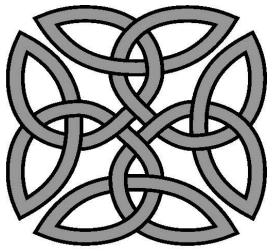


School of Mathematics

Solving Very Large Financial Planning Problems on Blue Gene

Andreas Grothey, University of Edinburgh

joint work with **Jacek Gondzio, Marco Colombo**

Overview

- Asset and Liability Management
  - Mean-Variance Formulation/Stochastic Programming
  - Structure of Problem
- OOPS (Interior Point Solver)
  - Interior Point Methods
  - Structure Exploitation through Object Oriented Design
- Challenges in Massively Parallel Implementation
  - Communications $\Rightarrow$ Data Management
  - Granularity of Computations
- Further Developments

Portfolio Optimization: Asset and Liability Management

- 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* (but distribution is known).

We have to make investment decisions: what to buy or sell, at which time stage

Objectives:

- maximize the final wealth
- minimize the associated risk

$\Rightarrow$ Mean Variance formulation:
$\max \mathbb{E}(X) - \rho \mathrm{Var}(X)$

Extensions (leading to nonlinear models)

- different risk measures (one-sided, expected shortfall/value at risk)
- (nonlinear) utility functions

Modelling: Multiperiod Mean-Variance Model

Variables:

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

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

Parameters:

v_j value of asset j

$r_{j,t}$ return of asset j when held at time t

L_t, C_t liabilities/cash contributions at time t

Objective:

$$\max I\!E(X) - \rho \mathrm{Var}(X), \qquad X = (1-c_t)\sum_j v_j x^h_{j,T}$$

Constraints:

$$\begin{aligned} x^h_{j,t} &= (1+r_{t-1,j})x^h_{j,t-1} - x^s_{j,t-1} + x^b_{j,t-1} && \text{(inventory)} \\ C_t + (1-c_t)\sum_j v_j x^s_{j,t} &= L_t + (1+c_t)\sum_j v_j x^b_{j,t} && \text{(cash balance)} \end{aligned}$$

Modelling: Multiperiod Mean-Variance Model

Variables:

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

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

Parameters:

v_j value of asset j

$r_{j,t}$ return of asset j when held at time t

L_t, C_t liabilities/cash contributions at time t

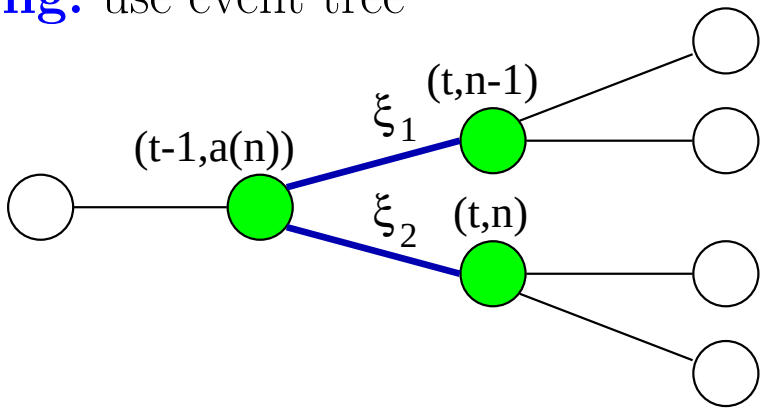
Objective:

$$\max I\!E(X) - \rho \mathrm{Var}(X), \qquad X = (1-c_t)\sum_j v_j x^h_{j,T}$$

Constraints:

$$\begin{aligned} x^h_{j,t} &= (1+r_{t-1,j})x^h_{j,t-1} - x^s_{j,t-1} + x^b_{j,t-1} && \text{(inventory)} \\ C_t + (1-c_t)\sum_j v_j x^s_{j,t} &= L_t + (1+c_t)\sum_j v_j x^b_{j,t} && \text{(cash balance)} \end{aligned}$$

Stochastic Programming: use event tree



- **Stages** represent points in time at which decisions are taken.
- **Nodes** represent possible futures (given by asset returns and probabilities):

$r_j^{(t,n)}$ return of asset j at node (t,n).

$p_{(t,n)}$ probability of reaching node (t,n).

$L_{(t,n)}$, $C_{(t,n)}$ liability payment/cash-contribution at node (t,n).

Asset and Liability Management as Stochastic Program

With every node $i = (t, n)$ we associate **variables**:

$$\begin{aligned} x^h_{j,i} &\text{ the position in asset } j \text{ at node } i; \\ x^b_{j,i}, x^s_{j,i} &\text{ the amount of asset } j \text{ bought/sold at node } i. \end{aligned}$$

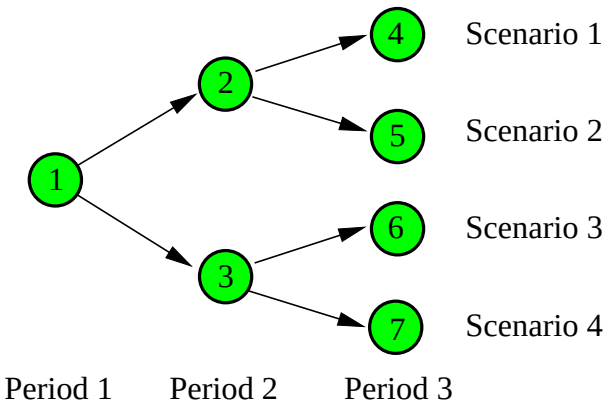
and **constraints**:

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

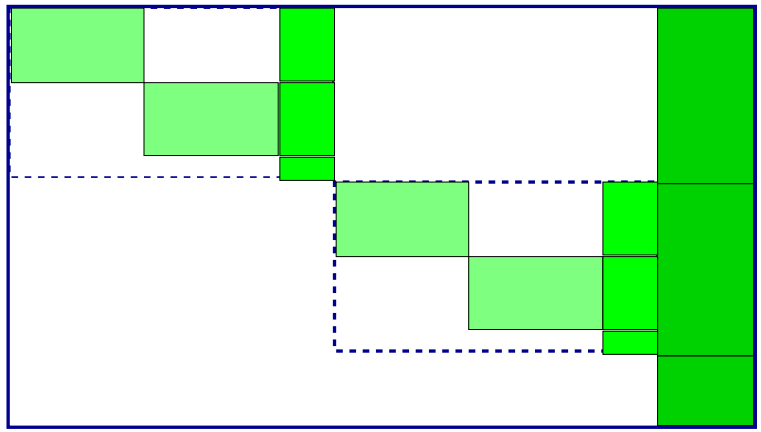
Objective is sum over final stage scenarios

$$\begin{aligned} \min I\!E(X) - \rho \mathrm{Var}(X) &= I\!E(X) - \rho I\!E[(X - E(X))^2] \\ &= (1 - c_t) \sum_{i \in L_T} p_i \sum_j v_j x^h_{i,j} - \rho \left[\sum_{i \in L_T} p_i [\cdots] \right] \end{aligned}$$

Multistage Stochastic Programming



Scenario Tree



Constraint Matrix

Symmetrical event tree with K realizations at each node and T periods corresponds to

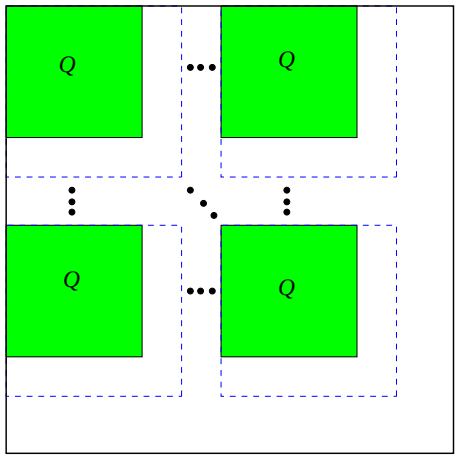
$$K^{T-1} \text{ scenarios} \qquad \frac{K^T - 1}{K - 1} \text{ nodes (blocks)}$$

Structure of Objective

$$\mathbb{E}(X) - \rho \mathrm{Var}(X) = \mathbb{E}(X) - \rho[\mathbb{E}(X^2) - \mathbb{E}(X)^2]$$

$$= \sum_{i \in L_T} p_i \sum_j v_j x_{ij}^h - \rho \left[\sum_{i \in L_T} p_i \sum_j (v_j x_{ij}^h)^2 - [\sum_{i \in L_T} p_i \sum_j v_j x_{ij}^h]^2 \right]$$

$\Rightarrow$

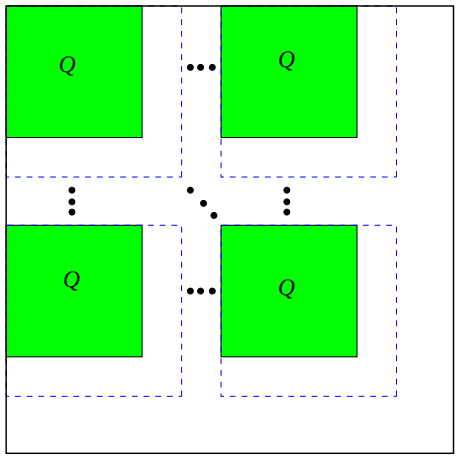


Dense, convex Hessian

Structure of Objective

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

$\Rightarrow$



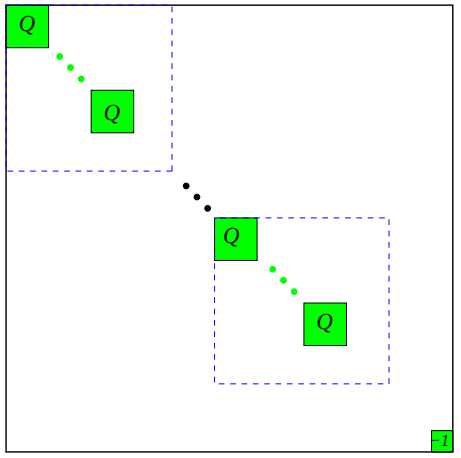
Dense, convex Hessian

Alternatively:

