Optimization under Uncertainty: Large Scale & Parallelisation

Andreas Grothey

School of Mathematics, University of Edinburgh

CDT Guest Lecture, October 2014

Overview

Research Interests

- Optimization under Uncertainty: Stochastic Programming
- Interior Point Methods
- Exploitation of Problem Structure
- High Performance Computing & Parallelisation
- Applications: Energy, Finance, Telecommunications

Example Problem: Asset and Liability Management (ALM)

Consider the following **multiperiod Financial Planning Problem** (e.g. for Pension Funds):

- A set of assets $\mathcal{J} = \{1, \ldots, J\}$ in which we can invest is given.
- At various points in time $t = 0, \ldots, T$ we can rebalance our portfolio (revise investment decisions). This will incur transaction costs.
- An asset j held between time periods t and $t+1$ will incur a return $r_{j,t}$. $[x_j \to (1 + r_{j,t})x_j]$
- At time period t we need to make a payment l_t and receive contributions c_t.
- We are given an initial amount b to invest.
- The objective is to maximize "financial health" of the fund.

ALM: Mathematical Model

Variables:

$x^h_{j,t}$ money invested in asset j at time t.

$x^b_{j,t}$ amount of asset j bought at time t.

$x^s_{j,t}$ amount of asset j sold at time t.

Constraints:

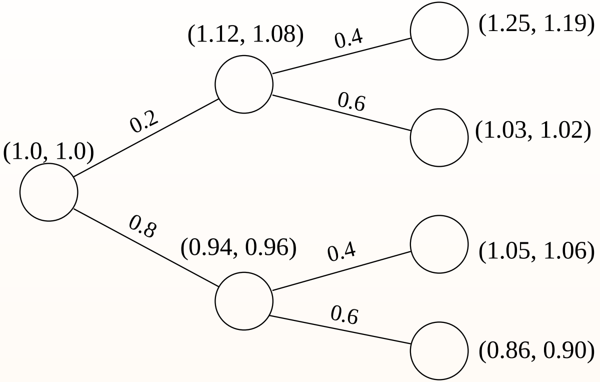
- Cash Balance
  (selling and buying must balance at every time stage)
- Inventory
  (Keep stock of assets we have from one period to the next)

Objective:

- Maximize final wealth

Scenario Tree

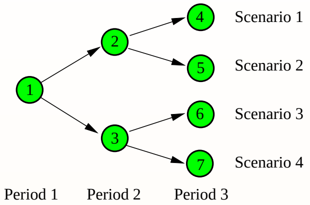
Asset returns are random: Capture evolution by scenario tree:



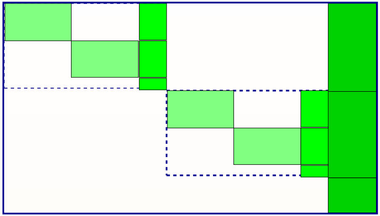
In year 1, assets A and B have two possible returns: (-6%,-4%) and (+12%,+8%) with probabilities 0.2 and 0.8, respectively.

In year 2, these returns are (-8%,-6%) and (+12%,+10%) with probabilities 0.4 and 0.6, respectively.

Multistage Stochastic Programming



Scenario Tree



Constraint Matrix

$\Rightarrow$ **nested** column bordered block-diagonal constraint matrix
Symmetrical event tree with K realizations/node and T periods corresponds to

