Dynamic Portfolio Replication Using Stochastic Programming

M. A. H. Dempster & G. W. P. Thompson*

RiskLab Cambridge
Centre for Financial Research
Judge Institute of Management Studies
University of Cambridge

January 22, 2001

Abstract

In this paper we consider the problem of tracking the value of a 'target' portfolio of European options with a range of maturities within the one year planning horizon using dynamic replicating strategies involving a small subset of the options. In defining a dynamic replicating strategy we only allow rebalancing decision points for the replicating portfolio at the payout dates of the options in the target, but for portfolio compression we measure the tracking error between the value of the two portfolios daily. The target portfolio value has a Bermudan path-dependency at these decision points and it is likely that a carefully chosen dynamic strategy will out-perform simpler static or quasi-static strategies. Here we construct trading strategies by solving appropriate stochastic programming formulations for two principal tracking problems of interest: portfolio compression for risk management calculations and dynamic replicating strategies for simplified replicating portfolios. We demonstrate the superior performance of dynamic strategies relative to more static strategies in a number of numerical tests and in an appendix describe briefly a prototype implementation of the approach in RiskWatch.

*The authors thank Dave Saunders of Algorithmics and the other members of the Centre for Financial Research for their comments and suggestions, in particular, Darren Richards for assistance with the RiskWatch dataset.

1 Introduction

In this paper we construct periodically rebalanced dynamic trading strategies for a portfolio containing a small number of tradable instruments which daily tracks the value of a large 'target' portfolio over a long period of time. A successful solution to this 'tracking' problem is practically useful in a number of financial applications. One such is the investment problem of index-tracking, where the target portfolio consists of the constituent assets of some equity or bond index such as the FTSE-100 or the EMBI+ (see e.g. Worzel, Vassiadou-Zeniou & Zenios, 1994) but here we will address a more complex problem involving nonlinear instruments from a risk management perspective. For risk managers the resolution of the tracking problem can be seen as a way of reducing extensive and complex daily *value at risk* (VaR) calculations for the target portfolio to the simpler task of evaluating the VaR of the tracking portfolio—we refer to this application as *portfolio compression*. In an investment bank the target portfolio would typically be very large. The approach can also be useful as a hedging tool—which extends Black–Scholes delta hedging replication for a single option to a typical large *portfolio* of options or other derivatives of various maturities by finding a *dynamic replication strategy* for the target portfolio. A practical application might involve using a collection of liquid instruments to track the value of a much less liquid target.

Here we analyze instances of the daily tracking problem involving a specified target portfolio of 144 European options on the S&P 500 index of different strikes and maturities *within* a one year planning horizon, using the technique of *dynamic stochastic programming* (DSP). Thus over the horizon considered in our experiments the values of both the target and dynamic tracking portfolios will exhibit a Bermudan path-dependency at rebalance points of the latter. Gondzio, Kouwenberg & Vorst (1998) have studied the use of DSP techniques to implement the Black–Scholes dynamic hedging strategy for a *single* European 0ption—a simple special case of the tracking problem studied here. The present paper is to our knowledge the first application of DSP to a realistically large portfolio containing instruments which mature within the planning horizon.

Over the last few years leading edge risk management practice has evolved from current mark-to-market to one period forward VaR and mark-to-future techniques (Jamshidian & Zhu 1997, Chisti 1999). When such static methodologies are applied over long horizons to target portfolios containing instruments with maturities within the horizon, they take no account of changes in portfolio composition due to instruments maturing—for replication of such portfolios dynamic trading strategies are required which may be found optimally

using dynamic stochastic programming techniques. DSPs are a form of stochastic dynamic programming problem—but solved by mathematical programming techniques—which allow very large numbers of decisions and high dimensional data processes at a smaller number of natural decision points or *stages* (such as option maturity dates) than the fine timestep typically considered in traditional dynamic programming problems. For practical purposes these latter are restricted by the large number of timesteps to only a few state and decision variables—Bellman's curse of dimensionality. DSPs are multi-stage stochastic programming problems for which the term *dynamic* signifies that the underlying uncertainties are being modelled as evolving in *continuous* time and the corresponding scenarios approximating the data process paths are to be simulated with a much finer timestep (here daily) than the (multi-day) interval between decision points. In this paper we demonstrate that the full use of such an approach—which we term *dynamic portfolio replication*—provides a better solution with respect to alternative definitions of tracking error to the daily tracking problem than simpler approaches. We also confirm in our context the general view in the literature (*cf.* Dempster *et al.*, 2000) regarding other DSP problems that the scenario trees required for such an approach reduce tracking error when initial branching is high.

The paper is organized as follows. In Section 2 we describe the DSP approach more fully and review the relevant recent literature. Section 3 discusses in detail the process of constructing dynamic trading strategies using DSP. We describe our particular problem and how the DSP approach is applied in this situation in Section 4. In Section 5 we report a number of numerical tests to compare the daily tracking performance of our dynamic trading strategies with simpler hedges including the portfolio delta hedge. Section 6 concludes and discusses current research directions. A brief description of a prototype implementation of the methods of this paper in Algorithmics software is given in an appendix.

2 The dynamic stochastic programming approach

Assuming underlying continuous time, the DSP approach to the optimization of a continuous state vector stochastic system is as follows (see e.g. Dempster (1980), Birge & Louveaux (1997) and Wets & Ziemba (1999)). First fix a sequence of times at which decisions will be made (with the current moment the first decision time). Now replace the law of the continuous time paths of the continuous state variables with a sampled 'scenario tree'; this has a single node representing the current moment from which a number of branches extend, representing possible discrete time transitions which the state variables may follow

from the current moment to the second decision time. Each of these branches ends at a node which itself has further branches, representing discrete time continuations of the paths of the state variables from the second decision time to the third. This process is repeated until every decision time is represented by a collection of nodes in a tree (see Figure 1). Once this *scenario tree* is constructed, we re-phrase our stochastic optimization problem as an *equivalent deterministic* optimization problem (typically a linear programming problem) by associating a set of decision variables with each node in the tree, and by expressing the objective and constraints of the problem in terms of these decision variables and the values of the state variables on the nodes. The drawback of this approach—which extends both classical decision tree analysis involving a finite number of possible decisions at each node and stochastic dynamic programming as noted above, is that the scenario tree (and hence the size of the optimization problem) grows exponentially as the number of decision times (stages) increases, often necessitating parallel computation and sequential sampling schemes (Dempster & Thompson, 1999).

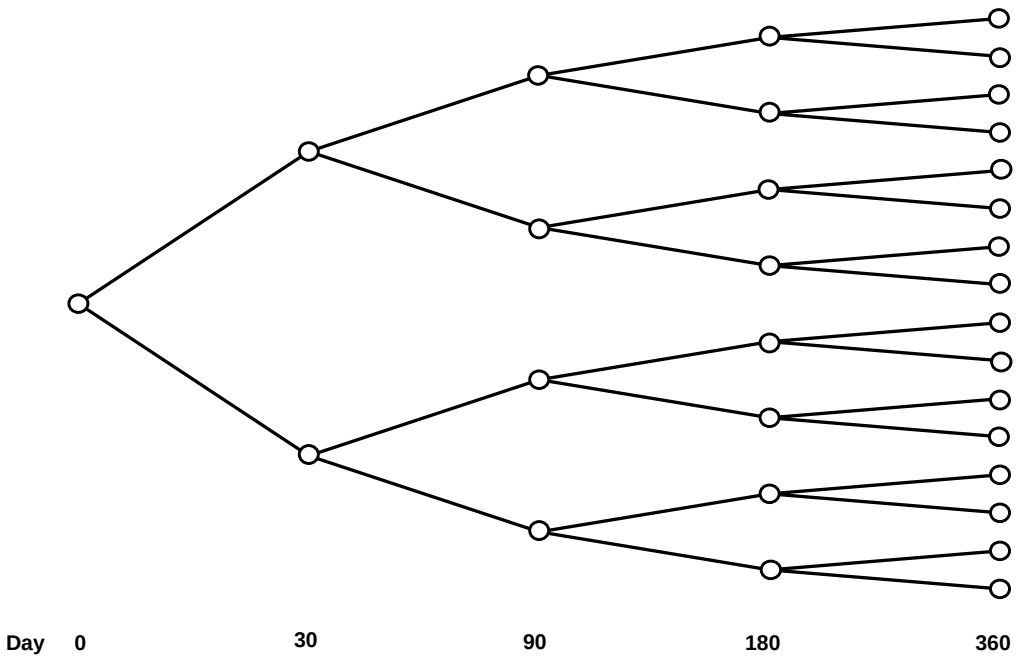

Figure 1: A binary branching scenario tree whose paths are generated by a data process simulator with daily timestep.

Although dating from the 1950s, the practical use of SP in financial problems is still fairly new since the associated optimization problems as noted above can become extremely large. Most stochastic programming problems in the current financial literature are of two types: portfolio insurance and asset liability management. A *portfolio insurance* problem also specifies a 'target portfolio' and asks the analyst to find a dynamic portfolio which when added to the target portfolio reduces the losses of the combined portfolio but retains as much as possible of the profits of the original. *Asset liability management* (ALM) problems consider a sequence of liabilities (which may change over time stochastically) and try to maximize some overall measure of profit after the liabilities have been paid by allocating investment capital between a set of asset classes such as stocks, bonds and cash (see e.g. Mulvey & Ziemba (1998)). In both cases the objective function does not usually penalize over-performance, in contrast to our *tracking problem* where we penalize both under- *and* over-performance relative to the target. This has subtle implications for the construction of the scenario tree.

As in Figure 1, the simplest scenario trees have the same number of branches at each node, and use random sampling to construct the state variable paths associated with each branch. A more sophisticated approach is to try to fit moments of a theoretical conditional distribution of the state variables at decision times (Kouwenberg 1998). In either case, obviously the more scenarios used to construct the stochastic programming problem the closer the approximation will be to the intractable stochastic optimization problem defined by the continuous time and state (vector) data process assumed to underly the situation being modelled. (In general weak convergence of the law defined by the scenario tree to the law of the underlying stochastic data process observed only at decision points can be established under reasonable conditions as the branching at each node tends to infinity.)

A problem can arise if the scenario tree contains an *arbitrage*: a trading opportunity within the deterministic equivalent of the DSP which can be seen to generate a positive profit from zero capital. If the tree and constraints of the problem permit such opportunities, it is likely that the optimization problem will be unbounded unless over-performance is penalized. Even if infinite profits are prohibited by other constraints in the model such as position limits (as is usually the case in ALM problems for example), it can be expected that the solution of a problem allowing arbitrages will perform poorly, although in practice this is intimately related to the methods used and to the number of scenarios generated. Klaassen (1997) discusses these issues in the context of ALM, and Kouwenberg & Vorst (1998) in portfolio insurance. A feature common to most of the scenario trees in the literature is that

nodes in early time periods have a higher number of branches than in later time periods. This is usually due to the way the solution of the DSP problem is expected to be used in practice—initial decisions are implemented and the whole problem is rolled forward to the next decision point.

