Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Journal of Economic Dynamics & Control 28 (2004) 1291–1315
JOURNAL OF Economic Dynamics & Control
www.elsevier.com/locate/econbase

Simulation and optimization approaches to scenario tree generation

Nalan Gülpınar, Berç Rustem, Reuben Settergren

Department of Computing, Imperial College of Science, Technology, and Medicine, 180 Queen's Gate, London SW7 2BZ, UK

Abstract

In this paper, three approaches are presented for generating scenario trees for financial portfolio problems. These are based on simulation, optimization and hybrid simulation/optimization. In the simulation approach, the price scenarios at each time period are generated as the centroids of random scenario simulations generated sequentially or in parallel. The optimization method generates a number of discrete outcomes which satisfy specified statistical properties by solving either a sequence of non-linear optimization models (one at each node of the scenario tree) or one large optimization problem. In the hybrid approach, the optimization problem is reduced in size by fixing price variables to values obtained by simulation. These procedures are backtested using historical data and computational results are presented.

Keywords: Stochastic programming; Scenario generation; Computational finance; Simulation; Non-linear optimization

1. Introduction

Stochastic programming requires a coherent representation of uncertainty. This is expressed in terms of a multivariate continuous distribution. Hence, a decision model is generated with internal sampling or a discrete approximation of the underlying continuous distribution. A method to obtain the discrete outcomes for the random variables is referred to as scenario tree generation. In multistage models, at each time period, new scenarios branch from old, creating a scenario tree.

The random variables are the uncertain return values of each asset on an investment. The discretization of the random values and the probability space leads to a framework

E-mail addresses: ng2@doc.ic.ac.uk (N. Gülpınar), br@doc.ic.ac.uk (B. Rustem), reuben@doc.ic.ac.uk (R. Settergren).

0165-1889/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0165-1889(03)00113-1

in which a random variable takes finitely many values. Thus, the factors driving the risky events are approximated by a discrete set of scenarios, or sequence of events. Given the event history up to a particular time, the uncertainty in the next period is characterized by finitely many possible outcomes for the next observation. This branching process is represented using a scenario tree.

The root node in the scenario tree represents the 'today' and is immediately observable from deterministic data. The nodes further down represent the events of the world which are conditional at later stages. The arcs linking the nodes represent various realizations of the uncertain variables. An ideal situation is that a generated set of scenarios represents the whole universe of possible outcomes of the random variable. Therefore, scenarios should include both optimistic and pessimistic projections.

Currently, there exist techniques to generate scenario trees for financial applications such as vector auto-regressive processes (Boender, 1997), approaches based on principal component analysis, (Mulvey and Vladimirou, 1992) and stochastic simulation of economic variables and asset returns (Dempster and Thorlacius, 1998; Carino et al., 1994; Mulvey, 1996). Stochastic forecasting methods are used for global financial planning by considering problem specific economic factors. Pflug (2001) constructs a scenario tree on the basis of a simulation model of the underlying financial process by using a stochastic approximation technique.

Hoyland and Wallace (2001) develop a scenario generation technique for multivariate scenario trees, based on optimization. This technique is also used for generating scenario trees for hydro inflow (Vitoriano et al., 2001), and dynamic portfolio insurance (Kouwenberg and Vorst, 1998). In these applications, different central moments are matched by solving a non-linear optimization problem at each node of the scenario tree. In Hoyland and Wallace (2001), a procedure to generate a scenario tree by solving a large non-linear optimization problem was also suggested, but was not fully exploited. Kouwenberg and Vorst (1998) compare random sampling, adjusted random sampling and optimization-based (fitting the mean and covariances) approaches to generate the scenario tree.

In this paper, we consider randomly generated scenarios as well as several variants of the moment matching procedure and report on the implementation results of these methods. Specifically, we investigate the basic overall optimization procedure and its relaxed variants such as sequential optimization and a hybrid simulation—optimization approach. Extensive out-of-sample backtesting on actual data of all these approaches are used to evaluate their relative merits. We also consider simulation and clustering-based approaches as possible alternatives to the optimization method. In this approach, the price scenarios at each node of the scenario tree are obtained as the centroids of randomly generated scenarios by clustering (e.g. Bratley and Fox, 1998; Christofides et al., 1999; Hansen and Jaumard, 1997). Parallel and sequential methods for simulating scenarios are investigated.

1.1. Multistage stochastic mean-variance model

Generating the scenario tree is important for the performance of multistage stochastic programming (Kouwenberg, 2001; Klaassen, 1997; Nielsen and Zenios, 1996).

Multistage stochastic programming is used to model the problem of financial portfolio management with transaction costs, given stochastic data provided in the form of a scenario tree. The mean or variance of total wealth at the end of planning horizon can be optimized in view of transaction costs. Multiperiod discrete-time optimal portfolio strategies are determined over a given finite investment horizon. In this section, we present multistage mean-variance optimization model; for more details, the reader is referred to Gülpınar et al. (2002, 2003).

We assume a portfolio of n risky assets and consider an investment horizon T. The portfolio is restructured over a period in terms of both return and risk. After the initial investment ($t=0$), the portfolio may be restructured at discrete times $t=1,\ldots,T-1$, and redeemed at the end of the period ($t=T$). Let $\boldsymbol{\rho}^t \equiv \{\boldsymbol{\rho}_1,\ldots,\boldsymbol{\rho}_t\}$ be stochastic events at $t=1,\ldots,T$. Following Dupacova et al. (2000) and others, the decision process is non-anticipative (i.e. decision at a given stage does not depend on the future realization of the random events). Past information and the probabilistic specifications $(\Omega,\mathscr{F},\mathscr{P})$ of the future uncertainty are used to determine the decision. In order to clarify the latter, let $\mathscr{F}_t \subseteq \mathscr{F}$ be the σ-field generated by the observation $\boldsymbol{\rho}^t \equiv \{\boldsymbol{\rho}_1,\ldots,\boldsymbol{\rho}_t\}$. The dependence of the decision at t on $\boldsymbol{\rho}^{t-1}$ means that the decision at t is $\mathscr{F}_{t-1}$ adapted (i.e. it is measurable with respect to $\mathscr{F}_{t-1}$). The constraints on a decision at each stage involve past observations and decisions.

A scenario is a possible realization of the stochastic variables $\{\boldsymbol{\rho}_1,\ldots,\boldsymbol{\rho}_T\}$. Hence, the set of scenarios corresponds to the set of leaves of the scenario tree, $\mathscr{N}_T$, and nodes of the tree at level $t \geqslant 1$ (the set $\mathscr{N}_t$) correspond to possible realisations of $\boldsymbol{\rho}^t$.

The set of all interior nodes of the scenario tree is $\mathscr{N}_I$. A node of the tree (or **event**) is denoted by $\mathbf{e}=(s,t)$, where s is a scenario (path from root to leaf), and time period t specifies a particular node on that path. The ancestor (parent) of event $\mathbf{e}=(s,t)$ is denoted $a(\mathbf{e})=(s,t-1)$, and the branching probability $p_{\mathbf{e}}$ is the conditional probability of event $\mathbf{e}$, given its parent event $a(\mathbf{e})$. The path to event $\mathbf{e}$ is a partial scenario with probability $P_{\mathbf{e}}=\prod p_{\mathbf{e}}$ along that path; since probabilities $p_{\mathbf{e}}$ must sum to one at each individual branching, probabilities $P_{\mathbf{e}}$ will sum up to one across each layer of tree-nodes $\mathscr{N}_t$; $t=0,1,\ldots,T$.

At each node of time period t, decisions for weights of each asset, transactions for buying and selling $\mathbf{w}_t$, $\mathbf{b}_t$, $\mathbf{s}_t$, respectively, must be determined. Due to the recourse nature of the multistage problem, decision variables $\mathbf{w}_t$, $\mathbf{b}_t$ and $\mathbf{s}_t$ are influenced by previous stochastic events $\boldsymbol{\rho}^t$, and hence $\mathbf{w}_t=\mathbf{w}_t(\boldsymbol{\rho}^t)$, $\mathbf{b}_t=\mathbf{b}_t(\boldsymbol{\rho}^t)$, and $\mathbf{s}_t=\mathbf{s}_t(\boldsymbol{\rho}^t)$. However, for simplicity, we shall use the terms $\mathbf{w}_t$, $\mathbf{b}_t$ and $\mathbf{s}_t$, and assume their implicit dependence on $\boldsymbol{\rho}^t$. Notice that $\boldsymbol{\rho}_t$ can take only finitely many values.