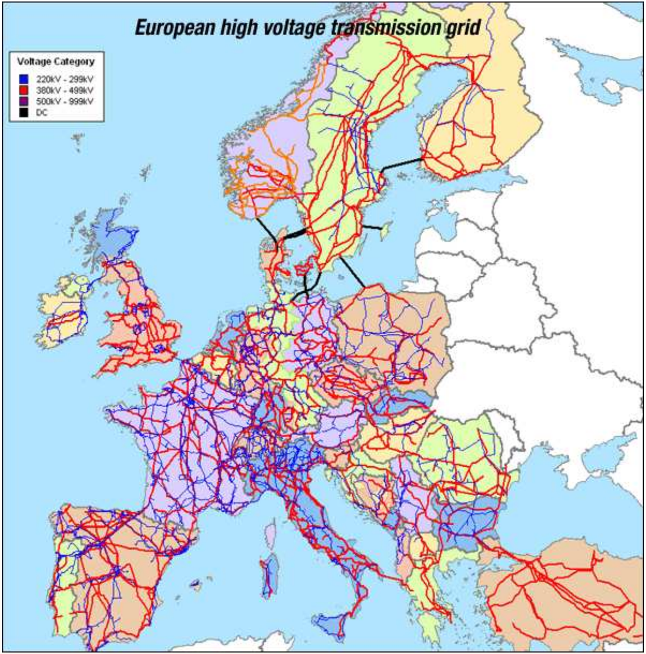
$$K^{T-1} \text{ scenarios} \qquad\qquad \frac{K^T - 1}{K - 1} \text{ nodes (blocks)}$$

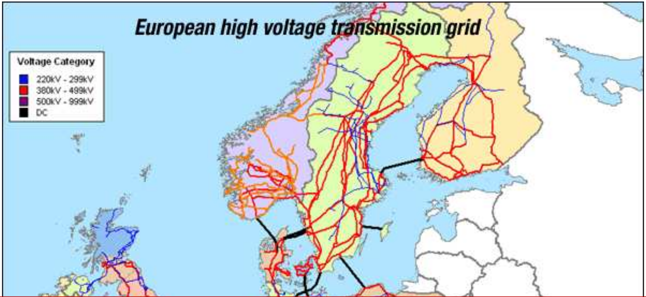
Realistic applications can have huge scenario trees!

Applications

(Multistage) Stochastic Programming has many applications

- Portfolio Optimization
  ("Asset and Liability Management", various risk measures)
- Robust Network Design with Uncertain Demand
  ("Security constrained optimal power flow" - Pan-European network has 20000 lines)
- Electricity Generation Planning (involving hydro or wind)
  ("Stochastic Unit Commitment")
- Cost-optimal routing in telecommunications with uncertain demand
  ("Top-percentile pricing")


European high voltage transmission grid
Voltage Category
220kV - 299kV
380kV - 499kV
500kV - 999kV
DC



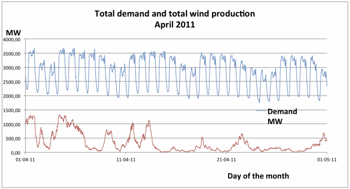
Security Contrained Optimal Power Flow

- "n-1"- (or even "n-2"-security) requires the inclusion of many contingency scenarios.
- Pan-European system has 13000 nodes and 20000 lines

$\Rightarrow$ Resulting SCOPF model would have $\approx 10^{10}$ variables.

- Only a few contingencies are critical for operation of the system (but which ones)?

Stochastic Unit Commitment with Wind Integration



Source: Udo,Wind energy in the Irish power system, 2011

Issues

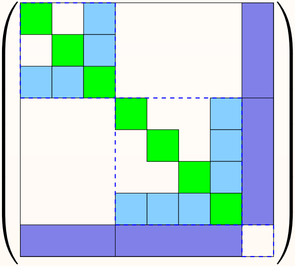
- How to plan power systems operations to deal with wind uncertainty?
- Network constraints
- Decomposition based solution methods

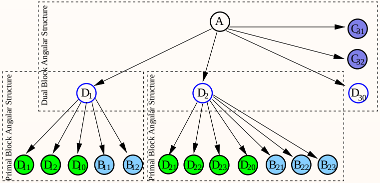
Ken McKinnon, Andreas Grothey, Tim Schulze

OOPS: Object Oriented Parallel Solver

OOPS

- OOPS is an IPM implementation, that can exploit (nested) block structures through object oriented linear algebra
- Solved (multistage) stochastic programming problems from portfolio management with over 10^9 variables
  ($\approx$ 2h on 1280 processors)





