

OptiRisk Systems: White Paper Series
Domain: Finance Reference Number: OPT 004

**SCENARIO GENERATION
FOR
STOCHASTIC PROGRAMMING**

Last Update
24 April 2008

OptiRisk
SYSTEMS



INVESTOR IN PEOPLE
BS EN ISO 9000 : 2000



A White Paper on Scenario Generation for Stochastic Programming

Sovan Mitra

July 2006

Acknowledgement

I gratefully acknowledge the comments and advice from Prof.G.Mitra , Dr.C.Lucas, Dr.P.Date and Asemali Fazal.

Contents

1	Scope and Purpose	5
1.1	Scope	5
1.2	Purpose	5
2	Background	6
3	Scenario Generation Introduction	7
3.1	Scenario Generation Definition	7
3.2	Scenario Generation Quality Measures	10
3.2.1	Sample Stability	10
3.2.2	Testing for Bias	10
3.3	Scenario Generation Computational Procedure	11
4	Uses of Scenarios	12
5	Scenario Generation by Statistical Approaches	13
5.1	Statistical Moment or Property Matching	13
5.2	Principal Components Analysis (PCA)	14
5.3	Regression and its variants	14
6	Scenario Generation by Sampling	15
6.1	Monte Carlo or Random sampling	15
6.2	Importance Sampling	15
6.3	Bootstrap Sampling	15
6.4	Internal Sampling	15
6.5	Conditional Sampling	16
6.6	Stratified Sampling	16
6.7	Markov Chain Monte Carlo Sampling (MCMC or Monte Carlo based on Markov Chain)	16
7	Scenario Generation by Simulation	18
7.1	Stochastic Process Simulation	18
7.2	Error Correction Model (ECM)	18
7.3	Vector Auto-regressive (VAR)	19
8	Other Scenario Generation Methods	20
8.1	Artificial Neural Networks	20
8.2	Clustering	20
8.3	Scenario Reduction	21
8.4	Hybrid Methods	21
9	Conclusion	22
10	Appendix 1:Introduction to Stochastic Programming	23
10.1	Importance of Stochastic Programming to Optimisation	23
10.1.1	Link between Stochastic and Linear Programming:the Expected Value Core Model	24

10.2	Conceptualisation of the Modelling Process	24
11	Appendix 2:Classes of Stochastic Programs	26
11.1	Distribution Problems	26
11.1.1	Expected Value Approach	27
11.1.2	Wait and See Approach	27
11.2	Recourse Problems	28
11.2.1	Multistage Recourse	29
11.2.2	Recourse Problem Categorisation	30
11.3	Chance Constraint Problems	31
11.4	Stochastic Programming Stochastic Measures	31
11.4.1	EVPI (Expected Value of Perfect Information)	31
11.4.2	VSS (Value of the Stochastic Solution)	32
	References	33

1 Scope and Purpose

1.1 Scope

This white paper surveys stochastic programming scenario generation methods. We introduce the basic concepts relating to scenario generation, the main scenario generation methods -by sampling, simulation and statistical approaches. We also review new scenario generation methods such as "hybrid" methods and a review of basic stochastic programming theory can be found in the appendices.

1.2 Purpose

The purpose of this white paper is to provide an informed survey of stochastic programming scenario generation methods. This informational analysis will be of value not only to academics but also modelling practitioners.

This paper will be of interest to

- quantitative analysts
- operational research analysts
- other professional involved in the area of stochastic modelling and optimisation

2 Background

In today's competitive and global business environment businesses are constantly confronted with the position of making decisions that involve a degree of uncertainty e.g. portfolio investment, power production and even agricultural planning (see [2]).



The ability therefore to be able to quantify and *optimise* such decisions is of paramount value. For instance Mulvey's celebrated Towers Perrin-Tillinghast ALM (Asset and Liability Management) Model saved US WEST "\$450 to \$1000 million in opportunity costs in its pension plan" [15]. The modelling technique of *Stochastic Programming* offers such a possibility.

Stochastic programming (from hereon SP) is a flexible and widely applicable modelling method that can incorporate a high degree of uncertainty, hence its popularity in financial modelling. Furthermore SP has been shown to provide more robust optimisation solutions compared to equivalent deterministic or linear programming models.

The modelling capability of SP rests heavily upon the ability to adequately model uncertainty, which is captured through scenarios. Incorporating little uncertainty reduces stochastic programs to deterministic or linear programming models. On the other hand arbitrarily incorporating any uncertainty can lead to unrealistic or uninformative modelling and analysis. Thus *scenario generation* is of central importance to any stochastic programming model.

In this white paper we explain the importance of scenario generation to stochastic programming and identify the key scenario generation problems. Next, a comprehensive survey of scenario generation methods is presented, specifically using statistical, sampling and simulation approaches and we also survey more recent techniques such as hybrid approaches. We finally end with a conclusion.

3 Scenario Generation Introduction

3.1 Scenario Generation Definition

Before reading the rest of the white paper it is assumed the reader has read or knowledgeable of basic stochastic programming theory, which can be found in the appendices.

For SP randomness is accounted for by the scenario tree. Thus we require the pdf of random parameters to be approximated by a discrete distribution of a limited number of outcomes. This modelling of data by using the scenario tree as its approximation is known as scenario generation and is central to the quality of the SP.

Surprisingly, scenario generation is not necessarily equivalent to finding a statistical approximation. Kaut and Wallace [11] mention that some scenario generation (from hereon SG) methods approximate distributions perfectly when the number of outcomes go to infinity but can perform poorly when using only a few outcomes. Similarly some methods do not guarantee convergence to the true distribution but perform well for SP modelling purposes.

Scenario generation is important in terms of improving computation; if we do not approximate distributions, computation can become infeasible ([10]). Although substantial sampling of some form may be employed (internal, stratified, random etc...) to improve computation, this still results in a large number of outcomes/computations and so causes computational difficulties. Scenario generation, by the very fact it has far fewer scenarios/outcomes, requires less computation and provides a significant computational advantage.

Additionally, scenario generation is important as a financial risk management tool. The Basel Committee propose using scenario based risk management, in particular for volatility [1].

