# Solving nonlinear financial planning problems with $10^9$ decision variables on massively parallel architectures *

Jacek Gondzio, Andreas Grothey
*School of Mathematics, University of Edinburgh*

## Abstract

Multistage stochastic programming is a popular technique to deal with uncertainty in optimization models. However, the need to adequately capture the underlying distributions leads to large problems that are usually beyond the scope of general purpose solvers. Dedicated methods exist but pose restrictions on the type of model they can be applied to. Parallelism make sthese problems potentially tractable, but is generally not exploited in todays general purpose solvers.
We apply a structure-exploiting parallel primal-dual interior-point solver for linear, quadratic and nonlinear programming problems. The solver efficiently exploitats the structure of these models. Its design relies on object-oriented programming principles, treating each substructure of the problem as an object carrying its own dedicated linear algebra routines. We demonstrate its effectiveness on wide range of financial planning problems, resulting in linear, quadratic or non-linear formulations.
Also coarse grain parallelism is exploited in a generic way that is efficient on any parallel architecture from ethernet linked PCs to massively parallel computers. On a 1280-processor machine with a peak performance of 6.2 TFlops we can solve a quadratic financial planning problem exceeding $10^9$ decision variables.
*Keywords: asset and liability management, interior point, massive parallelism, structure exploitation*

* Supported by the Engineering and Physical Sciences Research Council of UK, EPSRC grant GR/R99683/01.

## 1 Introduction

Decision making under uncertainty is an important consideration in financial planning. A promising approach to the problem is the multistage stochastic programming version of the asset liability management model as reported in [1, 2, 3, 4]. Its advantages include the ability to model the dynamic features of the underlying decision problem by allowing the rebalancing of the portfolio at different times as well as capturing possible dynamic effects of the asset distributions. Unfortunately realistic models tend to cause an explosion in dimensionality due to two factors: firstly the size of the problem grows exponentially with the number of portfolio rebalancing dates (or stages). Further a considerable number of realizations are required to capture the conditional distribution of asset returns with a discrete approximation. For $T$ stages and $p$ realizations the dimension of the resulting problem will be of order $p^T$.

The last decade has seen a rapid improvement of methods to solve large scale stochastic programs. However most of these are only applicable in a very special setting. Nested Benders Decomposition approaches [5, 6] are limited to LP formulations. Linear algebra approaches such as [7, 8] are usually limited to very special structures resulting for example from constraints on the allowed type of recurrence relation.

In this paper we discuss our experiences with the modern, general structure exploiting interior point implementation OOPS (Object-Oriented Parallel Solver) [9, 10]. We show that our approach makes the solution of general large nonlinear financial planning problems feasible. Furthermore it allows for fast computation of efficient frontiers and can exploit parallel computer architectures.

In the following Section 2 we state the asset liability management model that we are concerned with and present various nonlinear extensions. In Section 3 we give a brief description of the Object-Oriented Parallel Solver OOPS, while in Section 4 we report numerical results on the various problem formulations.

## 2 Asset Liability Management via Stochastic Programming

We are concerned with finding the optimal way of investing into assets $j = 1, \ldots, J$ over several time-periods $t = 0, \ldots, T$. The returns of the assets at each time-period are assumed to be uncertain but with a known joint distribution. An initial amount of cash $b$ is invested at $t = 0$ and the portfolio may be rebalanced at discrete times $t = 1, \ldots, T$. The objective is to maximize the expectation of the final value of the portfolio at time $T + 1$ while minimizing the associated risk measured by the variance of the final wealth.

The uncertainty in the process is described by an *event tree*: each node of the event tree at depth $t$ corresponds to a possible outcome of events at time $t$. Associated with every node $i$ in the event tree are returns $r_{i,j}, 1 \leq j \leq J$ for each of the assets and the probability $p_i$ of reaching this node. For every node, children of the node are chosen in such a way, that their combined probabilities and asset returns reflect the (known) joint distribution of all assets at the next time period, given the

sequence of events leading to the current node. The question how to best populate the event tree to capture the characteristics of the joint distribution of asset returns is an active research area, we refer the reader to [11].

