OOPS: a structure-exploiting parallel solver

Marco Colombo

School of Mathematics
University of Edinburgh

CARIPLO Workshop on Numerical Stochastic Programming
Edinburgh, 4 September 2008

Scope of this talk

Interior point methods

Exploiting structure and parallelism

Multi-period financial planning problem

Interior point methods

- KKT conditions for optimality

$$
\begin{aligned}
\min \quad & c^\top x + \tfrac{1}{2} x^\top Q x \\
\text{s.t.} \quad & Ax = b \\
& x \geq 0
\end{aligned}
\qquad
\begin{bmatrix} Ax - b \\ -Qx + A^\top y + s - c \\ XSe \end{bmatrix} = 0 \quad x, s \geq 0
$$

where $X = \mathrm{diag}(x),\ S = \mathrm{diag}(s)$

Interior point methods

- KKT conditions for optimality

$$\begin{array}{rl} \min & c^\top x + \frac{1}{2}x^\top Q x \\ \text{s.t.} & Ax = b \\ & x \geq 0 \end{array} \qquad \begin{bmatrix} Ax - b \\ -Qx + A^\top y + s - c \\ XSe \end{bmatrix} = 0 \quad x, s \geq 0$$

where $X = \mathrm{diag}(x),\ S = \mathrm{diag}(s)$

- Perturb complementarity and enforce strict positivity

$$XSe = \mu e \qquad x, s > 0$$

Solve the perturbed KKT conditions with Newton's method

$$\begin{bmatrix} A & 0 & 0 \\ -Q & A^\top & I \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \end{bmatrix} = \begin{bmatrix} b - Ax \\ c + Qx - A^\top y - s \\ -XSe + \mu e \end{bmatrix}$$

Interior point methods (cont.)

Perturb the complementarity conditions:

$$XSe = \mu e$$

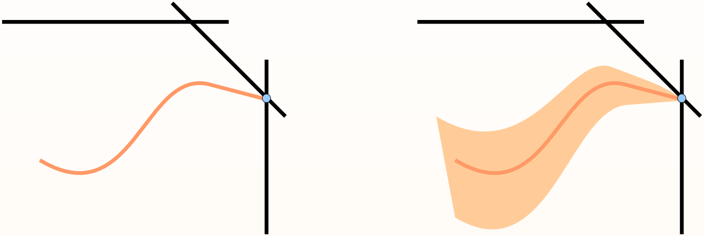
IPMs solve a sequence of problems parametrised by μ.

Let $\mu \to 0$:

- The perturbed conditions better approximate the original KKT conditions
- The solution traces a continuous path from the starting point to the optimal solution (central path)

Centrality

IPMs follow the central path to find the optimal solution.
The iterates lie in some neighbourhood of the central path.



Polynomial complexity:

in theory: $\mathcal{O}(\sqrt{n})$ or $\mathcal{O}(n)$ iterations

in practice: $\mathcal{O}(\ln n)$ iterations

Linear algebra

The Newton system can be reduced to

$$\underbrace{\begin{bmatrix} -Q-\Theta & A^\top \\ A & 0 \end{bmatrix}}_{\Phi} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r \\ h \end{bmatrix}, \qquad \Theta = X^{-1}S$$

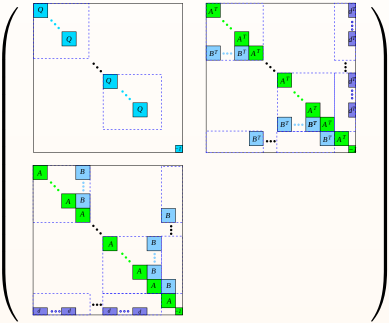
At each iteration:

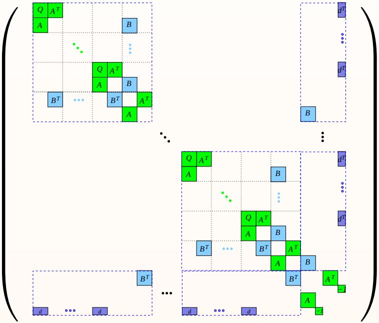
- factorize $\Phi = LDL^\top$
- backsolve to compute the search direction $(\Delta x, \Delta y)$