A final aspect of the DSP approach on which we should comment is the 'testing' of the quality of the solution to the optimization problem in the context of the original continuous time continuous state problem using simulated paths of the state variables. As noted above it is frequently assumed that after the optimization problem is solved, the optimal decision variables at the node representing the current moment will be implemented for some period of time, after which a new scenario tree will be constructed, and a new optimization problem solved. This process is repeated for as long as is required. A suitable testing procedure given this approach is to construct a large number of test scenarios involving the underlying stochastic processes (interest rates, stock prices, etc.), and for each test scenario and decision point, to construct a scenario tree and solve an optimization problem. Golub *et al.* (1997) and Fleten, Høyland & Wallace (1998) perform this type of test. It has the drawback that the scenario trees cannot be very large or the optimization problems involved will take too long to solve. A typical problem using this scheme might thus have weekly decisions, and generate scenario trees extending over three, four or five weeks. Gondzio & Kouwenberg (1999) give an example of an ALM problem where the scenario tree has 6 decision times and a branching of 13. Solving a single instance of this problem even with state-of-the-art software and hardware takes over 3 hours and improvements to such times are only possible by utilizing sophisticated parallel algorithms and hardware techniques. Another easier to implement testing procedure is to use each test scenario to construct a path through the scenario tree which, at each stage chooses the branch which is 'closest' to the path of the test scenario over the appropriate time interval and then uses the optimal decision variables at the nodes in the scenario tree, together with the genuine statespace variables from the test scenario (Dempster & Ireland, 1988). This is the approach we will adopt for the portfolio compression application which needs fast computation. Gondzio, Kouwenberg & Vorst (1998) use a similar approach, choosing test scenarios from a fine-grained tree which was previously 'aggregated' to form the scenario tree used in the optimization problem. Another approach is to generate test scenarios by picking a random path through a previously generated scenario tree or lattice, but this is likely to give rather optimistic answers, especially for trees if later time periods have very few branches or in the case of arbitrage-free lattices, since this property is destroyed by sampling. Gondzio *et al.* (1998) describe one way of

constructing smaller arbitrage-free trees starting from an initial (but possibly very large) arbitrage-free tree.

3 Constructing dynamic replicating strategies using DSP

In this section we describe in detail how we apply the DSP approach to tracking problems. Recall that a tracking problem defines a target portfolio, whose value is to be 'tracked' and asks us to find a self-financing tracking strategy using a prescribed set of trading instruments. We assume that the value of the target and tradables at time t are functions of the path over $[0,T]$ of some observable *state process* $\{\boldsymbol{S}(t) \in \mathcal{S} : 0 \leq t \leq T\}$, where boldface is used to denote random entities throughout.

Tracking problems

As mentioned above, we will assume a number of discrete *decision points* $0 = t(1) \leq t(2) \leq \cdots \leq t(T) \leq T$ in time and consider the data of our problem to be observations of the state process at these discrete time points which are for simplicity represented by integer labels $t = 1, 2, \ldots, T$.

We consider a multistage version of a static stochastic optimization problem whose deterministic equivalent was studied in Dembo & Rosen (1999). Given a discrete time process $\{\boldsymbol{\tau}_t \in \mathbb{R} : t = 1, \ldots, T\}$ of *values* of a *target portfolio* the stochastic dynamic *tracking problem* we study is given by

$$\inf_{\boldsymbol{x}} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\boldsymbol{y}_t^+ + \boldsymbol{y}_t^-\right] \tag{1}$$

such that

$$p_1' x_1 \leq p_1' x_0 \tag{2}$$

$$\boldsymbol{p}_t'[\boldsymbol{x}_t - \boldsymbol{x}_{t-1}] = 0 \qquad \text{a.s. } t = 2, \ldots, T \tag{3}$$

$$\boldsymbol{p}_t' \boldsymbol{x}_t - \boldsymbol{y}_t^+ + \boldsymbol{y}_t^- = \boldsymbol{\tau}_t \qquad \text{a.s. } t = 1, \ldots, T \tag{4}$$

$$\boldsymbol{x}_T - \boldsymbol{x}_{T-1} = 0 \qquad \text{a.s.} \tag{5}$$

$$\boldsymbol{y}_t^+ \geq 0, \quad \boldsymbol{y}_t^- \geq 0 \qquad \text{a.s. } t = 1, \ldots, T. \tag{6}$$

Here the fundamental discrete time *decision process* $\boldsymbol{x}$ is a set of portfolio positions (long and short) in the securities of the problem and the corresponding *price process* $\boldsymbol{p}$ is used to value

these positions at each decision point. At time point 1, prices p_1 and initial *endowment* $p_1' x_0$ are known, and an initial position x_1 must be taken. Subsequently all decisions are state dependent and hence stochastic; so that the tracking portfolio will have an *almost surely* (a.s.)—i.e. with probability one—upside ($\boldsymbol{y}_t^+ \geq 0$) or downside ($\boldsymbol{y}_t^- \geq 0$) *tracking error* (6) in value terms. The objective (1) is to minimize the *average absolute* (or L_1 norm) tracking error subject to *budget* (2), *self-financing* (3) and *tracking error* (4) constraints which must hold a.s. The stochastic tracking portfolio $\boldsymbol{x}_{T-1}$ set at the penultimate decision point must be held (5) over the period to the horizon at T. Here $\mathbb{E}$ denotes expectation and prime denotes transpose. Many variations on the problem are possible and several will be used in this paper.

First if (6) is replaced by

$$\rho \geq \boldsymbol{y}_t^+ \geq 0, \quad \rho \geq \boldsymbol{y}_t^- \geq 0 \quad \text{a.s. } t = 1, \ldots, T \tag{7}$$

and the objective (1) is replaced by

$$\inf_{\boldsymbol{x}} \rho \tag{8}$$

we obtain the problem (SPLINF) whose solution minimizes the *worst case* (or L_∞ norm) tracking error a.s. over both decision points and scenarios.

For either (SPL1) or (SPLINF) applied to the portfolio compression problem—since all positions in the tracking portfolio are *virtual*, i.e. for computational purposes only—we ignore transaction costs and allow an arbitrarily large initial endowment $p_1' x_0$. In other applications however an optimal dynamic replicating strategy must be implemented and hence proportional transaction costs are incurred. This results in a slightly more complex model involving a real budget constraint and *buy* and *sell* decisions $\boldsymbol{x}_t^+$ and $\boldsymbol{x}_t^-$ respectively. The L_1 variant of this model becomes

$$\text{(SPL1}') \qquad \inf_{\boldsymbol{x}, \boldsymbol{x}^+, \boldsymbol{x}^-} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\boldsymbol{y}_t^+ + \boldsymbol{y}_t^-\right] \tag{9}$$

such that

$$p_1' x_1 \leq p_1' x_0 \tag{10}$$

$$\boldsymbol{x}_{t-1} + \boldsymbol{x}_t^+ - \boldsymbol{x}_t^- = \boldsymbol{x}_t \qquad \text{a.s. } t = 1, \ldots, T \tag{11}$$

$$\boldsymbol{p}_t'(\boldsymbol{x}_t - \boldsymbol{x}_{t-1}) + \boldsymbol{p}_t' K(\boldsymbol{x}_t^+ + \boldsymbol{x}_t^-) = 0 \qquad \text{a.s. } t = 2, \ldots, T \tag{12}$$

$$\boldsymbol{p}_t' \boldsymbol{x}_t - \boldsymbol{y}_t^+ + \boldsymbol{y}_t^- = \boldsymbol{\tau}_t \qquad \text{a.s. } t = 1, \ldots, T \tag{13}$$

$$\boldsymbol{x}_t^+ \geq 0, \quad \boldsymbol{x}_t^- \geq 0, \quad \boldsymbol{y}_t^+ \geq 0, \quad \boldsymbol{y}_t^- \geq 0 \qquad \text{a.s. } t = 1, \ldots, T. \tag{14}$$

Here the new constraint (11) expresses position *inventory balance* over time, and the self-financing constraint (12) now involves a diagonal matrix $K = \mathrm{diag}(\kappa_1, \ldots, \kappa_I)$ of (two-way) proportional *transaction costs*.

To construct our dynamic replication strategy, we will approximate the law of the process $\boldsymbol{S}$ with a tree, solve an appropriate optimization problem on this tree, and then interpret the solution as a dynamic trading strategy for the tracking portfolio.

Scenario trees

To define the optimization problem used to construct our dynamic replication strategy, we first make precise the notion of a 'scenario tree' and how it approximates the law of the state process $\boldsymbol{S}$.

A *scenario tree* is a tree with nodes in a set $\mathcal{N}$, with one node, $r \in \mathcal{N}$ designated the *root node*. For $n \in \mathcal{N}, n \neq r$ we let $P(n)$ denote the *predecessor* of n: the unique node adjacent to n on the unique path from n to r, and refer to those nodes with a common predecessor as the *successors* of their predecessor. We define $P(r) = r$.

With each node $n \in \mathcal{N}$ we assume that there is given a (*decision*) *time* $t(n)$ and a *state* $s(n) \in \mathcal{S}$ satisfying: $t(r) = 1$ (or the current real time 0), $s(r)$ is the observed current state and $t(P(n)) < t(n)$ for all $n \neq r$. We will also assume that if $P(n) = P(n')$ then $t(n) = t(n')$. For each node n, we link $s(P(n))$ and $s(n)$ with an arc representing a path segment of a suitable discrete time approximation of $\boldsymbol{S}$ over the number of time points between the decision times corresponding to $P(n)$ and n respectively (see Figure 1). Thus we can associate a path of arcs $\{s(u) : u = t(1), \ldots, t(T)\}$ with each sequence of nodes $n_1, n_2, n_3, \ldots, n_T$ for which (with a slight abuse of notation) $t(1) = t(n_1) < t(n_2) < \cdots < t(n_T) = T$ and $P(n_{i+1}) = n_i$ for $i = 1, 2, \ldots, T-1$. In order to implement holding tracking portfolios over the period between the last two decision points we must distinguish *leaf nodes* $\ell \in \mathcal{L} \subset \mathcal{N}$ with $t(\ell) = T$. We assume that we are given a set of strictly positive real numbers $\{\pi(n) : n \in \mathcal{N}\}$ such that $\pi(r) = 1$ and if n has successors, then $\pi(n) = \sum_{n' : n = P(n')} \pi(n')$ for all $n \in \mathcal{N}$. In this situation the conditional probability of $s(n')$ being realized after $s(n)$ is $\pi(n')/\pi(n)$.

To show how to interpret a scenario tree as a discrete time stochastic process approximating the law of $\boldsymbol{S}$, it suffices to show how to generate a single sample path. Starting at the root node, select one of its successors n at random with conditional probability proportional to $\pi(n)$; then the arc from $s(r)$ to $s(n)$ represents a sample path for $\boldsymbol{S}$ over $[t(r), t(n)]$. Now

move to node n and select one of its successors $\tilde{n}$ with conditional probability proportional to $\pi(\tilde{n})$, and so on.

By identifying a node n with the path from the root node to n, and thence with a path $\{s(u) : u = t(1), \ldots, t(T)\}$ in $\mathcal{S}$, we can speak of the *value* of the target and the *price* of the tradable instruments at n. We denote these quantities $\tau(n)$ and $p(n)$ respectively, where $p(n) \in \mathbb{R}^I$ is a vector giving the prices of the I tradable instruments. We denote by T the *depth* of the tree: the number of nodes on a path of maximal length starting at the root node.