We use the following terminology: Let $L_t$ be the set of nodes in the event tree corresponding to time stage $t$. $L_T$ is the set of final nodes (leaves) and $L = \bigcup_t L_t$ the complete node set. An $i \in L$ denotes any node in the tree, with $i = 0$ corresponding to the root and $\pi(i)$ denotes the predecessor (parent) of node $i$. Let $v_j$ be the value of asset $j$, and $c_t$ the transaction cost. It is assumed that the value of the assets will not change throughout time and a unit of asset $j$ can always be bought for $(1+c_t)v_j$ or sold for $(1-c_t)v_j$. A unit of asset $j$ held in node $i$ (coming from node $\pi(i)$) will generate extra return $r_{i,j}$. Denote by $x^h_{i,j}$ the units of asset $j$ held at node $i$ and by $x^b_{i,j}, x^s_{i,j}$ the transaction volume (buying, selling) of this asset at this node, respectively. Similarly $x^h_{t,j}, x^b_{t,j}, x^s_{t,j}$ are the random variables describing the holding, buying and selling of asset $j$ at time stage $t$. The inventory constraints capture system dynamics: the variables (asset holdings) associated with a particular node and its parent are related

$$(1 + r_{i,j})x^h_{\pi(i),j} = x^h_{i,j} - x^b_{i,j} + x^s_{i,j}, \quad \forall i \neq 0, j. \tag{1}$$

We assume that we start with zero holding of all assets but with funds $b$ to invest. Further we assume that one of the assets represents cash, i.e. the available funds are always fully invested. Cash balance constraints describe possible buying and selling actions within a scenario while taking transaction costs into account:

$$\begin{array}{rcl} \sum_j (1+c_t)v_j x^b_{i,j} + l_i & = & \sum_j (1-c_t)v_j x^s_{i,j} + C_i \quad \forall i \neq 0 \\ \sum_j (1+c_t)v_j x^b_{0,j} & = & b, \end{array} \tag{2}$$

where $l_i$ are liabilities to pay at node $i$ and $C_i$ are cash contributions paid at node $i$. Further restrictions on the investment policy such as regulatory constraints or asset mix bounds can be easily expressed in this framework.

Markowitz portfolio optimization problem [12] combines two objectives of the investor who wants to: (i) maximize the final wealth, and (ii) minimize the associated risk. The final wealth $y$ is expressed as the expected value of the portfolio at time $T$ converted into cash [13]

$$y = I\!E((1-c_t)\sum_{j=1}^{J} v_j x^h_{T,j}) = (1-c_t)\sum_{i \in L_T} p_i \sum_{j=1}^{J} v_j x^h_{i,j}. \tag{3}$$

The risk is measured with the variance of return:

$$\mathrm{Var}((1-c_t)\sum_{j=1}^{J} v_j x^h_{T,j}) = \sum_{i \in L_T} p_i (1-c_t)^2 [\sum_j v_j x^h_{i,j}]^2 - y^2. \tag{4}$$

These two objectives are combined into a single concave quadratic function of the following form

$$f(x) = I\!E(F) - \lambda \mathrm{Var}(F), \tag{5}$$

where $F$ denotes the final portfolio converted into cash (3) and $\lambda$ is a scalar expressing investor's attitude to risk. Thus in the classical (multistage) Markowitz model we would maximize (5) subject to constraints (1), (2) and (3).
The need to well approximate the continuous joint distribution of asset returns leads to large event trees and subsequently very large problems. These models however display a regular structure which can be exploited by our solution methodology.