The 4 main scenario generation methods are:

- Sampling
- Statistical Approaches
- Simulation
- Other methods e.g. hybrids

Examples of scenario generation are given below from [4]:

Modelling Paradigm	SG Method	Origin	Application Field	Reference
Econometric Models and Time Series	AR (p)	Autoregressive Models and Generation of Data Trajectories.	Finance, Supply chains, Environment models	G. Box and F.M. Jenkins, 1976
	MA (q)	Moving Average Models and Generation of Data Trajectories	Finance, Supply chains, Environment models	G. Box and F.M. Jenkins, 1976
	ARMA (p,q)	Autoregressive Moving Average Models and Generation of Data Trajectories.	Finance, Supply chains, Environment models	G. Box and F.M. Jenkins, 1976
	GARCH	Generalised Autoregressive Conditional Heteroscedasticity and Generation of Data Trajectories.	Finance, Supply chains, Environment models	T. Bollerslev 1986, R. F. Engle 1982
	VAR	Vector Auto Regressive Models and Generation of Data Trajectories	Finance, Supply chains, Environment models	Fair and Shuller, 1990, Sims 1978
	BVAR	Bayesian Vector Auto Regressive Models and Generation of Data Trajectories.	Finance, Supply chains, Environment models	Ansley, C. F. and Kohn, R, 1986
	Reduced Rank Regression	Generation of Data Trajectories	Finance, Supply chains, Environment models	Engle and Granger, 1987
	Wiener Processes	Brownian Motion and Diffusion Processes	Finance and Environment models	D. Freedman 1983
	Generalised Wiener Processes	Brownian Motion with drift and Diffusion Processes	Finance and Environment models	T. Bollerslev 1986
	Geometric Brownian Motion			

Modelling Paradigm	SG Method	Origin	Application Field	Reference
Artificial Intelligence	Neural Gas	Neural Networks	Supply Chains, Energy, Environment models	Martinetz 1991, Futzke 1997, Ceasola 2004
	Property Matching	Statistical Approximation	Supply Chains, Energy models	Heyland and Wallace, 2001
Statistical Approaches	Moment Matching	Moment Fitting	Supply Chains, Energy, Environment models	Heyland et al. 2003
	Non Parametric Methods	Discretisation	Finance and Environment models	Heyland and Wallace, 2001
	SG Forecasting Methods	Quantile Regression and Forecasting Methods	Finance, Supply chains, Environment models	Tonnagaard et al. 2004
	Random Sampling	Discrete Sampling	Finance and Environment models	Jobst and Zenios 2001
	Stratified Sampling	Interval Sampling	Finance and Environment models	Jobst and Zenios 2001
Sampling	Bootstrap	Discrete Sampling	Finance models	Efron (1979)
	Monte Carlo	Sampling	Finance models	Jerrum and Sinclair 1996
	Markov Chains	Probability Interval Sampling	Finance, Energy, Supply Chains, Environment models.	Jerrum and Sinclair 1996
	VECM	Random path and Vector Error correction	Finance	Volosov et al. 2005

- Statistical Methods: the general principle behind applying statistical methods is to determine the values of particular statistical properties (e.g. 1st ,2nd moments) of some given (distribution of) data. Values of statistical properties can then in turn be used to determine the "best fit" theoretical distribution to data and so the theoretical distribution can be used for generating scenarios.

- Sampling: this method takes sample values from a given pdf, thus providing scenario values.
- Simulation: this involves the simulating of a process by inputting random numbers into its equation. The results give us the random variable's realisations, thus providing scenario values.
- Others: these include a variety of methods e.g. mixing scenario generation of sampling with moments matching.

3.2 Scenario Generation Quality Measures

Scenario Generation methods differ in their ability to model randomness. Kaut and Wallace in [11] discuss how to assess the quality of scenario generation and argue that it should be performance based rather than based on theoretical properties. In [11] they assess SG quality by 2 properties:

- in sample and out of sample stability
- testing for bias

We now explain these metrics.

3.2.1 Sample Stability

Sample stability can be seen as a robustness requirement; the *optimal* solution of the objective function should vary insignificantly to variations in the input. Note that the solutions may vary even when the same input is used as the SG method itself may give different random variable realisations e.g. random sampling from the same distribution (the input).

For in-sample stability we require the objective function's optimal solution to vary insignificantly as inputs are taken from the same sample. If the optimal solution does not change significantly for inputs taken from outside the sample, we have out of sample stability.

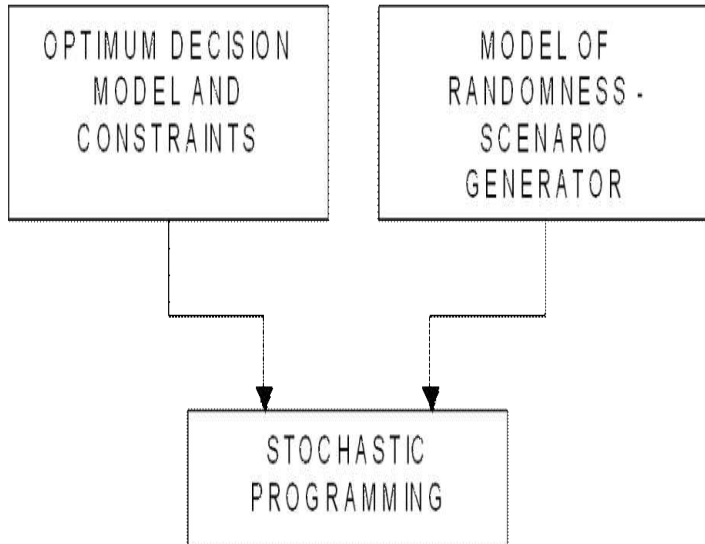
For out of sample testing one needs the random variable's complete pdf and so out of sample testing guarantees stability in the "real world", whereas in sample stability does not. Interestingly, out of sample stability does not imply in sample stability and vice versa -for a discussion on this see [11].

3.2.2 Testing for Bias

In testing for bias, we wish to determine if the scenario generation method itself introduces any bias producing the optimal solution. This can be theoretically achieved by comparing the optimal solution of the SP model to the optimal solution obtained from the "true" statistical process. Practically this is not applied as solving the theoretically optimal solution would imply the SP model is redundant.

3.3 Scenario Generation Computational Procedure

We can dissect a SP's components pictorially as below:



The optimum decision model together with the constraints constitutes the core of the problem that has to be solved and varies with respect to the specific characteristics of each individual application.

The second element is the model of randomness. Once the distribution is established, different scenarios that follow the underlying probability distribution are used to represent the uncertainty. Thus, by executing a scenario generator procedure, the random parameters of the decision model can be instantiated.