The optimization problem

We are now in a position to modify the deterministic equivalent of the static optimization problem introduced in Dembo & Rosen (1999) to state a recursive version of the dynamic deterministic equivalent of (1) applicable to our tree setting. Using the notation above, we consider the problem:

$$\text{(L1)} \qquad \underset{x(n):n\in\mathcal{N}}{\text{minimize}} \quad \frac{1}{T}\sum_{n\in\mathcal{N}} \pi(n)[y^+(n) + y^-(n)] \tag{15}$$

subject to, for all $n \in \mathcal{N}$, $\ell \in \mathcal{L}$,

$$\begin{gathered} p(r)'x(r) \le M \\ p(n)'[x(n) - x(P(n))] = 0 \\ p(n)'x(n) - y^+(n) + y^-(n) = \tau(n) \\ x(\ell) - x(P(\ell)) = 0 \\ y^+(n) \ge 0, \quad y^-(n) \ge 0, \end{gathered}$$

which is the problem of minimizing the expected total tracking error subject to an initial budget of M, and also the worst-case version:

$$\text{(LINF)} \qquad \underset{x(n):n\in\mathcal{N}}{\text{minimize}} \quad \rho \tag{16}$$

subject to

$$
\begin{aligned}
p(r)'x(r) &\leq M \\
p(n)'(x(n) - x(P(n))) &= 0 \\
p(n)'x(n) - y^+(n) + y^-(n) &= \tau(n) \\
y^+(n) \leq \rho, \quad y^-(n) &\leq \rho \\
x(\ell) - x(P(\ell)) &= 0 \\
y^+(n) \geq 0, \quad y^-(n) &\geq 0.
\end{aligned}
$$

In both cases, the variables $x(n) \in \mathbb{R}^N$ are interpreted as the holdings in the N tradable instruments to be used if the state process follows the path from the root node to n. The second constraint is the self-financing constraint at node n and the third defines $y^{\pm}(n)$ as the upside and downside tracking errors at n. Note that since $P(r) = r$, these constraints also make sense when $n = r$. Observe that at leaf nodes without successors the variables $x(\ell)$ must be identical to those at their predecessors $x(P(\ell))$.

We impose additional constraints in both forms of the problem. First, we restrict the holding in each tradable instrument to the interval $[-100{,}000{,}000, 100{,}000{,}000]$ (to ensure that the solution is finite), and secondly if any tradable instrument has value within 5×10^{-5} of zero at node n, or at every successor of n, then we impose $x(n) = 0$. This second condition stops the optimal strategy from holding large positions which appear to have low or zero tracking error at the nodes of the tree on the paths corresponding to the law of implied by the scenario tree, but which have very high daily tracking error under the discrete time approximation to the true law in between decision points. We also impose proportional transaction costs in the deterministic equivalent of (9), which we now state for our particular problem.

The transaction cost model

The deterministic equivalent form of (9)-(14) to be used in our particular problem has objective the same as that of the previous problem (L1). Now however we must distinguish between the different components of the decision variables x: the vector $x(n)$ has components denominating the holdings in the underlying, cash and the options respectively. The values of the tradable instruments at node n are given by the vector $p(n)$. T denotes the 'depth' of the scenario tree. $P(n)$ is the predecessor node of node n; r denotes the root

node. There are two transaction costs: κ_{snp} is the cost associated with purchases/sales of the index, while κ_{opt} applies to options transactions. The initial portfolio holdings are given by the vector h. Thus we have

$$\text{(L1}') \qquad \underset{x(n):n\in\mathcal{N}}{\text{minimize}} \quad \frac{1}{T}\sum_{n\in\mathcal{N}} \pi(n)[y^+(n)+y^-(n)] \tag{17}$$

subject to, for all $n \in \mathcal{N}$, $n \neq r$

$$h + x^+(r) - x^-(r) = x(r)$$
$$x(P(n)) + x^+(n) - x^-(n) = x(n)$$
$$p(r)'(x(r)-h) = -\kappa_{\text{snp}} p_0(r)(x_0^+(r)+x_0^-(r)) - \kappa_{\text{opt}} \sum_{i=2}^{5} p_i(r)(x_i^+(r)+x_i^-(r))$$
$$p(n)'(x(n)-x(P(n))) = -\kappa_{\text{snp}} p_0(n)(x_0^+(n)+x_0^-(n)) - \kappa_{\text{opt}} \sum_{i=2}^{5} p_i(n)(x_i^+(n)+x_i^-(n))$$
$$p(n)'x(n) - y^+(n) + y^-(n) = \tau(n)$$
$$x^+(n) \geq 0, \quad x^-(n) \geq 0, \quad y^+(n) \geq 0, \quad y^-(n) \geq 0.$$

We also impose the constraint that options cannot be held once they have expired.

Turning the solution into a trading strategy

Once one of the above optimization problems has been solved, the problem arises of interpreting the solution as a replicating trading strategy for the tracking portfolio. We will evaluate the tracking error of our replicating portfolio by simulating many independent realizations of the underlying (daily) discrete time data process and then valuing both target and tracking portfolios at each timestep. We can use the portfolio associated with the root node of an optimal solution up to the first branch time, but then we must decide how to re-balance. If the value of the state process at that time were exactly equal to the value of this process at one of the successors of the root node, we would just use the optimal portfolio associated with that node, but unfortunately this is unlikely to be the case.

In the context of dynamic portfolio replication, we will re-solve the stochastic programming problem by sampling a new scenario tree with the actual value of the process at the second decision point assigned to the root, and then take the calculated optimal portfolio

at the new root node (see Figure 2). This would be far too time consuming in portfolio compression applications.

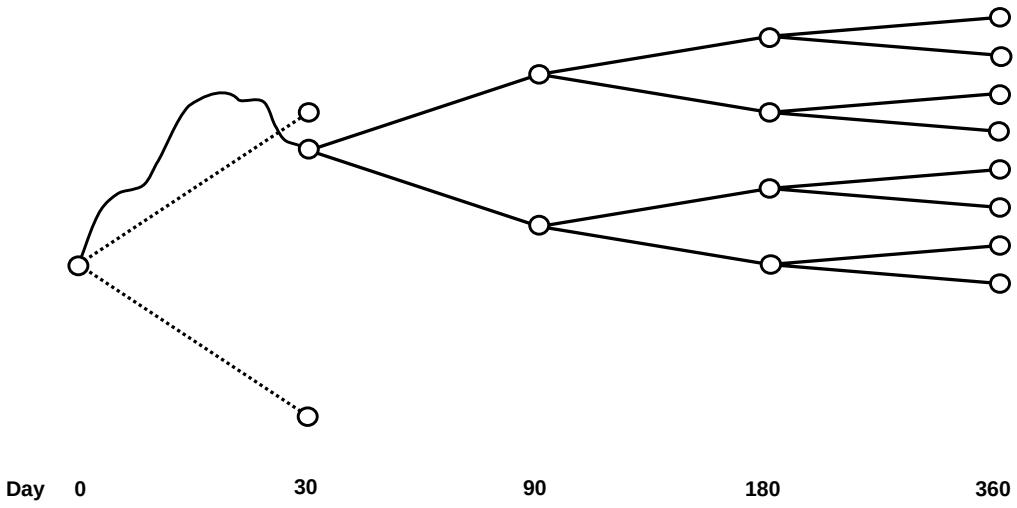

Figure 2: Graphical illustration of the full dynamic replication strategy showing part of the initial tree and the tree used at the second decision point.

In this latter case we will use a simple procedure based on having a metric $m(\cdot,\cdot)$ on $\mathcal{S}$ for measuring the distance from a simulated realization of $\boldsymbol{S}$ to the nearest node. For example, if the state distribution of the process $\boldsymbol{S}$ at t were Gaussian with covariance matrix V, then the choice of the *Malanobis metric* $m(s,s') = (s-s')^T V^{-1}(s-s')$ would be very natural. Denoting the solution to the optimization problem by x^*, the *portfolio compression trading strategy* is defined as follows (see Figure 3): let $n = r$ initially, let t' be the (common) time at the successors of node n, and use the portfolio $x^*(n)$ over $[t(n), t']$. At time t' consider each of the successors of n which themselves have successors and find the successor $\tilde{n}$ which minimizes $m(S(t'), s(\tilde{n}))$ where $S(t')$ is the observed value of the state process at time t', and implement $x^*(\tilde{n})$. Then replace n with $\tilde{n}$ and repeat from the start. Once a node n is reached which has no successors, use the portfolio $x^*(n)$ for all future time. If this strategy turns out to be non-self-financing at any times of decision points,

any 'slack' is to be absorbed by investing/borrowing using one of the tradables, chosen arbitrarily (preferably one with a low volatility). Thus this method is based on constructing a dynamic trading strategy from the solution of a single *fixed* DSP with sufficiently many scenarios (and hence tree nodes) to give finely resolved number of alternative portfolios at each decision point.

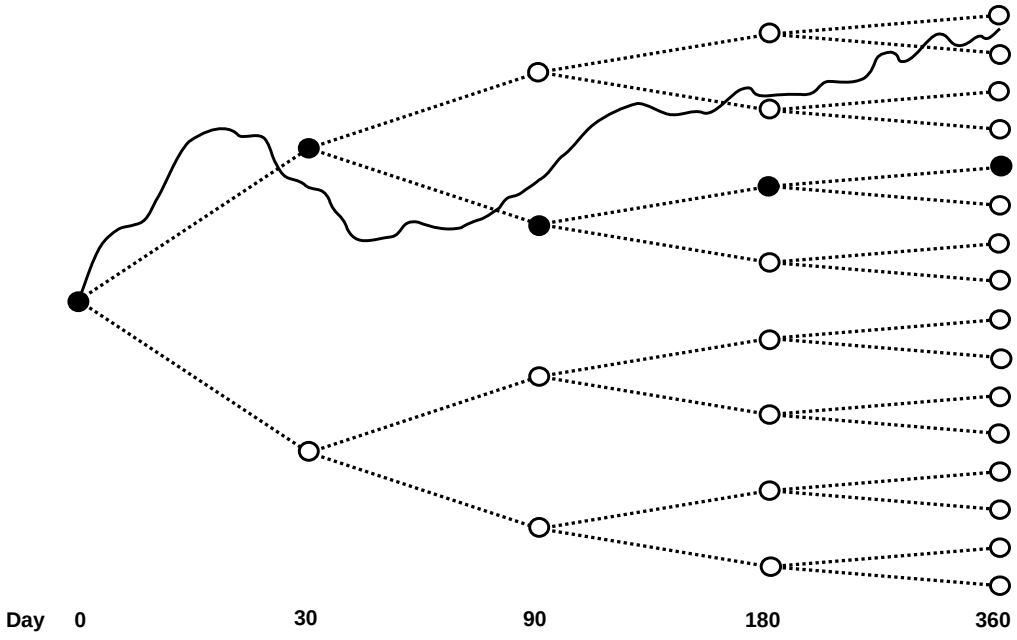

Figure 3: Graphical illustration of the compression trading strategy (black nodes).