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

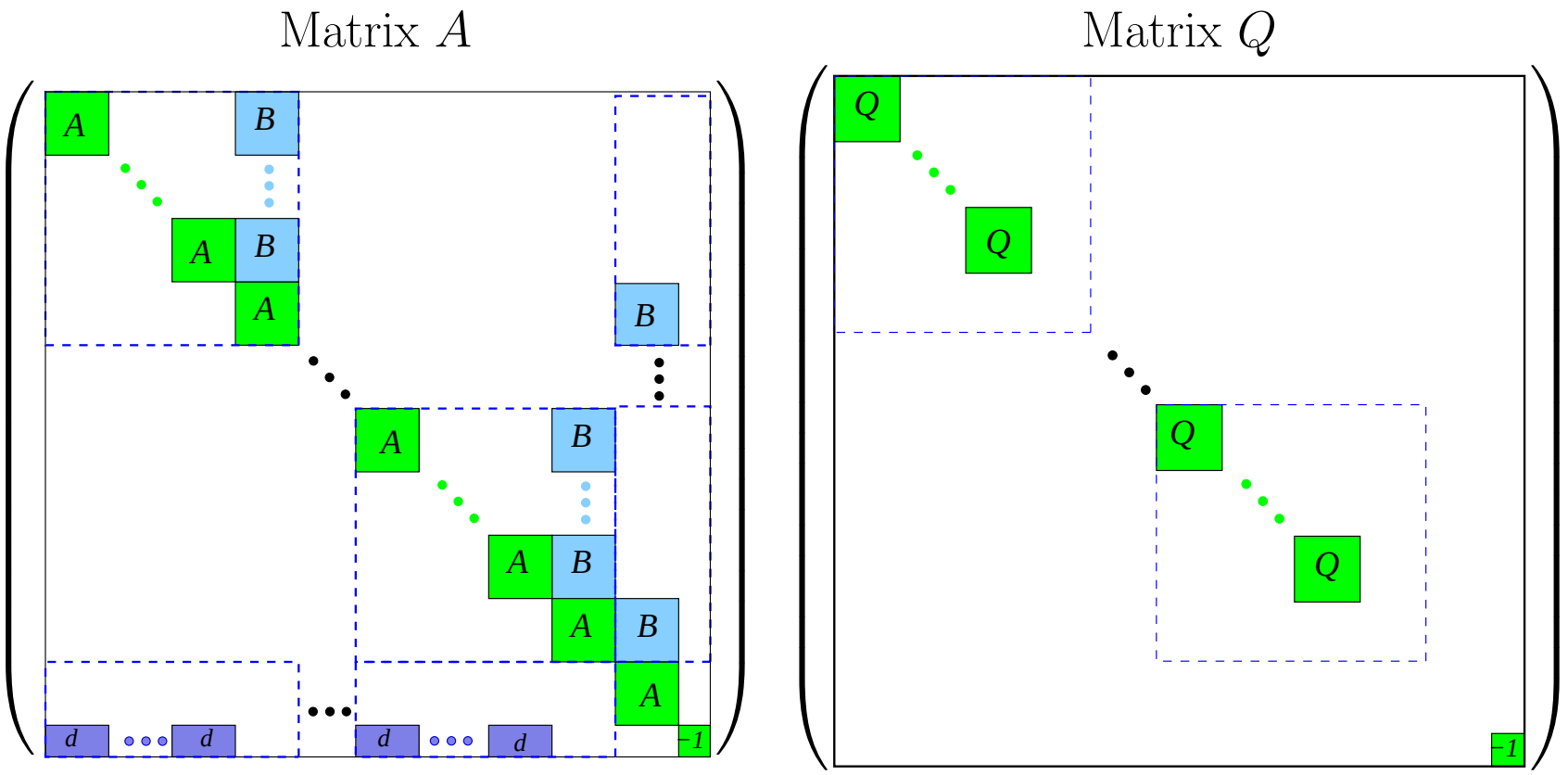
$$\mathbb{E}(X) - \rho \mathrm{Var}(X) = y - \rho \left[\sum_{i \in L_T} p_i \sum_j (v_j x_{ij}^h)^2 - y^2 \right]$$

$\Rightarrow$



Sparse, nonconvex Hessian

ALM: Structure of matrices A and Q:



Stochastic Programming approach to Financial Planning

- Popular with academics, practitioners seem suspicious
  - Large problem sizes ✘
  - Places constraints on model (?) ✘
- Traditionally solved by decomposition
  - Nested Benders Decomposition aka L-shaped method
  - Works well for LP models ✔
  - Unclear how to extend to quadratic/nonlinear models ✘
- Or solved by Interior Point Methods
  - Use taylored linear algebra
  - Only certain types of constraints are allowable (?) ✘

$\Rightarrow$ Structure Exploitation is key to succesful implementation

OOPS - Object Oriented Parallel (Interior Point) Solver

- Mantra: "Truly large scale problems are not only sparse but structured" (due to e.g. dynamics, uncertainty, spatial distribution etc.)
- Exploiting structure is key to building efficient IPMs for large problems:
  - Faster linear algebra
  - Reduced memory use (by use of implicit factorization)
  - Possibility to exploit (massive) parallelism
  - **We assume that structure is known!** $\Rightarrow$ no automatic detection.
- OOPS is a general purpose (parallel) Interior Point solver
  - Not tuned to any particular hardware
  - Not tuned to any particular problem (structure).
- OOPS currently solves LP/QP problems.
- NLP extension solves nonlinear financial planning problems

Interior Point Methods

$$\min c^\top x + \tfrac{1}{2} x^\top Q x \quad \text{s.t.} \quad \begin{aligned} Ax &= b \\ x &\geq 0 \end{aligned} \qquad \text{(QP)}$$

Optimality conditions:

$$\begin{aligned} c + Qx - A^\top y - z &= 0 \\ Ax &= b \\ XZe &= 0(\mu e) \\ x, z &\geq 0 \end{aligned}$$

$\Rightarrow$ Newton Step:

$$\begin{bmatrix} -Q & A^\top & I \\ A & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} \xi_c \\ \xi_b \\ r_{xz} \end{bmatrix} = \begin{bmatrix} c + Qx - A^\top y - z \\ b - Ax \\ \mu e - XZe \end{bmatrix} \qquad \text{(NS-QP)}$$

$\Rightarrow$ Reduced to

$$\begin{bmatrix} -Q - \Theta & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \xi_c - X^{-1} r_{xz} \\ \xi_b \end{bmatrix}$$

where $\Theta = X^{-1}Z, \quad X = \text{diag}(x), \quad Z = \text{diag}(z)$

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

$\Rightarrow$ 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

Example: Bordered Block-Diagonal Structure (Schur complement)

$$\underbrace{\begin{pmatrix} \Phi_1 & & & B_1^\top \\ & \ddots & & \vdots \\ & & \Phi_n & B_n^\top \\ B_1 & \cdots & B_n & \Phi_0 \end{pmatrix}}_{\Phi} = \underbrace{\begin{pmatrix} L_1 & & & \\ & \ddots & & \\ & & L_n & \\ L_{1,0} & \cdots & L_{n,0} & L_0 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} D_1 & & & \\ & \ddots & & \\ & & D_n & \\ & & & D_0 \end{pmatrix}}_{D} \underbrace{\begin{pmatrix} L_1^\top & & & L_{1,0}^\top \\ & \ddots & & \vdots \\ & & L_n^\top & L_{n,0}^\top \\ & & & L_0^\top \end{pmatrix}}_{L^\top}$$

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

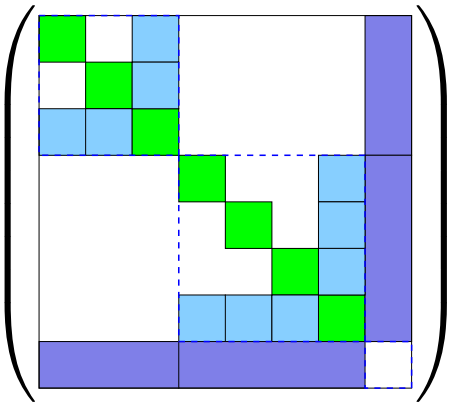
$$\begin{aligned} \Phi_i &= L_i D_i L_i^\top & L_{i,0} &= B_i L_i^{-\top} D_i^{-1},\ i = 1, \ldots n \\ C &= \Phi_0 - \textstyle\sum_{i=1}^n L_{i,0} D_i L_{i,0}^\top & C &= L_0 D_0 L_0^\top \end{aligned}$$

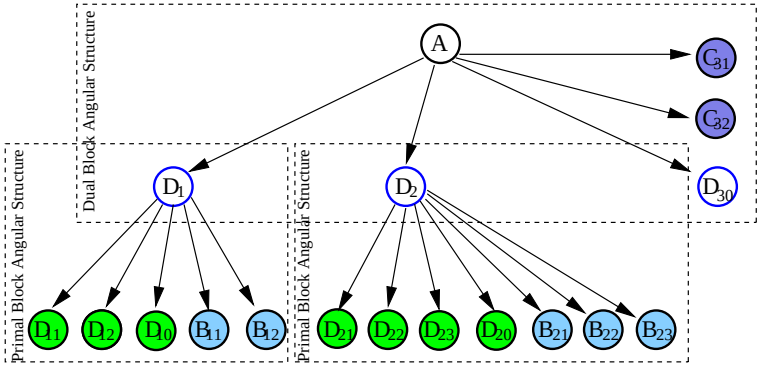
- And the system $\Phi x = b$ can be solved by

$$\begin{aligned} z_i &= L_i^{-1} b_i \\ z_0 &= L_0^{-1}(b_0 - \textstyle\sum L_{i,0} z_i) \\ y_i &= D_i^{-1} z_i \end{aligned} \qquad \begin{aligned} x_0 &= L_0^{-\top} y_0 \\ x_i &= L_i^{-\top}(y_i - L_{i,0}^\top x_0) \end{aligned}$$

