Multi-period Financial Planning Problem

A set of assets $\mathcal{J} = \{1, ..., J\}$ is given (e.g. bonds, stock, real estate).
At every stage $t = 0, ..., T-1$ we can buy or sell different assets.
The return of asset j at stage t is *uncertain*.

We have to make investment decisions:
what, when and how much to buy or sell

Objectives:

- maximize the final wealth
- minimize the associated risk

Example: *Asset Liability Management*
problem of crucial importance to *pension funds* and *insurance companies*.

- W. Ziemba and J. Mulvey, *Worldwide Asset and Liability Modeling*, Cambridge University Press, 1998.

J. Gondzio OOPS: Parallel IPM Solver for NI

School of Mathematics

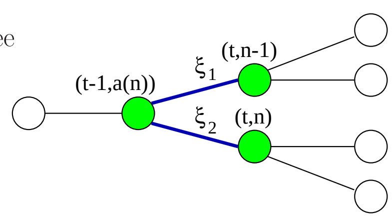
Parallel Solution Techniques in Financial Planning Problems

Jacek Gondzio, Andreas Grothey

ISMP, Copenhagen, August 2003

J. Gondzio OOPS: Parallel IPM Solver for NLP

Modelling: use event tree



and decision variables associated with its nodes (t, n).
Let $a(t, n)$ denote the *ancestor* of node (t, n).
With asset $j \in \mathcal{J}$ at node (t, n) we associate:

$x_{j,t,n}$ the position in asset j in node (t, n);
$x^b_{j,t,n}$ the amount of asset j bought in (t, n);
$x^s_{j,t,n}$ the amount of asset j sold in (t, n).

For any $t: \ 1 \leq t \leq T$, we write the *inventory equation* for asset j at node (t, n)

$$x_{j,t,n} = (1 + r_{j,t,n}) \cdot x_{j,t-1,a(t,n)} + x^b_{j,t,n} - x^s_{j,t,n},$$

where $r_{j,t,n}$ is a return of asset j corresponding to moving from node $(t-1, a(t, n))$ to node (t, n) in the event tree.

J. Gondzio OOPS: Parallel IPM Solver for NI

Financial Planning Problems

- dynamics: multiple decision stages
- stochastics: uncertainty of returns
- curse of dimensionality
- very large-scale optimization
  - sparsity
  - nested block-structure

Solution Techniques

- structure exploitation
- decomposition & parallelisation
- where to decompose?
  - the algorithm (nested Benders decomposition), or
  - the linear algebra in the IPM
- OOPS: Object-Oriented Parallel IPM Solver

Decomposition: Yes, but where to decompose?

Decomposing the linear algebra

- Use interior point methods, because:
  - they are predictable (number of iterations $\mathcal{O}(\log n)$)
  - they can take advantage of the problem structure
  - their linear algebra operations are parallelisable
- Object-Oriented Parallel IPM Solver (OOPS):
  - uses abstract `Matrix` class
  - allows modelling of very complicated structures (including nested ones)
  - uses fast parallel linear algebra
  - reduces memory use
  - runs on any platform which supports `MPI`

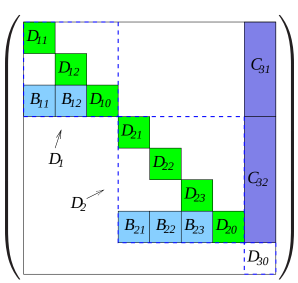
Linear Algebra of IPMs

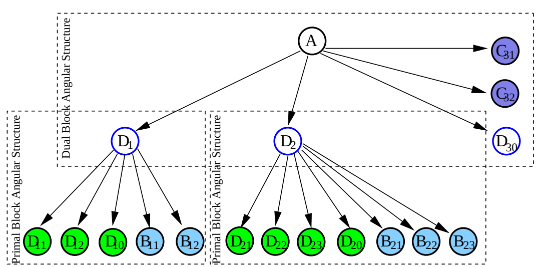
Solve

$$\underbrace{\begin{bmatrix} -Q-\Theta & A^\top \\ A & 0 \end{bmatrix}}_{\Phi\,(QP)} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r \\ h \end{bmatrix} \quad \text{or} \quad \underbrace{\begin{bmatrix} -Q & A^\top \\ A & \Theta \end{bmatrix}}_{\Phi\,(NLP)} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r \\ h \end{bmatrix}$$

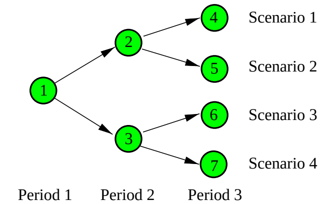
for several right-hand-sides at each iteration

Tree representation of matrices Q and A:

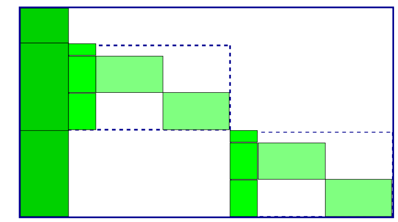




Multistage Stochastic Programming



Scenario Tree



Constraint Matrix

Symmetrical event tree with p realizations at each node and T periods correspond
to

$$p^{T-1}$$

scenarios.