In general, a scenario generation procedure (for multistage problems) involves some or all of the following steps:

- Collecting historical observations of the parameter values, assumptions of a model, which explain the behaviour of the random parameters (for instance, econometric models for interest rates and retail price index)
- Estimation/Calibration of parameters for the chosen model, which uses historical data/subjective judgement.
- Generation of data trajectories (paths) according to the chosen model or discretisation of the distributions using approximation of statistical properties.

4 Uses of Scenarios

Scenarios in general are useful for 2 main model types: descriptive and decision models (for instance see Mitra [14]). Descriptive models are defined by a set of mathematical operations that can predict how a mathematical system will behave e.g. markov models. Decision Models on the other hand do not fully describe a system but are sufficiently descriptive to enable decision making e.g. linear and stochastic programming models. Consequently scenarios are necessary not just in stochastic programming/decision models but also descriptive models to enable 2 types of modelling, that is, ex ante and ex decision modelling.

Ex-ante decision modelling: "ex-ante" is Latin for "before the event" thus ex-ante analysis is analysis of potential future events (e.g. future asset returns) and so useful for understanding the possible impact of decisions. Hence in stochastic programming theory ex ante analysis amounts to in sample scenario generation.

Ex-post decision modelling:ex-post translated from Latin means "after the event" and can be considered the opposite of ex ante analysis; ex-post analysis is the analysis of possible future events using historical/"real" data (e.g. historical asset returns). Historical data or ex post analysis traditionally has been the most common method of future event analysis. In stochastic programming ex post analysis amounts to out of sample scenario generation.

5 Scenario Generation by Statistical Approaches

In this section we survey the scenario generation methods by statistical approaches.

5.1 Statistical Moment or Property Matching

Statistical moment or property matching is a generic estimation technique, whereby we do not assume knowledge of a random variable's pdf. Instead we describe the distribution by its statistical moments or other properties e.g. percentiles. From such properties we can then construct a discrete pdf or scenario tree consistent with such properties. Alternatively, we can construct the best theoretical distribution fitting the statistical properties and thus use a theoretical distribution to obtain scenarios e.g. by sampling.

Hoyland and Wallace in [10] describe a scenario generation method using statistical property matching, in particular moment matching. A scenario tree actually specifies a (discrete) pdf, consequently Hoyland and Wallace propose a method of constructing a scenario tree such that it matches the uncertain random variable's pdf as closely as possible.

Rather than minimising the error difference at every point between the scenario tree and a desired distribution, Hoyland and Wallace minimise the error difference between each pdf's moments.

In order to find the best set of scenarios that minimises moment error differences, we first must specify the number of scenarios we wish to have. Naturally, the more scenarios one uses, the lower the total distance value. Then the scenario tree, values and branch probabilities are found by non-linear optimisation, where the objective is to minimise the error between the scenario tree's resultant moments and the random variable's moments.

Interestingly, the moments can be well-matched only with a handful of scenarios. This is an important advantage over sampling, whereby sampling of a pdf for scenario generation can result in too many scenarios.

Hoyland and Wallace state that non-moment statistical properties can be used instead [10] but they argue that the matching properties should be problem dependent.

Additional benefits of statistical property matching include:

- can incorporate asset correlations
- can incorporate inter-period statistical dependencies
- can specify a worst case scenario as a statistical property to satisfy
- the method's performance is not dependent on the pdf but on the pdf properties e.g. moments

5.2 Principal Components Analysis (PCA)

PCA is a generic method of data analysis, enabling identification of key factors in data trends of many dimensions. This is achieved by identifying the data's eigenvectors, eigenvalues and covariances and thus is suited to computational implementation.

Dr.L.Smith's notes "A Tutorial on Principal Components Analysis" (from Cornell University) is a good tutorial available free on the internet.

5.3 Regression and its variants

Regression is the method of statistically "fitting" a given equation to data. Variants on the regression method include quantile regression and reduced rank regression [6].

6 Scenario Generation by Sampling

In this section we survey the scenario generation methods by sampling approaches. For each sampling method, the main principle is to take a sample from a pdf so that for a given value x we have an associated probability. The value x and its probability give us our scenario value and its branch probability.

6.1 Monte Carlo or Random sampling

In some situations we are given a random variable's pdf, $p(x)$. In random or Monte Carlo sampling we randomly pick numbers for x and using $p(x)$ we can obtain x 's probability. In this way scenarios are obtained whereby x is scenario values with associated probability $p(x)$. Essentially this SG approximates the pdf using a set of samples as scenarios.

6.2 Importance Sampling

By the CLT (central limit theorem), random sampling implies we should get a good approximation of the distribution and this approximation improves as the number of samples increases. However, we can improve the accuracy of our approximation if we restrict random samples to areas which contribute to the distribution more than others. The basic concept is essentially to give more weight or importance to sampling some regions more than others.

Consider for example estimating the mean and variance of a normal-like pdf but whose probability is 0 outside the range $x=[-1,1]$. Thus random sampling the distribution for ALL x values, including those outside $x=[-1,1]$ would yield as accurate an approximation as random sampling within $x=[-1,1]$ only. Thus it would be better to sample only x values within $x=[-1,1]$.

6.3 Bootstrap Sampling

The word bootstrap literally means "to undertake something with minimal outside help". Hence bootstrap sampling is sampling from a sample itself i.e. sampling within a sample. This is used as frequently researchers only have 1 sample available. Consequently, we are assuming the sample is representative of the whole population.

6.4 Internal Sampling

In internal sampling, sampling is performed *during* the execution of an optimisation algorithm itself that solves the SP [11]. Normally samples are obtained prior to executing any optimisation algorithm for SP. This method provides computational advantages compared to other sampling methods. Examples of this method are [11]: stochastic decomposition, where sampling is done within Bender's decomposition, and stochastic quasigradient methods.

6.5 Conditional Sampling

This method obtains its name from its relation to conditional probability. In random sampling, current sample choices are independent of previous choices whereas in conditional sampling the next sample choice will be statistically dependent (i.e. conditional) on past choice(s). This approach is useful when the underlying process possesses some statistical dependency on past values and hence the process is not entirely random.

6.6 Stratified Sampling

Stratified sampling can be thought of as a variation on importance sampling. For a given statistical population, the population may be subdivided into "groupings" for which little variation in sample values occur within the group itself. However, if one were to sample from the entire population, large variation in sample values would occur. Consequently, to improve sampling accuracy, we divide the population into such groups (called strata) and sample each group independently. Note that the strata groups should be mutually exclusive (1 element should not belong to another group) and the total set of groups should cover the entire population.