Let the expected returns $\hat{\mathbf{r}}_{\mathbf{e}}$, and covariance matrix Λ[1] be given for $\mathbf{e}\in\mathscr{N}$. The transaction costs for buying and selling assets are denoted by $\mathbf{c}_b$ and $\mathbf{c}_s$ and the vector $\mathbf{p}$ is the current portfolio position. Let α_t denote the discount factor for the risk. α_t is allowed to decrease with t. Hence, the multistage portfolio reallocation problem can

[1] In the original formulation discussed in Gülpınar et al. (2003), we consider different Λ for each $\mathbf{e}$. In the present discussion, this is simplified to a common Λ.

be expressed as follows:

$$\min_{\mathbf{w},\mathbf{b},\mathbf{s}} \quad \sum_{t=1}^{T} \alpha_t \sum_{\mathbf{e}\in\mathcal{N}_t} P_{\mathbf{e}}[(\mathbf{w}_{a(\mathbf{e})} - \bar{\mathbf{w}}_{a(\mathbf{e})})'(\Lambda + \hat{\mathbf{r}}_{\mathbf{e}}\hat{\mathbf{r}}_{\mathbf{e}}')(\mathbf{w}_{a(\mathbf{e})} - \bar{\mathbf{w}}_{a(\mathbf{e})})]$$

$$\text{s.t.} \quad \mathbf{p} + (1 - \mathbf{c}_b)\mathbf{b}_0 - (1 + \mathbf{c}_s)\mathbf{s}_0 = \mathbf{w}_0,$$

$$\mathbf{1}'\mathbf{b}_0 - \mathbf{1}'\mathbf{s}_0 = 1 - \mathbf{1}'\mathbf{p},$$

$$\hat{\mathbf{r}}_{\mathbf{e}} \circ \mathbf{w}_{a(\mathbf{e})} + (1 - \mathbf{c}_b) \circ \mathbf{b}_{\mathbf{e}} - (1 + \mathbf{c}_s) \circ \mathbf{s}_{\mathbf{e}} = \mathbf{w}_{\mathbf{e}}, \quad \mathbf{e} \in \mathcal{N}_I,$$

$$\mathbf{1}'\mathbf{b}_{\mathbf{e}} - \mathbf{1}'\mathbf{s}_{\mathbf{e}} = 0, \quad \mathbf{e} \in \mathcal{N}_I,$$

$$\sum_{\mathbf{e}\in\mathcal{N}_T} P_{\mathbf{e}}(\hat{\mathbf{r}}_{\mathbf{e}}'(\mathbf{w}_{a(\mathbf{e})} - \bar{\mathbf{w}}_{a(\mathbf{e})})) \geqslant \mathscr{W},$$

$$\mathbf{w}_{\mathbf{e}}^L \leqslant \mathbf{w}_{\mathbf{e}} \leqslant \mathbf{w}_{\mathbf{e}}^U, \quad \mathbf{e} \in \mathcal{N},$$

$$\mathbf{0} \leqslant \mathbf{b}_{\mathbf{e}} \leqslant \mathbf{b}_{\mathbf{e}}^U, \quad \mathbf{e} \in \mathcal{N}_I \cup \mathbf{0},$$

$$\mathbf{0} \leqslant \mathbf{s}_{\mathbf{e}} \leqslant \mathbf{s}_{\mathbf{e}}^U, \quad \mathbf{e} \in \mathcal{N}_I \cup \mathbf{0}.$$

If $\alpha_t = 1$, then risk in each time period is weighted equally. The box constraints on $\mathbf{w_e}$, $\mathbf{b_e}$, $\mathbf{s_e}$ for each event $\mathbf{e} \in \mathcal{N}$ are to prevent short sales. The constant parameter $\mathscr{W}$ is supplied to constrain final expected wealth to a particular value. The optimization will find the lowest-variance (least risky) investment strategy to achieve that specified expected wealth. Varying $\mathscr{W}$ and reoptimizing will generate points along the efficient frontier. Therefore, the efficient investment strategies which are not totally risk seeking are calculated.

2. Simulation and randomized clustering approach

We describe a procedure based on simulation and randomized clustering to generate the event tree which is the input to the financial optimization problem. The basic data structure is the scenario tree node, which contains a cluster of scenarios (vectors in $\mathbb{R}^n$), one of which is designated as the centroid. The final tree consists of the centroids of each node, and their branching probabilities.

An approach similar to our clustering of parallel simulations (described below) is introduced in Dupacova et al. (2000), but without a detailed clustering algorithm. Instead of expending, effort to find clusterings which are optimal by some measures, we introduce a randomized clustering algorithm which can be repeated until an acceptable clustering is found. The main steps of our algorithm can be outlined as follows:

Step 1 (*Initialization*): Create a root node, with N scenarios. Initialize all the scenarios (including the centroid) with the desired starting point ('today's' prices). Form a job queue consisting of the root node.

Step 2 (*Simulation*): Remove a node from the job queue. Simulate one time period of growth (from 'today' to 'tomorrow') in each scenario.

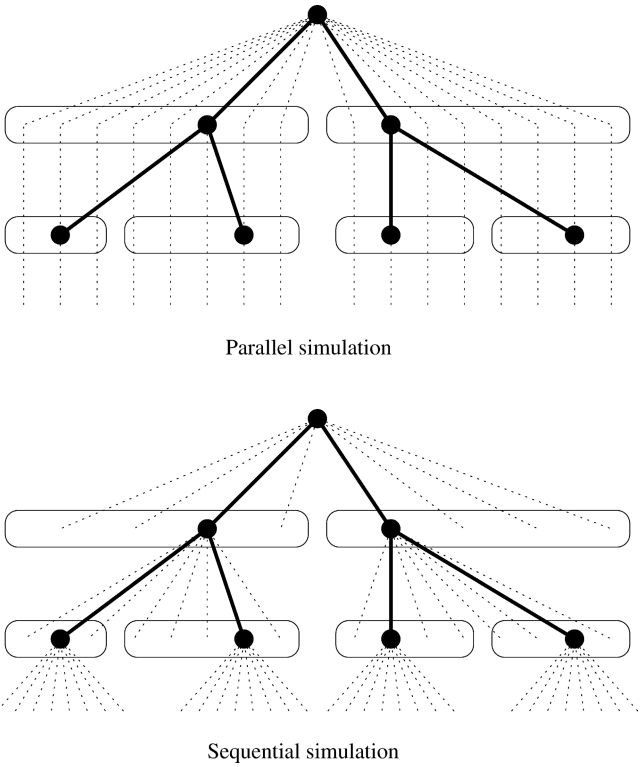

Fig. 1. Two methods of simulating scenarios. Dotted lines represent simulation paths, ovals represent how the scenarios are clustered, and the heavy lines depict the resulting scenario tree. The parallel simulation uses root branching $N = 16$ (with cluster size diminishing down the tree), and the sequential uses a constant $N = 8$ from every node.

Step 3 (*Randomized seeds*): Randomly choose a number of distinct scenarios around which to cluster the rest: one per desired branch in the scenario tree.

Step 4 (*Clustering*): Group each scenario with the seed point to which it is the closest. If the resulting clustering is unacceptable, return to step 3.

Step 5 (*Centroid selection*): For each cluster, find the scenario which is the closest to its center, and designate it as the centroid.

Step 6 (*Queueing*): Create a child scenario tree node for each cluster (with probability proportional to the number of scenarios in the cluster), and install its scenarios and centroid. If the child nodes are not leaves, append to the job queue. If the queue is non-empty, return to step 2. Otherwise, terminate the algorithm.

2.1. Parallel and sequential simulations

There are two methods of scenario simulation, parallel and sequential, which are illustrated in Fig. 1. Both methods rely on a probabilistic model which describes the

probability density function, pdf, $f(x_{t+1}|x_t)$. Given this pdf, the process of simulating a 'tomorrow' event, given 'today', will be called an update. In both methods, we consider pseudo-random numbers and low-discrepancy quasi-random Sobol sequences (e.g., Sobol, 1967; Bratley and Fox, 1998).

With parallel simulation, each of the scenarios belonging to the node is updated from their current values. Although memory is saved by performing these updates only as needed, the parallel method is equivalent to generating a number of linear scenarios for the entire time horizon before the process of clustering them into a scenario tree even begins. This is a common form of scenario generation, see also Dupacova et al. (2000). The time 0 elements of all those scenarios (which will probably be the same) form the root of the tree; the clusters of the time 1 elements formed by the first iteration of the algorithm are the level one nodes; those clusters are then divided into sub-clusters which represent level two nodes, etc. Thus, the total number of scenarios in all nodes in the job queue is always equal to the original number of scenarios. The other method is sequential simulation, in which all the scenarios of a node are initialized to that node's centroid before updating. Since this cannot be done until clustering at the previous level determines the centroid, the scenarios cannot be generated out to the time horizon before clustering (hence the designation as sequential).

The main practical difference between the methods stems from the following contrast: in parallel simulation, a large number of scenarios N (at least a multiple of the number of leaves) is necessary at the root, and clusters become smaller and smaller further down the tree. In sequential simulation, however, there is no need to use more scenarios at the root than at the leaf, so the same number N of scenarios can be simulated from every node. Because, in parallel simulation, scenarios must remain in memory for all jobs on the queue, whereas in sequential simulation, only scenarios for the present node are necessary, the sequential method will generally require less memory. However, the methods are largely incomparable in terms of speed. For the same value of N, the parallel method has the tremendous advantage of smaller descendants; on the other hand, with values of N adjusted to achieve similar resolution at the leaves, the sequential method has the tremendous advantage of not having to use an exponentially (in the depth of the tree) larger root value of N.