- Operations (Cholesky, Solve, Product) are only performed on sub-blocks
  $\Rightarrow$ Can also exploit structure in sub-blocks

Tree Representation of Problem Structure

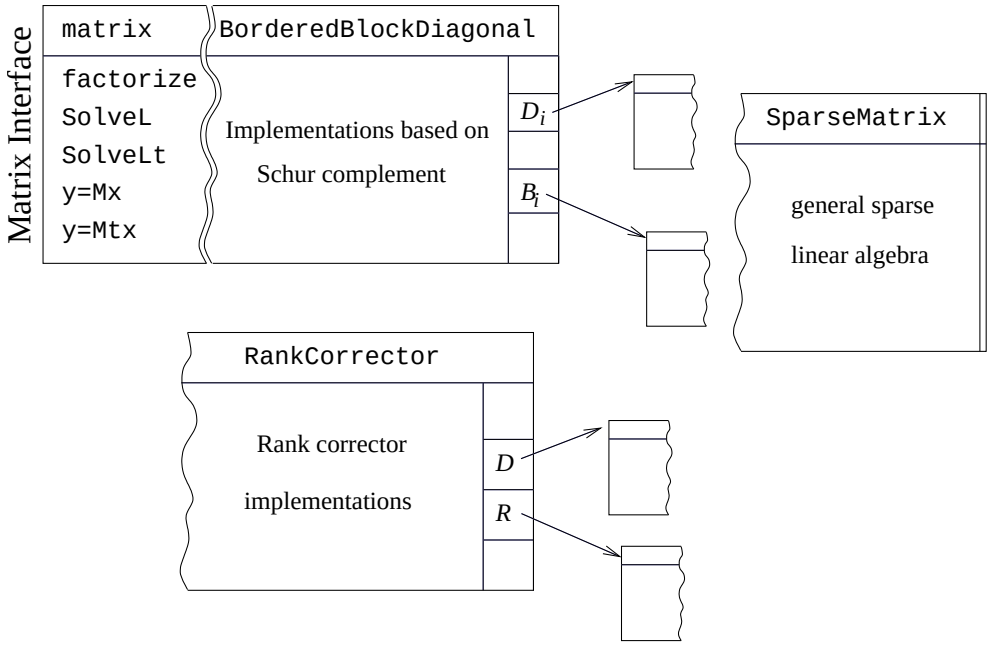




$\Rightarrow$ Organisation of Linear Algebra, Parallelism

OOPS: Object-oriented linear algebra implementation

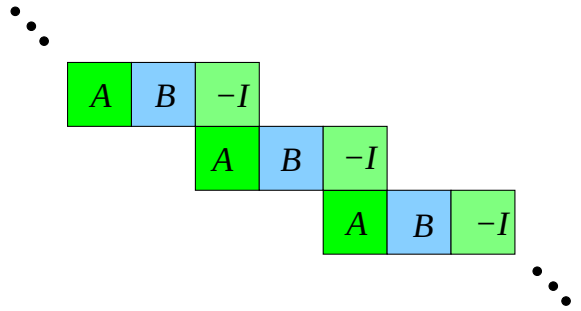
- Every node in *block elimination tree* has own linear algebra implementation (depending on its type)
- Implementation is realisation of an abstract linear algebra interface.
- Different implementations for different structures are available.



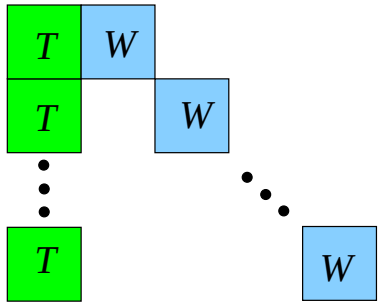
$\Rightarrow$ Rebuild *block elimination tree* with matrix interface structures

Sources of Structure

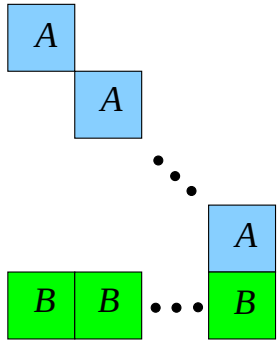
(linear) dynamics: $x_{t+1} = Ax_t + Bu_t$



uncertainty:
common 1st stage decision (today)
recourse action in 2nd stage (tomorrow)
$Tx + W_i\, y_i = h_i$

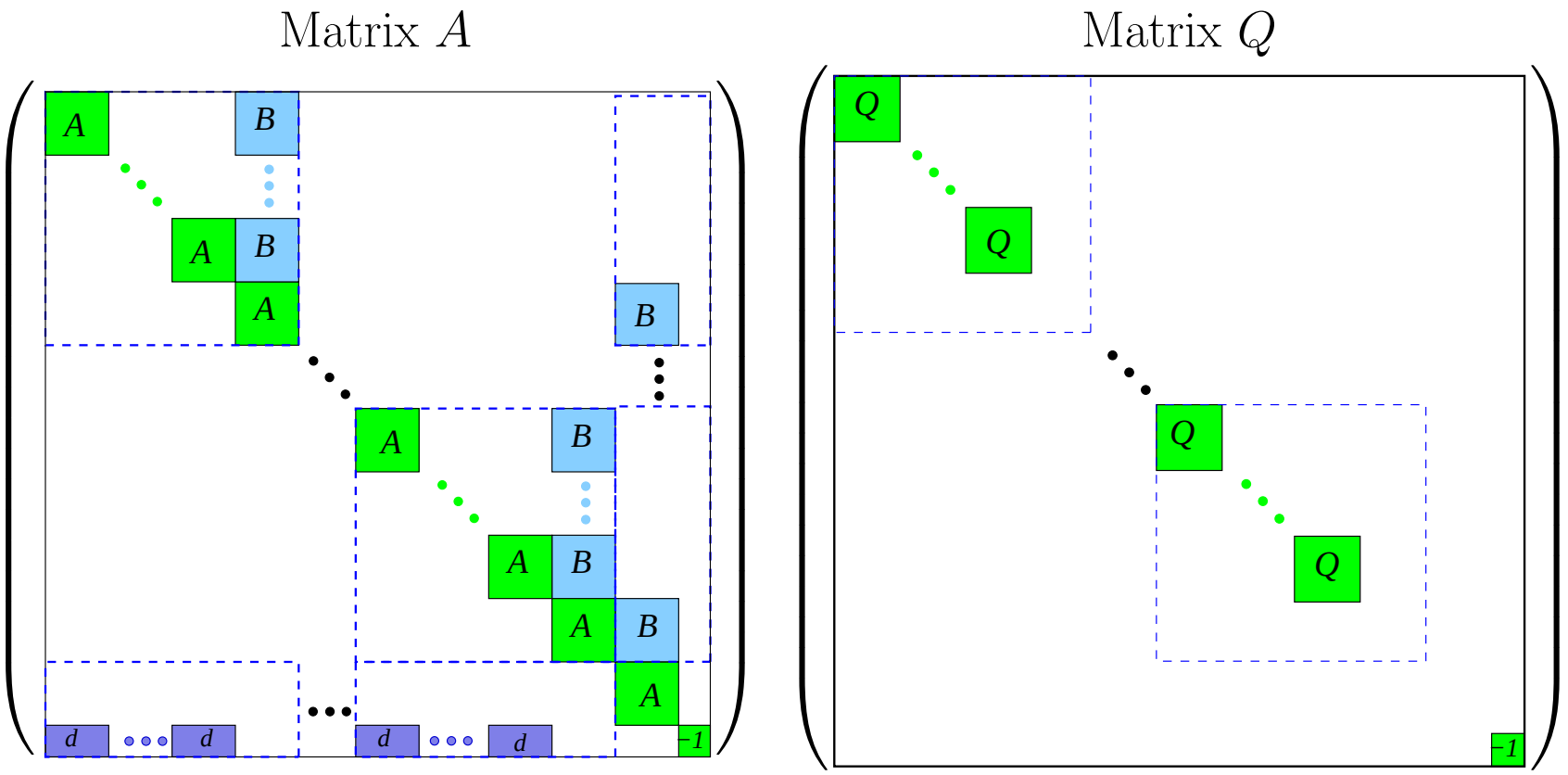


almost "independent" divisions
share a **common resource**
$\sum_{i=1}^{k} B_i\, x_i = b$

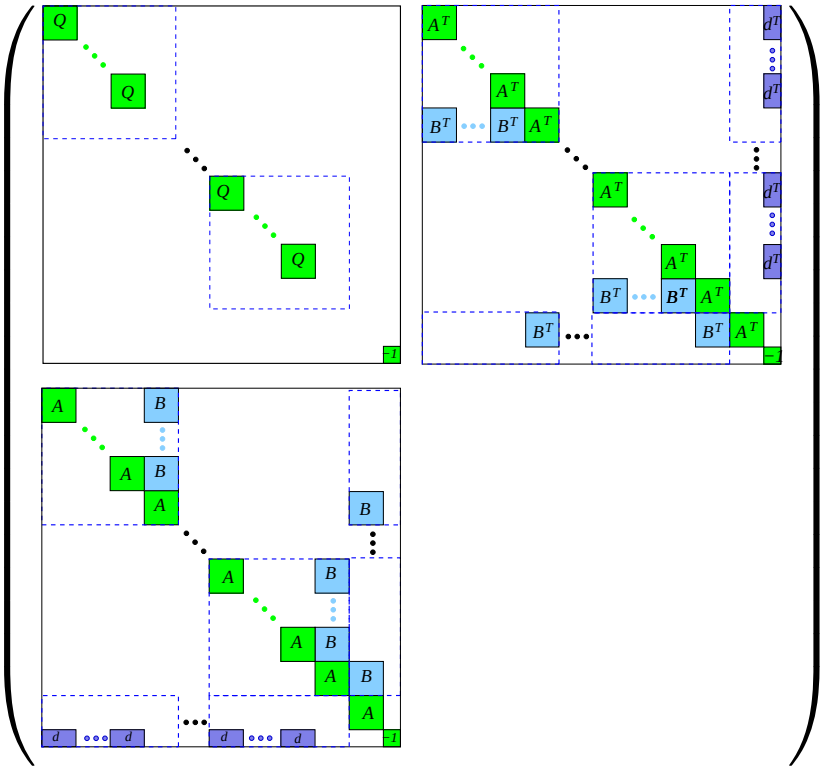


And many others. . . . Also **nested** structures.