By random sampling within each strata we reduce the overall population sampling error. Each strata is given a weighting, which can be assigned in 2 ways. One is to make the weighting proportional to the percentage of elements of the whole population the strata contains. A second way is to weight proportionately to the standard deviation of the strata.

6.7 Markov Chain Monte Carlo Sampling (MCMC or Monte Carlo based on Markov Chain)

In Monte Carlo sampling we are already given the pdf from which to sample. However in some circumstances we still may wish to perform Monte Carlo sampling but we may need to construct the pdf itself, which can be difficult if the distribution is not simple. We call this difficult desired distribution $\pi(x)$.

Surprisingly, we can construct difficult distributions $\pi(x)$ by using markov chains whose stationary distribution (its long term equilibrium distribution), denoted by $\phi(x)$, approximately matches $\pi(x)$ i.e. $\pi(x) \approx \phi(x)$. We also want transitions in the markov chain to approximately correspond to simple random perturbations in $\pi(x)$.

We can randomly sample $\pi(x)$ now by sampling from $\phi(x)$ in the following way. Whereas in Monte Carlo sampling random numbers are drawn to sample the distribution, in MCMC the numbers drawn are not random but follow a markov chain. Let us say the random numbers drawn in Monte Carlo are X_1, X_2, X_3 etc... then all numbers are independent of past drawings and equiprobable. In MCMC, a number is drawn and then the next number will be conditional on the last number i.e. $p(X_2)=p(X_2 | X_1)$. From this conditional distribution a number is drawn at random to give X_2 .

A caveat to this procedure is that markov chain transition probabilities are dependent on the initial starting point in the chain. This dependency can be removed if the chain is run long enough so that it "forgets" its initial starting point. This is known as "burn in" as samples obtained during burn-in are not used due to their dependency on the starting point.

The procedure for MCMC can be described as follows:

1. find a markov chain so that $\pi(x) \approx \phi(x)$
2. run the markov chain sufficiently long enough to forget its initial starting point (burn in)
3. draw the next value to sample by randomly drawing from the conditional distribution i.e. $p(X_2) = p(X_2 | X_1)$.

7 Scenario Generation by Simulation

In this section we survey the scenario generation methods by simulation approaches. Simulation was invented out of the need to calculate solutions to analytically intractable equations and where other numerical techniques would be computationally infeasible.

Simulation is where some underlying mathematical process (e.g. Brownian Motion) is "simulated"; random numbers are inputted into the random component of an equation and the result is recorded. For stochastic programming, scenario generation by simulation results in a "path by path" production of the scenario tree where each path is considered equiprobable. To reduce the number of paths sometimes paths are bounded together by some method (see [8] for instance).

We now describe the simulations used for scenario generation.

7.1 Stochastic Process Simulation

In Finance many random variables follow a stochastic process and hence are simulated to provide scenarios. Popular stochastic processes include geometric Brownian motion and its variants. Geometric Brownian motion is defined by:

$$dS(t) = \mu S(t)dt + \sigma S(t)dB(t)$$

$S(t)$ is asset price, μ, σ are the drift and volatility respectively. The term $dB(t)$ is a Brownian motion i.e. $B(t_2) - B(t_1) \sim N(0, |t_2 - t_1|)$. Thus we can simulate stochastic processes over a time interval by randomly inputting numbers to the Brownian motion and calculating $S(t)$.

7.2 Error Correction Model (ECM)

The ECM originates from Econometrics, from modelling Economic behaviour.

$$\Delta Y_t = \theta \Delta X_t + \delta(Y_{t-1}^* - Y_{t-1}) + u_t$$

The (dependent) variable we wish to model Y_t is dependent on X_t where θ is a constant. Y_{t-1}^* represents the predicted Y_{t-1} value and Y_{t-1} represents the actually *observed* value. Thus the term $\delta(Y_{t-1}^* - Y_{t-1})$ represents the "error correction" and δ is a constant by which error is correction is increased by on the next prediction of Y .

Simulation of ECM is achieved by firstly fitting the equation to a given set of data where the term u_t is an error term. Then we forecast one period into the future by taking random drawings from the distribution of u_t . Each drawing provides a "data path" for the series and for each data path we re-calculate all the previously fitted terms. This process is then repeated so that one essentially obtains a scenario tree. If too many branches occur then we can reduce the scenario tree size by applying a grouping method of some sort.

7.3 Vector Auto-regressive (VAR)

A variable Y_t is a VAR model of order p , denoted VAR(p), if:

$$Y_t = C_0 + C_1Y_{t-1} + \dots + C_pY_{t-p} + u_t$$

where C_n are constants for all n and u_t is an error term. VAR frequently models stochastic variables in Financial and Economic literature; Kouwenberg [13] uses VAR for scenario generation. Variants of VAR exist e.g. BVAR (Bayesian VAR).

Simulation of VAR is achieved by firstly fitting it to historical data so that values for C_n are found. From this fitted model we then forecast one period into the future by taking random drawings from the distribution of u_t . Each drawing provides a "data path" for the series and for each data path we re-calculate C_n (for all n). This process is then repeated so that one essentially obtains a scenario tree. If too many branches occur then we can reduce the scenario tree size by applying a grouping method of some sort.

8 Other Scenario Generation Methods

The importance of SG has stimulated approaches from a variety of fields. For example, Boender in [3] uses scenarios to incorporate risk drivers rather than incorporate probability distributions. In fact Boender uses the scenario generation process itself as a method of iterative learning the best asset management strategy.

We now survey some new and uncommon scenario generation methods.

8.1 Artificial Neural Networks

An Artificial Neural Network (ANN) is an information processing method, copied from the method by which animal brains process information. They differ from other computer information processing methods in terms of their structure because neural network structure is taken from the brain. The brain is principally composed of a large number of neurons; neurons are highly interconnected elements. Neurons receive a number of input signals (either from other neurons or an external source e.g. eye) and produce an output signal outputted to other neurons. The complex network of interconnecting neurons enables neural network structures to tackle complex inputs and subsequently provide a final output signal.

The advantage of neural networks is that they can engage in adaptive learning in that they can learn how to do tasks based on the data given. They can also solve complex problems that many other methods cannot.