Solution Approaches: (nonexhaustive list, obviously)

For LPs:

- Benders decomposition and its extensions:
  - Benders, *Numerische Mathematik* 4 (1962).
  - Van Slyke and Wets, *SIAM J on Appl. Maths* 17 (1969).
  - Birge, *Operations Research* 33 (1985).
  - Ruszczyński, *Mathematical Programming* 33 (1985).
  - Gassmann, *Mathematical Programming* 47 (1990).
  - Mulvey and Ruszczyński, *Operations Research* 43 (1995).
  - Gondzio and Kouwenberg, *Operations Research* 49 (2001).
  - Linderoth and Wright, *Computational Opt. and Appl.* 24 (2003).
- Interior point methods:
  - Birge and Qi, *Management Science* 34 (1988).
  - Jessup, Yang and Zenios, *SIAM J on Opt.* 4 (1994).
  - Vladimirou and Zenios, *Annals of OR* 90 (1999).

For NLPs:

- Specialized interior point methods:
  - Steinbach, *Hierarchical Sparsity ...*, Uryasev and Pardalos (eds) 2000.
  - Blomvall and Lindberg, *A Riccati Solver ...*, *EJOR* 143, *OMS* 17 (2002)
  - Gondzio and Grothey, *OOPS: Exploiting structure in QPs and NLPs ..*

ALM (continued)

Denote $x_i = (x^s_{i,1}, x^b_{i,1}, x^h_{i,1}, \dots, x^s_{i,J}, x^b_{i,J}, x^h_{i,J})$, and define matrices

$$A = \begin{pmatrix} 1 & -1 & 1 & & & & \\ & & & \ddots & & & \\ & & & & 1 & -1 & 1 \\ -c^s_1 & c^b_1 & 0 & \cdots & -c^s_J & c^b_J & 0 \end{pmatrix}, \quad B_i = \begin{pmatrix} 0 & 0 & 1+r_{i,1} & & & & \\ & & & \ddots & & & \\ & & & & 0 & 0 & 1+r_{i,J} \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{pmatrix}$$

$$Q_i \in I\!R^{3J\times 3J} : \begin{cases} (Q_i)_{3j,3k} = p_i(1-c_t)^2 v_j v_k, \; j,k = 1,\dots,J, & i \in L_T \\ Q_i = 0, & i \notin L_T \end{cases}$$

$$d_i \in R^{1\times 3J} : (d_i)_{3j} = (1-c_t)p_i v_j,$$

where $c^b_j = (1+c_t)\,v_j,\; c^s_j = (1-c_t)\,v_j$.

Rewrite ALM problem as

$$\max_{x,y\geq 0} \; y - \rho[\sum_{i\in L_T} x_i^T Q_i x_i - y^2] \quad \text{s.t.} \quad \begin{aligned} \textstyle\sum_{i\in L_T} d_i^T x_i &= y \\ B_{a(i)} x_{a(i)} &= A x_i \quad \forall i \neq 0 \\ A x_0 &= b e_{J+1}. \end{aligned}$$

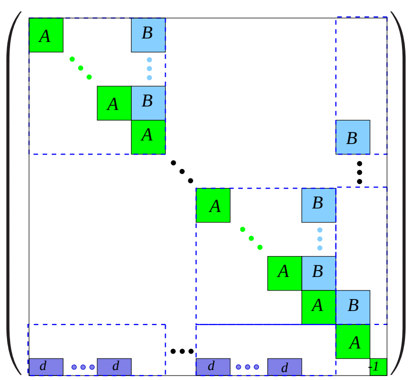
ALM (continued)

Assemble the vectors $x_i, i\in L$ and y into a vector $x = (x_{\sigma(0)}, x_{\sigma(1)}, \dots, x_{\sigma(|L|-1)}, y)$, where σ is a permutation of the nodes $0,\dots,|L|-1$ in a reverse depth-first order.