A comparison is more useful in terms of the character of the scenario trees produced by either method: the sequential method will produce much more homogeneous trees, and the parallel much more extreme, for two reasons. First, with N growing smaller down the tree in the parallel method, the centroids that eventually represent the scenario clusters are drawn from a smaller sample size; this can be justified by viewing forecasts nearer the time horizon as less critical than those for the immediate future. Second, in the sequential method, at every stage the simulated scenarios in all of the clusters are discarded, and the next simulation restarted from the centroid, which will prevent any extreme variation. If the user wants scenario trees that contain more realistic extreme events, he should use the parallel method, setting N so that simulation at the leaves has an appropriate level of resolution. If the number of scenarios needed at the leaves makes N prohibitively large, then the sequential method should be used instead. Alternatively, the methods could be mixed: begin with the parallel method with a smaller N, and

generate more scenarios as they are needed – when clusters become too small to divide into sub-clusters of appropriate sizes.

2.2. Randomized seeds

If k branches are desired from the current scenario tree node, then k clusters will need to be formed. Initially the seed points around which the clusters are built might as well be chosen to be the first k scenarios, since the scenarios are independently generated, and are in arbitrary order. If the resulting clustering fails to meet the criteria applied in the test stage, new seed points will have to be chosen, and the clustering process repeated, until the criteria are met.

It is possible each time to choose k completely new seed points, but that would discard all of the information earned in the current iteration. If only seeds of problem clusters are replaced, the next iteration will have an improved chance of success, and total execution times improved in the long run, because of the reduction in long strings of failures. Strategies for intelligent seed replacement would be tied to the constraints that are violated. For instance, if the clusters are not uniform enough in size, effective strategies could include replacing seeds for only the smallest and/or largest clusters, or replacing seeds of some small clusters with scenarios randomly chosen from the largest clusters.

2.3. Clustering

The distance measure to determine which seed each scenario is closest to can be chosen with great flexibility. Possible choices include Euclidean, Manhattan, or any other p-norm. In the implementation described in this paper, the square of the Euclidean was chosen for efficient calculation. A number of criteria can be applied to the clustering which results from the randomized selection of seeds. One of the most important is the relative sizes of the clusters. Not only do they affect the character of the tree (allowing smaller clusters will allow more extreme, low probability, events to be represented in the scenario tree), but, with the parallel simulation method, it is necessary to ensure that any cluster will contain enough scenarios to divide into sub-clusters in all future levels. Bounds for the number of scenarios necessary for a node at any level can be computed bottom-up, starting from a desired number of scenarios per leaf; from this the minimum possible N at the root in a parallel simulation can be calculated. Practically, considerably more scenarios will be needed, since the clusters are chosen randomly, and cannot be expected to be uniformly the bare minimum in size.

Constraints can be placed on the sizes of the clusters in many ways: larger or smaller than certain sizes, having a ratio of largest/smallest within a certain range, etc. Constraints can also be set on the character of the clusters, not just the size. Bounds can be placed on statistical properties such as the means of the assets, or the variances within the clusters, etc. It is important to note that, the stricter the constraints, the less likely they are to be satisfied by this random clustering process, so it could take many iterations to reach an acceptable clustering. The user can adjust the acceptance criteria to reach an acceptable tradeoff between speed of execution, and quality of the

resulting scenario tree, but if the necessary constraints make this randomized procedure impractically slow, it would probably be more appropriate to apply a non-randomized method that seeks more directly to satisfy those constraints, such as the other methods in this paper, or Christofides et al. (1999).

2.4. Centroid selection

Once an acceptable clustering is found, it is necessary to represent each cluster with a single point, which becomes the data in the scenario tree. In addition to a measure of distance, a notion of 'center' needs to be fixed: mean, median, center of gravity, etc. Although the points in each cluster are closer to their seed point than to any other, there is no particular reason that a seed should be near the center of its cluster; it could be an extreme outlying event that gathered a cluster of all the points that were away from the center in a similar direction. Also, in a space of high dimension, even a large cluster of points may be quite sparse, with no points actually near the center. To prevent the scenario tree from containing scenarios that are not consistent with the simulation parameters, the centroid should be taken to be not be the measured center of the cluster, but rather the simulated scenario closest to the center.

3. Optimization approach

In an optimization approach to generate a scenario tree, the decision maker specifies the market expectations by the statistical properties that are relevant for the problem to be solved. The event tree is constructed so that these statistical properties are preserved. This is done by letting stochastic returns and probabilities in the scenario tree be decision variables in a non-linear optimization problem where the objective is to minimize the square distance between the statistical properties specified by the decision maker and the statistical properties of the constructed tree Hoyland and Wallace (2001).

For the general description of the scenario generation approach, we assume a symmetric tree, meaning that the number of branches is the same for all conditional distributions in the same period. However, there is no theoretical restriction on the tree topology we might wish to generate. Let S denote the set of all specified statistical properties and SV_i be the value of specified statistical property i, $i \in S$. If x and p denote the price vector and the probability vector, respectively, the mathematical expression of statistical property i can be defined as a function of these unknown random variables, $f_i(x, p)$. The objective is to construct x and p so that sum of square deviations between the statistical properties of the constructed distribution and the specifications is minimized. The scenario generation model, therefore, can be stated as follows:

$$\min_{x,p} \quad \sum_{i \in S} w_i (f_i(x, p) - SV_i)^2$$

$$\text{s.t.} \quad \sum p_i = 1, \quad p \geqslant 0,$$

where w_i is the weight for statistical property i.

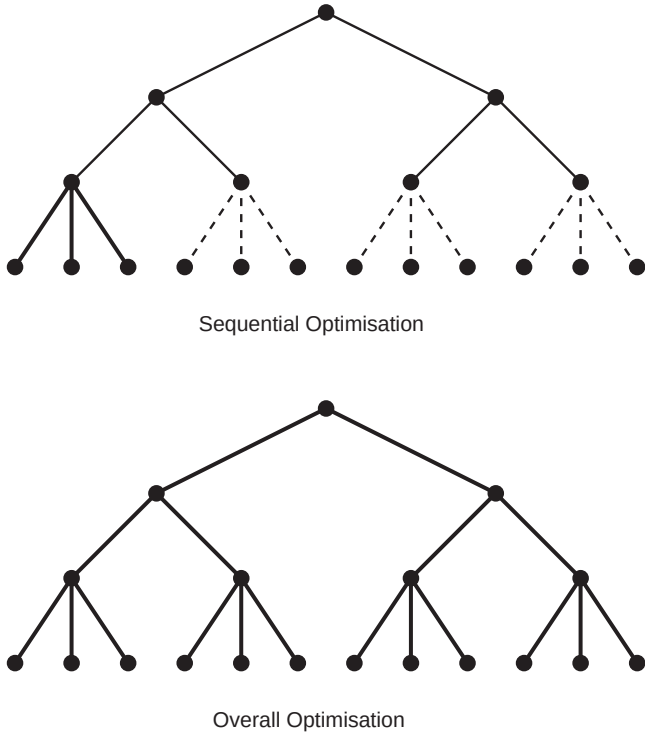

Fig. 2. Two procedures for generating scenarios based on optimization approach. For the sequential, solid lines represent optimization problems previously solved, dotted lines future problems to be solved, and heavy lines the current optimization problem. In overall optimization, the entire tree is optimized at once.

In general, this non-linear optimization problem is not convex, therefore the solution might be a local optimum. However, it is satisfactory for our purposes to have a solution with distribution properties equal or close to the specifications of the continuous distribution. An objective value equal to or close to zero indicates that distribution of the scenarios has a perfect or good match with the specifications, respectively. In case of the lack of a perfect match (because of non-convexities or inconsistent specifications), the weights w_i can incorporate the relative importance of satisfying the different specifications.

In this paper, we adopt the basic idea described above to generate the scenario tree which is input to the financial portfolio allocation problem. The decision variables of the optimization problem are the prices (or returns) of a set of assets and probabilities of the event tree. Two alternative ways of applying the optimization approach are investigated to construct the event tree, which are illustrated in Fig. 2. If the scenario tree is constructed by considering the branching at each node separately, then we call it *sequential optimization*. In this case, a small non-linear optimization problem is constructed and solved at each node of the scenario tree. An alternative approach is to

consider all nodes of the event tree and generate the whole tree in one large non-linear optimization problem, which we call *overall optimization*.

3.1. Sequential optimization

Any central moments and co-moments can be part of the statistical specifications of the distribution. We consider the first four central moments, namely expected value, standard deviation, skewness and kurtosis, and covariances as co-moments. Let $I = \{1,2,\ldots,n\}$ denote the set of n assets. Let M_{ik}, for $k = 1,2,3,4$, be the first four central moments of the current continuous distribution for the asset i. The covariance of assets i and l (such that $i,l \in I$ and $i < l$) is denoted by C_{il}.

