide Asset and Liability Modeling (W. T. Ziemba and J. M. Mulvey, eds.), Cambridge University Press, 1997.*

Charnes A. and Cooper A.A. (1959): Chance Constrained Programming, *Management Science, Vol.6, pp.73-79.*

Condevaux-Lanloy C. Fragnière E. (2000): An approach to deal with uncertainty in energy and environmental planning: the MARKAL case, *Environmental Modelling and Assessment 5, Issue 3 145-155.*

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

Corvera-Poiré X, (1995): STOCHGEN User's Manual, Departement of Mathematics, University of Essex.

Dempster M.A.H. (1988): On Stochastic Programming: dynamic problems under risk, *Stochastics 25.*

Dempster M.A.H. and Thompson R.T. (1999): EVPI-based importance sampling solution procedures for multistage stochastic linear programmes on parallel MIMD architectures, *Annals of Operational Research 90, 161-184.*

Dominguez-Ballesteros B., Mitra G., Koutsoukis N-S (1999): Modern Mathematical Programming Modelling Languages: A Comparative Review, *Technical Report in preparation, Brunel University.*

Dupač V. (1965): A dynamic stochastic approximation method, *Annals of Mathematical Sciences 6, 1695-1702.*

Ellison E.F.D., Hajian M., Levkovitz R., Maros I., Mitra G. e Sayers D. (1999): FortMP Manual, *OptiRisk Systems and Brunel University.*

Entriken R. (1997): Languages constructs for modelling stochastic linear programs, *SOL Report 97-1, Department of EESOR, Stanford University.*

Eppen G.D. Martin R.K. and Schrage L. (1989): A Scenario Approach to Capacity Planning, *Operations Research 37, 517-527.*

Escudero L.F., Galindo E., Garcia G. Gomez E., Sabau V. (1999): Schumann, a modelling framework for supply chain management under uncertainty, *European Journal of Operational Research (119) 1, 14-34.*

Frantzeskakis L. and Powell W.B. (1990): A Successive Linear Approximation Procedure for Stochastic, Dynamic Vehicle Allocation Problems, *Transportation Science 24, 40-57.*

Fourer R., Gay D.M., Kernighan B.W. (1993): Ampl: A modelling language for mathematical programming, *Duxbury Press/Brooks/Cole Publishing Company.*

Fourer R., Gay D.M. (1997): Stochastic Programming in the AMPL Modeling Language, *Session WE4-G-IN11 Modelling Support for Stochastic Programming, Intl. Symposium on Mathematical Programming, Lausanne, Aug 1997.*

Gaivoronski A.A. (1995): Stochastic Programming approach to the network planning under uncertainty, *in A. Sciomachen (ed.), Optimization in Industry, vol 3, Wiley and Sons.*

Gassmann H.I., (1990): MSLiP: A Computer Code for the Multi-Stage Stochastic Linear Programming Problem, *Mathematical Programming 47, 407-423.*

Gassmann H.I., Ireland A.M. (1995): Scenario formulation in an algebraic modelling language, *Annals of Operations Research 59, pp. 45-75.*

Gassmann H.I., Ireland A.M. (1996): On the formulation of stochastic linear programs using algebraic modelling languages, *Annals of Operations Research 64, pp. 83-112.*

Geoffrion A.M., (1992): Indexing in Modelling Languages for Mathematical Programming, *Management Science 38, No. 3.*

Glassey C.R. (1973): Nested decomposition and multi-stage linear programs, *Management Science, 20, 282-292.*

Higle J.L, Sen S. (1996): Stochastic Decomposition: A Statistical Method for Large Scale Stochastic Linear Programming, *Kluwer, Boston.*

Ho J.K., Loute E. (1974): Nested decomposition for dynamic models, *Mathematical Programming, 6, 121-140.*

Høyland K. and Wallace S.W. (2001): Generating scenario trees for multistage problem, *Management Science 47 (2), 295-307.*

IBM World Trade Corporation (1976): IBM Mathematical Programming System Extended /370 (MSPX/370): *Program Reference No. SH19-1095-1, New York & Paris.*

Infanger, G. (1994): Planning Under Uncertainty: Solving Large-Scale Stochastic Linear Programs, *The Scientific Press series, Boyd and Fraser.*

Infanger, G. (1997): DECIS User's Guide, *Dr. Gerd Infanger, 1590 Escondido Way, Belmont, CA 94002.*

Jobst N.J., Horniman M.D., Lucas C.A. and Mitra G. (2001): Computational Aspects of Alternative portfolio selection Models in the presence of Discrete asset choice Constraints, *to appear in The Journal of Quantitative Finance.*

Kall P. and Mayer J. (1996): SLP-IOR: An interactive model management system for stochastic linear programs, *Mathematical Programming 75, 221-240.*

King A. (1994): SP/OSL Version 1.0 Stochastic Programming Interface Library: User' s Guide, *Research Report RC 19757 (87525) 9/26/94 Mathematics, T.J Watson Research Center, Yorktown Heights, NY.*