Key to efficient implementation is exploiting structure of Φ

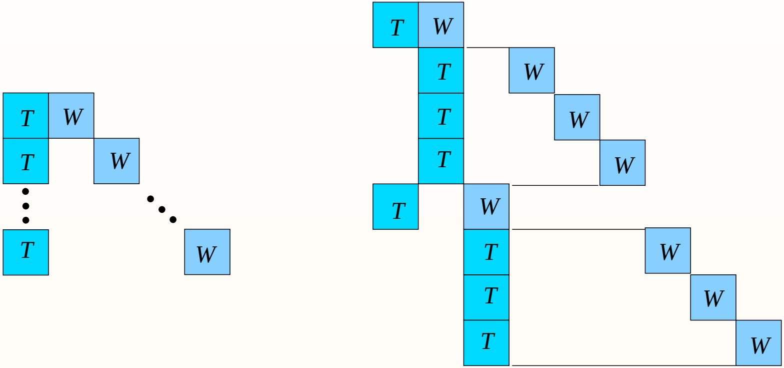
Structures of A and Q imply structure of Φ

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

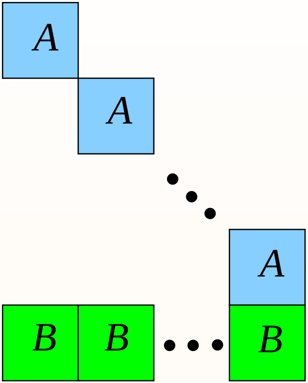




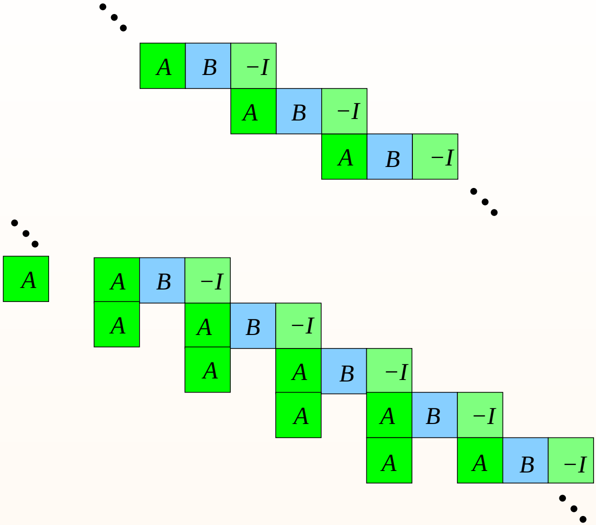
Sources of structure I: Uncertainty



Sources of structure II: Common resources



Sources of structure III: Dynamics



OOPS - Object Oriented Parallel (Interior Point) Solver

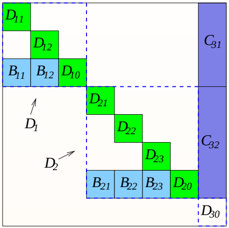
Key advantages of exploiting the structure in the problem:

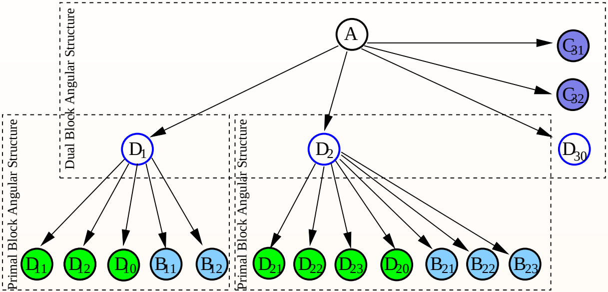
- Faster linear algebra
- Reduced memory use (by use of implicit factorization)
- Possibility to exploit (massive) parallelism
- We assume that structure is known!

OOPS is a general purpose (parallel) Interior Point solver

- Written in C with an object-oriented design
- Not tuned to any particular hardware or problem
- OOPS currently solves LP/QP problems
- NLP extension solves nonlinear financial planning problems

Tree representation of the matrix structure





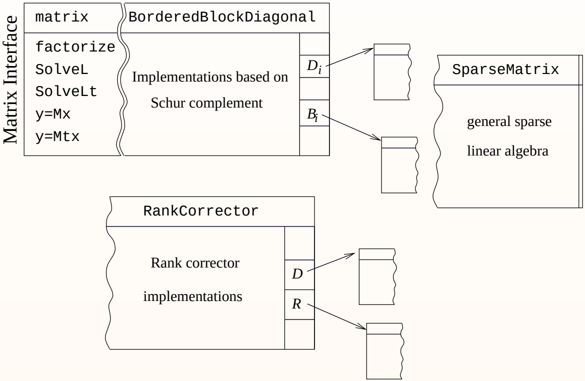
Every block should have a structure-exploiting linear algebra:

- Blocks may be nested
- Blocks may have different structure

Object-oriented linear algebra implementation

Every node in the tree has its own linear algebra implementation

- An implementation realises an abstract linear algebra interface
- Different implementations for different structures are available



Example: bordered block-diagonal structure

Factorize $\Phi = LDL^\top$

$$\Phi = \begin{bmatrix} \Phi_1 & & & B_1^\top \\ & \ddots & & \vdots \\ & & \Phi_n & B_n^\top \\ B_1 & \cdots & B_n & \Phi_c \end{bmatrix} L = \begin{bmatrix} L_1 & & & \\ & \ddots & & \\ & & L_n & \\ L_{1,c} & \cdots & L_{n,c} & L_c \end{bmatrix} D = \begin{bmatrix} D_1 & & & \\ & \ddots & & \\ & & D_n & \\ & & & D_c \end{bmatrix}$$

Cholesky-like factors can be obtained by Schur-complement:

$$\begin{aligned} \Phi_i &= L_i D_i L_i^\top \\ L_{i,c} &= B_i (D_i L_i^\top)^{-1} \\ C_i &= L_{i,c} D_i L_{i,c}^\top \\ C &\equiv \Phi_c - \sum_i C_i = L_c D_c L_c^\top \end{aligned}$$

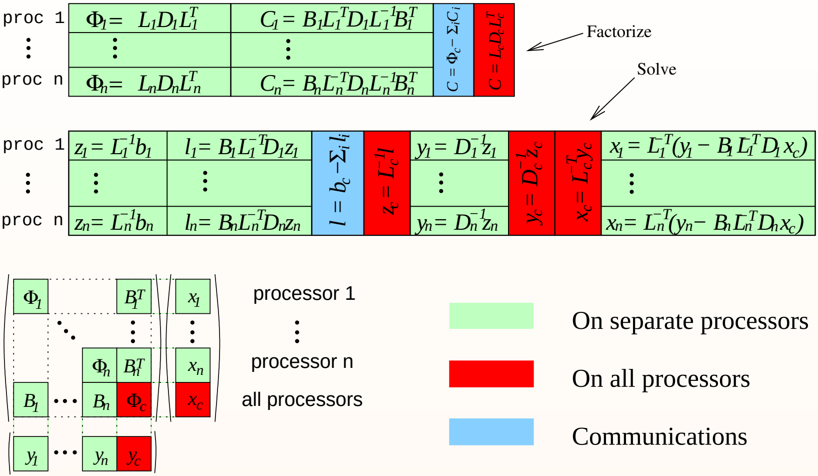
Example (cont.)

System $\Phi x = b$ can then be solved by

$$\begin{aligned} z_i &= L_i^{-1} b_i & \qquad x_c &= L_c^{-\top} D_c^{-1} z_c \\ z_c &= L_c^{-1}(b_c - \sum L_{i,c} z_i) & \qquad x_i &= L_i^{-\top}(D_i^{-1} z_i - L_{i,c}^{\top} x_c) \end{aligned}$$

- Operations (Cholesky, Solve, Product) are only performed on sub-blocks
- We can also exploit structure in sub-blocks

Exploiting parallelism in computations and storage



Multi-period financial planning problem

- A set of assets $\mathcal{J} = \{1, ..., J\}$ is given.
- Initial investment b.
- At every stage $t = 0, \ldots, T-1$ we can buy or sell any assets.
- The return of asset j at stage t is uncertain.

Competing objectives:

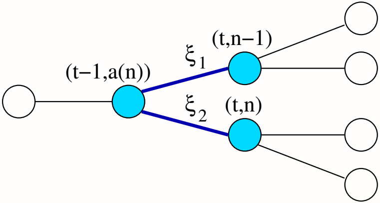
- maximize the final wealth
- minimize the associated risk

Mean-Variance formulation (Markowitz): $\max \boldsymbol{E}(X) - \rho \mathrm{Var}(X)$.