Let N_t be the number of branches from a node at stage $t = 1,\ldots,T-1$. The price scenarios x_{ij} for asset $i \in I$ and probabilities p_j for $j = 1,\ldots,N_t$ of the continuous distribution are decision variables in the following non-linear optimization problem:

$$\min_{x,p} \quad \sum_{i=1}^{n}\sum_{k=1}^{4} w_{ik}(m_{ik} - M_{ik})^2 + \sum_{i,l\in I,\ i<l} w_{il}(c_{il} - C_{il})^2$$

$$\text{s.t.} \quad \sum_{j=1}^{N_t} p_j = 1,$$

$$m_{i1} = \sum_{j=1}^{N_t} x_{ij}\, p_j, \qquad i \in I,$$

$$m_{ik} = \sum_{j=1}^{N_t} (x_{ij} - m_{i1})^k\, p_j, \qquad i \in I,\ \ k = 2,3,4,$$

$$c_{il} = \sum_{j=1}^{N_t} (x_{ij} - m_{i1})(x_{lj} - m_{l1})\, p_j, \qquad i,l \in I \text{ and } i < l,$$

$$p_j \geqslant 0, \qquad j = 1,\ldots,N_t,$$

where $w_{ik} = w'_k/M_{ik}^2$ and $w_{il} = w'_{il}/C_{il}^2$ are weights in which w'_k for $(k = 1,\ldots,4)$ are the relative importance of the central moments and w'_{il} for covariances of assets $i,l \in I$.

In this formulation, the first constraint shows that probabilities must sum to one at each individual branching, while the rest is the formulations of the first four central moments and the co-moments. The last constraints ensure that probabilities are non-negative. Notice that if the central moments of the current distribution m_{ik} $(k=1,\ldots,4)$, and co-moments c_{il} are substituted in the objective function, the problem becomes a non-linear optimization problem with linear constraints.

3.1.1. Estimation of conditional moments

It is important to note that M_{ik} and C_{il} are computed conditional on past history and associated path of the scenario tree. At the root node of the scenario tree, we consider

the price history $H_0 = (O_1, \ldots, O_n)$ which is the list of n vectors corresponding to n assets. The historical price data O_i is observed for each asset i ($i \in I$) monthly or weekly from a specific date up to the present. Let h_{obs} be number of historical observations.

At the node $\mathbf{e} \in \mathcal{N}_t$ for $t = 1, \ldots, T-1$, the historical data set H_0 is updated by including the sequence of price vector scenarios from the root to the current node, $H_{\mathbf{e}} = H_0 \cup (X_0, \ldots, X_{\mathbf{e}})$. Assume that the original historical data H_o and the price sequence $(X_0, \ldots, X_{\mathbf{e}})$ have, respectively, h_{obs} and n_{obs} number of observations. The new updated price list $H_{\mathbf{e}}$ has data of length $h_{\text{obs}} + n_{\text{obs}}$. The price list $H_{\mathbf{e}}$ is fit to an exponential growth curve

$$y_{ij} = a_i \mathrm{e}^{b_i z_{ij}}, \quad i \in I, \ j = 1, \ldots, h_{\text{obs}} + n_{\text{obs}}, \tag{1}$$

where y_{ij} is a dependent variable (an observed price data), z_{ij} is an independent variable (time periods) and a_i and b_i are constant parameters to be estimated.

The residuals measure the vertical distance of estimated price values to the exponential curve and are essentially calculated as difference between the estimated and observed price values using the following the formula:

$$y_{ij} - a_i \mathrm{e}^{b_i z_{ij}}, \quad i \in I, \ j = 1, \ldots, h_{\text{obs}} + n_{\text{obs}}. \tag{2}$$

From the residuals obtained from the exponential fit of the historical price data $H_{\mathbf{e}}$, the central moments and the co-moments can be calculated. For the projected first moment from any node $\mathbf{e} \in \mathcal{N}_t$, we use the following property:

$$M_{i1\mathbf{e}} = \mathrm{e}^{r_i} x_{i\mathbf{e}}, \quad \forall i \in I. \tag{3}$$

where $x_{i\mathbf{e}}$ is the last element of the historical data and r_i denotes the growth rate for asset i. For variance, skewness, kurtosis and covariances are computed by the following formulae:

$$M_{ik\mathbf{e}} = \frac{1}{h_{\text{obs}} + n_{\text{obs}} - k + 1} \sum_{j=1}^{h_{\text{obs}}+n_{\text{obs}}} (y_{ij} - M_{i1\mathbf{e}})^k, \quad \forall i \in I, \ k = 2, 3, 4 \tag{4}$$

and covariance between assets i and l is

$$C_{il\mathbf{e}} = \frac{1}{h_{\text{obs}} + n_{\text{obs}} - k + 1} \sum_{j=1}^{h_{\text{obs}}+n_{\text{obs}}} (y_{ij} - M_{i1\mathbf{e}})(y_{lj} - M_{i1\mathbf{e}}),$$

$$\forall i \neq l \in I, \ k = 2, 3, 4. \tag{5}$$

3.1.2. Algorithm

At the root node, only one non-linear optimization problem is constructed and solved. Therefore, N_0 discrete realizations of the price scenarios and their probabilities (that are consistent with the first four central moments and co-moments specified at the root node) are obtained.

For each generated outcome at root node, we specify the conditional distribution properties for the period $t = 1$ (by using the data $H_{\mathbf{e}} = H_0 \cup X_{\mathbf{e}}$, $\mathbf{e} \in \mathcal{N}_1$). Therefore at $t = 1$, we have N_0 non-linear optimization programs whose solutions give the outcomes at $t = 2$. Thus, for each scenario we obtain the solutions for consecutive stages after

generating scenarios conditional on the previous outcome. The same procedure is carried out until the last period. In general terms, the sequential optimization scenario generation approach can be outlined as follows:

Repeat for $\mathbf{e} \in \mathcal{N}_t$ and $t = 0, \ldots, T-1$.

Step 1: Fit data $H_{\mathbf{e}}$ to an exponential growth curve.

Step 2: Calculate the central moments and co-moments from the residuals of the data $H_{\mathbf{e}}$ from the fitted curve.

Step 3: Create and solve the optimization problem of finding new price vector scenarios and probabilities to fit the above moment data.

Step 4: Update the historical data.

The last issue we discuss in this section is the negative prices that can arise in the solution of the non-linear optimization problem because of high volatility. To prevent this, we can bound the price of an asset 'today' to at least α% of the price of the same asset 'yesterday'. Hence, if the negative prices are encountered, the following constraints are added to the above optimization problem and solved again

$$x_{ij\mathbf{e}} \geqslant \left(\frac{\alpha}{100}\right) x_{ia(\mathbf{e})}, \quad \forall i \in I, \ j \in N_t, \tag{6}$$

where $a(\mathbf{e})$ denotes the parent node of node $\mathbf{e}$.

3.2. Overall optimization

Since the sequential optimization approach requires the distribution properties to be specified only local to each node of the tree, the approach lacks direct control of the statistical properties defined 'overall' outcomes in the later periods ($t > 0$). Also the sequential optimization involves a more rigid optimization scheme. First period trees might satisfy the first period specifications, but lead to conditional second period specifications which make it impossible to obtain a perfect match. Therefore, we can say that the sequential optimization procedure generates a suboptimal scenario tree. In this aspect a model that generates the whole event tree in one large optimization is more realistic than sequential optimization.

The main disadvantage of the overall optimization approach is that the degree of non-convexity increases and a perfect match is difficult to compute. Also the size of the non-linear optimization problem increases in terms of the number of variables and constraints, depending on the number of assets, the depth and branching structure of the scenario tree at each stage as well as the number of moments and co-moments matched. Finally, sequential optimization allows easy updating of the statistical properties at each node of the tree as input to the problem, whereas overall optimization includes an updating procedure as a function in the optimization problem.

The overall optimization procedure follows the same idea explained in Section 3.1. However, instead of generating tree node by node, the whole tree is generated by solving one large optimization problem. Thus, all possible nodes of the tree are considered in order to model the optimization problem such that the structure of the event tree matches with the properties of the historical data. Given the notation in the previous

section, the non-linear programming model for the overall optimization procedure can be formulated as follows:

$$\min_{x,p} \quad \sum_{t=0}^{T-1} \sum_{\mathbf{e}\in\mathcal{N}_t} \left(\sum_{i=1}^{n} \sum_{k=1}^{4} w_{ike}(m_{ike} - M_{ike}(H_{\mathbf{e}}))^2 + \sum_{i,l\in I,\ i<l} w_{ile}(c_{ile} - C_{ile}(H_{\mathbf{e}}))^2 \right)$$

$$\text{s.t.} \quad \sum_{j=1}^{N_t} p_{je} = 1, \quad \mathbf{e}\in\mathcal{N}_t,\ t=0,1,\ldots,T-1,$$

$$m_{i1e} = \sum_{j=1}^{N_t} x_{ije}\, p_{je}, \quad i\in I,\ \mathbf{e}\in\mathcal{N}_t,\ t=0,1,\ldots,T-1,$$

$$m_{ike} = \sum_{j=1}^{N_t} (x_{ije} - m_{i1e})^k\, p_{je}, \quad i\in I,\ k=2,3,4,\ \mathbf{e}\in\mathcal{N}_t,\ t=0,1,\ldots,T-1,$$

$$c_{ile} = \sum_{k=1}^{N_t} (x_{ike} - m_{i1e})(x_{lke} - m_{l1e})\, p_{ke},$$

$$i,l\in I,\ i<l,\ \mathbf{e}\in\mathcal{N}_t,\ t=0,1,\ldots,T-1,$$

$$p_{je} \geqslant 0, \quad j=1,\ldots,N_t,\ \mathbf{e}\in\mathcal{N}_t,\ t=0,1,\ldots,T-1.$$

The main steps of the computational procedure can be summarized in the following algorithm:

3.2.1. Algorithm

Step 1: Fit data H_0 to an exponential growth curve.