### 2.1 Extensions of Asset Liability Management Problem

There are several disadvantages associated with the standard mean-variance formulation of the asset liability model as described in the previous section. It has been observed, for example, that the mean-variance model does not satisfy the second order stochastic dominance condition [14]. Furthermore, by using variance to measure risk, this model penalizes equally the overperformance and the underperformance of the portfolio. A portfolio manager is interested in minimizing the risk of loss hence a semi-variance (downside risk) seems to be a much better measure of risk.
To allow more flexibility for the modelling we introduce two more (nonnegative) variables $s_i^+, s_i^-$ per scenario $i \in L_t$ as the positive and negative variation from the mean and add the constraint

$$(1-c_t)\sum_{j=1}^{J} v_j x_{i,j}^h + s_i^+ - s_i^- = y, \quad i \in L_T \tag{6}$$

to the model. The variance can be expressed as

$$\mathrm{Var}(X) = \sum_{i\in L_t} p_i (s_i^+ - s_i^-)^2 = \sum_{i\in L_t} p_i ((s_i^+)^2 + (s_i^-)^2), \tag{7}$$

since $(s_i^+)^2, (s_i^-)^2$ are not both positive at the same time. Using (6) we can easily express the semivariance $\mathrm{sVar}(X) = I\!E[(X - I\!EX)_-^2] = \sum_{i\in L_t} p_i (s_i^+)^2$ to measure downside risk. The standard Markowitz model can be written as

$$\max_{x,y,s\geq 0} y - \rho[\sum_{i\in L_T} p_i ((s_i^+)^2 + (s_i^-)^2)] \quad \text{subject to} \quad (1),(2),(3),(6). \tag{8}$$

In this paper we are concerned with its extensions (we implicitly assume constraints (1), (2), (3), (6) in all of these):

- Downside risk (measured by the semi-variance) is constrained:

$$\max_{x,y,s\geq 0} y \quad \text{s.t.} \quad \sum_{i\in L_t} p_i (s_i^+)^2 \leq \rho. \tag{9}$$

- Objective in a form of a logarithmic utility function captures risk-adversity:

$$\max_{x,y,s\geq 0} \sum_{i\in L_t} p_i \log(\sum_{j=1}^{J} v_j x_{i,j}^h) \quad \text{s.t.} \quad \sum_{i\in L_t} p_i (s_i^+)^2 \leq \rho. \tag{10}$$

- Following Konno et al. [15] objective function takes skewness into account and captures the investors preference towards positive deviations in the case of non-symmetric distribution of returns for some assets

$$\max_{x,y,s\geq 0} y + \gamma \sum_{i\in L_t} p_i (s_i^+ - s_i^-)^3 \quad \text{s.t.} \quad \sum_{i\in L_t} p_i ((s_i^+)^2 + (s_i^-)^2) \leq \rho. \quad (11)$$

All these extensions have attractive modelling features but they inevitably lead to nonlinear programming formulations. It is worth noting that to date only a few algorithms have ever been considered for these formulations [7, 8] and it is not obvious if they can be extended to the general settings. Our approach can easily handle all these models.

### 2.2 Efficient Frontier

The standard Markowitz objective function $f(x) = I\!E(F) - \lambda \mathrm{Var}(F)$, uses the risk-aversion parameter $\lambda$ to trade off the conflicting aims of maximizing return while minimizing risk. However a risk-aversion parameter is not an intuitive quantity, a better picture of the possible options would be gained from the complete trajectory $(\mathrm{Var}(F,\lambda), I\!E(F,\lambda))$ for all values of $\lambda$, that is knowing how much extra expected return could be gained from an increase in the still acceptable level of risk. This $(\mathrm{Var}(F,\lambda), I\!E(F,\lambda))$ trajectory is known as the *efficient frontier*.
The efficient frontier can be calculated by repeatedly solving the ALM model for different values of $\lambda$. However it would be desirable if this computation could be sped up by the use of warm-starts; after all we seek to solve a series of closely related problems. Unfortunately both proposed solution approaches for multistage stochastic programming, namely decomposition and interior point methods suffer from a perceived lack of efficient warmstarting facilities. We will show that OOPS comes with a warm starting facility that allows a significant decrease in computational cost when calculating the efficient frontier.