8.2 Clustering

In clustering firstly some underlying process is simulated (e.g. VAR) so that a set of data "paths" are created. Then a set of paths are then clustered or grouped together by some method. These clusters then form the scenarios. Gulpinar in [8] provides a detailed clustering algorithm. Gulpinar's clustering algorithm [8] is:

1. choose a root node scenario value, typically today's stock price. Then decide the number of simulated values you require starting from this root node; denote this number by $N-1$. Therefore there will be $N-1$ simulated values + 1 root node value, giving a total of N values.
2. simulate, using an underlying process, by 1 time period those $N-1$ values so that we end up with $N-1$ simulated values and the root node's value. Gulpinar refers to these N obtained values as scenarios even though they are *not at all* the final scenarios of the tree.
3. randomly choose a number of scenarios to represent the value to which the remaining scenarios are to be clustered around. Then group each scenario; if the resulting clustering is unacceptable go back to the previous step.
4. for each cluster find the scenario that is closest to the centre. This scenario's value becomes the centroid of the cluster.

5. the cluster's centroid becomes the actual ST's scenario value, with its probability proportional to the number of scenarios within the cluster.

Gulpinar states that the clustering can be performed sequentially or in parallel [8].

8.3 Scenario Reduction

In some problems we are already given a scenario tree however, the tree size is computationally too large and so must be reduced. One method of scenario reduction is to employ sampling methods, as described previously for a pdf but applied to a scenario tree itself, rather than a pdf. An example is Zenios's scenario reduction method in [17], who reduces the scenario tree obtained from an interest rate "binomial tree" model.

Similar to random sampling, scenario reduction by random sampling can be improved by introducing some order into the sampling process. For example, stratified sampling can be applied by grouping particular scenario tree paths in stratas (see [12] for instance)

8.4 Hybrid Methods

Hybrid methods are simply a combination of scenario generation methods. For example, one uses the clustering method to give scenario values but then one determines the scenarios' probabilities by optimising moment matching properties to the actual random variables's pdf. Gulpinar [8] provides a detailed explanation.

9 Conclusion

This whitepaper has given a survey of stochastic programming scenario generation methods as well as overviewed basic stochastic programming theory. We have explained how scenario generation forms a central role in any stochastic programming problem as it is responsible for uncertainty modelling.

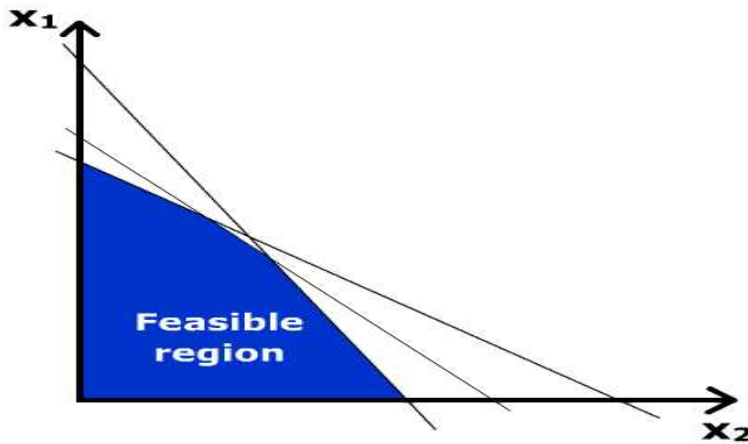
In the whitepaper we surveyed how statistical properties can be incorporated and exploited to produce scenarios trees consistent with such properties. The whitepaper also discussed how various sampling approaches can provide scenario generation methods, for example stratified sampling.

The paper reviews how simulation techniques have been applied to various models and then applied to scenario generation. We also cover some more uncommon and new scenario generation techniques like hybrid methods. In conclusion the importance of scenario generation to stochastic programming can be considered paramount and offers a rich area for potential research and investigation.

10 Appendix 1: Introduction to Stochastic Programming

10.1 Importance of Stochastic Programming to Optimisation

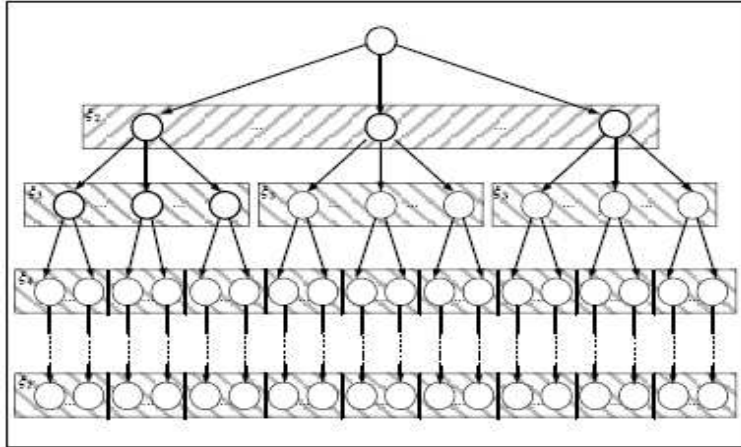
Linear programming (from hereon LP) optimisation has made significant contributions to a variety of real world applications from agricultural planning to financial portfolio optimisation.



However linear programs, being suited to largely deterministic problems, have been unable to provide adequate solutions to problems with substantial degrees of uncertainty. A particularly important example being optimising portfolio returns where asset prices are renowned for unpredictability.

An early approach to accommodate uncertainty in LP was sensitivity analysis; examining how solutions varied in response to input variable changes. However, as demonstrated by Hight and Wallace [9] such sensitivity analyses imposes a number of limitations and could in fact be misleading. Sensitivity analysis was not a suitable method for incorporating uncertainty in models.

SP emerged out of the need to adequately incorporate uncertainty that cannot be reformulated in a LP. The SP with recourse makes use of event trees, which define states of the world. This structure primarily facilitates modelling of randomness by incorporating probabilistic decisions, scenarios and decision nodes with (branch) probabilities associated with them.



As Birge and Louveaux [2] point out, scenarios are particularly useful *when the optimal solution varies considerably with changes in value of the stochastic variables*. If this were not the case, the solution obtained for 1 stochastic variable realisation ought to be a good solution for most realisations.

SP has been proven to provide more practical and optimal solutions to those obtained through LP. A well-known example is Mulvey et al. Russell-Yasuda Kasai Asset Liability Model where SP could take into account the stochastic nature of the model.