Step 2: Calculate the central moments and co-moments from the residuals of the data H_0 from the fitted curve.

Step 3: Model the non-linear optimization problem and solve it to generate price scenarios and probabilities of branches of the scenario tree. The estimation of conditional moments are discussed below.

In order to ensure against negative prices, the inequalities in (6) are imposed as further constraints and the model is reoptimized. Notice that now both $x_{ij\mathbf{e}}$ and $x_{ia(\mathbf{e})}$ are decision variables, so inequality (6) is a linear constraint, not simply a variable bound.

3.2.2. Estimation of conditional moments

The statistical targets $M_{ik\mathbf{e}}$ and $C_{il\mathbf{e}}$ are no longer constant values computed prior to the optimization problem in which they are used as in the sequential optimization approach. Since they depend on the statistical measurement of the past, they are functions of the history $H_{\mathbf{e}}$, which makes them dependent on the values of decision variables from ancestor nodes. This functional dependence cannot be written as an

expression, as the calculation of targets $M_{ik\mathbf{e}}$ and $C_{il\mathbf{e}}$ is not a simple function, but rather an algorithmic process applying the following steps.

Repeat for $\mathbf{e} \in \mathcal{N}_t$ and $t = 0, \ldots, T-1$:

- fit asset histories to an exponential curve given in (1),
- calculate residuals as presented in (2),
- calculate higher central- and co-moment targets from residuals using formulae introduced (3), (4) and (5).

This process has to be carried out for every statistical target: at every node, for every asset, for every central- and co-moment, for every evaluation of the objective function (including evaluations for the purposes of numerical gradient calculation). This complicated objective function has a very high cost in time (as detailed in Section 5), but without it, the overall optimization method could be nothing more than solving all of the sequential optimization problems in one larger separable problem.

3.3. Arbitrage constraints

Some financial applications, such as option pricing, require scenario trees which are free from arbitrage. An arbitrage opportunity is a self-financing trading strategy that generates a strictly positive cash flow between 0 and T in at least one state and does not require an outflow of funds at any date. It is clear that investors would engage in such a trading strategy as much as possible if we assume that investors always prefer more to less. Therefore, such a trading opportunity cannot exist if market is in equilibrium. Formally, it is said that there are no arbitrage opportunities in the market if and only if there exists an unique risk-neutral (Martingale) probability measure; for further details see, for instance Taqqu and Willinger (1987). The issue of an arbitrage-free economic market is often neglected in stochastic programming models for financial applications. There are few examples in the financial stochastic programming literature which consider arbitrage constraints.

Klaassen (1998) analyses the effect of arbitrage in the case of asset liability management for banks by using internal sampling. He also illustrates that the presence of arbitrage opportunities may cause substantial biases in the optimal investment strategy in Klaassen (1997). Kouwenberg and Vorst (1998) generate an arbitrage-free scenario tree while fitting the mean and variance of the underlying distribution.

In the optimization-based scenario tree generation approach, we can take into account simultaneously both arbitrage opportunities and statistical properties of the event tree. Therefore, not only do generated price scenarios at each time period have the statistical properties which match the first four central moments and co-moments, but also they are arbitrage-free. The following hard constraints, which we call arbitrage constraints, can be imposed a priori to the non-linear optimization problem:

$$x_{i\mathbf{e}} = \mathrm{e}^{-r_i} \sum_{s \in S(\mathbf{e})} x_{is}\pi_s, \quad \forall i \in I,\ \mathbf{e} \in \mathcal{N}_t,\ t = 0, \ldots, T-1, \tag{7}$$

where $S(\mathbf{e})$ is the set of successor nodes of $\mathbf{e}$ and π_s is the strictly positive probability measure for $s \in S(\mathbf{e})$, see Kouwenberg and Vorst (1998).

3.4. Discounted mean constraints

Let M_{i0} be the mean of the historical data for each asset i at $t=0$. The decision variable $x_{i\mathbf{e}}$ is the price of asset $i \in I$ at node $\mathbf{e} \in \mathcal{N}_t$ for $t = 1, \ldots, T$. The following constraints, we call discounted mean constraints, for each asset are imposed into the non-linear optimization model at time periods t $(t > 1)$ or only at the last period $t = T$,

$$\sum_{e \in \mathcal{N}_t} P_{\mathbf{e}} x_{i\mathbf{e}} = \mathrm{e}^{-r_i t} M_{i0}, \quad \forall i \in I, \tag{8}$$

where $P_{\mathbf{e}}$ denotes the probability of a partial scenario of event $\mathbf{e}$. A partial scenario is described by the path from the root to the event $\mathbf{e}$ and its probability $P_{\mathbf{e}}$ is defined as the product of the probabilities of branching along that path. Notice that probabilities $P_{\mathbf{e}}$ will sum up to one across each layer of tree-nodes N_t, $(t = 0, 1, \ldots, T)$, since probabilities $p_{\mathbf{e}}$ must sum to one at each individual branching.

4. Hybrid approach

In the optimization approach, economic scenarios are generated to ensure that their statistical properties match with the distribution of the returns while in the simulation approach scenarios are the centroids of simulations. A *hybrid* approach combines the main ideas of the simulation and optimization approaches. In this approach, prices are obtained as the centroids of clustering of simulations and substituted for decision variables in the optimization problem. The probabilities are then determined by solving the optimization problem, whose size has been greatly reduced.

Consider an event $\mathbf{e}$ at time period t $(t = 0, \ldots, T-1)$. Let $r_{ij\mathbf{e}}$ be the return simulated for asset $i \in I$, at node $\mathbf{e}$ and at branch $j = 1, \ldots, N_t$. Probabilities $p_{j\mathbf{e}}$ of the continuous distribution at node $\mathbf{e}$ are variables of the optimization problem. The hybrid approach for the sequential optimization procedure can be outlined as follows:

Repeat for $\mathbf{e} \in \mathcal{N}_t$ and $t = 0, \ldots, T-1$.

Step 1: Fit data $H_{\mathbf{e}}$ to an exponential growth curve.

Step 2: Calculate the central moments and co-moments from the residuals of the data $H_{\mathbf{e}}$ from the fitted curve.

Step 3: Simulate N_t price vector scenarios with clustering, as described in Section 2.

Step 4: Fix price decision variables to the simulated values in the optimization problem, and solve to find probabilities.

Step 5: Update the historical data.

For the hybrid approach with overall optimization, price scenarios are generated by simulation procedures a prior and probabilities are calculated by solving the reduced non-linear optimization problem. If the optimization problem does not have a feasible solution, then simulation is repeated to obtain different scenarios which might match the statistical properties of the historical data.

5. Computational experiments

5.1. Implementation

The simulation-based scenario generation approach was coded in C++. For clustering and centroid selection, the distance measure is Euclidean 2-norm, and the center of a cluster of scenarios is their vector mean. Two parameters, the number of scenario simulations at the root N and the maximum ratio between largest and smallest cluster sizes, are specifiable. If the parallel method is used, an additional parameter, m, the minimum number of scenarios per leaf, can be specified to ensure no clusters will be too small for subsequent division. With both simulation procedures, we apply the following rule to update. If today's prices are x_{tod}, then tomorrow's prices are simulated from a multivariate lognormal distribution with mean $\exp\{\mathbf{r}\} \otimes x_{\text{tod}}$ and covariance matrix Λ (where growth rate vector $\mathbf{r}$ and covariance matrix Λ are specified as data).

The optimization-based scenario generation approach was implemented in Fortran 77 and the NAG Library (2003) was used to solve non-linear optimization problems. The following parameters need to be specified for the optimization approach:

- the number of stages,
- the number of successors per node branching at each stage.

For the hybrid approach, the number of scenario simulations at root node of the tree needs to be specified. Scenarios at each node can be output either in terms of prices or returns. In the latter case, rates of return are obtained by dividing the current node's price by the price of the parent scenario node. The resulting scenario tree is output in the input format of multistage portfolio optimization program *foliage*, Gülpınar et al. (2002). Thus, generated scenario trees can be easily optimized over, and backtested with historical data.