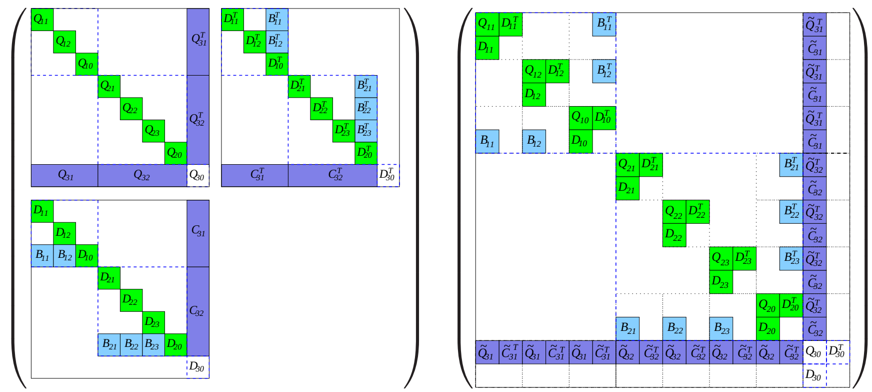
Nested block-diagonal matrix $Q = \mathrm{diag}(Q_{\sigma(0)}, Q_{\sigma(1)}, \dots, Q_{\sigma(|L|-1)}, -1)$.

Constraint matrix A has the form

$$\begin{pmatrix} A & & & B_i & & & & & & 0 \\ & \ddots & & \vdots & & & & & & \vdots \\ & & A & B_i & & & & & & 0 \\ & & & A & & & & & & B_i \\ & & & & \ddots & & & & & \vdots \\ & & & & & A & & & B_i & 0 \\ & & & & & & \ddots & & \vdots & \vdots \\ & & & & & & & A & B_i & 0 \\ & & & & & & & & A & B_i \\ & & & & & & & & & A \\ d_i \cdots d_i & 0 & \cdots & & d_i \cdots d_i & 0 & & & & -1 \end{pmatrix}$$



Structures of A and Q imply structure of Φ:



$$\begin{pmatrix} Q & A^\top \\ A & 0 \end{pmatrix} \qquad P \begin{pmatrix} Q & A^\top \\ A & 0 \end{pmatrix} P^{-1}$$

Asset and Liability Management Problem (ALM)

Final wealth y is the expected value of the final portfolio converted into cash

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

while risk is expressed as its variance $[\mathrm{Var}(X) = I\!E(X^2) - (I\!E(X))^2]$:

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

The ALM problem can then be expressed as

$$\begin{aligned} \max_{x,y\geq 0} \quad & y - \rho[\sum_{i\in L_T} p_i[(1-c_t)\sum_j v_j x^h_{i,j}]^2 - y^2] \\ \text{s.t.} \quad & (1-c_t)\textstyle\sum_{i\in L_T} p_i \sum_j v_j x^h_{i,j} = y \\ & (1+r_{i,j})x^h_{a(i),j} = x^h_{i,j} - x^b_{i,j} + x^s_{i,j}, \quad \forall i \neq 0, j \\ & \textstyle\sum_j (1+c_t)v_j x^b_{i,j} = \sum_j (1-c_t)v_j x^s_{i,j}, \quad \forall i \neq 0 \\ & \textstyle\sum_j (1+c_t)v_j x^b_{0,j} = b. \end{aligned}$$

see, e.g., Steinbach, *Markowitz revisited …*, SIAM Review 43 (2001).

ALM: Extensions

Introduce two more (nonnegative) variables per final scenario $i \in L_t$ to model the positive and negative variation from the mean

$$(1-c_t)\sum_{j=1}^{J} v_j x_{i,j}^h + s_i^+ - s_i^- = y.$$

Since $(s_i^+)^2, (s_i^-)^2$ cannot both be positive the variance is expressed as

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

We model **downside risk** using a semi-variance $I\!E[(X - I\!EX)_-^2]$

$$I\!E[(X - I\!EX)_-^2] = \sum_{i\in L_t} p_i (s_i^+)^2.$$

Downside risk can be taken into account

- in the objective, or
- as a constraint.

Extensions ($x_i = (x_{i,1}, ..., x_{i,J})$ denotes the portfolio in node i)

Standard Markowitz formulation:

$$\max\ y - \sigma \sum_{i\in L_t} p_i (d_i^\top x_i - y)^2 \ \text{s.t.} \quad \begin{array}{lrcll} (C1) & \sum_{i\in L_t} p_i d_i^\top x_i - y & = & 0 & \\ (C2) & B x_{a(i)} - A x_i & = & 0, & i \neq 0 \\ (C3) & A x_0 & = & b & \end{array} \quad \text{(QP)}$$

Risk exposure constrained:

$$\max\ y \quad \text{s.t.} \quad \begin{array}{c} \sum_{i\in L_t} p_i (d_i^\top x_i - y)^2 \le \rho \\ (C1) - (C3) \end{array} \quad \text{(NLP)}$$