X value of the final portfolio

ρ investor's attitude to risk

Modelling with event tree



With asset $j \in \mathcal{J}$ at node $i = (t, n)$ we associate:

$x^h_{i,j}$	position in asset j at node i
$x^b_{i,j}, x^s_{i,j}$	amount of asset j bought/sold at node i
v_j	value of asset j
$r_{j,t}$	return of asset j when held at time t
L_i, C_i	liabilities/cash contributions at node i

Asset and Liability Management Problem I

Objective:

$$\begin{aligned} \boldsymbol{E}(X) &= (1-c_t)\sum_{i\in L_T} p_i \sum_j v_j x^h_{i,j} = y \\ \mathrm{Var}(X) &= \sum_{i\in L_T} p_i (1-c_t)^2 \Big[\sum_j v_j x^h_{i,j}\Big]^2 - y^2 \end{aligned}$$

Constraints at each node i:

$$\begin{aligned} x^h_{i,j} &= (1+r_{i,j})x^h_{a(i),j} + x^b_{i,j} - x^s_{i,j} \quad (\mathit{inventory}) \\ \sum_j (1+c_t) v_j x^b_{i,j} + L_i &= \sum_j (1-c_t) v_j x^s_{i,j} + C_i \qquad (\mathit{cash\ balance}) \end{aligned}$$

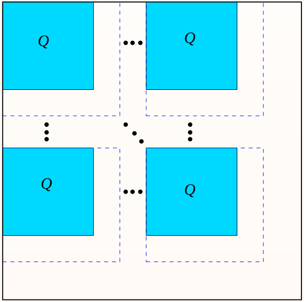
Asset and Liability Management Problem II

$$
\begin{array}{llcll}
\max\limits_{x,y\geq 0} & y - \rho\Big[\sum\limits_{i\in L_T} p_i[(1-c_t)\sum\limits_j v_j x^h_{i,j}]^2 - y^2\Big] & & & \\
\text{s.t.} & (1-c_t)\sum\limits_{i\in L_T} p_i \sum\limits_j v_j x^h_{i,j} & = & y & \\
 & (1+r_{i,j})x^h_{a(i),j} & = & x^h_{i,j} - x^b_{i,j} + x^s_{i,j} & \forall i, \forall j \\
 & \sum\limits_j (1+c_t)v_j x^b_{i,j} + L_i & = & \sum\limits_j (1-c_t)v_j x^s_{i,j} + C_i & \forall i \\
 & \sum\limits_j (1+c_t)v_j x^b_{0,j} & = & b &
\end{array}
$$

Structure of the objective I

Straightforward representation:

$$\begin{aligned} \boldsymbol{E}(X) - \rho \mathrm{Var}(X) &= \boldsymbol{E}(X) - \rho[\boldsymbol{E}(X^2) - \boldsymbol{E}(X)^2] \\ &= \sum_{i \in L_T} p_i \sum_j v_j x_{ij}^h - \rho \left[\sum_{i \in L_T} p_i \sum_j (v_j x_{ij}^h)^2 - [\sum_{i \in L_T} p_i \sum_j v_j x_{ij}^h]^2 \right] \end{aligned}$$



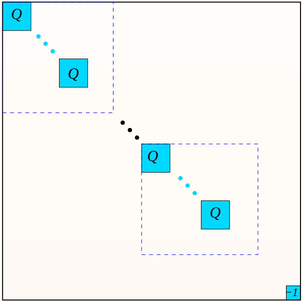
Dense, positive semidefinite Hessian

Structure of the objective II

Alternative representation:

$$\boldsymbol{E}(X) - \rho \mathrm{Var}(X) = y - \rho \Big[\sum_{i \in L_T} p_i \sum_j (v_j x_{ij}^h)^2 - y^2 \Big]$$

$$\text{where: } y = \sum_{i \in L_T} p_i \sum_j v_j x_{ij}^h$$



Sparse, indefinite Hessian

Performance of OOPS

Problem	Stgs	Blks	Assets	Scens	Cons	Vars	iter	time	procs
ALM1	5	10	5	11k	66k	166k	14	86	1
ALM2	6	10	5	111k	666k	1.6M	22	387	5
ALM3	6	10	10	111k	1.2M	3.3M	29	1638	5
ALM4	5	24	5	346k	2.1M	5.2M	33	856	8
ALM5	4	64	12	266k	3.4M	9.6M	18	1195	8
ALM6	4	120	5	1.7M	10.4M	26.1M	18	1470	16
ALM7	4	120	10	1.7M	19.1M	52.2M	19	8465	16
BG/L1	7	128	6	12.8M	64.1M	153.9M	42	3923	512
BG/L2	7	64	14	6.4M	96.2M	269.4M	39	4692	512
BG/L3	7	128	13	12.8M	179.6M	500.4M	45	6089	1024
HPCx	7	128	21	16.0M	352.8M	1,010M	53	3020	1280

Conclusions and future work

- OOPS provides an efficient implementation of a structure-exploiting parallel software
- Structure can be exploited both at the linear algebra level and algorithmically (structured warmstarts)
- Application to grid computing
- Incorporation of iterative solvers (strucured preconditioners)
- Integration within a structured modelling language