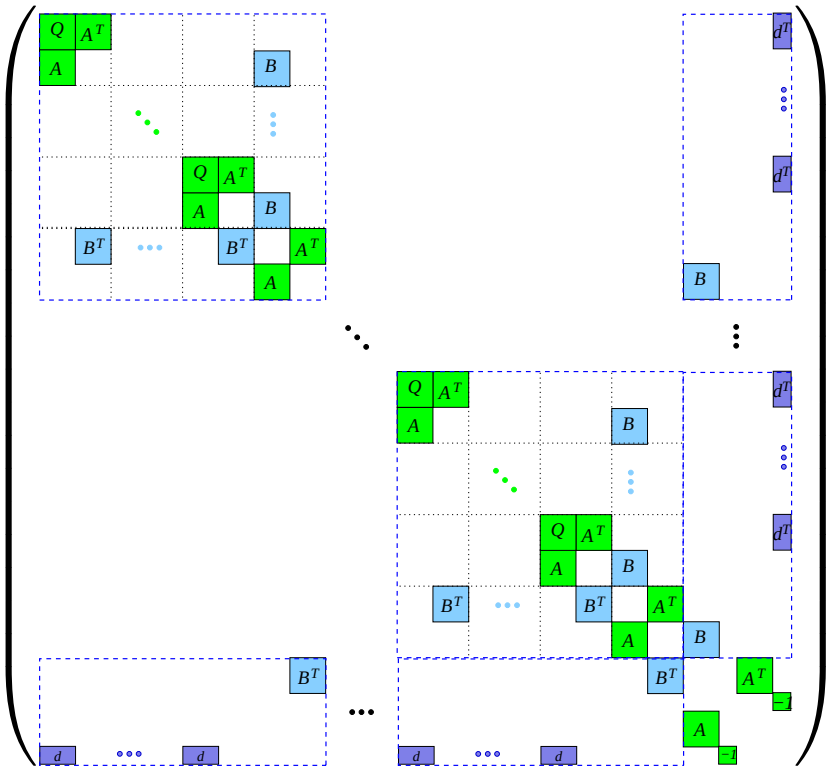
ALM: Structure of matrices A and Q:



Structures of A and Q imply structure of Φ



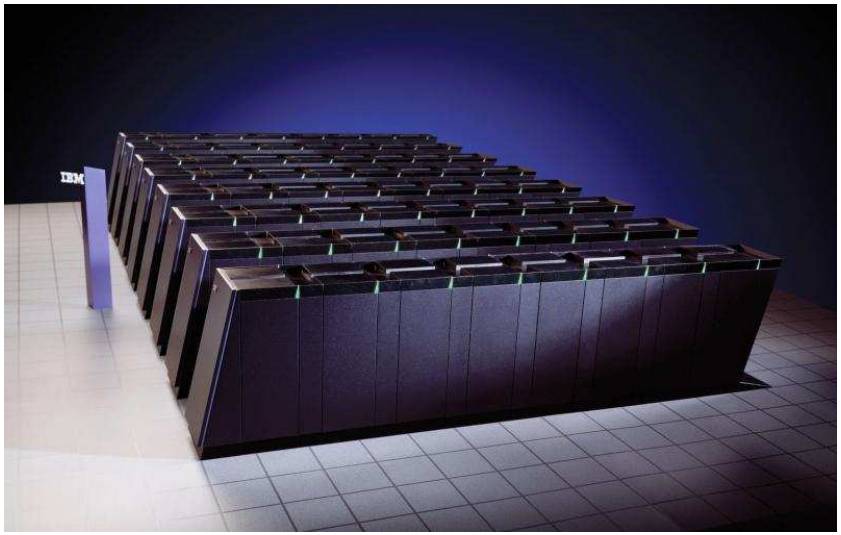
$$\begin{pmatrix} Q & A^\top \\ A & 0 \end{pmatrix}$$



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

Results (ALM: Mean-Variance QP formulation):

Problem	Stages	Blk	Assets	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	"



BlueGene/L (Edinburgh, Scotland)

- 2048 Processors
- 0.7GHz, 256Mb
- $R_{max} = 4.7$ TFlops
- #148 in `top500.org` list

HPCx (Daresbury, England)

- 1600 IBM Power-4 Processors
- 1.7GHz, 800Mb
- $R_{max} = 6.2$ TFlops
- $\approx$ #100 in `top500.org` list

ALM: Size of largest problem

Optimization of 21 assets (stock market indices)

$\Rightarrow$ Scenario tree needs to capture:

- correlations of 21 assets
- correlations over time periods

$\Rightarrow$ 100 branches over 6 stages $= 10^{12}$ scenarios

Better:

- Scenario tree geometry: 128-30-16-10-5-4 $\Rightarrow$ 16 million scenarios.
- Scenario tree generated using simulated geometric brownian motion with mean reversion.
- $\Rightarrow$ 1.01 billion variables, 353 million constraints

Issues for Massively Parallel Implementation

- Communications

  $\Rightarrow$ Memory/Data management

- Granularity of Computations

  $\Rightarrow$ Sparsity of multilevel linear algebra

- Bootstrapping

  $\Rightarrow$ How to get the data onto the processors

Example: Bordered Block-Diagonal Structure (Schur complement)

$$\underbrace{\begin{pmatrix} \Phi_1 & & & B_1^\top \\ & \ddots & & \vdots \\ & & \Phi_n & B_n^\top \\ B_1 & \cdots & B_n & \Phi_0 \end{pmatrix}}_{\Phi} = \underbrace{\begin{pmatrix} L_1 & & & \\ & \ddots & & \\ & & L_n & \\ L_{1,0} & \cdots & L_{n,0} & L_0 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} D_1 & & & \\ & \ddots & & \\ & & D_n & \\ & & & D_0 \end{pmatrix}}_{D} \underbrace{\begin{pmatrix} L_1^\top & & & L_{1,0}^\top \\ & \ddots & & \vdots \\ & & L_n^\top & L_{n,0}^\top \\ & & & L_0^\top \end{pmatrix}}_{L^\top}$$

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

$$\begin{aligned} \Phi_i &= L_i D_i L_i^\top & L_{i,0} &= B_i L_i^{-\top} D_i^{-1}, \ i = 1, \dots n \\ C &= \Phi_0 - \textstyle\sum_{i=1}^n L_{i,0} D_i L_{i,0}^\top & C &= L_0 D_0 L_0^\top \end{aligned}$$

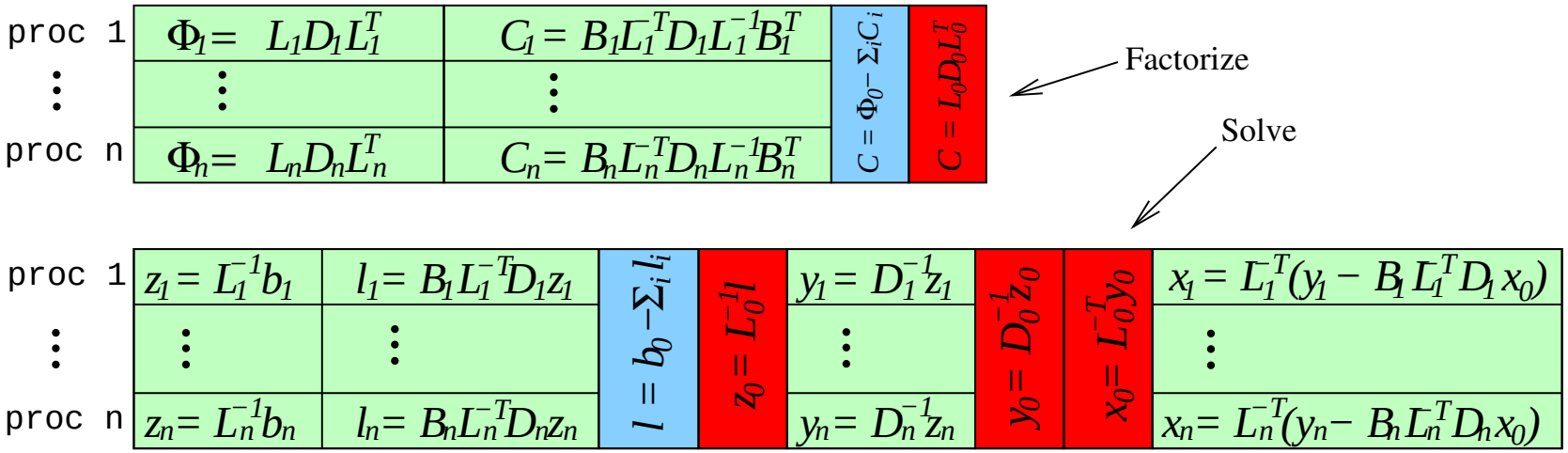
- And the system $\Phi x = b$ can be solved by

$$\begin{aligned} z_i &= L_i^{-1} b_i \\ z_0 &= L_0^{-1}(b_0 - \textstyle\sum L_{i,0} z_i) \\ y_i &= D_i^{-1} z_i \end{aligned} \qquad \begin{aligned} x_0 &= L_0^{-\top} y_0 \\ x_i &= L_i^{-\top}(y_i - L_{i,0}^\top x_0) \end{aligned}$$