Other methods are also worth exploring, such as interpolating between optimal portfolios on paths through the scenario tree which are close to the observed path or fitting a parameterized trading rule to the optimal solution, but we will not consider them here.

4 The options problem

We consider a portfolio of standard European options on a stock index $\boldsymbol{I}_t$ with a wide range of strikes and maturities—each of which is taken as a decision point—and try to find a dynamic trading strategy which uses cash, the underlying index and possibly a small

subset of the options. The index is assumed to follow the SDE defining *Geometric Brownian motion*, *viz.*

$$d\boldsymbol{I}_t = \boldsymbol{I}_t(\sigma\, d\boldsymbol{B}_t + \mu\, dt), \tag{18}$$

where $\boldsymbol{B}_t$ is a standard Brownian motion, $\sigma \approx 20\%$, $\mu \approx 10\%$ and $I_0 = \$1275$. The risk-free interest rate is taken to be 5%.

The target

The target consists of 0.299527 units of the underlying index, an initial $-\$8.74774$ in cash, and a large number (144) of call options (see Table 1).

The target was constructed by including, for each strike and maturity in Table 1, either a put option or a call option (choosing one or the other at random with equal probability) and then selecting the size of the position from a uniform distribution on $[-1, 1]$. The inclusion of puts is accomplished using put-call parity and leads to the non-zero holdings in the underlying index and the cash account.

Figure 4 shows the highly nonlinear payouts of this portfolio as its options mature.

The tradable instruments

We will assume that the set of tradable instruments consists of the underlying index, cash, and perhaps also a single option for each of the four maturities. The strikes of these options are chosen to be close to the expected price of the index at that maturity, and are given in Table 2. We set the proportional cost for index transactions κ_{snp} to be 1% and the corresponding cost for options κ_{opt} transactions to be 2.5%.

The scenario tree

To generate the scenario tree we will use either *random sampling* of the discretization of (18) with daily timestep or the following simple *discretization procedure* at decision points: if $\tilde{n}$ is the kth successor of node n, $k = 0, 1, \ldots, K-1$, set

$$I(\tilde{n}) := I(n) \exp\bigl(\Phi^{-1}(\tfrac{1+2k}{2K})\sqrt{\Delta t}\,\sigma + (\mu - \tfrac{1}{2}\sigma^2)\Delta t\bigr), \tag{19}$$

where $\Delta t := t(\tilde{n}) - t(n)$ is the time difference between n and its successors, Φ is the normal distribution function and $I(n)$ denotes the value of the index at node n. The probability of a particular successor $\tilde{n}$ of n is taken to be $1/K$. The aim here is to set up a uniform grid

	Maturity (days)			
Strike	30	90	180	360
750	0.499804	0.15194	0.998693	0.113471
950	0.74093	−0.700248	0.391195	−0.24525
1150	0.571599	0.302505	0.972411	0.579243
1165	−0.262467	0.909999	0.309761	0.648487
1180	0.49019	0.571087	0.36354	0.456401
1195	−0.292544	0.142507	−0.336683	−0.475631
1210	−0.479556	0.925102	−0.681272	−0.0208652
1225	0.55358	−0.126283	0.205513	0.699144
1240	0.151576	0.846815	0.962714	−0.276082
1255	−0.833992	0.53772	−0.0719259	0.209064
1270	−0.780106	0.181939	0.689277	−0.118359
1285	−0.218627	0.284286	0.473013	−0.567698
1300	0.918133	−0.861963	−0.0463901	−0.387226
1315	−0.673621	0.548627	−0.00481466	−0.820417
1330	−0.479278	−0.757467	−0.111879	0.867178
1345	−0.128155	−0.850984	−0.336303	0.237092
1360	−0.744766	−0.270088	0.16514	−0.773939
1375	0.881284	−0.610828	0.349776	−0.85157
1390	−0.691578	−0.676415	−0.528584	−0.531592
1405	−0.690527	0.101874	0.916708	0.869105
1420	0.753697	0.0554727	−0.85114	0.148518
1435	−0.472188	−0.572643	−0.23437	−0.758292
1450	0.540183	−0.997261	0.433229	0.997215
1465	0.542085	−0.045124	−0.957911	0.727558
1480	−0.562721	0.524317	0.257921	0.459888
1495	−0.176574	−0.970444	0.0958814	−0.0225336
1510	0.859913	0.689543	−0.214078	−0.775679
1525	0.374881	−0.817983	−0.383831	0.398134
1540	0.216602	0.577868	0.222183	0.765986
1555	0.265243	−0.801022	−0.678699	−0.579844
1570	0.264488	0.494393	0.60596	0.37497
1585	0.921523	0.0242848	0.435875	−0.471075
1600	−0.106634	0.923895	−0.716491	−0.0146881
1800	0.905168	−0.0522193	−0.514874	−0.10789
2000	−0.203584	0.30069	0.742959	−0.406423
2200	0.910171	0.171869	0.508814	0.499529

Table 1: The static target portfolio's option holdings.

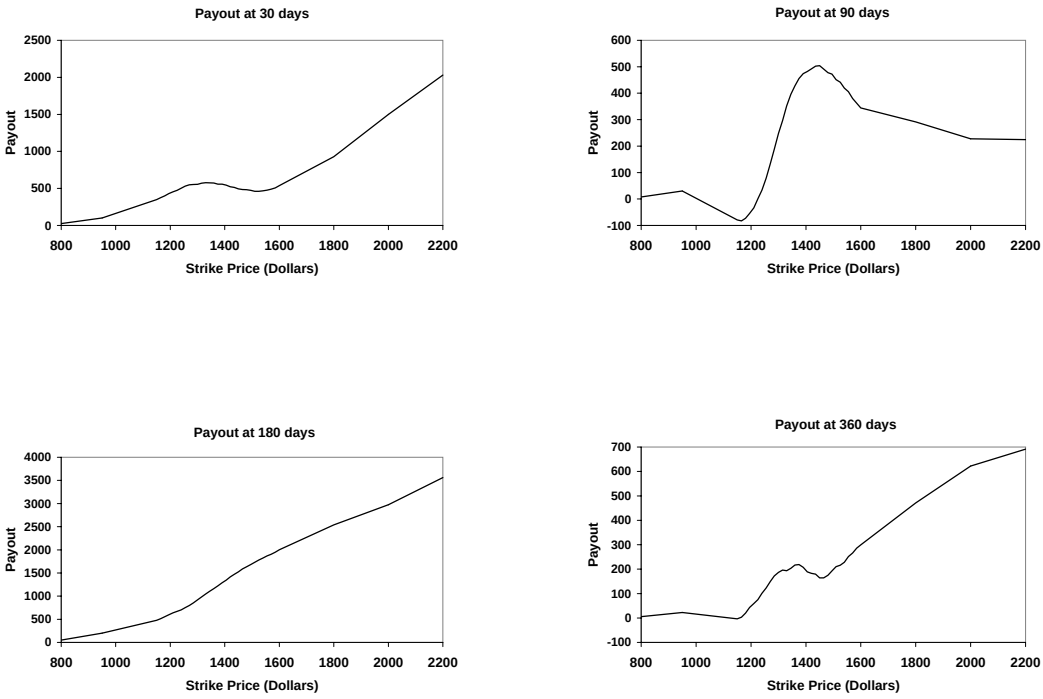

Figure 4: Target payouts at the four maturities.

Maturity	30	90	180	360
Strike	1285	1300	1330	1405

Table 2: The options available to replicating portfolios.

of $2K$ levels and use the inverse Gaussian distribution function to create the corresponding grid for the logarithm of the index values. In both cases, the depth of the tree — i.e. the number of decision points — is T.

5 Numerical Experiments

We consider two types of experiment illustrating our two intended applications: portfolio compression, where we replicate the target with a trading strategy which can be simulated very quickly, and dynamic portfolio replication, in which we model future trading decisions and optimize the current trading decision taking the effects of future decisions into account.

In both cases we will consider the effects of allowing dynamic strategies to use just cash and the underlying index as tradable instruments versus allowing additionally trading in the four options described in Section 4.

As a 'benchmark' we will compare our dynamic replication strategies to a static simple strategy: which uses a simple scenario tree branching only after the first decision point and optimizes a *single* trading decision which is used at every node of the tree (see Figure 5). A second alternative to the full dynamic replication strategy is a strategy which is allowed to rebalance at each decision point of a test scenario according to a static simple strategy, but *assumes* at each such point that the implemented portfolio of the static simple solution will not subsequently change. We term such a strategy *quasi-static* (see Figure 6).

A further natural benchmark is to construct on each test scenario the portfolio *delta hedge* which trades only in cash and the underlying to rebalance at each decision point and holds the new portfolio to the next decision point.

Portfolio compression

Our first experiments compress the target by solving the first two optimization problems presented in Section 3, with either L_1 or L_∞ objectives, and interpreting the solutions as *dynamic* trading strategies using the nearest node technique described in Section 3. For this purpose we ignore the budget constraint and could even ignore the self-financing constraint except that we would like to know whether we can use the nearest node method of Section 3 to construct a dynamic trading strategy so we must impose it. We consider two alternative scenario trees for the optimization problems: one with a branching of five at each node (625 scenarios), and one with a branching of ten (10,000 scenarios). To generate the value for the

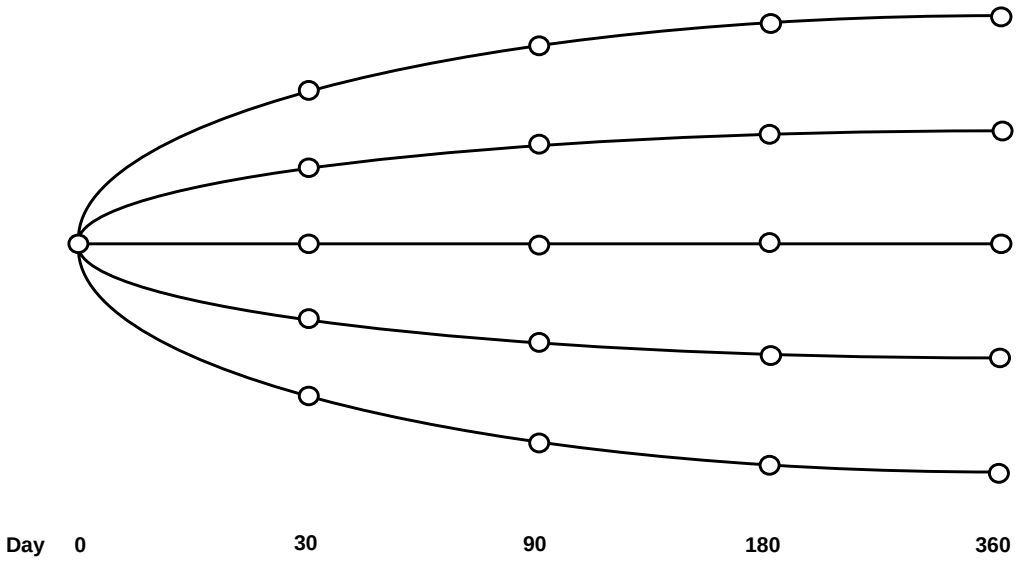

Figure 5: Graphical illustration of the static simple trading strategy in which a common portfolio is used at all nodes of the scenario tree.