## 3 Object-Oriented Parallel Solver (OOPS)

Over the years, interior point methods for linear and nonlinear optimization have proved to be a very powerful technique. We review basic facts of their implementation in this section and show how OOPS uses the special structure in stochastic programming problems to enable the efficient (and possible parallel) solution of very large problem instances.
Consider the nonlinear programming problem

$$\min \quad f(x) \quad \text{s.t.} \quad g(x) + z = 0,\ z \geq 0$$

where $f : \mathcal{R}^n \to \mathcal{R}$ and $g : \mathcal{R}^n \to \mathcal{R}^m$ are assumed sufficiently smooth. Interior point methods proceeed by replacing the nonnegativity constraints with logarith-

mic barrier terms in the problem objective to get

$$\min \quad f(x) - \mu \sum_{j=1}^{n} \ln z_j \quad \text{s.t.} \quad g(x) + z = 0,$$

where $\mu \geq 0$ is a barrier parameter. First order stationary conditions of this problem are

$$\begin{aligned} \nabla f(x) - \nabla g(x)^T y &= 0 \\ g(x) + z &= 0, \\ YZe &= \mu e, \end{aligned}$$

where $Z = \mathrm{diag}\{z_1, \ldots, z_n\}$. Interior point algorithms for nonlinear programming apply Newton method to solve this system of nonlinear equations and gradually reduce the barrier parameter $\mu$ to guarantee convergence to the optimal solution of the original problem. The Newton direction is obtained by solving the system of linear equations:

$$\begin{bmatrix} Q(x,y) & A(x)^T & 0 \\ A(x) & 0 & I \\ 0 & Z & Y \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} -\nabla f(x) - A(x)^T y \\ -g(x) - z \\ \mu e - YZe, \end{bmatrix}, \quad (12)$$

where $Q(x,y) = \nabla^2 f(x) + \sum_{i=1}^{m} y_i \nabla^2 g_i(x) \in \mathcal{R}^{n \times n}$ and $A(x) = \nabla g(x) \in \mathcal{R}^{m \times n}$ are the Hessian of Lagrangian and the Jacobian of constraints, respectively. After substituting $\Delta z = \mu Y^{-1} e - Ze - ZY^{-1}\Delta y$ in the second equation we get

$$\begin{bmatrix} -Q(x,y) & A(x)^T \\ A(x) & \Theta_D \end{bmatrix} \begin{bmatrix} \Delta x \\ -\Delta y \end{bmatrix} = \begin{bmatrix} \nabla f(x) + A(x)^T y \\ -g(x) - \mu Y^{-1} e \end{bmatrix}, \quad (13)$$

where $\Theta_D = ZY^{-1}$ is a diagonal matrix. Interior point methods need to solve several linear systems with this *augmented system* matrix at every iteration. This is by far the dominant computational cost in interior point implementations.

In many important applications (such as stochastic programming) the augmented system matrix displays a nested block structure. Such a structure can be represented by a matrix tree, that closely resembles the event tree of the corresponding stochastic program. Every node in the matrix tree represents a particular block-component of the augmented system matrix. OOPS exploits this structure by associating with each node of the event/matrix tree a linear algebra implementation that exploits the corresponding block matrix structure in operations such as matrix factorizations, backsolves and matrix-vector-products. It also enables the exploitation of parallelism, should several processors be available to work on a node.

In effect, all linear algebra operations required by an interior point method are performed in OOPS recursively by traversing the event tree, where several processors can be assigned to a particular node, if required. More details can be found in [9, 16, 10].