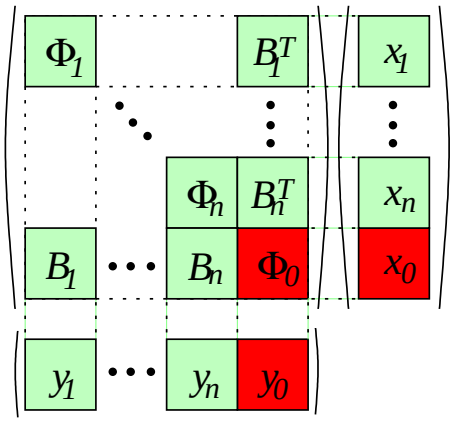
- Operations (Cholesky, Solve, Product) are only performed on sub-blocks
  $\Rightarrow$ Can also exploit structure in sub-blocks

Exploiting Parallelism: Bordered Block-Diagonal Structure:

- Distribution of computations:



- Storage:



processor 1

processor n

all processors

On separate processors

On all processors

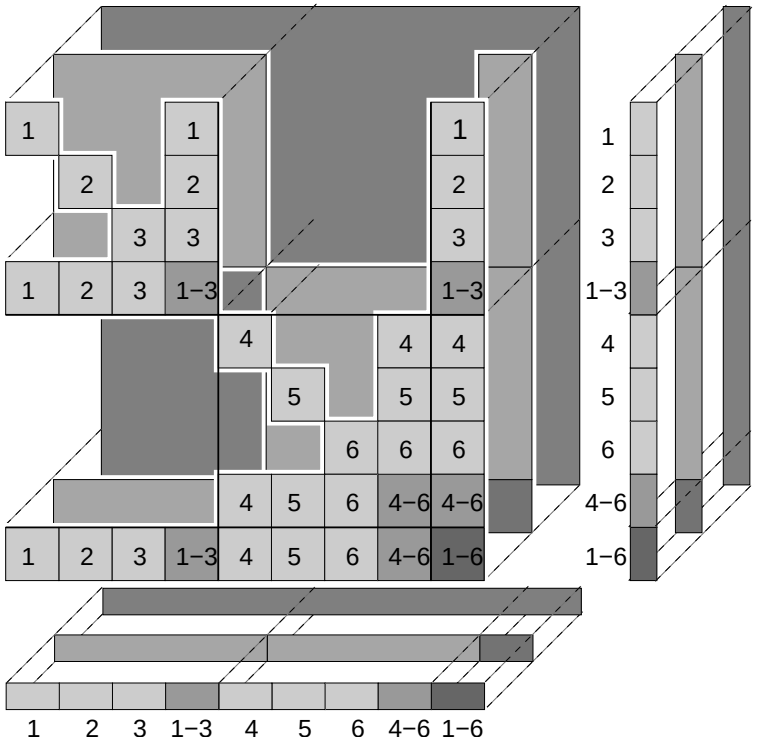
Communications

Memory Management

- Data for problem requires 2.6GB of memory.
  $\Rightarrow$ need to split information between processors
- To each node in block-elimination tree a set of processors is assigned
- Linear Algebra is implemented so that processors communicate when needed

Distribution of leading matrix blocks among processors implies

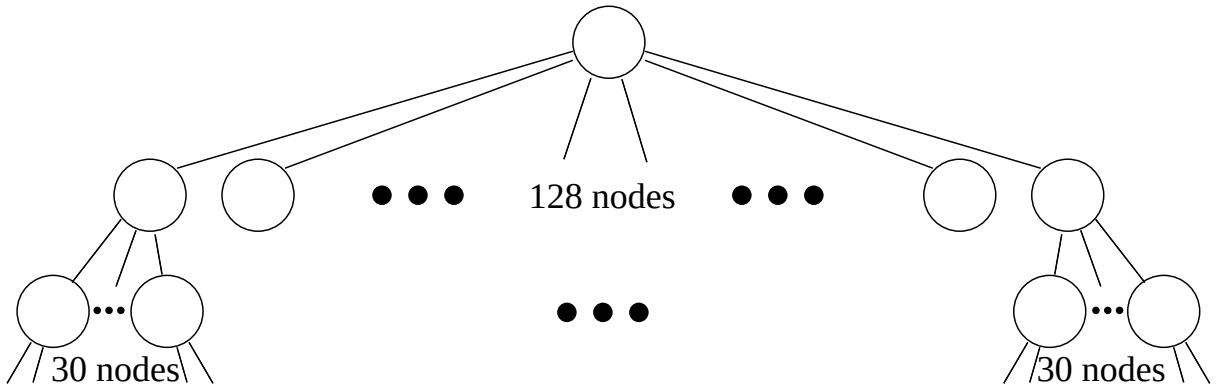
- Distribution of subordinate blocks
- Distribution of row/column vector contributions



Granularity of Computations: Parallel division of the problem

- In ALM problems matrices up to $\approx 500.000 - 1.000.000$ variables can be treated as unstructured sparse matrices on one processor
- Problem has:
  - 128 first level nodes with 10.000.000 variables each.
  - 3840 second level nodes with 350.000 variables each.

$\Rightarrow$ need to decompose problem at second level
(with 1280 processors $\Rightarrow$ 3 blocks per processor)



Sparsity of Linear Algebra (nested Bordered Block-Diagonal)

$$\Phi = \begin{bmatrix} \Phi_1 & & & B_1^T \\ & \ddots & & \vdots \\ & & \Phi_n & B_n^T \\ B_1 & \cdots & B_n & \Phi_0 \end{bmatrix}, \Phi = LL^T, L = \begin{bmatrix} L_1 & & & \\ & \ddots & & \\ & & L_n & \\ L_{0,1} & \cdots & L_{0,n} & C \end{bmatrix}$$

Factorize matrix by Schur complement:

- Factor $\Phi_i = L_i D_i L_i^T$
- Build $C = \Phi_0 - \sum_i (L_i^{-1} B_i^T)^T D_i (L_i^{-1} B_i^T)$
- Factor $C = L_c D_c L_c^T$

Work tends to be dominated by forming $V_i = L_i^{-1} B_i^T$ and $V_i^T V_i$.

$\Rightarrow$

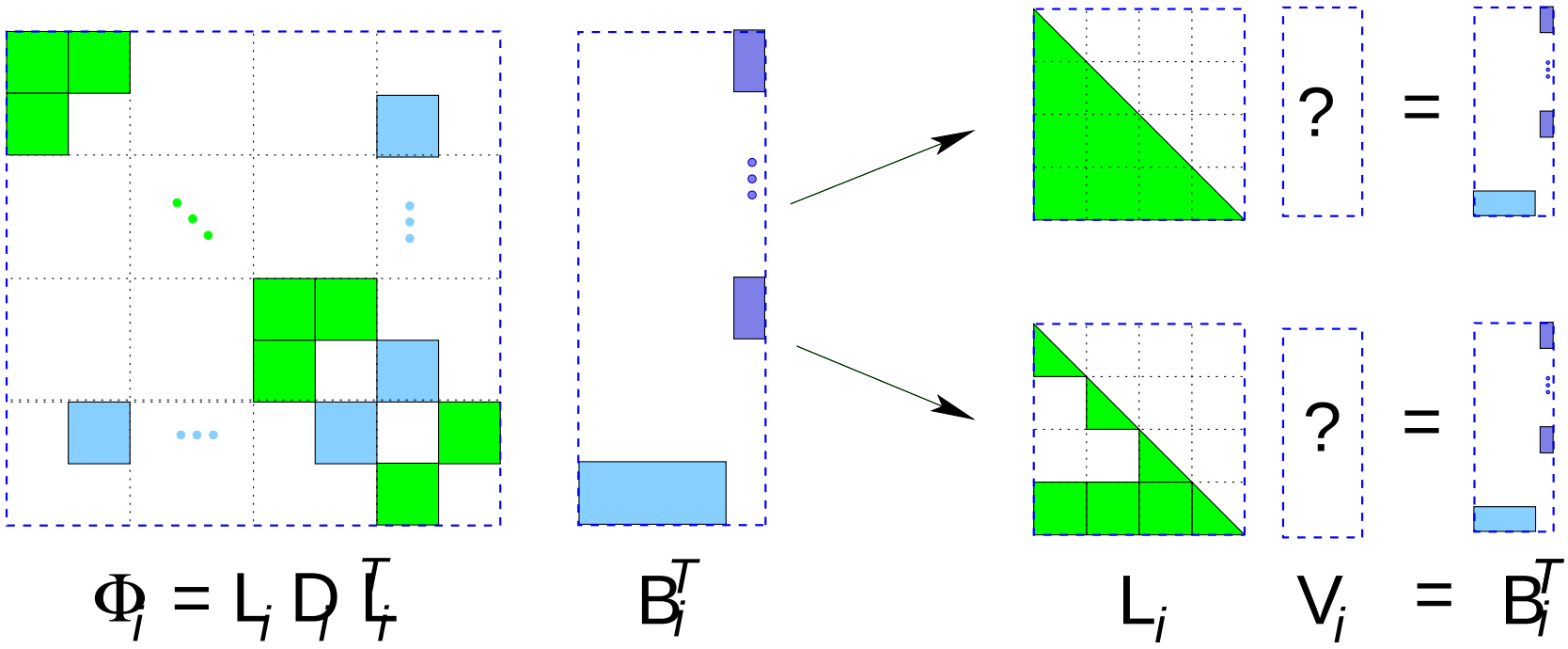
- Keep V_i as sparse as possible.
- Requires careful exploitation of sparsity when L_i itself is available through Schur-complement.