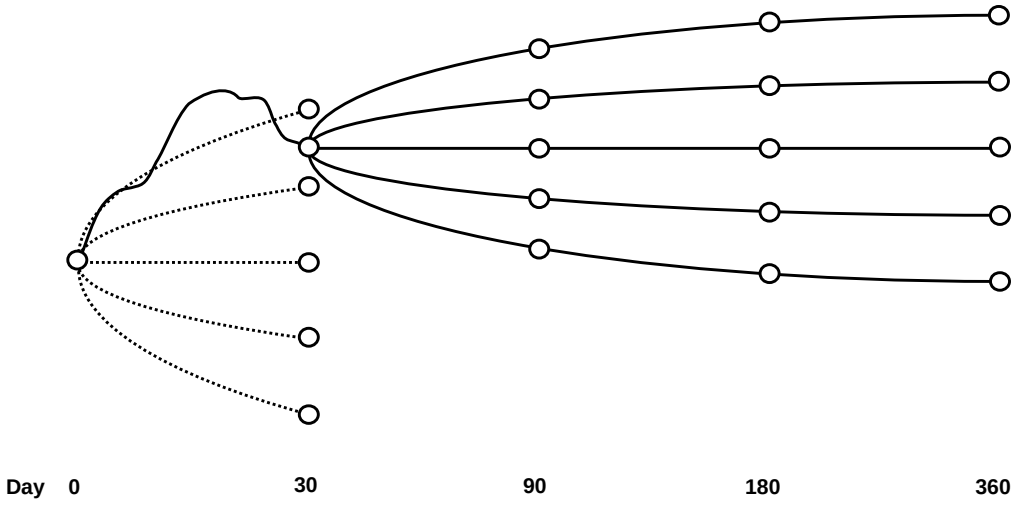

Figure 6: Graphical illustration of the quasi-static simple trading strategy which uses a static simple trading strategy from the realized simulation path at the second and subsequent decision points.

index at nodes in the scenario tree we use the inverse distribution function discretization procedure described in Section 4.

We also consider two *static* simple strategies obtained by minimizing the expected average absolute tracking error (the objective of (L1)), and by minimizing the worst absolute tracking error (the objective of (LINF)) over a set of 1000 scenarios for the index path, sampled at time 0 and the four maturity dates in the target.

Our experimental design with each strategy type is as noted above to solve the DSP versions of the tracking problem for an appropriately generated scenario tree whose nodes are at option maturity dates and then test the quality of these decisions with a large number of randomly generated *test scenarios* simulating the index with a daily timestep up to the last option payout date at 360 days ($T = 5$). The numerical results evaluating the tracking performance of the various approaches were obtained by running 10,000 test simulations of the path of the index and considering the tracking error between the portfolios *each day* from the initial time 360 days. For the evaluation of the daily tracking error between the two portfolios for of all alternative trading strategies for the tracking portfolio at daily timesteps *between* option maturity dates we use the Black–Scholes formula for all options in the portfolio with the true volatility utilised for the underlying index simulation. The average absolute tracking error for a single index simulation path is taken to be the average of the daily absolute tracking errors. The *expected* average absolute daily tracking error is the average over the 10,000 simulated index paths of the individual path average absolute tracking errors. We also record the worst (absolute) tracking error observed, and the minimum and maximum values attained by the target, the compressed portfolios and the simple strategies.

A summary of the numerical tracking error evaluation results for portfolio compression is presented in Tables 3 to 6. The 'objective' column shows the optimal value for the initial optimization problem. This value is usually very different from the expected average absolute daily tracking error obtained when the solution is implemented as a trading strategy, since the simulated paths of the index are highly unlikely to pass through the nodes in the scenario tree. All optimization problem CPU total solution times shown are for an Athlon 650 Mhz PC with 256 MB memory.

Note that allowing options trading improves the simple strategies but is detrimental to the dynamic strategies when the nearest node strategy is used. This suggests that using the nearest node technique of Section 3 to interpret the solution of the initial optimization problem as a trading strategy is questionable with non-linear instruments unless the scenario

Method	Objective	Expected average absolute daily tracking error	Optimization CPU time (s)
static simple, L1	209.8	301.0 (1.52)	13
static simple, LINF	1542.0	500.0 (2.60)	17
5-branch dynamic, L1	49.8	97.9 (0.96)	1
5-branch dynamic, LINF	218.4	112.0 (0.89)	1
10-branch dynamic, L1	57.4	90.5 (0.91)	73
10-branch dynamic, LINF	301.0	117.0 (0.72)	126

Table 3: Portfolio compression evaluation results: strategies using only cash and the index as tradable instruments. One standard error in the estimate of the expected average tracking error is indicated in brackets.

tree is very large (well in excess of the 10,000 scenarios used here to give a much more finely resolved strategy).

The dynamic strategies using just cash and the underlying index not suprisingly have a significantly lower expected average absolute daily tracking error than any of the other strategies except the delta hedge, particularly for the versions minimizing the (L1) objective. The frictionless delta hedge strategy has the next best performance to such a 10-branch dynamic L_1 strategy but takes over twice as long to evaluate on a test scenario due to Black–Scholes option evaluations at decision points (Table 5). Thus for portfolio compression used in Monte-Carlo VaR calculations it would be significantly inferior.

Figures 7 and 8, showing respectively the density functions of the average absolute daily tracking error and the worst tracking error for the 10-branch dynamic and simple strategies with each objective, demonstrate that although the dynamic worst case (L_∞) trading strategy better controls the upper tails of both the average and worst case absolute daily tracking error distributions, the average (L_1) trading strategy appears better overall with respect to both criteria.

It is also of interest to see if the compressed portfolios still track the target in extreme market conditions. Figure 9 shows the distribution functions of the *minimum* value of the target and of the four tracking strategies as before, while Figure 10 shows the distribution functions of the corresponding *maxima*. These figures show that the dynamic strategies are also better at estimating both the lower tail of the minimum value of the target and the

Method	Objective	Expected average absolute daily tracking error	Optimization CPU time (s)
static simple, L1	139.4	187.0 (1.5)	98
static simple, LINF	774.0	330.0 (1.0)	59
5-branch dynamic, L1	8.2	387.0 (22.2)	1
5-branch dynamic, LINF	62.3	441.0 (22.4)	2
10-branch dynamic, L1	16.1	209.0 (22.2)	117
10-branch dynamic, LINF	112.0	237.6 (22.4)	1729

Table 4: Portfolio compression evaluation results: using cash, the index and four options as tradable instruments. One standard error in the estimate of the expected average tracking error is indicated in brackets.

Method	Expected average absolute daily tracking error	Expected worst absolute daily tracking error	Test scenario CPU times (s)
Static simple, L1	187.0	494.0	0.0228
Static simple, LINF	330.0	496.6	0.0228
Dynamic L1	90.5	200.3	0.0004
Dynamic LINF	117.0	218.3	0.0004
Delta hedge	93.3	170.2	0.0009

Table 5: Portfolio compression results: A comparision of the best static and dynamic hedges with delta hedging

Method	Cash	Underlying	30-day	60-day	180-day	360-day
static simple, L1	−3172.5	4.21	0.73	1.78	3.37	−3.51
static simple, LINF	518.1	1.00	1.36	1.65	3.36	−0.13
10-branch dynamic, L1	−5960.9	6.43	N/A	N/A	N/A	N/A
10-branch dynamic, LINF	−5193.8	5.78	N/A	N/A	N/A	N/A

Table 6: Portfolio compression results: Optimal initial portfolio holdings using only cash and the underlying in the dynamic strategies. Reported are the best static hedges (i.e. those from Table 4) and the best dynamic hedges (i.e. those from Table 3).

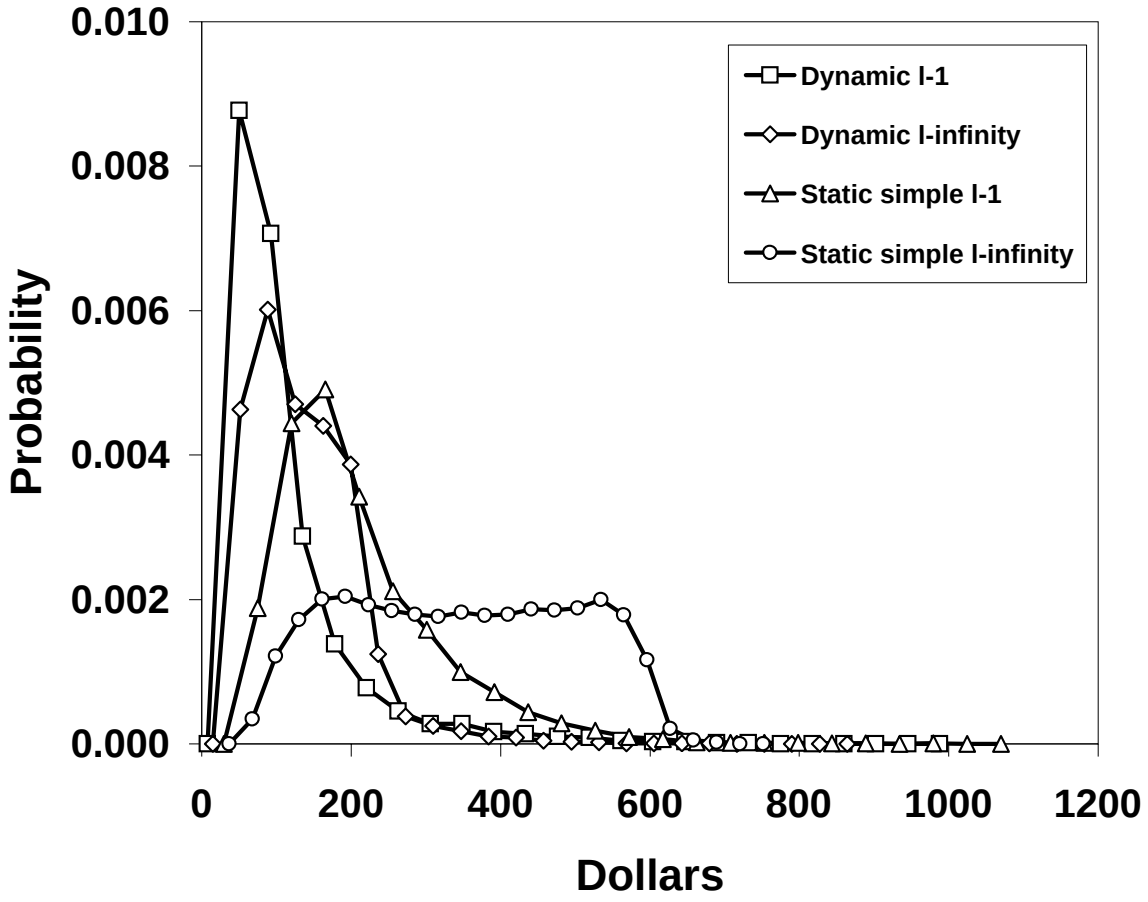

Figure 7: Density functions of the average absolute daily tracking error for two compression techniques and two static simple strategies.