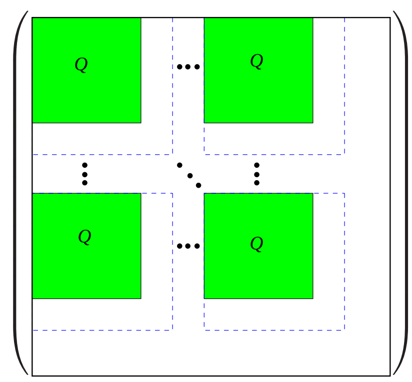
Downside risk constrained:

$$\max\ y \quad \text{s.t.} \quad \begin{array}{rcll} \sum_{i\in L_t} p_i (s_i^+)^2 & \le & \rho & \\ d_i^\top x_i + s_i^+ - s_i^- - y & = & 0, & i \in L_t \\ (C1) - (C3) & & & \end{array} \quad \text{(NLP)}$$

Nonlinear utility function:

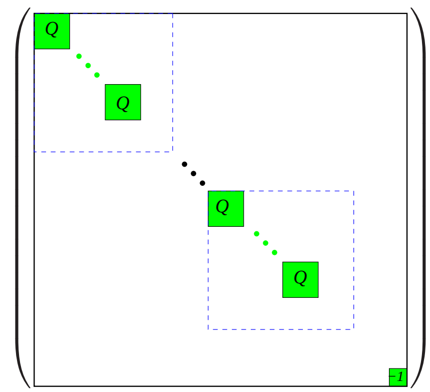
$$\max\ \log(1+y) \quad \text{s.t.} \quad \begin{array}{rcll} \sum_{i\in L_t} p_i (s_i^+)^2 & \le & \rho & \\ d_i^\top x_i + s_i^+ - s_i^- - y & = & 0, & i \in L_t \\ (C1) - (C3) & & & \end{array} \quad \text{(NLP)}$$

Variance representation



$\sum_{i\in L_T} p_i [(1-c_t)\sum_j v_j x_{i,j}^h - y]^2$

dense, convex QP



$\sum_{i\in L_T} p_i (1-c_t)^2 [\sum_j v_j x_{i,j}^h]^2 - y^2$

sparse, nonconvex QP

Results (nonconvex QP formulation):

Problem	Stages	Blocks	Assets	Total Nodes	constraints	variables
ALM1	5	10	5	11111	66.667	166.666
ALM2	6	10	5	111111	666.667	1.666.666
ALM3	6	10	10	111111	1.222.222	3.333.331
ALM4	5	24	5	346201	2.077.207	5.193.016
ALM5	4	64	12	266305	3.461.966	9.586.981
UNS1	5	35	5	360152	2.160.919	5.402.296
ALM6	4	120	5	1742521	10.455.127	26.137.816
ALM7	4	120	10	1742521	19.167.732	52.275.631

Problem	1 proc		2 procs		k procs		
	time (s)	iter	time (s)	speed-up	time (s)	speed-up	k
ALM1	72.8	12	35.2	2.07	12.2	5.97	6
ALM2	1528	19	758	2.01	309	4.95	5
ALM3	7492	29	3661	2.04	1464	5.12	5
ALM4	5434	31	2717	2.00	905	6.00	6
ALM5	6842	11	3480	1.97	1150	5.95	6
UNS1	5252	15	2823	1.86	1108	4.74	5
ALM6		15			1294	-	16
ALM7		23			7058	-	16

24 750MHz UltraSparc-III processors, 48GB of shared memory

Results (NLP formulation): textbook SQP implemented

Semi-variance constrained → quadratically constrained problem.

Problem	Stages	Blocks	Assets	Total Nodes	constraints	variables
ALM1	5	10	5	11111	76.668	186.667
ALM2	6	10	5	111111	766.668	1.866.667
ALM4	5	24	5	346201	2.408.984	5.856.569
UNS1	5	35	5	360152	2.503.994	6.088.445

Problem	1 proc		2 procs		k procs		
	iter	time (s)	time (s)	speed-up	time (s)	speed-up	k
ALM1	36	218	107	2.04	44	4.95	5
ALM2	45	3456	1737	1.98	703	4.92	5
ALM4	67	11744	5902	1.98	1973	5.95	6
UNS1	42	14705	7949	1.85	3109	4.73	5

24 750MHz UltraSparc-III processors, 48GB of shared memory

Conclusions: IPMs for NLP offer:

- flexibility: complicated (nested) structures handled
- efficiency: QPs/NLPs with up to 50 million variables solved
- parallelism: near perfect speed-ups achieved

→ can solve complicated financial planning problems.

Object-Oriented Parallel IPM Solver (OOPS):

`http://www.maths.ed.ac.uk/~gondzio/parallel/solver.html`

- Gondzio and Sarkissian, *Mathematical Programming* 96 (2003).
- Gondzio and Grothey, *SIAM J. on Optmization* 13 (2003).
- Gondzio and Grothey, *Parallel IPM solver for structured QPs: applicatio to financial planning problems*, Tech. Rep. MS-03-001, School of Math University of Edinburgh, April 2003.

Papers available from:
`http://www.maths.ed.ac.uk/~gondzio/`