Linear Algebra of IPMs

Main work: solve

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

for several right-hand-sides at each iteration

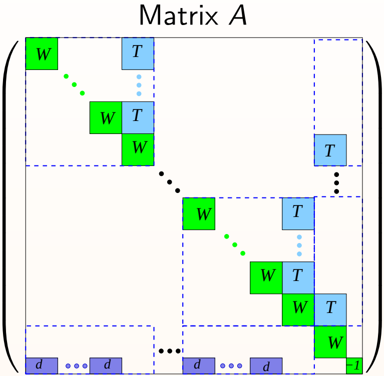
Two stage solution procedure

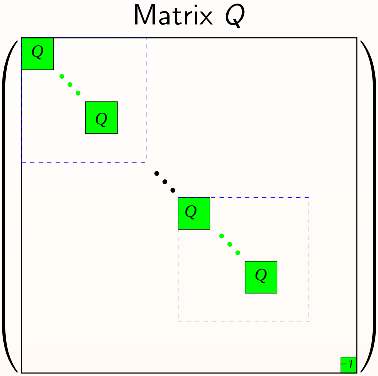
- factorize $\Phi = LDL^\top$
- backsolve(s) to compute direction $(\Delta x, \Delta y)$ + corrections

$\Rightarrow$ Φ changes numerically but not structurally at each iteration

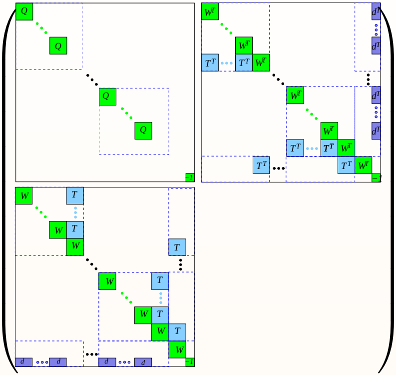
Key to efficient implementation is exploiting structure of Φ in these two steps

ALM: Structure of matrices A and Q:

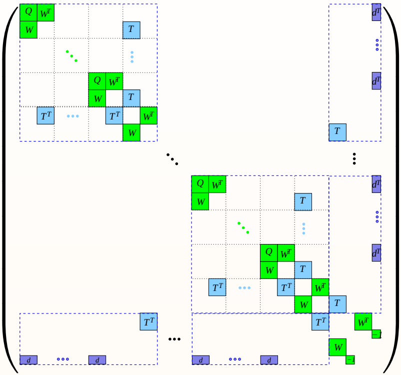




Structures of A and Q imply structure of Φ:

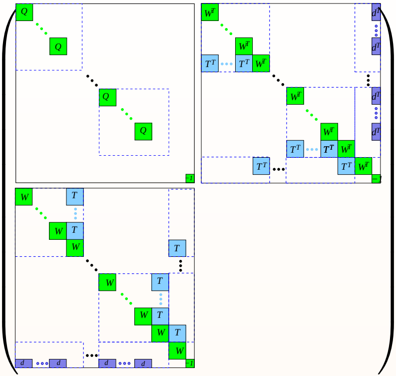


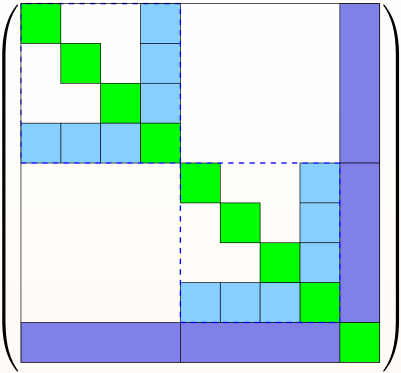
$$\begin{pmatrix} Q & A^\top \\ A & 0 \end{pmatrix}$$



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

Structures of A and Q imply structure of Φ:





$$\begin{pmatrix} Q & A^\top \\ A & 0 \end{pmatrix}$$

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

Nested bordered block-diagonal structure in Augmented System!