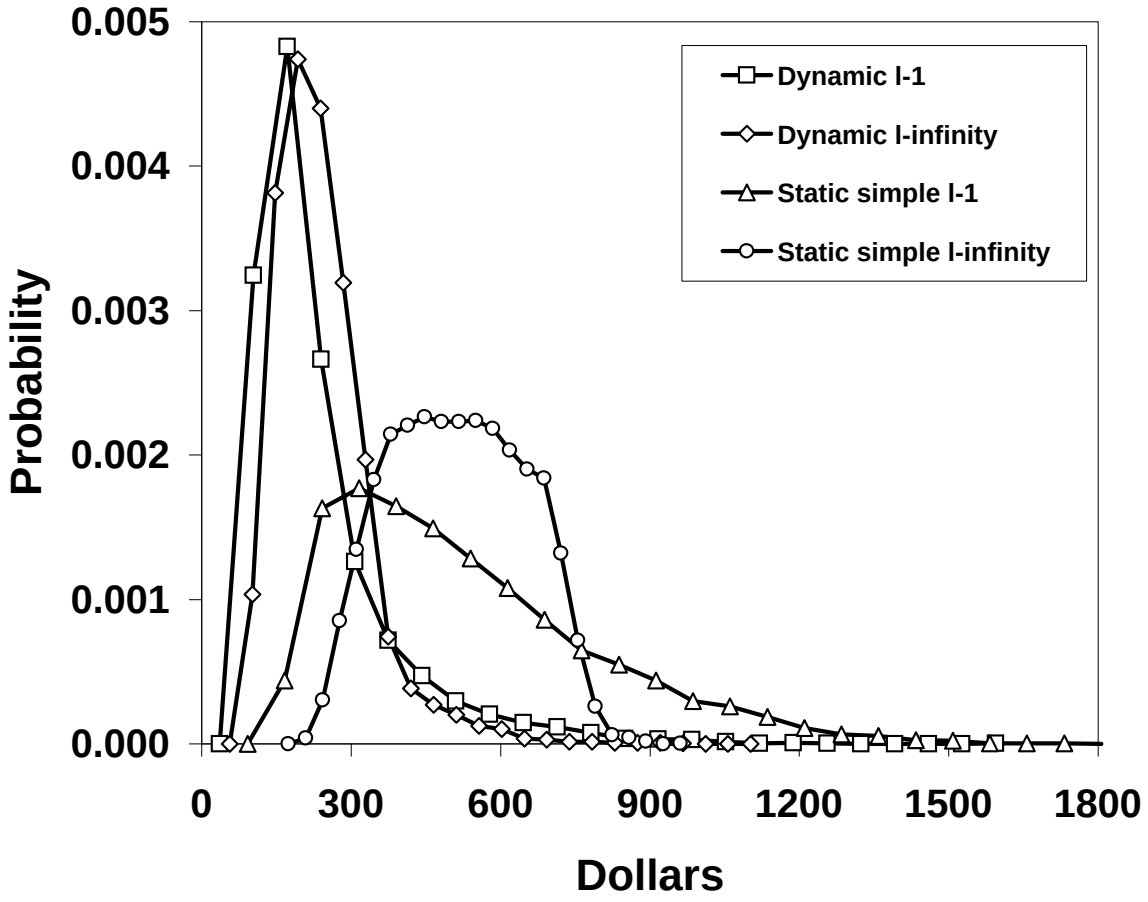

Figure 8: Density functions of the worst absolute daily tracking error for two compression techniques and two static simple strategies.

upper tail of the target value's maximum.

Dynamic portfolio replication

The second set of experiments use dynamic stochastic programming iteratively to construct a dynamic replication strategy for the target using a prescribed set of tradable instruments. For each test scenario at each of the four maturity dates we construct a new scenario tree, re-solve the stochastic programming problem and re-balance to the portfolio associated with the initial node of this problem (*cf.* Figure 2).

For this dynamic replication strategy we assume a proportional transaction cost of 1% on trades involving the underlying index and 2.5% on trades involving options, and impose an initial budget constraint of 1.025 times the initial value of the target as discussed in Section 4. (These transaction costs are higher than would be paid for hedging an S&P 500 options portfolio with S&P 500 futures contracts, but are used here for illustrative purposes.) The tracking errors to be minimised in the objectives of the DSP problems are taken to be the absolute differences between the mark-to-market values of the replicating strategy and the target evaluated at times 0, the three successive rebalance dates and the horizon (again using Black–Scholes valuation). Here we use 1,000 test simulations to estimate the expected average absolute at these 5 dates.

We try a large number of different scenario trees varying both in size and the extent to which branching occurs near the root node. We found little advantage in using the inverse distribution function discretization technique of Section 4 to generate the scenario tree and instead use purely random index path sampling as described above. Again we will consider the effect on the dynamic strategies of restricting the strategy to using just cash and the underlying index. In all cases we will minimize the (L1′) objective.

Benchmark results

Our benchmark replication strategies will be quasi-static simple strategies based on respectively 100, 200 and 300 random scenarios for the path of the index from its initial value and the delta hedge rebalanced at decision points *with* transactions costs. We allow the quasi-static strategies to trade using cash, the underlying index and the four tradable options. Note that the simple strategies are quasi-static in the sense that the optimization problem solved at each step *assumes* no further re-balancing, but is in fact allowed to rebalance the portfolio according to the root node solution of the static simple problem corresponding to

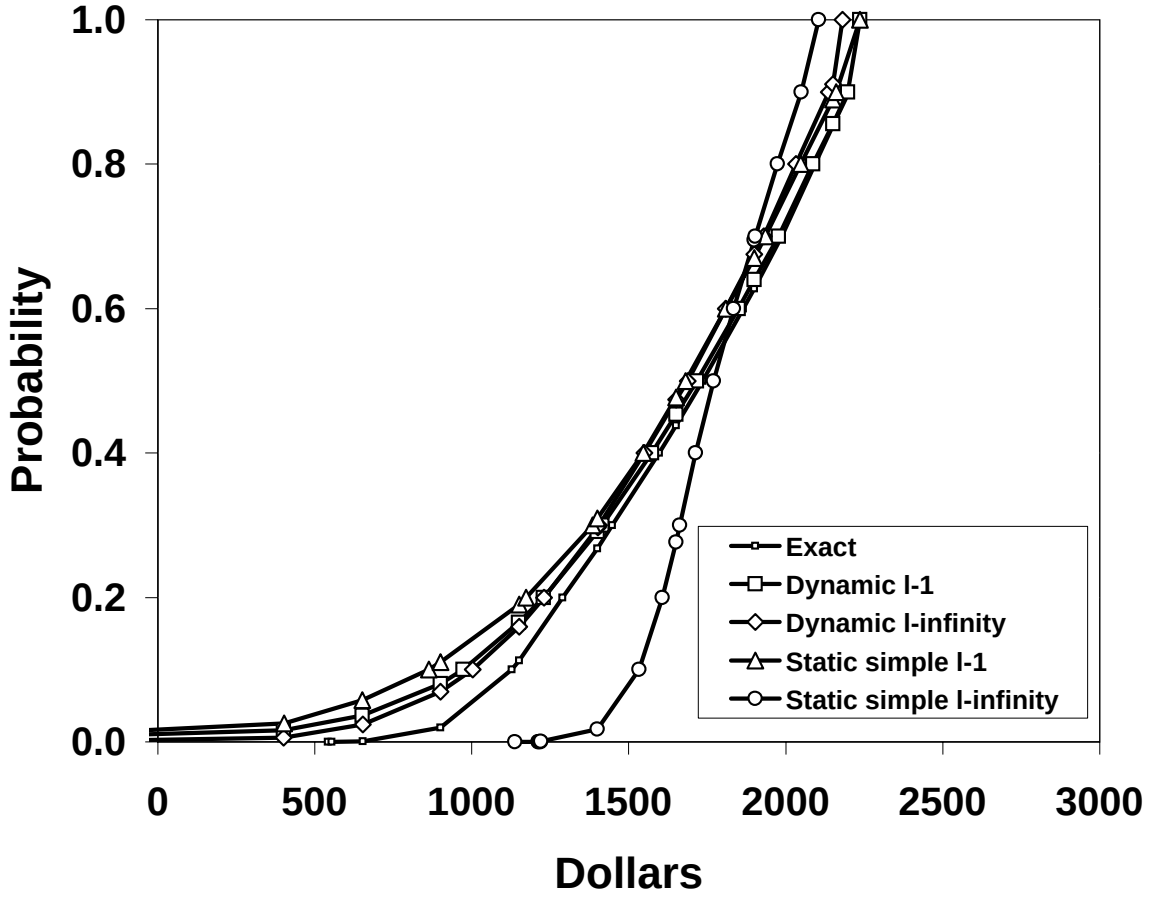

Figure 9: Distribution functions of the minimum value of the target and of several compressed portfolios.

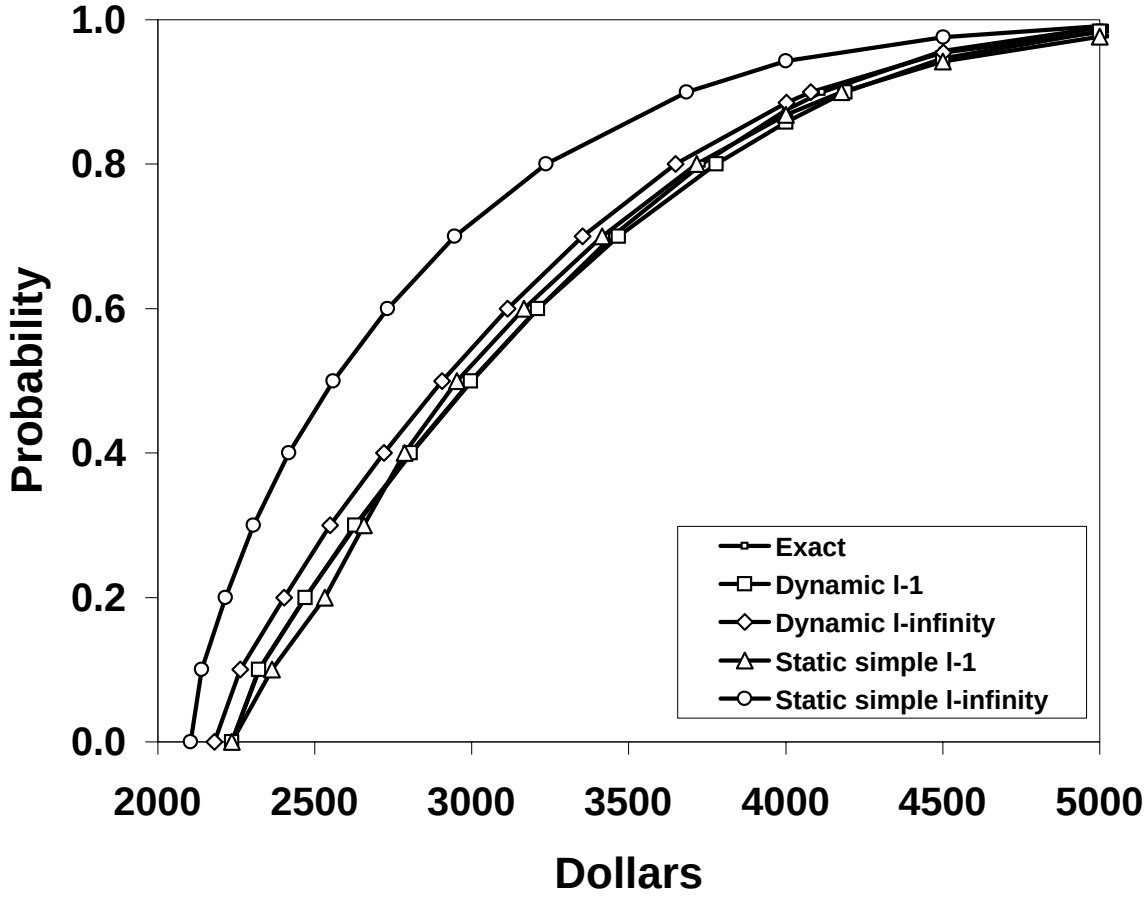

Figure 10: Distribution functions of the maximum value of the target and of several compressed portfolios.