5.2. Testing

The above implementation was tested by generating scenario trees from historical price data, then using multistage portfolio optimization software *foliage*, Settergren (2000), to determine optimal investment strategies (for specified levels of risk), and measuring the success of those strategies by their performance with the historical data. See Gülpınar et al. (2002), for more specific details of the backtesting procedures, but a summary follows. The historical data consisted of monthly price data of 10 FTSE stocks through the 1990s. At any particular 'present' time, the previous 10 time periods were fit to exponential growth curves yielding parameters $\mathbf{r}$, and residuals from that fit yielded covariance matrix Λ and the third and fourth central moments. A scenario tree was generated, optimized over with *foliage*, and the resulting investment strategy implemented at the 'present' prices, and portfolio value updated according to 'tomorrow' prices. Then the present was moved forward one time period, and the process repeated. Note that, for any given scenario tree, the optimizer can yield the entire range of efficient strategies, from risk seeking to risk averse (in other words

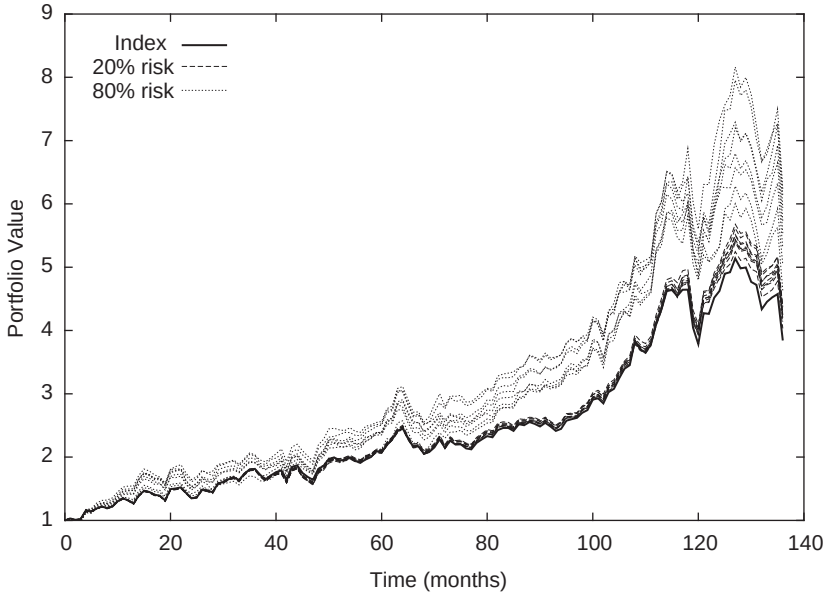

Fig. 3. Multiple randomized backtesting experiments, at low risk and high risk.

from the highest expected return to the lowest risk), so the desired risk level is another parameter that needs to be specified during a backtesting experiment.

5.2.1. Backtesting with simulation

It is important to investigate the robustness of this procedure, since it involves randomness. Fig. 3 illustrates backtesting Fig. 3 results at 20% and 80% risk for a number of back-testing runs using randomly generated scenario trees (in this situation, using a fixed price history and stochastic investment strategies can be viewed as kind of a backwards perspective of market volatility). The parameters of the simulations are 10 time periods of past history, 3 time periods forecast, with branching of 4 (64 leaves of the scenario tree) with $N = 5000$ simulations. Two contrasting features of the sets of low- and high-risk curves are as they should be: simulations at 80% risk have both a higher mean return, as well as a higher volatility. It is interesting to note that the crash near month 120 in all of the backtesting reflects the crash in the FTSE in late 1998.

Although different randomized backtesting experiments do yield different results (sometimes the randomized scenario trees perform better, sometimes they perform worse) the trend is consistent. One way to combat this is to use, instead of pseudo-random numbers, low-discrepancy Sobol sequences, Sobol (1967), and Bratley and Fox (1998). In addition to being deterministic, the low-discrepancy property causes the number of simulations N to be as effective as a much larger number of randomly generated numbers. Fig. 4 plots backtesting performed using a deterministic Sobol sequences against eight experiments with random generation.

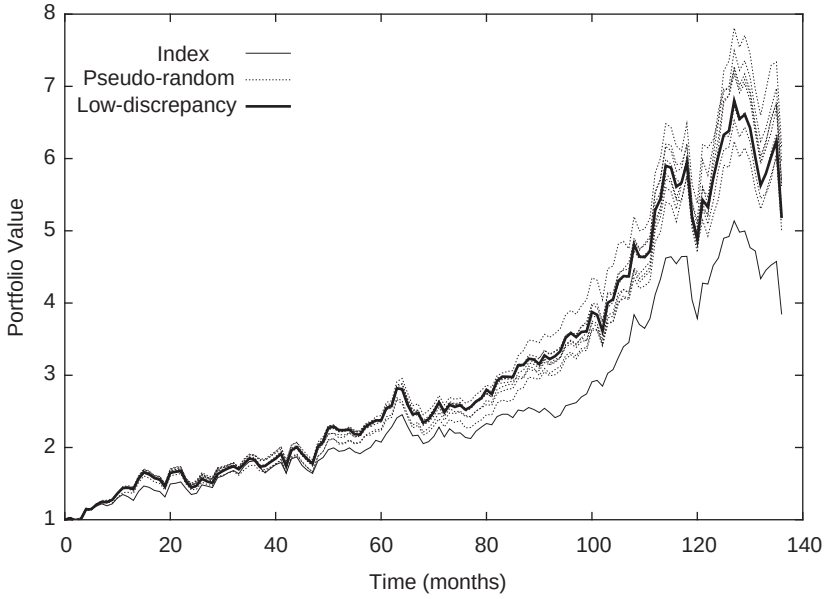

Fig. 4. Backtesting using deterministic low-discrepancy (Sobol) quasirandom sequences, vs. randomized backtesting at 75% risk level.

Figs. 5 and 6 contrast the results of using sequential vs. parallel simulation at one particular setting of the remaining parameters: 10 time periods of past history, 3 time periods forecast, with branching of 4 (64 leaves of the scenario tree). $N = 5000$ simulations were used for both the sequential and parallel methods; also, Sobol sequences were used to reduce the effects of randomness. For each set of simulated scenario trees, backtesting was performed at risk levels 25%, 50%, 75%, and 99%, and the results plotted against an equally weighted index of the assets involved.

In Fig. 6, the performance is slightly better (except the 50% risk curve), perhaps because the parallel simulation produces more extreme scenarios (see Section 2.1), and with those in mind, the optimizer hedges against these extremes with more conservative strategies. In both cases, however, there is a fairly clear ordering of the results of the varying-risk strategies, with higher risk yielding higher return – on average, with occasional violations due to inevitably higher volatility.

5.2.2. Backtesting with optimization

Figs. 7 and 8 are the results of using sequential optimization and overall optimization with 10 time periods of past history, 3 time periods forecast, with branching of 3 (27 leaves of the scenario tree). Similarly, for each set of simulated scenario trees, backtesting was performed at risk levels 25%, 50%, 75%, and 99%, and the results plotted against an equally weighted index of the assets involved. Although, we can see a fairly clear ordering of the results of the varying-risk strategies in sequential optimization procedure, this is not case for the overall optimization. The result in Fig. 8 indicates that

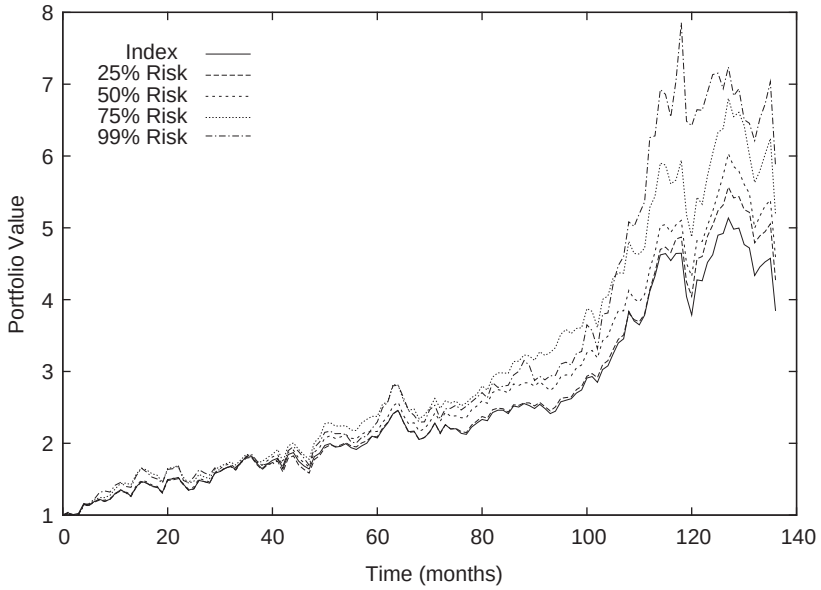

Fig. 5. Backtesting using the sequential method of simulation.