## 4 Numerical Results

We will now present the computational results that underpin our claim that very large nonlinear portfolio optimization problems are now within scope of a modern structure exploiting implementation of general mathematical programming algorithms like OOPS.
We have used OOPS to solve the three variants (9), (10) and (11) of the Asset and Liability Management problem. All test problems are randomly generated using a symmetric scenario tree with 3-4 periods and between 24-70 realizations per time stage (Blocks). The data for the 20-40 assets used are also generated randomly. Statistics of the test problems are summarized in Table 1. As can be seen problem sizes increase to just over 10 million decision variables. Computational results for the three ALM variants (9), (10), (11) are collected in Table 2. Computations were done on the SunFire 15K at Edinburgh Parallel Computing Centre (EPCC), with 48 UltraSparc-III processors running at 900MHz and 48GB of shared memory. Since the parallel implementations relies solely on MPI we expect these results to generalize to a more loosely linked network of processors such as PCs linked via Ethernet. We used an optimality tolerance of $10^{-5}$ throughout.
All problems can be solved in a reasonable time and with a reasonable amount of interior point iterations - the largest problem needing just over 7 hours on a single 900MHz processor. OOPS displays good scalability, achieving a parallel efficiency of up to 0.96 on 8 processors. With the event of multi-core architectures even for desktop PCs, this shows that large nonlinear portfolio management problems are tractable even on modest computing hardware.

### 4.1 Comparison with CPLEX 9.1

We wish to make the point that a structure exploiting solver is an absolute requirement to solve very large stochastic nonlinear programming problems. To demonstrate this we have compared OOPS with the state-of-the-art commercial solver CPLEX 9.1. Since CPLEX has only the capability to solve QPs and we do not have a parallel CPLEX license, we compare CPLEX with OOPS for the QP model (8) on a single 3GHz processor with 2GB of memory. Results are reported in Table 3. As can bee seen OOPS needs consistently less memory than CPLEX (which actually fails to solve problem C70 due to running out of memory - the time for this problem has been extrapolated from the number of nonzeros in the factorization as reported by CPLEX). The smallest problem C33 is solved slightly faster

| Problem | Stages | Blk | Assets | Total Nodes | Constraints | Variables |
|---|---|---|---|---|---|---|
| ALM1 | 3 | 70 | 40 | 4971 | 208,713 | 606,322 |
| ALM2 | 4 | 24 | 25 | 14425 | 388,876 | 1,109,525 |
| ALM3 | 4 | 55 | 20 | 169456 | 3,724,953 | 10,500,112 |

Table 1: Asset and Liability Management Problems: Problem Statistics.

| Problem | 1 proc | | 2 procs | | 4 procs | | 8 procs | |
|---|---|---|---|---|---|---|---|---|
| | iter | time (s) | time (s) | pe | time (s) | pe | time (s) | pe |
| variant (9): semi-variance | | | | | | | | |
| ALM1 | 35 | 568 | 258 | 1.10 | 141 | 1.01 | 92 | 0.76 |
| ALM2 | 30 | 1073 | 516 | 1.04 | 254 | 1.05 | 148 | 0.91 |
| ALM3 | 43 | 18799 | 9391 | 1.00 | 4778 | 0.98 | 2459 | 0.96 |
| variant (10): logarithmic utility | | | | | | | | |
| ALM1 | 25 | 448 | 214 | 1.05 | 110 | 1.02 | 72 | 0.78 |
| ALM2 | 31 | 1287 | 618 | 1.04 | 306 | 1.05 | 179 | 0.90 |
| ALM3 | 60 | 24414 | 12480 | 0.98 | 6275 | 0.97 | 3338 | 0.91 |
| variant (11): skewness | | | | | | | | |
| ALM1 | 50 | 820 | 390 | 1.05 | 208 | 1.02 | 130 | 0.79 |
| ALM2 | 43 | 1466 | 715 | 1.03 | 396 | 0.93 | 207 | 0.89 |
| ALM3 | 62 | 23664 | 11963 | 0.99 | 6131 | 0.97 | 3097 | 0.96 |

Table 2: Results for nonlinear ALM variants.

| Problem | Constraints | Variables | Blk | CPLEX 9.1 | | OOPS | |
|---|---|---|---|---|---|---|---|
| | | | | time | memory | time | memory |
| C33 | 57,274 | 168,451 | 33 | 292 | 497MB | 344 | 156MB |
| C50 | 130,153 | 382,801 | 50 | 1361 | 1.3GB | 828 | 345MB |
| C70 | 253,522 | 745,651 | 70 | (5254) | OoM | 1627 | 664MB |

Table 3: Comparison of OOPS with CPLEX 9.1.