No. Scenarios	Objective	Expected average absolute tracking error	Optimization CPU time (s)
100	87.2	110.9 (8.03)	4
200	91.1	115.5 (0.97)	12
300	92.4	111.9 (0.95)	27
delta hedge	N/A	115.9 (0.89)	N/A

Table 7: Dynamic portfolio replication results: quasi-static simple and delta hedge benchmarks. One standard error in the estimate of the expected average tracking error is indicated in brackets.

the current timestep. As the problem is re-solved at each timestep, the positions in the portfolio will generally change.

A summary of the numerical results for the benchmark strategies is given in Table 7. As the number of scenarios increases, although the objective of the optimization problem used in the quasi-static strategeies increases, the expected average absolute tracking error of the resulting strategy decreases (as we would expect) and is comparable to that of the delta hedge. Similar remarks apply to the expected *worst* absolute tracking error.

Experiments varying tree size

For dynamic portfolio replication we consider scenarios trees which have the same branching factor at each node, and we gradually increase the branching factor expecting that larger trees should lead to better replicating trading strategies.

The results are presented in Tables 8 and 9. Again we see that allowing the dynamic strategy to trade in options is detrimental, although the difference decreases as the branching-factor increases. None of the dynamic strategies based on balanced scenario trees with equal branching at each node and trading in only cash and the underlying index beat the simple benchmarks. However we will see that by varying the branching structure over decision points these results can be improved.

Experiments varying initial branching

Since with the dynamic replication strategy although we re-solve the optimization problem at each timestep using a new scenario tree, we ultimately only use the optimal portfolio

Branching	Objective	Expected average absolute tracking error	CPU time (s)
2-2-2-2	15.6	414.8 (27.0)	1
3-3-3-3	51.5	260.4 (20.3)	1
4-4-4-4	43.5	210.4 (7.4)	1
5-5-5-5	58.4	163.6 (6.0)	1
6-6-6-6	50.0	145.5 (3.7)	2
7-7-7-7	54.3	142.4 (3.6)	8

Table 8: Dynamic portfolio replication results: dynamic strategy with varying tree sizes. Only cash and the underlying index are available as trading instruments. One standard error in the estimate of the expected average tracking error is indicated in brackets.

Branching	Objective	Expected average absolute tracking error	time (s)
2-2-2-2	0.0	987.0 (98.9)	1
3-3-3-3	23.6	703.2 (74.9)	1
4-4-4-4	1.5	694.2 (83.8)	1
5-5-5-5	21.3	433.7 (29.0)	4
6-6-6-6	18.6	312.5 (37.0)	9
7-7-7-7	26.0	188.5 (16.0)	61

Table 9: Dynamic portfolio replication results: dynamic strategy with different branching structures. Cash, the underlying index and four options are available as trading instruments. One standard error in the estimate of the expected average tracking error is indicated in brackets.

associated with the root node appropriate to the timestep. Given this procedure, it seems sensible to make scenario trees branch more near the root node of each successive DSP problem. Here we keep the total number of nodes in the tree roughly constant (in an attempt to keep the optimization time constant across experiments) and gradually increase the branching at the root node of each successive DSP problem.

The results are summarized in Tables 10 and 11. As the initial branching increases, both the objective and the expected average tracking error tend to decrease, but with some sampling fluctuation due to the randomly generated scenario trees. This random decrease is more prominent when the dynamic trading strategy is allowed to use options (Table 11), when the final strategies in the table (in which almost all branching occurs at the root node), give a *significant improvement* over the best benchmark strategy in less than one third the computing time (*cf.* Table 7). Initial positions chosen by the various strategies are set out in Table 12.

Finally, a comparison of the distributions of the average and worst absolute tracking errors (measured at the initial time and the four subsequent decision points) for the quasi-static simple benchmarks and the dynamic replication strategies based on the tree with highest branching in the initial time stage, both with and without the availability of trading options, is shown in Figures 11 and 12. Again this shows that the best dynamic replication strategies continue to track the target in extreme market conditions.

6 Conclusions

For both applications considered in this paper—portfolio compression and dynamic portfolio replication—the solutions from a stochastic programming approach are superior to optimized 'quasi-static' approaches and a delta hedge. For compression however, is not clear how the solution of the stochastic programming problem should be interpreted as a trading strategy; the method used here is not robust enough to allow the strategy to trade in options as well as cash and the underlying.

From Figures 9 and 10 we see that the extreme values taken by the target portfolio over the planning horizon have medians close to 1750 and 3000 respectively, indicating that the expected average absolute tracking error achieved by the best dynamic hedge (90.5) is approximately 3%–5% of the target value.

In the dynamic portfolio replication problem, very different trees should be used from those effective for portfolio compression: almost all the branching should occur at the root

Branching	Objective	Expected average absolute tracking error	CPU time (s)
6-6-6-6	50.0	145.5 (3.7)	2
8-6-5-4	43.6	137.3 (3.3)	3
9-7-5-3	40.5	134.0 (3.9)	5
13-8-4-2	47.5	135.8 (3.1)	3
21-9-3-1	38.8	122.4 (2.6)	2
22-8-3-1	30.1	119.7 (2.6)	2
31-8-2-1	34.1	118.6 (2.4)	4
43-9-1-1	25.9	122.8 (2.5)	7
55-7-1-1	28.9	120.6 (2.5)	4
75-5-1-1	24.7	119.3 (2.5)	5
120-3-1-1	23.0	115.7 (2.0)	6
300-1-1-1	17.1	116.5 (2.3)	5

Table 10: Dynamic portfolio replication results: dynamic strategy with different branching structures. Only cash and the underlying index are available as trading instruments. One standard error in the estimate of the expected average tracking error is indicated in brackets.

Branching	Objective	Expected average absolute tracking error	CPU time (s)
6-6-6-6	18.7	312.5 (37.0)	9
8-6-5-4	9.3	261.5 (29.7)	5
9-7-5-3	7.1	215.9 (17.2)	7
13-8-4-2	19.9	132.5 (5.3)	8
21-9-3-1	16.9	133.4 (14.3)	10
22-8-3-1	12.2	107.2 (6.0)	7
31-8-2-1	17.9	116.7 (4.5)	8
43-9-1-1	10.2	103.9 (7.2)	6
55-7-1-1	12.2	132.1 (34.4)	73
75-5-1-1	10.2	94.8 (2.0)	6
120-3-1-1	8.1	96.1 (2.6)	7
300-1-1-1	3.6	93.0 (1.8)	8

Table 11: Dynamic portfolio replication results: dynamic strategy with different branching structures. Cash, the underlying index and four options are available as trading instruments. One standard error in the estimate of the expected average tracking error is indicated in brackets. The anomalous runtime is due to chance degeneracy of the deterministic equivalent linear programme.

Method	Cash	Underlying	30-day	60-day	180-day	360-day
quasi-static simple, 100 scenarios	−329.2	1.73	0.27	2.97	3.85	0.00
quasi-static simple, 200 scenarios	−171.4	1.62	0.00	2.97	3.65	0.00
quasi-static simple, 300 scenarios	−387.6	1.80	0.00	2.41	3.95	0.00
dynamic with options	−1426.8	2.16	−4.23	0.00	0.00	13.45
dynamic w/o options	−5194.0	5.81	N/A	N/A	N/A	N/A

Table 12: Dynamic portfolio replication results: Optimal initial portfolio holdings.

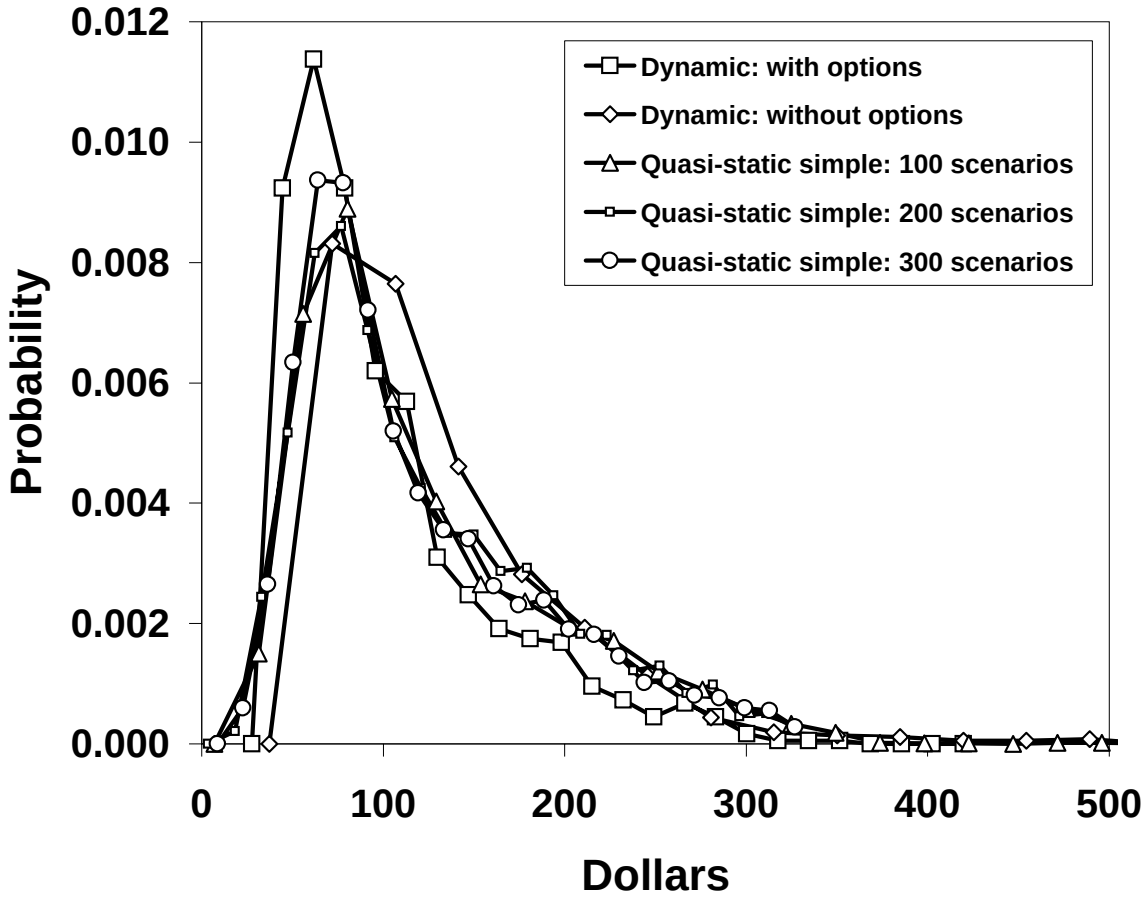

Figure 11: Dynamic portfolio replication results: density functions of the average tracking error for two dynamic hedging techniques and three benchmark strategies.