Kushner H. (1971): Introduction to Stochastic Control, *Holt, New York, NY.*

Kusy M.I., Ziemba W.T. (1986): A Bank Asset and Liability Model, *Operations Research 34, 356-376.*

Kyriakis T. (2001): A Stochastic Programming approach to asset and liability management, *Ph.D Thesis.*

Levkovitz, R, Mitra G. (1993): Solution of large-scale linear programs: A review of hardware, software and algorithmic issues. *In: T.A. Ciriani & R.C. Leachman (eds) Optimisation in Industry. John Wilsey and sons, 139-171.*

Lucas C., Messina E. and Mitra G. (1996): Risk and Return analysis of a multi-period strategic planning problem, *in L. Thomas and A. Christers (eds.) Stochastic Modelling in Innovative Manufacturing, Springer Verlag,(81-96):*

Luenberger D.G. (1997): Investment Science, *Oxford University Press, New York.*

Markowitz H. (1952): Portfolio Selection. *Journal of Finance 7, 77-91*

Maximal Software Incorporation (2000): MPL Modelling System, *Release 4.11, USA.*

Messina E., Mitra G. (1997): Modelling and analysis of multistage Stochastic Programming problems: A software environment, *European Journal Of Operational Research 101 2, pp. 343-359.*

MirHassani S.A, Lucas C., Mitra G., Messina E. and Poojari C.A, (2000): Computational Solution of Capacity Planning Models under Uncertainty, *Parallel Computing 26, pp.511-538.*

Mitra G. Lucas Poojari C.A. Dominguez D-B (2001): A combined model for production capacity planning & inventory scheduling under uncertainty, *Informs Hawaii 2001.*

Mulvey J.M. (1996): Generating Scenarios for the Towers Perrin Investment System, *Interfaces 26, 1-13.*

Mulvey J.M. and Ruszczyński A. (1995): A new scenario decomposition method for large scale stochastic optimisation, *Operational Research 43, 3, 477-490.*

Mulvey J.M. and Vladimirou H. (1991): Applying the progressive hedging algorithm to stochastic generalised networks, *Annals of Operations Research 31,399-424.*

Mulvey J.M. and Ziemba W.T., editors (1998): Worldwide Asset Liability Management. *Cambridge University Press.*

Pereira M.V.F. and Pinto L.M.V.G. (1991): Multi-stage Stochastic Optimization Applied to Energy Planning, *Mathematical Programming 52, 359-375.*

Poojari C.A. and Mitra G. (2001): A solver system for multi-stage Stochastic Programming problem, *Working paper, Department of Mathematical Sciences, Brunel University.*

Poojari C.A., Lucas C.A., MirHassani A. and Mitra G. (2000): Solving two-stage Stochastic Programming problem using lagrangean relaxation: An enumerative approach, *Submitted to Journal of Global Optimisation.*

Poojari C.A. and Mitra G. (2001): A solution technique for two-stage stochastic program with first–stage integer variables. *Presented at the Ninth International conference on Stochastic Programming, Berlin.*

Powell W.B. (1988): A Comparative Review of Alternative Algorithms for the Dynamic Vehicle Allocation Program, *Vehicle Routing: Methods and Studies, Golden and Assad (eds.): North-Holland.*

Rockafellar R.T. and Uryasev S. (2000): Optimization of Conditional Value at Risk, *The Journal of Risk 2, (#3).*

Ruszczyński A. (1993): Parallel decomposition of multistage Stochastic Programming problems, *Mathematical Programming 58,201-228.*

Ruszczyński A. (1986): A regularised decomposition for minimising a sum of polyhedral functions, *Mathematical Programming 35, 309-333.*

Sen S., Doverspike R.D., and Cosares S. (1994): Network planning with random demand, *Telecommunication Systems, 3:11-30.*

Shapiro J. (2001): Modeling the Supply Chain, *Duxbury Press, California.*

Valente P., Fourer R. and Mitra G. (2001): Extending algebraic modelling languages for Stochastic Programming, *Working Paper, Brunel University and Northwestern University.*

Van Slyke R.M., Wets R.J.-B (1969): L-Shaped Linear Programs with Applications to Optimal Control and Stochastic Programming, *SIAM Journal on Applied Mathematics 17,638-663.*

Zenios S.A (1993): Financial Optimisation, Essays dedicated to G.B. Dantzig, Medal of the National Academy of Sciences, USA, and H. Markowitz, Nobel Prize in Economics 1990, *Cambridge University Press.*

Zenios S.A. Censor Y. (1997): Parallel Optimization: Theory, Algorithms, and Applications (Numerical Mathematics and Scientific Computation): *Oxford, Oxford University Press.*

Wright S (2001): Solving Optimization Problems on Computational Grids, *Optima Mathematical Programming Society Newsletter, May 2001.*