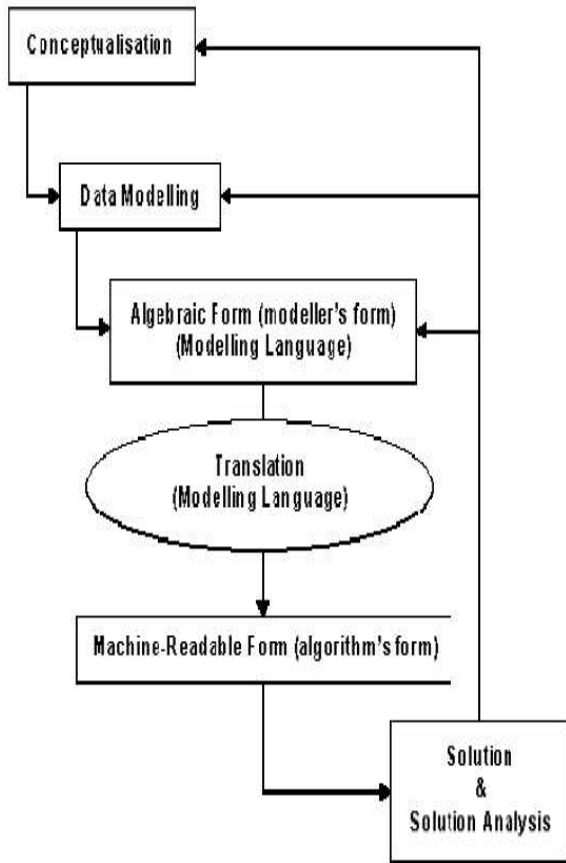
Note that scenarios must obey the **non-anticipativity constraint**: we do not know the future and so decisions can only be based on past information.

10.1.1 Link between Stochastic and Linear Programming: the Expected Value Core Model

When one numerically computes solutions to a SP, we must assign or *realise* actual values and probabilities to the random variables in scenarios. Consequently for a particular realisation of random variables, in terms of calculation the SP model resembles an LP calculation. This underlying deterministic LP model is called the "core model" and captures the essential optimisation problem.

10.2 Conceptualisation of the Modelling Process

Translating a real world decision problem into an equivalent stochastic programming problem poses numerous challenges e.g. what probabilities accurately reflect which scenarios, how many decision stages should exist? To appreciate such difficulties it is worthwhile to understand the modelling process as detailed by Dinguozzi-Ballesteros (see [16]). The modelling process is depicted diagrammatically below [16]:



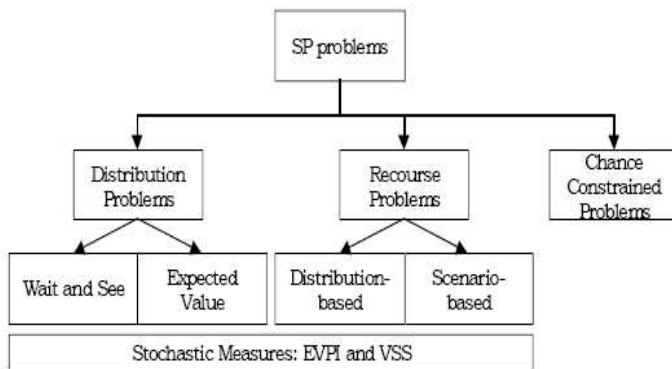
- Conceptualisation stage: this involves intellectually understanding and formulating the essential real decision problem. For SP this means devising a mathematical model, capable of capturing randomness.
- Data Modelling stage: involves finding a method of representing data that realistically reflects its randomness. For SP this typically involves identifying the random variables and using some scenario generation method to specify their values.
- Algebraic formulation stage (normally implemented in an AML (algebraic modelling language)): the mathematical model is implemented in some AML for computer processing
- Solution and Solution Analysis: this is essentially evaluating the results from one's model/program, then appraising its validity and consequently improving upon it.

11 Appendix 2:Classes of Stochastic Programs

In this section we briefly overview the fundamental aspects of Stochastic Programming but this should be read with additional texts for a comprehensive understanding. For a detailed explanation the reader is referred to Birge and Louveaux [2]. SP problems can be broadly grouped into 3 classes:

- Chance Constraint problems
- Recourse problems
- Distribution Problems

Each group can be subdivided further, as depicted below:



Note that as pointed out by Gassmann and Ireland [7], these broad classifications are *not* mutually exclusive.

11.1 Distribution Problems

Distribution in this definition refers to the distribution of SP the solutions or objective function outcomes as inputs are varied. Distribution here does not refer to the probability distribution function (from hereon pdf) from which scenarios can be extracted. The SP distribution problem can be considered the equivalent of sensitivity analysis in LP problems, where the inputs are varied to determine the change (or sensitivity) in the output. Hence the distribution problem enables one to learn about the model's robustness.

To calculate the objective function's distribution we can adopt 2 approaches: expected value or wait-and-see.

11.1.1 Expected Value Approach

In the expected value approach, all random variables are replaced with their expected value, thus the SP "reduces" to a LP. Sometimes this expected value substitution is done anyway to gain some intuitive insight into the model. We can reconsider the optimisation problem now as:

$$\begin{aligned} E[\xi(\omega)] &= \sum_{\omega \in \Omega} p(\omega)\xi(\omega) \\ &= \bar{\xi} \end{aligned}$$

If for $\bar{\xi}$ we have $(\bar{A}, \bar{b}, \bar{c})$ then we can rewrite the optimisation problem as:

$$\begin{aligned} Z_{EV} &= \min \bar{c}x \\ \text{where } x &\in \bar{F} = \{x | \bar{A}x = \bar{b}, x \geq 0\} \end{aligned}$$

For the above problem of $Z_{EV} = \min \bar{c}x$, consider now if we had an optimal solution for it and denote it by x_{EV}^* (we define x as optimal for the problem $Z = \min cx$ (with all the constraints) if the following is true: $cx \geq cx_{EV}^*$ for any feasible x). Using x_{EV}^* we can calculate the objective function's set of values (i.e. Z) for different realisations of $\xi(\omega)$. Consequently we obtain a distribution of objective function values, from which we can calculate its expectation, or in other words the expectation of the expected value solution, denoted Z_{EEV} :

$$Z_{EEV} = E[c(\omega)x_{EV}^*]$$

Note if for some $\omega \in \Omega$, x is not feasible i.e. $x_{EV}^* \notin F^\omega$ then we set:

$$Z_{EEV} \rightarrow +\infty$$

11.1.2 Wait and See Approach

Under Wait and See we assume we have fully complete and perfect information about the random variables (no uncertainty exists) and then the decision is optimised. So, we calculate the objective function's value for each different realisation of ω , giving a distribution of objective function values. This approach is used to "build up a picture" of the objective function's distribution as the realisations change. We restate the optimisation problem now as:

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

Essentially we are evaluating a set of LP calculations, each associated with a particular realisation of random variables (i.e. a scenario). The expected value of the wait and see solution, Z_{WS} , is defined:

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

Wait and See can be interpreted as the SP solution's monetary value when we have absolutely fully perfect information about the future.