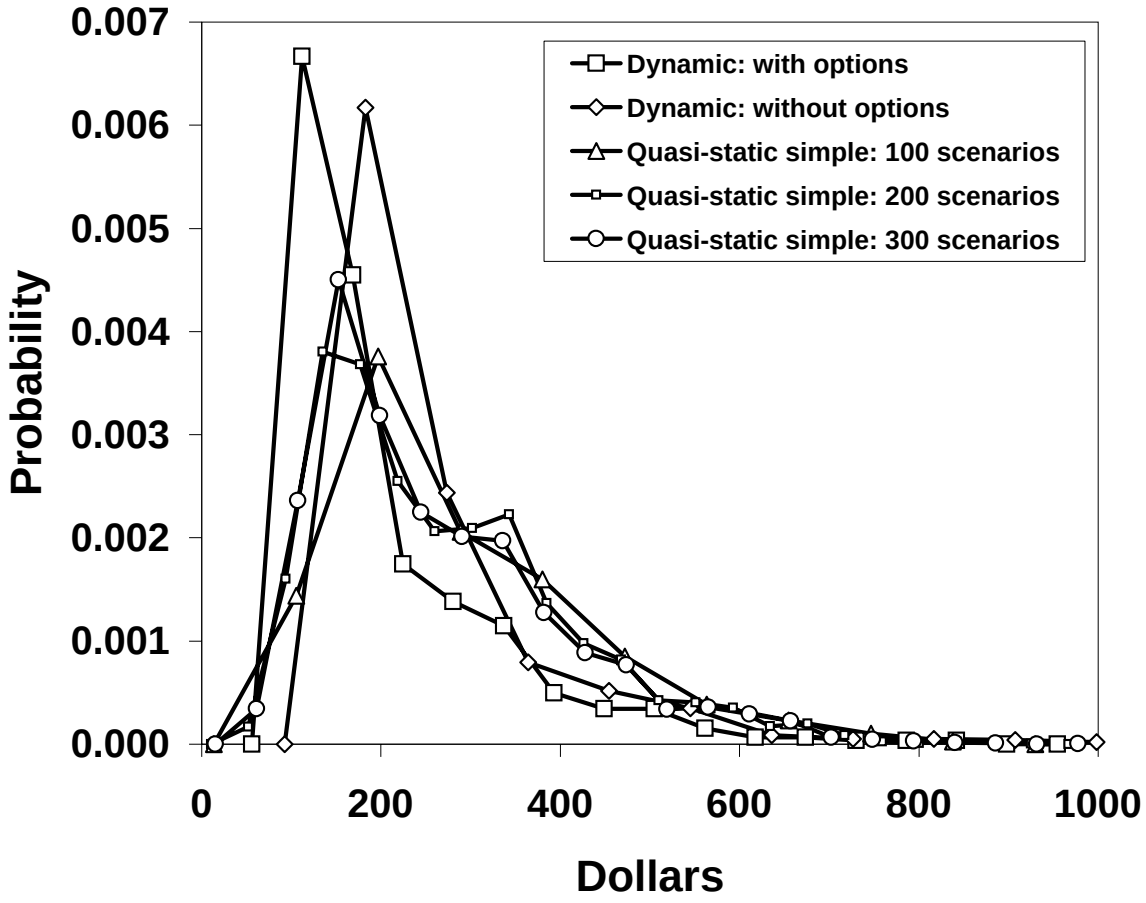

Figure 12: Dynamic portfolio replication results: density functions of the worst absolute tracking error for two dynamic hedging techniques and three benchmark strategies.

node. In this case the method is robust enough to allow the dynamic strategy to trade options in order to reduce tracking error and produce an overall significantly best strategy in a running time three times faster than the best quasi-static benchmark alternative.

In a companion paper we will describe the extension of these ideas to a swaption portfolio where, unfortunately, further ingenuity is required.

References

Birge, J. R. & Louveaux, F. (1997), *Introduction to Stochastic Programming*, Springer, Berlin.

Dembo, R. & Rosen, D. (1999), 'The practice of portfolio replication', *Ann. Op. Research* **85**, 267–284.

Chisti, A. (1999), 'Simulation of fixed-income portfolios using grids', *Algo Research Quarterly* **2**(2), 41–50.

Dempster, M. A. H., ed. (1980), *Stochastic Programming*, Academic Press, London.

Dempster, M. A. H. & Ireland, A. M. (1988), A financial decision support system, *in* G. Mitra, ed., *Mathematical Models for Decision Support Systems*, Springer, Berlin, pp. 415–440.

Dempster, M. A. H. & Thompson, R. T. (1999), 'EVPI-based importance sampling solution procedures for multi-stage stochastic linear programmes on parallel MIMD architectures', *Ann. Op. Research* **90**, 161–184.

Dempster, M. A. H., Hicks-Pedron, N., Medova, E. A., Scott, J. E. & Sembos, A. (2000), 'Planning logistics operations in the oil industry', *Journal of the Operational Research Society* **51**, 1271–1288.

Fleten, S., Høyland, K. & Wallace, S. (1998), 'The performance of stochastic dynamic and fix mix portfolio models', Working Paper, Norwegian University of Science and Technology.

Golub, B., Holmer, M., Polman, L. & Zenios, S. (1997), 'Stochastic programming models for money management', *European Journal of Operations Research* **85**, 282–296.

Gondzio, J. & Kouwenberg, R. (1999), 'High performance computing for asset liability management', Working Paper, Erasmus University.

Gondzio, J., Kouwenberg, R. & Vorst, T. (1998), 'Hedging options under transaction costs and stochastic volatility', Working Paper.

Jamshidian, F. & Zhu, Y. (1997), 'Scenario simulation: Theory and methodology', *Finance and Stochastics* **1**, 43–67.

Klaassen, P. (1997), 'Discretized reality and spurious profits in stochastic programming models for asset/liability management', Working Paper, Vrije Universiteit, Amsterdam.

Kouwenberg, R. (1998), 'Senario generation and stochastic programming models for asset liability management', Working Paper, Econometric Institute, Erasmus University Rotterdam.

Kouwenberg, R. & Vorst, T. (1998), 'Dynamic portfolio insurance: A stochastic programming approach', Working Paper, Department of Finance, Erasmus University Rotterdam.

Mulvey, J. & Ziemba, W. T., eds (1998), *Worldwide Asset Liability Modelling*, Cambridge University Press.

Wets, R. J.-B. & Ziemba, W. T., eds (1999), 'Stochastic Programming: State of the Art, 1998', *Ann. Op. Research* **85**.

Worzel, K. J., Vassiadou-Zeniou, C. & Zenios, S. A. (1994), 'Integrated simulation and optimization models for tracking indices of fixed-income securities', *Operations Research* **42**(2), 223–233.

Appendix: A prototype implementation in RiskWatch

To demonstrate the practicality of using a stochastic programming approach to dynamic portfolio replication, we have implemented a general stochastic programming system using RiskWatch for instrument and portfolio valuation. At present the system is in prototype form, using a number of external programs and our stochastic programming model generation suite STOCHGEN 2.3. In future these will be integrated seamlessly with RiskWatch,

allowing the whole process of simulation, solution and solution analysis to be performed from behind a RiskWatch GUI.

An overview of the prototype's design is given in Figure 13. The aim is to let the user write a quite general stochastic programming problem using STOCHGEN, which employs the modelling language AMPL, with reference to the value of RiskWatch entities such as instruments and portfolios. This allows the user to combine the flexibility of STOCHGEN/AMPL for expressing optimization problems (which covers all the optimization functionality currently in RiskWatch) with that of RiskWatch for valuing financial instruments. The design also makes it easy to implement alternative sampling techniques such as quasi-random sampling or even to allow part of the scenario-tree to be constructed by hand, to include crash scenarios for example.

In more detail: we use a Perl script to create the scenario tree, outputting a scenario set containing each scenario present in the scenario tree, load this into RiskWatch, run a simulation to value the instruments and portfolios, and use a RiskScript macro to write all the simulation data to a single data-file.

This data-file is read by another program which creates AMPL-format data-files containing the data for each scenario, and a file of auxiliary information, such as the names of the RiskWatch entities for which data is present, the number of timesteps, the simulation dates and so on. Other information such as a non-uniform weighting of scenarios could easily be passed via this file.

These data-files, together with the stochastic programming problem expressed as an AMPL model, are fed into the STOCHGEN system, producing a single linear program (the deterministic equivalent form of the stochastic programming problem) in standard MPS or SMPS format for solvers.

Once solved, the optimal decision variables at each node in the tree can be extracted from solver output files and represented as a RiskWatch scenario set.

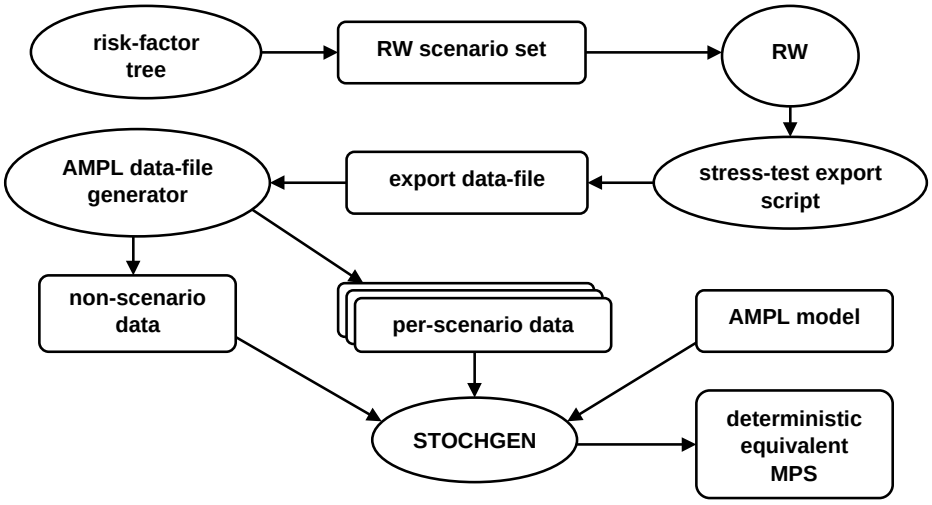

Figure 13: Stochastic programming using RiskWatch and STOCHGEN (prototype design)