Sparsity of Multilevel Algebra

Dominant cost is forming Schur complement:

$$C = \Phi_0 - \sum_{i=1}^{n} V_i^T D_i V_i, \text{ where } V_i = L_i^{-1} B_i^T$$

$\Rightarrow$ Need to solve a system for each column of B_i^T

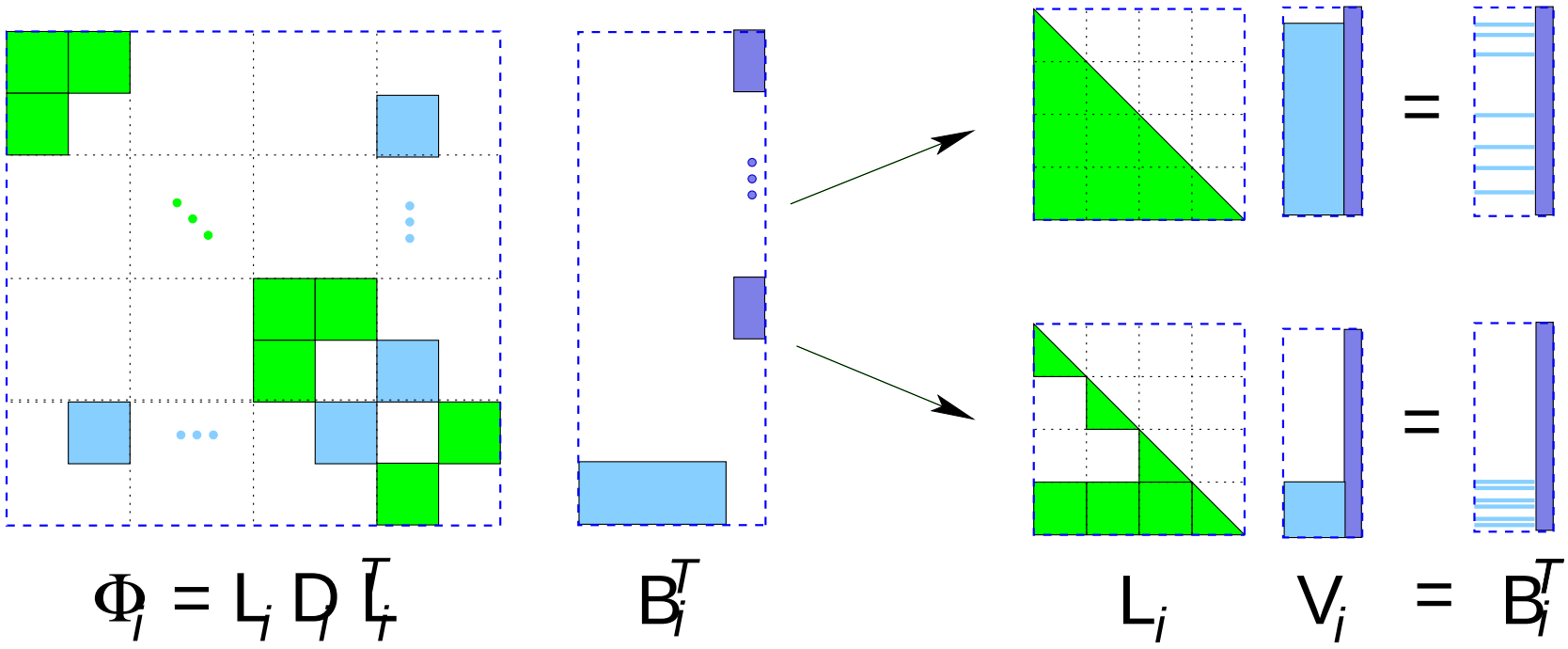


Sparsity of Multilevel Algebra

Dominant cost is forming Schur complement:

$$C = \Phi_0 - \sum_{i=1}^{n} V_i^T D_i V_i, \text{ where } V_i = L_i^{-1} B_i^T$$

$\Rightarrow$ Need to solve a system for each column of B_i^T



Bootstrapping

Problem: Data for the problem is $\approx$ 25 GB. We have 512MB/node on BlueGene.
How to get the data on the processors ?

Data can be represented as

- core: 64 variables, 22 constraints
- scenario tree (topology, probability, returns for each node): 2.6 GB

Still does not fit on one processor

- Scenario tree data (probability, returns) can be generated/read in on each processor as needed
- Scenario tree topology needs $\approx$ 30 MB

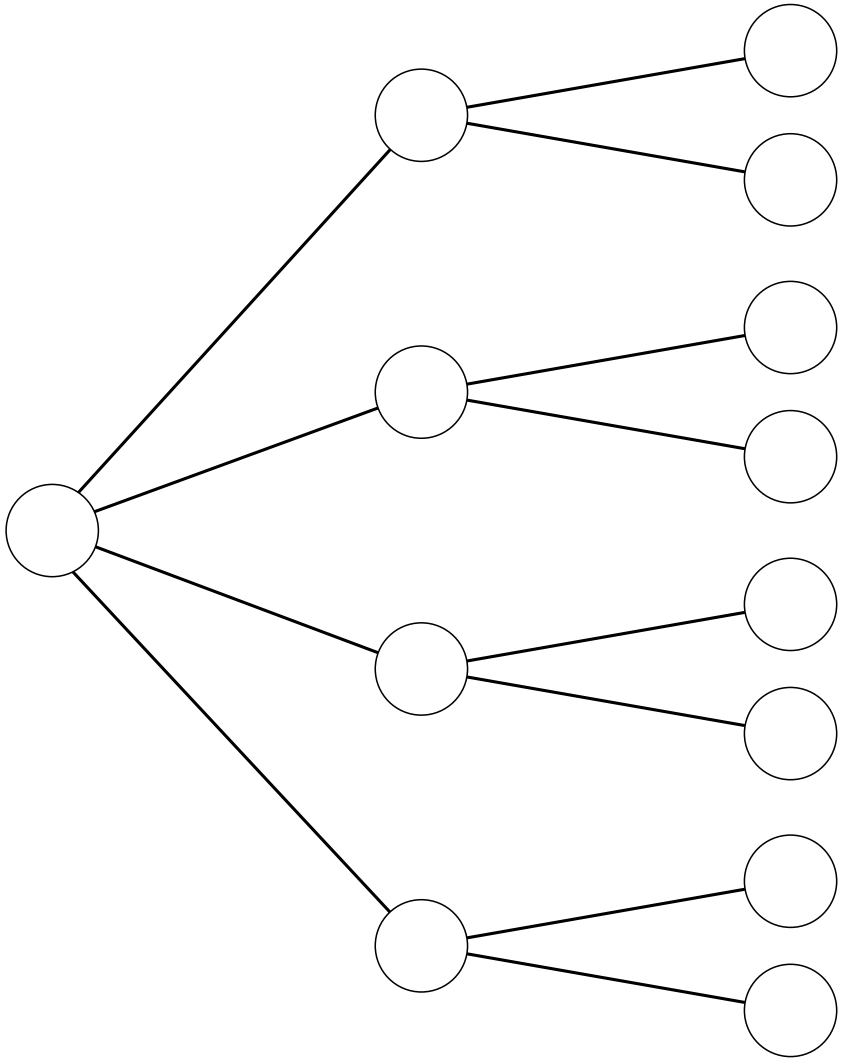
This is all that OOPS needs to decide on the allocation of blocks to processors

Bootstrapping II

1. Read problem dimensions,
   generate scenario tree topology and matrix tree (30MB)
2. Call OOPS: Divide matrix tree among processors,
   set up data structures on each processor
3. Generate/Read in scenario tree data on every processor
4. On every processor: Call-back to generator to fill in SparseMatrix data
5. Start solution process