Exploiting Structure: Bordered block-diagonal matrix

$$\underbrace{\begin{pmatrix} \Phi_1 & & & B_1^\top \\ & \ddots & & \vdots \\ & & \Phi_n & B_n^\top \\ B_1 & \cdots & B_n & \Phi_0 \end{pmatrix}}_{\color{red}\Phi} = \underbrace{\begin{pmatrix} L_1 & & & \\ & \ddots & & \\ & & L_n & \\ L_{1,0} & \cdots & L_{n,0} & L_c \end{pmatrix}}_{\color{red}L} \underbrace{\begin{pmatrix} D_1 & & & \\ & \ddots & & \\ & & D_n & \\ & & & D_c \end{pmatrix}}_{\color{red}D} \underbrace{\begin{pmatrix} L_1^\top & & & L_{1,0}^\top \\ & \ddots & & \vdots \\ & & L_n^\top & L_{n,0}^\top \\ & & & L_c^\top \end{pmatrix}}_{\color{red}L^\top}$$

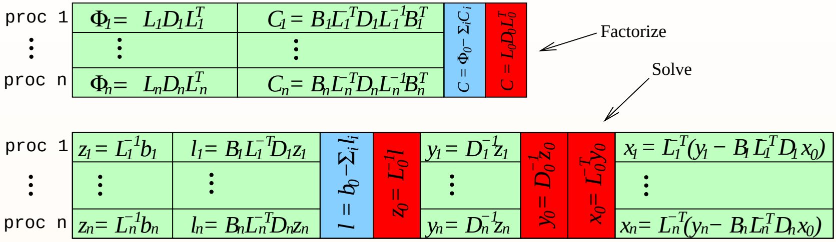
- Cholesky-like factors can be obtained by Schur-complement:

$$\begin{aligned} \Phi_i &= L_i D_i L_i^\top & \qquad L_{i,0} &= B_i L_i^{-\top} D_i^{-1}, \quad i = 1, \dots n \\ C &= \Phi_0 - \textstyle\sum_{i=1}^n L_{i,0} D_i L_{i,0}^\top & \qquad C &= L_c D_c L_c^\top \end{aligned}$$

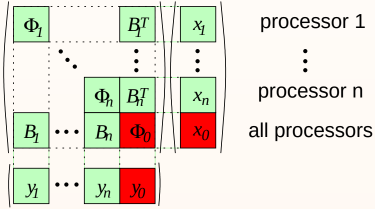
- And the system $\Phi x = b$ can be solved by

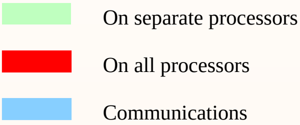
$$\begin{aligned} z_i &= L_i^{-1} b_i \\ z_0 &= L_c^{-1}(b_0 - \sum L_{i,0} z_i) \\ y_i &= D_i^{-1} z_i \end{aligned} \qquad \begin{aligned} x_0 &= L_c^{-\top} y_0 \\ x_i &= L_i^{-\top}(y_i - L_{i,0}^\top x_0) \end{aligned}$$

- Distribution of computations:

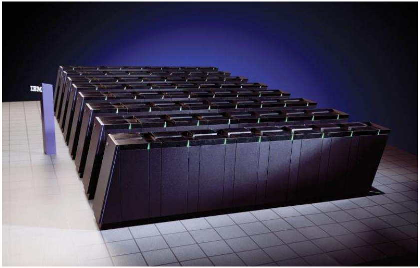


- Storage:





High Performance Computing



BlueGene/L (Edinburgh, Scotland)