11.2 Recourse Problems

In recourse problems we take at least 2 decisions, where one must be taken before the random event(s) is realised and one must be taken after it. We say this decision is under recourse because we can correct the decision taken before uncertainty and after uncertainty has passed. Birge ([2],p52) defines recourse as: *"recourse programs are those in which some decisions or recourse actions can be taken after uncertainty is disclosed"*.

An example of recourse would be the following. Imagine we are required to produce a product X to meet consumer demand, which is uncertain, and production of X is more costly after consumer demand uncertainty is realised. This is a realistic example as production costs could increase due to supplies becoming more scarce with increased demand (oil prices behave like this). We also we would like to maximise the amount produced without over-producing as any unsold products would be regarded as a loss.

We decide today, prior to any uncertainty realisation of consumer demand, to produce an amount M of product X. We then observe the uncertainty being realised (consumer demand) and consequently produce an additional amount N of product X (the recourse action). So the total amount produced is M+N.

It is standard to call the first stage decisions as those decisions before uncertainty is realised, typically represented by vector x. Second stage decisions are decisions after uncertainty is realised, typically represented by y(ω), y or y(x, ω) if y decision has dependence on 1st stage decision x. As stated by Birge ([2],p52) the sequence of events are:

$$x \rightarrow \xi(\omega) \rightarrow y(x, \omega)$$

In the above example the first stage decision x was to produce N amounts of X, the second stage decision y was to produce M amounts of X. Notice that y was dependent on our first stage decision x and the uncertain event ω , hence $y=y(x,\omega)$.

Mathematically, the 2 stage SP with recourse is:

$$\min Z = cx + E[Q(x, \omega)]$$

$$\text{subject to } Ax=b$$

$$x \geq 0$$

and:

$$Q(x, \omega) = f(\omega)y(\omega)$$

$$\text{subject to } D(\omega)y(\omega)=d(\omega)+B(\omega)$$

$$y(\omega) \geq 0$$

$$\omega \in \Omega$$

D(ω):recourse or technology matrix

f(ω):objective function coefficient

B(ω):interstage linking matrix

$d(\omega)$: the right hand side

The term cx appears in the first stage, therefore before uncertainty occurs and having to satisfy the first stage constraint $Ax=b$. $Q(x,\omega)$ is called the recourse function. As we are viewing this equation before uncertainty has passed, the second stage expressions use ω to represent uncertainty. We use the expectation, $E[Q(x,\omega)]$, to evaluate the best decision to make in the face of uncertainty.

In the recourse function definition $Q(x,\omega)$ as above, notice its similarity to the first stage problem definition with x "replaced" by $y(\omega)$. The second stage problem can be solved as a LP once ω is realised.

In the equations above, A and b are known with certainty but D, f, B and d are stochastic. For notational convenience, we could split A, b or c into 2 if they differed before and after uncertainty is realised. For example, we could split c into c^1 and c^2 , where c^1 is in the first stage but c^2 would be uncertain i.e. $c^2 = c^2(\omega)$.

In recourse modelling we can determine x 's optimal value as the random variable's realisations are varied. In fact, we can rewrite the SP recourse problem as its deterministic equivalent:

$$E[Q(x, \omega)] = \sum_{i=1}^N p^i Q(x^i, \omega^i)$$

where i =number of scenarios (N in total) and p^i is the probability associated with event ω^i .

Finally it is worth noting that as expectations can be represented in integral form (provided they can be integrated) we can thus rewrite $E[(Q(x,\omega))]$:

$$E[(Q(x, \omega))] = \int (Q(x, \omega) P(d\omega))$$

11.2.1 Multistage Recourse

The previous discussion involved only 1 stage of recourse i.e. after some uncertainty was realised 1 corrective action was taken. In multistage recourse we have more than 1 stage and at every stage we have the possibility to engage in recourse. Thus decisions are based on taking into account previous decisions and previous (partial) uncertainties being resolved. Dupacova [5] specifies the order of events for a T -stage stochastic process where $\omega = (\omega_1, \omega_2, \dots, \omega_T)$ and the set of decisions at each stage are $x=(x_1, x_2, \dots, x_T)$. Then the order of events is:

$$(x_1, \omega_1, x_2(x_1, \omega_1), \omega_2, \dots, x_T(x_1, \omega_1, x_2, \dots, \omega_{T-1}))$$

The multistage recourse problem is expressed in terms of nested expectations and so typically are more complex to solve. Note that the decisions are made with knowledge of past information.

$$Z = \min_{x_1} \{c_1 x_1 + E_{\xi_2} [\min_{x_2} c_2 x_2 + E_{\xi_3|\xi_2} [\min_{x_3} c_3 x_3 + \dots E_{\xi_T|\xi_{T-1}|\dots|\xi_2} \min_{x_T} c_T x_T]]]\}$$

subject to:

$$\begin{aligned} A_{11}x_1 &= b_1 \\ A_{21}x_1 + A_{22}x_2 &= b_2 \\ A_{31}x_1 + A_{32}x_2 + A_{33}x_3 &= b_3 \\ \vdots &\quad \ddots \quad \vdots \\ A_{T1}x_1 + A_{T2}x_2 + A_{T3}x_3 + \dots &= b_T \end{aligned}$$

Note that multiperiod does not necessarily mean multistage. A new stage occurs when some realisation of a random variable is observed, whereas a new time period does not necessarily imply any partial realisation.

11.2.2 Recourse Problem Categorisation

Following Gassman and Ireland's definitions [7], recourse problems are split into 2 categories, based on how the scenarios are obtained.

If the entire scenario is explicitly defined, including random variable realisations and dependent data, the recourse problem is known as scenario based. If the scenarios are extracted from an a priori pdf by some method (e.g. sampling) then the recourse problem is known as distribution based.