| $T$ | Blk | $J$ | Scenarios | Constraints | Variables | Iter | Time | Procs | Mach |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 128 | 6 | 12,831,873 | 64,159,366 | 153,982,477 | 42 | 3923 | 512 | BG/L |
| 7 | 64 | 14 | 6,415,937 | 96,239,056 | 269,469,355 | 39 | 4692 | 512 | BG/L |
| 7 | 128 | 13 | 12,831,873 | 179,646,223 | 500,443,048 | 45 | 6089 | 1024 | BG/L |
| 7 | 128 | 21 | 16,039,809 | 352,875,799 | 1,010,507,968 | 53 | 3020 | 1280 | HPCx |

Table 4: Dimensions and solution statistics for very large problems.

by CPLEX, while for larger problems OOPS becomes much more efficient than CPLEX.

### 4.2 Massively Parallel Architecture

In this section we demonstrate the parallel efficiency of our code running on a massively parallel environment. We have run the QP model (8) on two supercomputers: the BlueGene/L service at Edinburgh Parallel Computing Centre (EPCC) in co-processor mode, consisting of 1024 IBM-PowerPC-440 processors running at

| Procs | Mem | time | Cholesky | Solves | MatVectProd |
|---|---|---|---|---|---|
| 16 | 426MB | 2587 (1.00) | 1484 (1.00) | 956 (1.00) | 28.8 (1.00) |
| 32 | 232MB | 1303 (0.99) | 743 (1.00) | 485 (0.98) | 18.0 (0.80) |
| 64 | 132MB | 688 (0.94) | 377 (0.98) | 270 (0.88) | 13.0 (0.55) |
| 128 | 84MB | 348 (0.93) | 187 (0.99) | 139 (0.86) | 9.0 (0.40) |
| 256 | 56MB | 179 (0.90) | 93 (0.99) | 73 (0.82) | 5.8 (0.31) |
| 512 | 46MB | 94 (0.86) | 47 (0.98) | 39 (0.76) | 3.9 (0.23) |

Table 5: Parallel efficiency of OOPS.

| Constraints | Variables | Procs | 0.001 | 0.01 | 0.05 | 0.1 | 0.5 | 1 | 5 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 533,725 | 198,525 | 1 | 14 | 14 | 14 | 14 | 15 | 18 | 18 | 17 |
| | | | 14 | 5 | 5 | 6 | 5 | 5 | 9 | 10 |
| 5,982,604 | 16,316,191 | 32 | 23 | 24 | 23 | 25 | 22 | 24 | 23 | 24 |
| | | | 24 | 11 | 13 | 11 | 13 | 12 | 12 | 14 |
| 70,575,308 | 192,478,111 | 512 | 52 | 45 | 43 | 44 | 42 | 44 | 46 | 46 |
| | | | 52 | 13 | 15 | 15 | 16 | 16 | 23 | 25 |

Table 6: Warmstarting OOPS on efficient frontier problems for a series of $\lambda$.

700Mhz and 512MB of RAM each. The second machine was the 1600-processor HPCx service at Daresbury, with 1GB of memory and 1.7GHz for every processor. Results for these runs are summarized in Table 4. As can be seen OOPS is able to solve a problem with more than $10^9$ variables on HPCx in less than one hour. Table 5 also gives the parallel efficiency for a smaller problem scaling from 16-512 processors on BlueGene. OOPS achieves a parallel efficiency of 86% on 512 processors as compared to 16 processors, with the dominant factorization part of the code even achieving 98% parallel efficiency.

### 4.3 Efficient Frontier

Finally we have run tests calculating the efficient frontier for several large problems with up to 192 million decision variables on BlueGene. For every efficient frontier calculation the mean-variance model was solved for 8 different values of the risk-aversion parameter $\lambda$ using OOPS' warmstarting facilities [16]. Results are gathered in Table 6. For every problem instance, the first line gives iteration numbers for computing points on the efficient frontier from coldstart, while the bottom line gives the iteration count for the warmstarted method. The last two large problems have been solved using 32 and 512 processors (procs), respectively. As can be seen OOPS' warmstart was able to save 45% - 75% percent of total iterations across the different problem sizes, demonstrating that warmstarting capabilities for truly large scale problems are available for interior point methods.