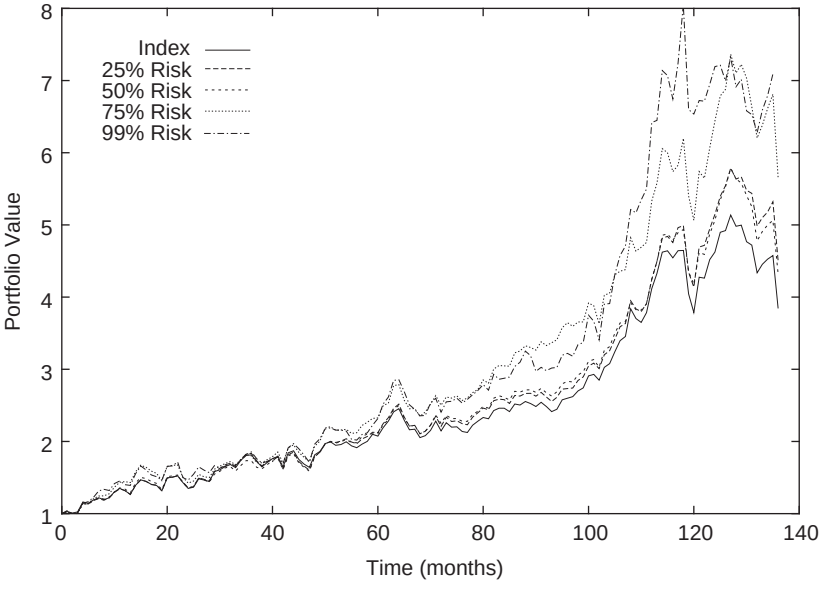

Fig. 6. Backtesting using the parallel method of simulation.

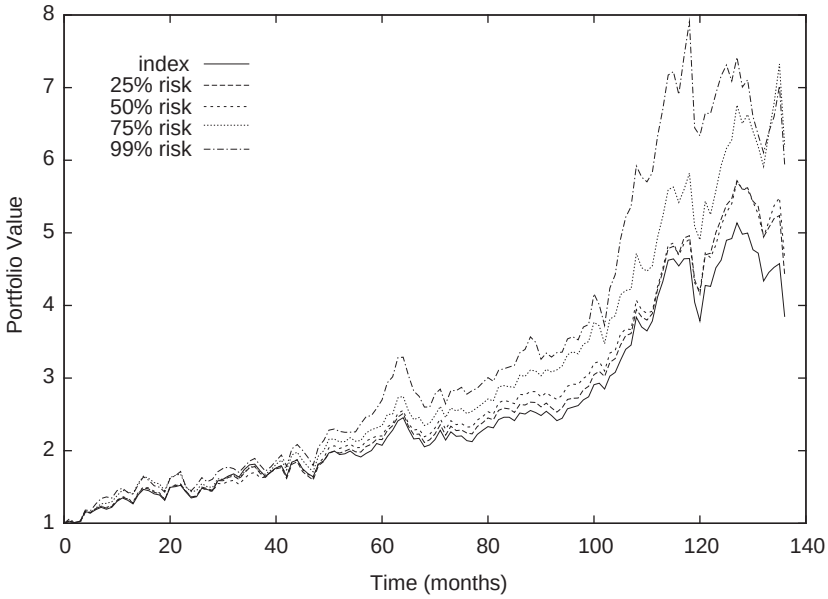

Fig. 7. Backtesting using sequential optimization procedure.

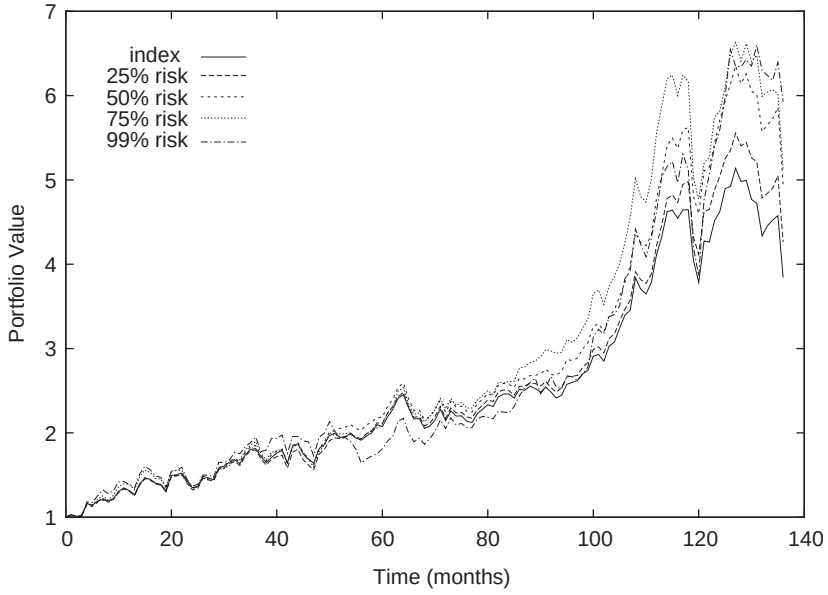

Fig. 8. Backtesting using overall optimization procedure.

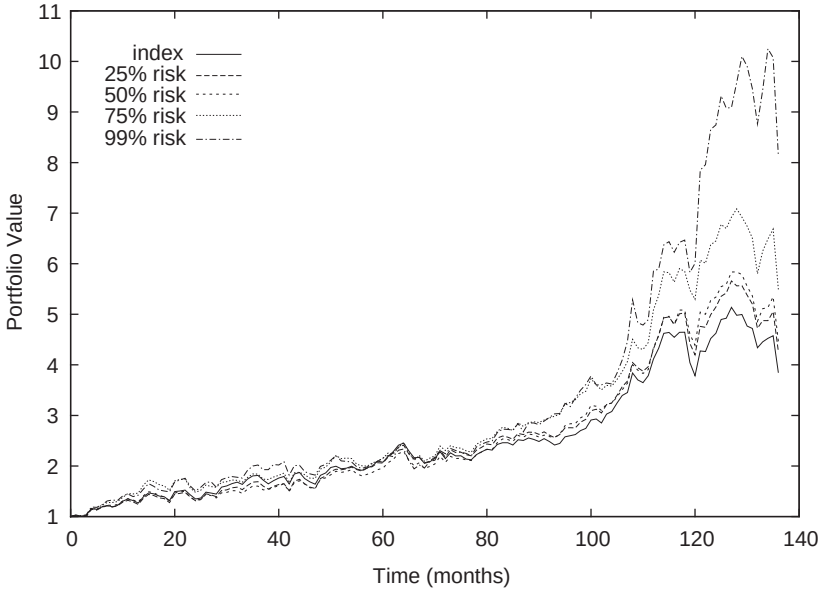

Fig. 9. Backtesting using hybrid approach with sequential optimization.

lower risk yields a better actual result. This shows that a risk-averse strategy performs almost as well in these cases and would have been able to respond to various random shocks, had they arisen. The reason for this is that the optimization-based scenario generation procedure does not take care of the extreme negative or positive events which influence the solution of the portfolio optimization problem.

5.2.3. *Backtesting with simulation and optimization (hybrid)*

Figs. 9 and 10 present backtesting results using the hybrid approach with sequential and overall optimization with 10 time periods of past history, 3 time periods forecast, with branching of 3. Price scenarios are generated by sequential simulations of $N = 10\,000$ branches with Sobol at each node. These results show that, for this set of data, the hybrid approach with sequential optimization is superior to other approaches in terms of the portfolio value.

5.3. *Processing time*

Since in terms of actual performance (backtesting) all scenario generation strategies are reasonably similar, the efficiency of these approaches measured in terms of CPU seconds spent for generating scenario trees is considered. For a fair comparison, 3- and 4-stages scenario trees with 2, 3 and 4 branchings are generated by all approaches using the same historical data: 10 time periods of history for 10 assets. For the simulation approaches, $N = 10\,000$ was held constant for all instances. This number was chosen as appropriate for the problem which needs the largest number of simulations, which

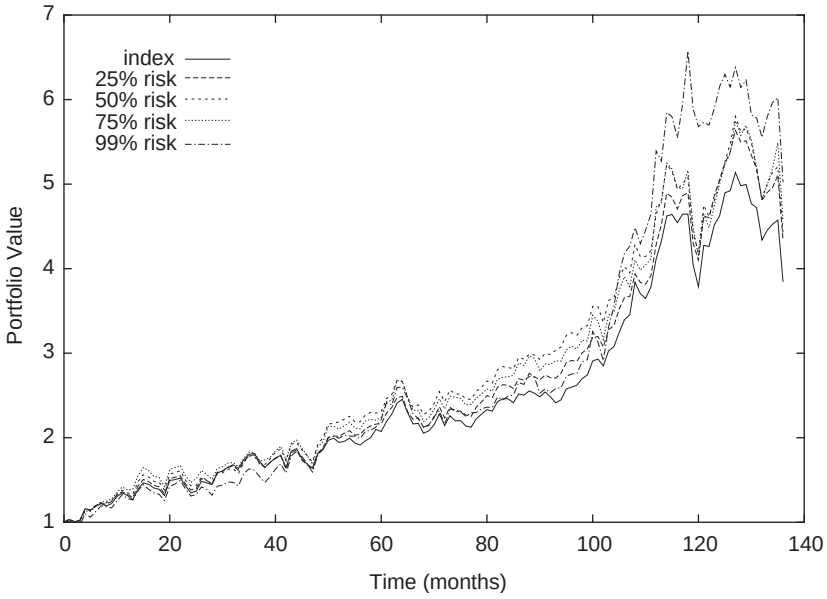

Fig. 10. Backtesting using hybrid approach with overall optimization.