Gassmann and Ireland [7] describe 6 categories of "distribution-based" problems for recourse problems, where scenarios are extracted from a pdf. Essentially, this is a classification of scenario generation problems based upon the pdf's statistical properties. We now explain them:

- 1) Type 1 (independent random variables): random variables are independent of any point in time.
- 2) Type 2 (period-to-period independence): problems for which distributions are correlated within a short time periods but are independent of distant past realizations. An example may be traffic congestion; random variables may be dependent on realisations from previous hours but not from previous years
- 3) Type 3 (random-walk problems): the random variable follows a "random walk" in time. Most financial models assume random variables follow a random walk of some form e.g. stock prices
- 4) Type 4 (dependence on past data): problems for which the random variables depend on past events -this is conditional probability. An example would be the probability of picking a red ball from a bag with red and black balls but without replacing balls after each pick.
- 5) Type 5: random problem dimension problem. Problems which allow the number of decision variables and/or constraints to be dependent on prior realizations. This is also observed in markov processes where there exists a self-terminating state known as the self-absorbing state. If one enters the self-absorbing state one cannot escape it and so the process ends. The equivalent of a self-absorbing state can occur in SP too. An example is modelling business performance by scenarios, where the scenario

of bankruptcy would terminate the entire SP.

6) Type 6 (dependence on past decisions): problems for which random variable realisations depend on past decisions (rather than just past events). An example of this would be when a trader transacts a high volume trade and the volume of the trade causes the asset's price to move (known as the liquidity effect)

11.3 Chance Constraint Problems

Whereas in linear programs the constraints never alter and must always be satisfied e.g. $x \geq 90$, in chance constraint the constraints only need to be satisfied with probability e.g. $p(x \geq 90)=0.6$. Mathematically we express chance constraints as:

$$\min c(\omega)x$$

$$\begin{aligned} &\text{subject to } p(A_i x \geq h_i) \geq \beta_i \\ &i=1, \dots, I \text{ (I=number of constraints)} \\ &\text{where } 0 \leq \beta_i \leq 1 \\ &\quad \xi_i = (A_i, h_i) \end{aligned}$$

In the above expressions β_i is the probability for the i^{th} constraint and ξ_i is a random vector. If A_i is:

a row vector then the i^{th} constraint is called an individual chance constraint
 a $r \times c$ matrix then the i^{th} constraint is called a joint chance constraint

11.4 Stochastic Programming Stochastic Measures

Given an SP model, it is advantageous to have some measure of its performance. Two such measures are EVPI and VSS.

11.4.1 EVPI (Expected Value of Perfect Information)

To calculate EVPI we need to calculate the here and now value, Z_{HN} . In here-and-now, we want to calculate the *minimum expected* cost of the optimisation problem without allowing any uncertainty to be realised. All decisions must be made now, prior to any uncertainty realisations; one can think of Z_{HN} as the "opposite" of Z_{WS} . We express Z_{HN} mathematically as:

$$Z_{HN} = \min E[c(\omega)x]$$

$$\begin{aligned} &\text{where } F = \bigcap_{\omega \in \Omega} F^\omega \\ &\text{and } x \in F \end{aligned}$$

If the objective function Z represents cost, then one can think of Z_{HN} as the minimum expected cost of the stochastic optimisation problem.

We define EVPI by:

$$EVPI = Z_{HN} - Z_{WS}$$

EVPI represents the monetary cost one would pay to have perfectly complete information (no uncertainty). Hence a small EVPI implies little cost saving is obtained by better forecasting, whereas a large EVPI implies forecasting offers significant savings.

In Birge and Louveaux [2] it is shown EVPI is bounded:

$$0 \leq EVPI \leq Z_{HN} - Z_{EV} \leq Z_{EEV} - Z_{EV}$$

It also worth noting the following relation:

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

11.4.2 VSS (Value of the Stochastic Solution)

$$VSS = Z_{EEV} - Z_{HN}$$

VSS is a measure of a model's usefulness in terms of how much monetarily is saved by taking into account the stochastic nature of variables. Alternatively, VSS gives the value of including randomness in our model.

In Birge and Louveaux [2] it is shown VSS is bounded by:

$$0 \leq VSS \leq Z_{EEV} - Z_{EV}$$

References

- [1] Carol Alexander. *Market Models*. 2001.
- [2] J.R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. 1997.
- [3] G.C.E. Boender. A hybrid simulation/optimisation scenario model for asset/liability management. *European Journal of Operational Research*, 1997.
- [4] Nico Di Domenica. *Stochastic Programming and Scenario Generation: Decision Modelling Simulation and Information Systems Perspective*. 2005.
- [5] Jitka Dupacova. Applications of stochastic programming: Achievements and questions. *European Journal of Operations Research*, 2002.
- [6] R. Engle and C.W.J. Granger. Cointegration and error correction: Representation, estimation and testing. *Econometrica*, 1987.
- [7] H.I. Gassmann and A.M. Ireland. On the formulation of stochastic linear programs using algebraic modelling languages. *Annals of Operations Research*, 1996.
- [8] Rustem B. Gulpinar, N. and R. Settergren. Simulation and optimization approaches to scenario tree generation. *Journal of Economic Dynamics and Control*, 28, 2004.
- [9] J. Higle and S. Wallace. Managing risk in the new power business -a sequel. *Computer Applications in Power*, 2002.
- [10] K. Hoeyland and S.W. Wallace. Generating Scenario Trees for Multistage Decision Problems. *Management Science*, 47(2):295–307, 2001.
- [11] M. Kaut and S.W. Wallace. Evaluation of scenario generation methods for stochastic programming. ???, 2003.
- [12] J. Kim. Event tree sampling. *Computers & Operations Research*, 2006.
- [13] R. Kouwenberg. Scenario generation and stochastic programming for asset liability management. *European Journal of Operational Research* 134, 2001.
- [14] Gautum Mitra. Models for decision making: an overview of problems tools and major issues. pages 17–53, 1988.
- [15] Gould G. Mulvey, J. and C. Morgan. An asset and liability management system for towers perrin-tillinghast. *INTERFACES*, 30, January-February 2000, pp.96-114, 2000.
- [16] Mitra G. Poojari C. Valente, P. and T. Kyriakis. Software tools for stochastic programming: A stochastic programming integrated environment (spine). 2001.
- [17] S. Zenios. Asset/liability management under uncertainty for fixed-income securities. *Annals of Operations Research*, 1995.