## 5 Conclusion

We have presented a case for solving nonlinear portfolio optimization problems by general purpose structure exploiting interior point solver. We have concentrated on three variations of the classical mean-variance formulations of an Asset and Liability Management problem each leading to a nonlinear programming problem. While these variations have been recognized for some time for their theoretical value, received wisdom is that these models are out of scope for mathematical programming methods. We have shown that in the light of recent progress in structure exploiting interior point solvers, this is no longer true. Indeed nonlinear ALM problems with several of millions of variables are within grasp of the next generation of Desktop PCs, while massively parallel machines can tackle problems with over $10^9$ decision variables.

## References


[1] Consigli, G. & Dempster, M., Dynamic stochastic programming for asset-liability management. *Annals of Operations Research*, **81**, pp. 131–162, 1998.

[2] Mulvey, J. & Vladimirou, H., Stochastic network programming for financial planning problems. *Management Science*, **38**, pp. 1643–1664, 1992.

[3] Zenios, S., Asset/liability management under uncertainty for fixed-income securities. *Annals of Operations Research*, **59**, pp. 77–97, 1995.

[4] Ziemba, W.T. & Mulvey, J.M., *Worldwide Asset and Liability Modeling*. Publications of the Newton Institute, Cambridge University Press: Cambridge, 1998.

[5] Birge, J.R., Decomposition and partitioning methods for multistage stochastic linear programs. *Operations Research*, **33**, pp. 989–1007, 1985.

[6] Ruszczyński, A., Decomposition methods in stochastic programming. *Mathematical Programming B*, **79**, pp. 333–353, 1997.

[7] Blomvall, J. & Lindberg, P.O., A Riccati-based primal interior point solver for multistage stochastic programming. *European Journal of Operational Research*, **143**, pp. 452–461, 2002.

[8] Steinbach, M., Hierarchical sparsity in multistage convex stochastic programs. *Stochastic Optimization: Algorithms and Applications*, eds. S. Uryasev & P.M. Pardalos, Kluwer Academic Publishers, pp. 363–388, 2000.

[9] Gondzio, J. & Grothey, A., Parallel interior point solver for structured quadratic programs: Application to financial planning problems. Technical Report MS-03-001, School of Mathematics, University of Edinburgh, Edinburgh EH9 3JZ, Scotland, UK, 2003. Accepted for publication in *Annals of Operations Research*.

[10] Gondzio, J. & Grothey, A., Solving nonlinear portfolio optimization problems with the primal-dual interior point method. Technical Report MS-04-001, School of Mathematics, University of Edinburgh, Edinburgh EH9 3JZ, Scotland, UK, 2004. Accepted for publication in *European Journal of Operational Research*.

[11] Høyland, K., Kaut, M. & Wallace, S.W., A heuristic for moment-matching scenario generation. *Computational Optimization and Applications*, **24(2/3)**, pp. 169–186, 2003.

[12] Markowitz, H.M., *Portfolio Selection: Efficient Diversification of Investments*. John Wiley & Sons, 1959.

[13] Steinbach, M., Markowitz revisited: Mean variance models in financial portfolio analysis. *SIAM Review*, **43(1)**, pp. 31–85, 2001.

[14] Ogryczak, W. & Ruszczynski, A., Dual stochastic dominance and related mean-risk models. *SIAM Journal on Optimization*, **13(1)**, pp. 60–78, 2002.

[15] Konno, H., Shirakawa, H. & Yamazaki, H., A mean-absolute deviation-skewness portfolio optimization model. *Annals of Operational Research*, **45**, pp. 205–220, 1993.

[16] Gondzio, J. & Grothey, A., Reoptimization with the primal-dual interior point method. *SIAM Journal on Optimization*, **13(3)**, pp. 842–864, 2003.