- 2048 Processors
- 0.7GHz, 256Mb
- $R_{max} = 4.7$ TFlops

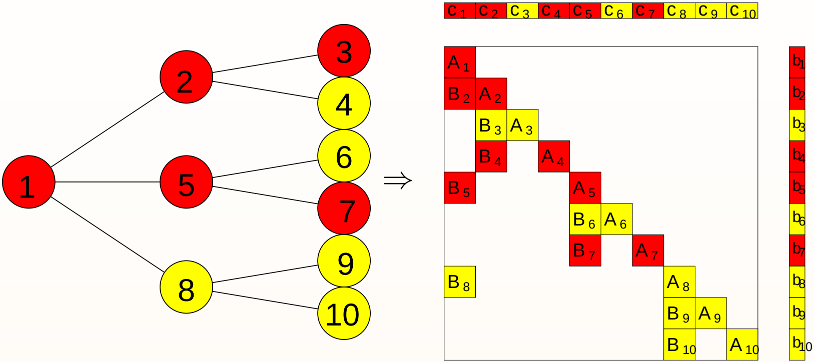
HPCx (Daresbury, England)

- 1600 IBM Power-4 Processors
- 1.7GHz, 800Mb
- $R_{max} = 6.2$ TFlops

Results

Problem	Stgs	Blk	$\|J\|$	Scenarios	Constraints	Variables	iter	time	procs	machine
ALM1	5	10	5	11.111	66.667	166.666	14	86	1	SunFire 15K
ALM2	6	10	5	111.111	666.667	1.666.666	22	387	5	"
ALM3	6	10	10	111.111	1.222.222	3.333.331	29	1638	5	"
ALM4	5	24	5	346.201	2.077.207	5.193.016	33	856	8	"
UNS1	5	35	5	360.152	2.160.919	5.402.296	27	872	8	"
ALM5	4	64	12	266.305	3.461.966	9.586.981	18	1195	8	"
ALM6	4	120	5	1.742.521	10.455.127	26.137.816	18	1470	16	"
ALM7	4	120	10	1.742.521	19.167.732	52.275.631	19	8465	16	"
ALM8	7	128	6	12.831.873	64.159.366	153.982.477	42	3923	512	BlueGene
ALM9	7	64	14	6.415.937	96.239.056	269.469.355	39	4692	512	BlueGene
ALM10	7	128	13	12.831.873	179.646.223	500.443.048	45	6089	1024	BlueGene
ALM11	7	128	21	16.039.809	352.875.799	1.010.507.968	53	3020	1280	HPCx

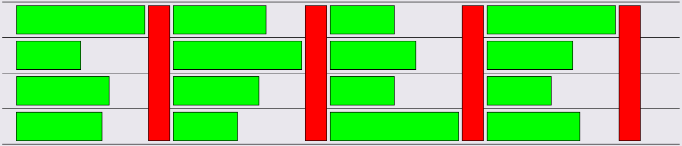
(Multilevel) Scenario Tree Approximations



- Approximate large problem on reduced tree
- Can we to successive approximations
- Very successfully done for problems in physical space (multigrid), can this be done for probability space?

Asynchronous computation

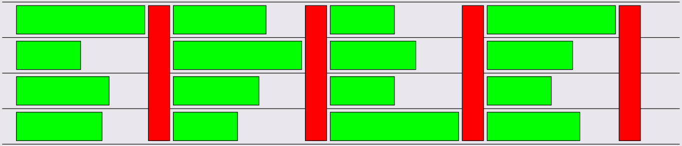
Synchronous Parallel Computation



- Global operations result in parallelisation barriers
- Especially pronounced for massive parallelism (> 1000 procs)

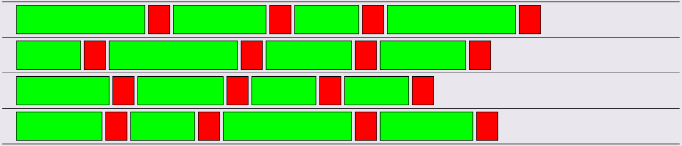
Asynchronous computation

Synchronous Parallel Computation



- Global operations result in parallelisation barriers
- Especially pronounced for massive parallelism ($>$ 1000 procs)

Asynchronous Parallel Computation



- How to organise communications?
- Does this still converge?