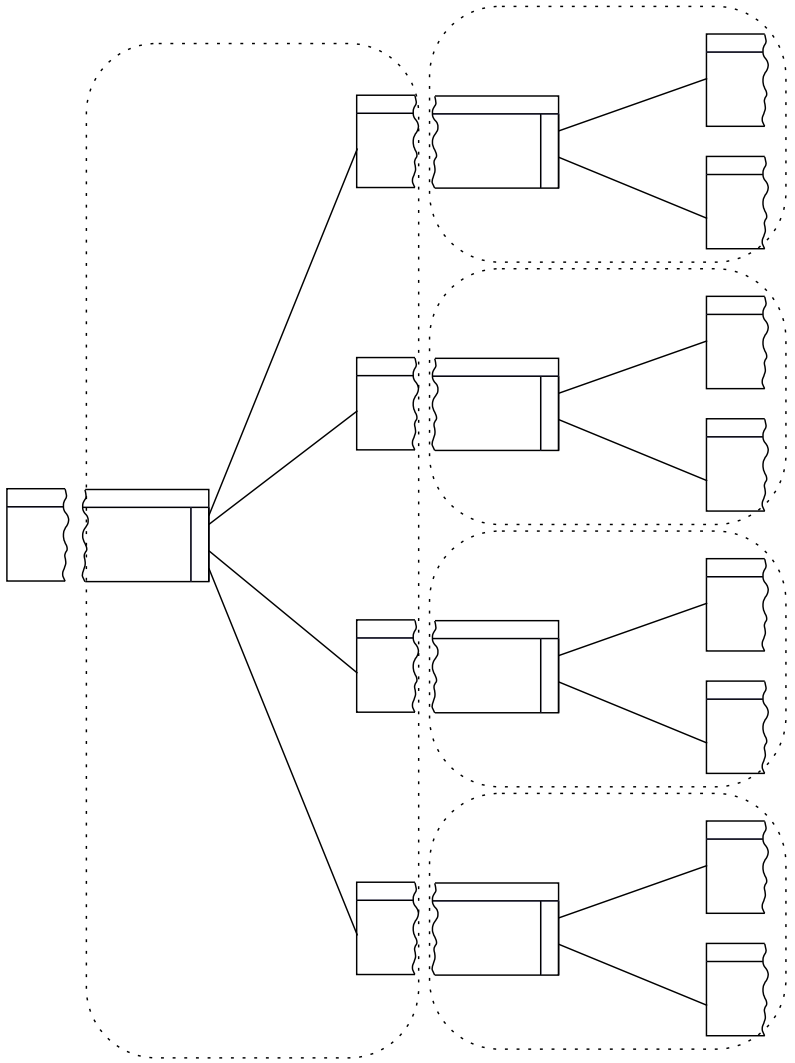
Bootstrapping

- Set up (shape of) Scenario Tree



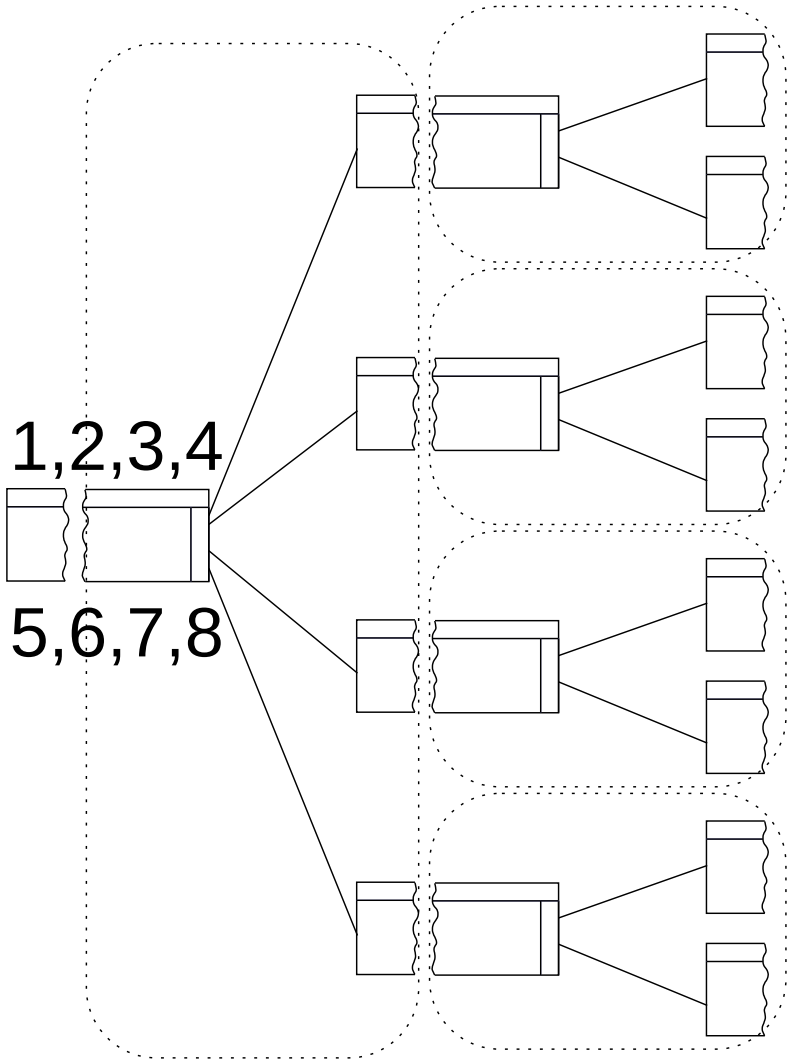
Bootstrapping

- Set up (shape of) Scenario Tree
- Convert to Algebra Tree (without data)



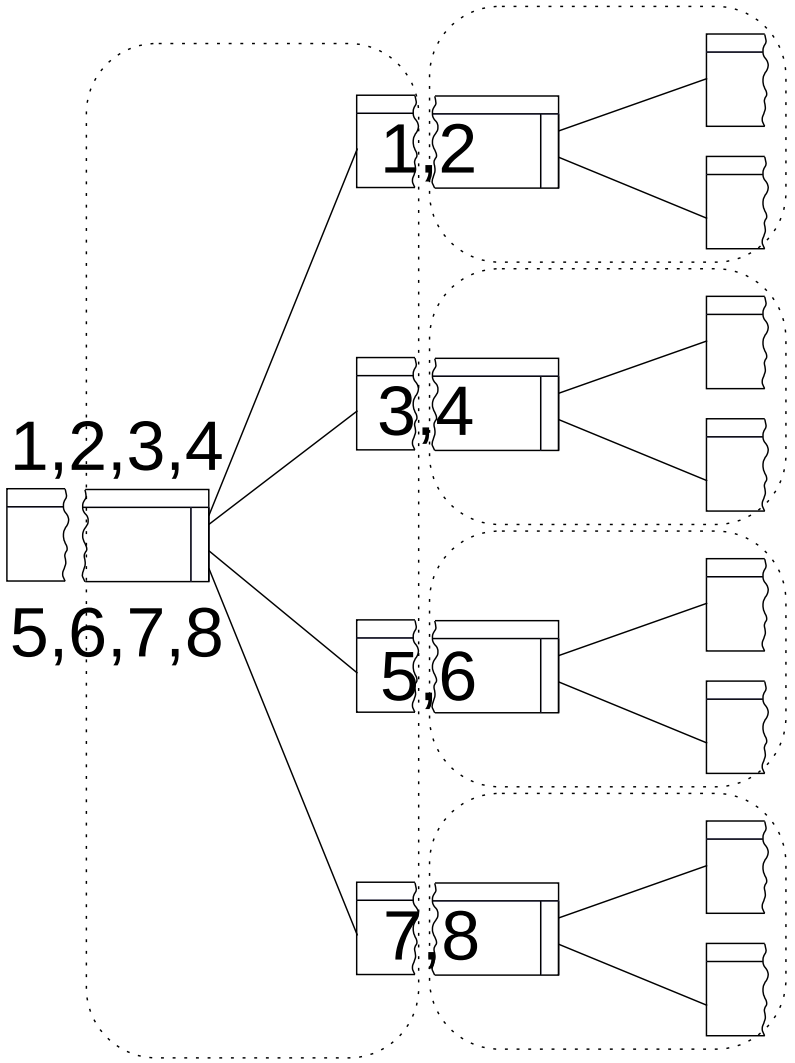
Bootstrapping

- Set up (shape of) Scenario Tree
- Convert to Algebra Tree (without data)
- Assign processors



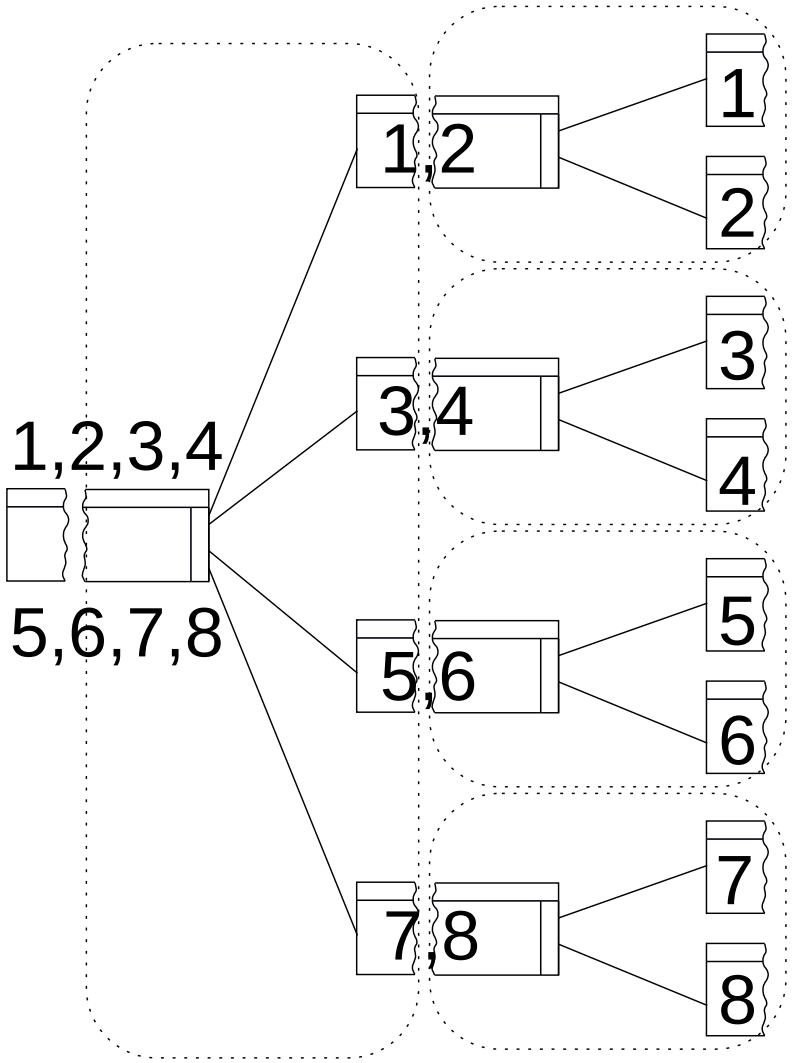
Bootstrapping

- Set up (shape of) Scenario Tree
- Convert to Algebra Tree (without data)
- Assign processors