Table 1
CPU time (s) taken to generate scenario trees by different approaches

Method (Variant)	Depth—branching					
	3-stage tree			4-stage tree		
	2	3	4	2	3	4
Sim. (parallel)	2.93	3.01	3.20	3.95	4.10	4.26
Sim. (sequential)	6.97	13.57	22.57	14.82	41.46	94.19
Hyb. (sequential opt.)	13.42	31.36	69.12	28.74	104.62	291.24
Hyb. (overall opt.)	48.30	1021.80	6732.85	334.00	—	—
Opt. (sequential)	3.22	20.85	513.20	6.53	641.19	547.77
Opt. (overall)	422.30	2857.92	—	40860.20	—	—

is parallel simulation on the largest tree (branching of for 4 time periods yields 256 leaves of the scenario tree, therefore the average simulations in each leaf's cluster is about 39). The scenario trees were all generated on a 200 MHz Sun Sparc II with 256 Mb of memory. The results of CPU time in seconds taken to generate scenario trees by different approaches are presented in Table 1.

It is interesting to note that, since in parallel simulation the number of scenarios processed per time stage is constant (N), the timings are almost exactly the same to generate trees of the same depth. In addition, the time increase to generate trees of depth

4 instead of 3 is approximately $\frac{4}{3}$. Sequential simulation, however, does increase in time for larger trees; as expected, the amount of work per node is approximately invariant. From $6.97/(1+2+4)=0.9957$ s/node to $94.19/(1+4+16+64)=1.1081$ s/node, the time to process $N=10\,000$ simulations remains similar to that in the parallel simulation, about 1 s on this computer. It is also important to note that the time required by the simulation methods can be quite variable: increasing or decreasing N will affect the timing proportionately, and parameters affecting the clustering rejection rate (and thus the number of re-clusterings necessary to generate the tree) can also have a significant impact on timing.

The main factor affecting the efficiency of the optimization-based approach is the size of the non-linear optimization problem (in overall optimization) or the number of optimization problems to solve (in sequential optimization). The size of the problem depends on the number of assets, depth and branching specifications of scenario tree. Although the problem size is small at each node of the scenario tree for the sequential optimization, it is slow for large scenario trees compared to the simulation-based approach. However, for moderate size scenario trees, it is still faster than other optimization approaches.

The hybrid approach with sequential optimization requires less time than sequential optimization to generate larger scenario trees, since the hybrid approach takes advantage of the simulation procedure. In addition, this approach is comparable with the simulation-based approach in terms of efficiency to generate the trees. It is difficult to estimate the time spent to solve the non-linear optimization problem at each node; it varies since the input to the optimizer, as well as the objective function, are changing at each node of the scenario tree since the historical data and resulting exponential fit are updated, even though the size of the problem is the same as long as the same branching is specified at each time period.

The results show that the overall optimization is by far the slowest method, and the added effort required does not seem to be justified by our backtesting results. Apart from the size of the problem, the updating procedure for every statistical target – at every node, for every asset, for every central- and co-moment, for every evaluation of the objective function – slows down the overall optimization procedure. Because of the extremely complicated objective function of overall optimization problems, even reduction of the number of variables using the hybrid technique is not always enough to allow problem solution. As can be seen in the table, the 3-stage, 4-branching problem was solvable by the hybrid method (but not overall), but beyond that size, even the hybrid problem was not solvable.

It is worthwhile to mention that the efficiency of optimization-based approaches also depends on the relative weight for each asset's central moments and co-moments. The user must be aware of the input to the non-linear optimization problem and choose the weights accordingly. If, for instance, the mean and variance have higher priority than the other moments, then their relative weights must be higher than others.

As a result we can easily claim that, although there are timing differences between the other approaches, they are all quick enough to be used for trees of moderate size; for very large trees however, the speed advantage of simulation would become more significant; for instance, a 11-stage tree with branching 2 (1024 scenarios) can be

generated with sequential simulation ($N = 1000$) in 138.81 s, sequential optimization in 2305.90 s and hybrid approach with sequential optimization in 375.12 s.

6. Conclusions

In this paper, three approaches for generating price scenarios for portfolio optimization are investigated. For the simulation-based approach, scenarios are clustered from simulations generated sequentially or in parallel, while in the optimization approach scenarios are obtained by solving a non-linear optimization problem at each node or a large non-linear programming problem. In the hybrid approach of simulation and optimization, simulated prices are fixed in the optimization problem to find the probability of branches in the scenario tree. These procedures are tested and their performances are measured by backtesting in terms of the value of portfolio as well as the CPU time spent on generating a scenario tree.

The results of this paper demonstrate clearly the diminishing returns of efforts to predict the future. Although the overall optimization model is arguably the most theoretically sound way to accurately generate scenario trees, the immense effort expended yielded no perceptible gains in backtesting over the faster heuristics of sequential optimization, simulation, and a hybrid simulation/optimization method.

Acknowledgements

This research was supported by EPSRC grant GR/M41124. We would like to acknowledge the financial support of Fujitsu European Centre for Information Technology (FECIT). The authors are also grateful to Dr Benedict Tanyi for valuable discussions and for providing a low-discrepancy generator. We also acknowledge helpful comments of two anonymous referees.

References

Boender, G., 1997. A hybrid simulation and optimisation scenario model for asset liability management. European Journal of Operational Research 99, 126–135.

Bratley, P., Fox, B.L., 1998. Implementing Sobol's quasirandom sequence generator. Algorithm 659, 88–100.

Carino, D.R., Kent, T., Myers, D.H., Stacy, C., Syvlvanus, M., Turner, A., Watanabe, K., Ziemba, W.T., 1994. The Russell–Yasuda Kasai model: an asset liability model for a Japanese Insurance Company using multi-stage stochastic programming. Interfaces 24, 29–49.

Christofides, A., Tanyi, B., Christofides, S., Whobrey, D., Christofides, N., 1999. The optimal discretisation of probability density functions. Computational Statistics & Data Analysis 31, 475–486.

Dempster, M., Thorlacius, A.E., 1998. Stochastic simulation of internal economic variables and asset returns: the Falcon asset model. Proceedings of the 8th International AFIR Colloquium, London Institute of Actuaries, pp. 29–45.

Dupacova, J., Consigli, G., Wallace, S.W., 2000. Scenarios for multistage stochastic programs. Norwegian Academy of Science and Letters, Technical Report, http://www.iot.ntnu.no/iok-html/users/sww/reports.htm.

Gülpınar, N., Rustem, B., Settergren, R., 2002. Multistage stochastic programming in computational finance. In: E.J. Kontoghiorghes, B. Rustem, S. Siokos (Eds.), Computational Methods in Decision Making, Economics and Finance: Optimization Models. Kluwer Academic, Dordrecht, pp. 33–45.

Gülpınar, N., Rustem, B., Settergren, R., 2003. Multistage stochastic mean-variance portfolio analysis with transaction cost. Forthcoming in: Financial and Economic Networks, 3 (2003), Ed. A. Nagurney.

Hansen, P., Jaumard, B., 1997. Cluster analysis and mathematical programming. Mathematical Programming 79, 191–215.

Hoyland, K., Wallace, S.W., 2001. Generating scenario trees for multistage problems. Management Science 47 (2), 295–307.

Klaassen, P., 1998. Financial asset-pricing theory and stochastic programming models for asset-liability management: a synthesis. Management Science 44, 31–48.

Klaassen, P., 1997. Discretized reality and spurious profits in stochastic programming models for asset liability management. European Journal of Operational Research 101, 374–392.

Kouwenberg, R., 2001. Scenario generation and stochastic programming models for asset liability management. European Journal of Operational Research 134 (2), 279–293.

Kouwenberg, R., Vorst, T., 1998. Dynamic portfolio insurance: a stochastic programming approach. Working Paper.

Mulvey, J.M., 1996. Generating scenarios for the tower perrin investment system. Interfaces 26, 1–13.

Mulvey, J.M., Vladimirou, H., 1992. Stochastic network programming for financial planning problems. Management Science 38 (11), 1642–1664.

NAG Library, 2003. http://www.nag.co.uk, The Numerical Algorithms Group, Oxford.

Nielsen, S.S., Zenios, S.A., 1996. Solving multistage stochastic network programs on massively parallel computers. Mathematical Programming 73, 227–250.

Pflug, G.Ch., 2001. Scenario tree generation for multiperiod financial optimization by optimal discretization. Mathematical Programming 89, 251–271.

Settergren, R., 2000. Manual for foliage. Department of Computing, Imperial College of Technology, Science and Medicine.

Sobol, I.M., 1967. The distribution of points in a cube and the approximate evaluation of integrals. U.S.S.R. Computational Mathematics and Mathematical Physics 7 (4), 86–112.

Taqqu, M.S., Willinger, W., 1987. The analysis of finite security markets using martingales. Advances in Applied Probability 19, 1–25.

Vitoriano, B., Cerisola, S., Ramos, A., 2001. Generating scenario trees for hydro inflows. Working Paper.