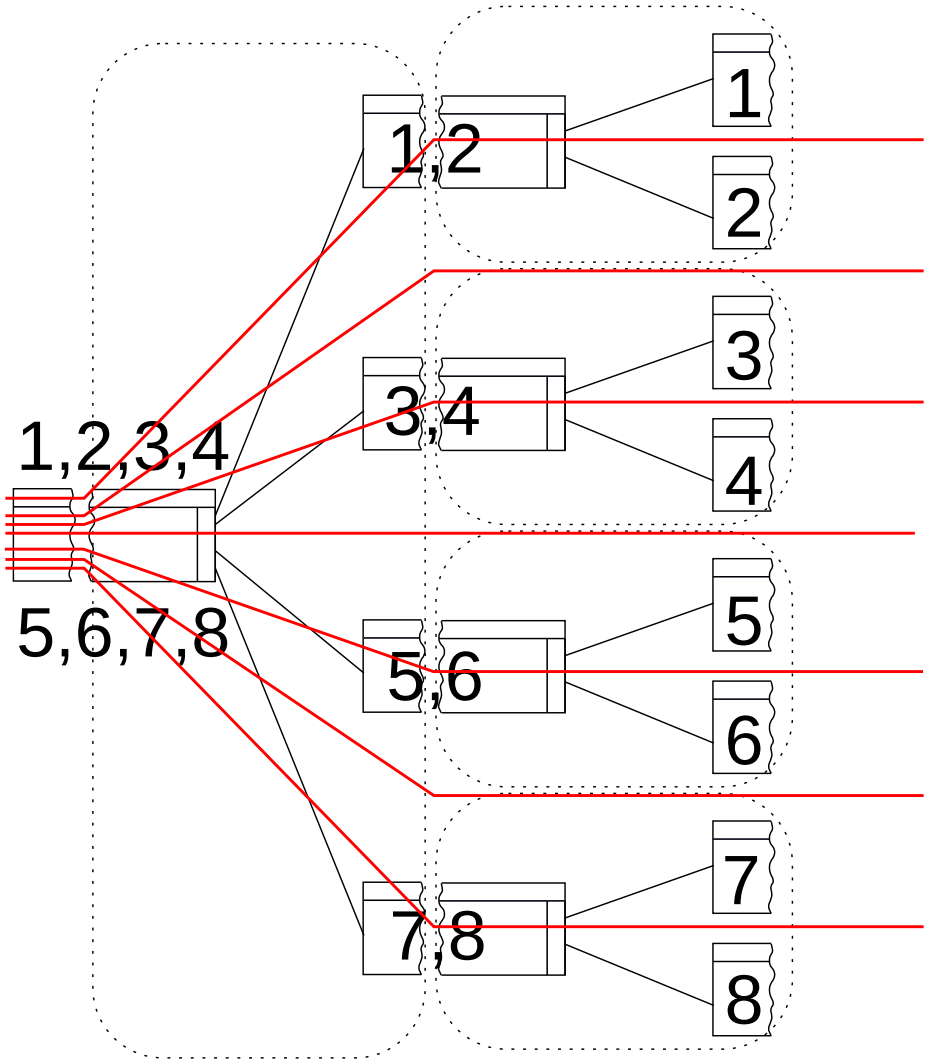
Bootstrapping

- Set up (shape of) Scenario Tree
- Convert to Algebra Tree (without data)
- Assign processors



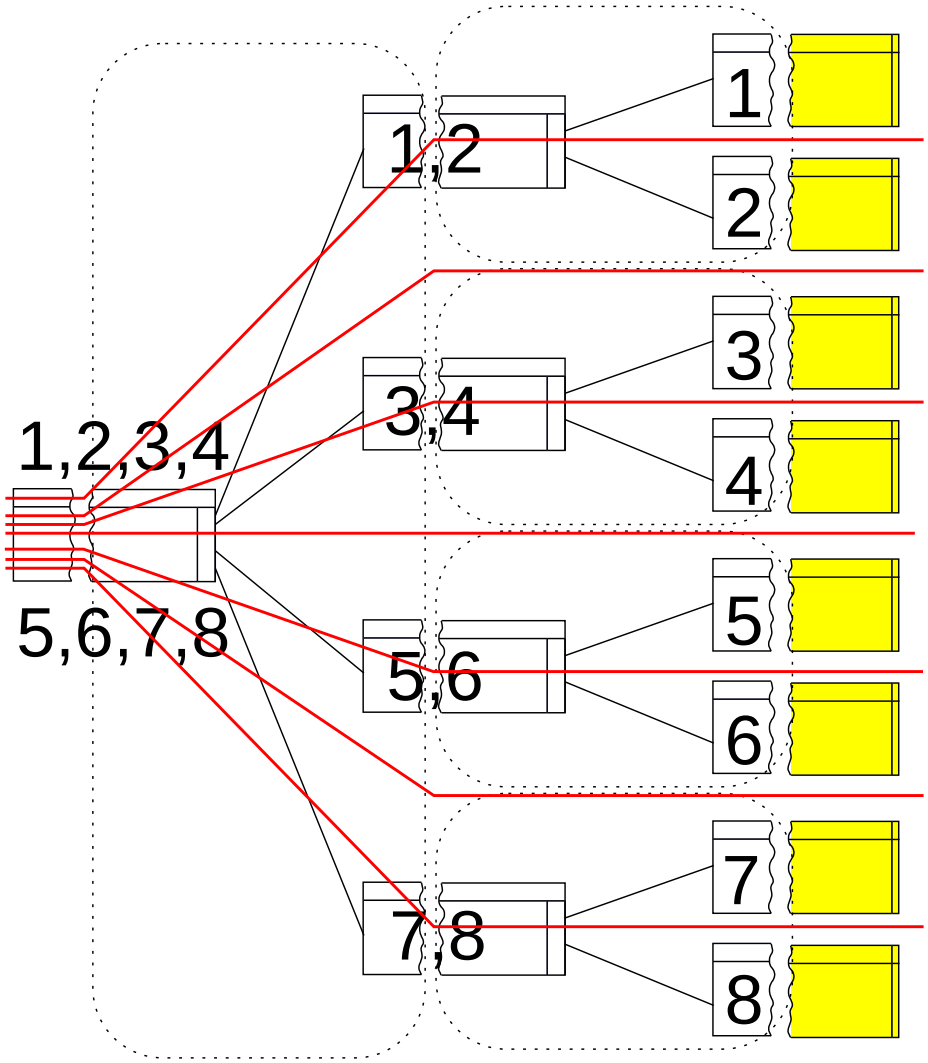
Bootstrapping

- Set up (shape of) Scenario Tree
- Convert to Algebra Tree (without data)
- Assign processors
- Divide between processors



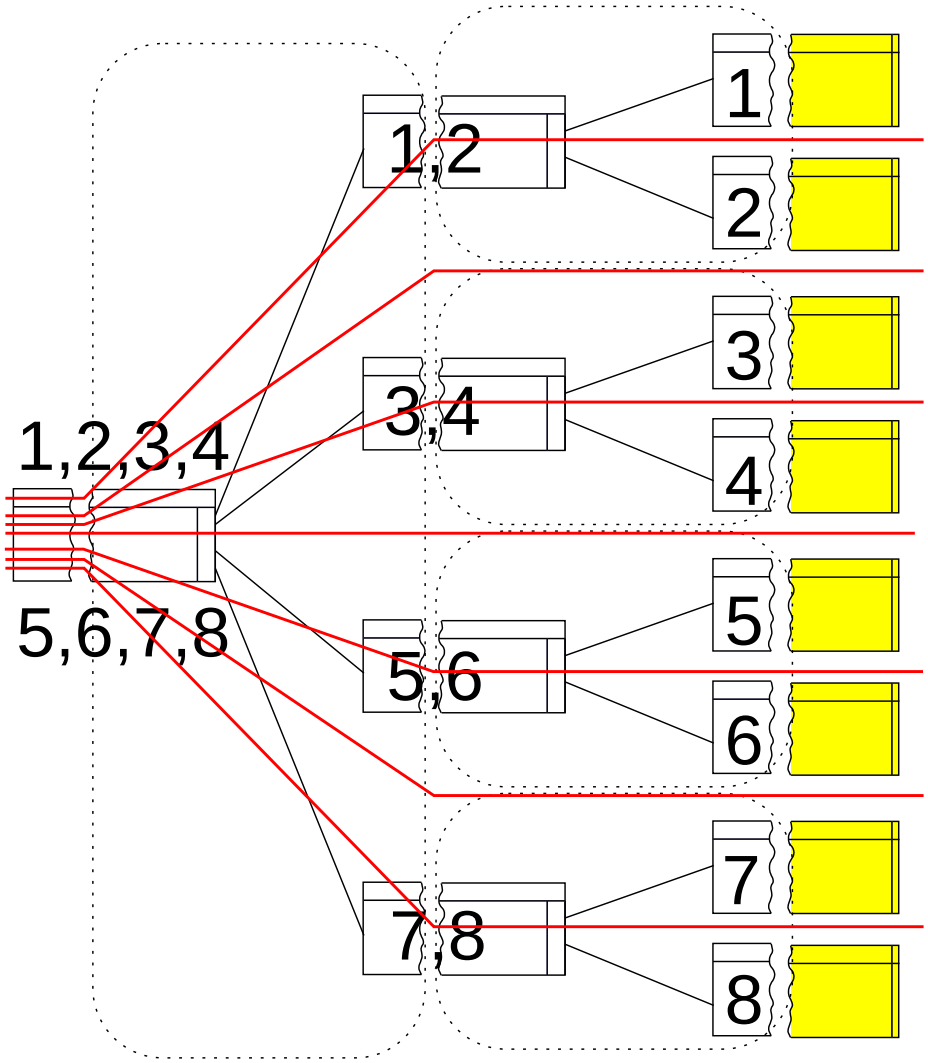
Bootstrapping

- Set up (shape of) Scenario Tree
- Convert to Algebra Tree (without data)
- Assign processors
- Divide between processors
- Load data on each processor (core + generate scenarios)



Bootstrapping

- Set up (shape of) Scenario Tree
- Convert to Algebra Tree (without data)
- Assign processors
- Divide between processors
- Load data on each processor (core + generate scenarios)
- Solve



Results

Stages	Blk	Assets	Scenarios	Constraints	Variables	iter	time	procs	machine
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

Results

Stages	Blk	Assets	Scenarios	Constraints	Variables	iter	time	procs	machine
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

Parallel Efficiency/Scaling

nodes	peak Mem	time	Comm	Cholesky	Solves	MatVectProd
16	426MB	2587 (1.00)	24	1484 (1.00)	956 (1.00)	28.8 (1.00)
32	232MB	1303 (0.99)	13	743 (1.00)	485 (0.98)	18.0 (0.80)
64	132MB	688 (0.94)	6	377 (0.98)	270 (0.88)	13.0 (0.55)
128	84MB	348 (0.93)	3	187 (0.99)	139 (0.86)	9.0 (0.40)
256	56MB	179 (0.90)	3	93 (0.99)	73 (0.82)	5.8 (0.31)
512	46MB	94 (0.86)	2	47 (0.98)	39 (0.76)	3.9 (0.23)

Further Developments: Warmstarting

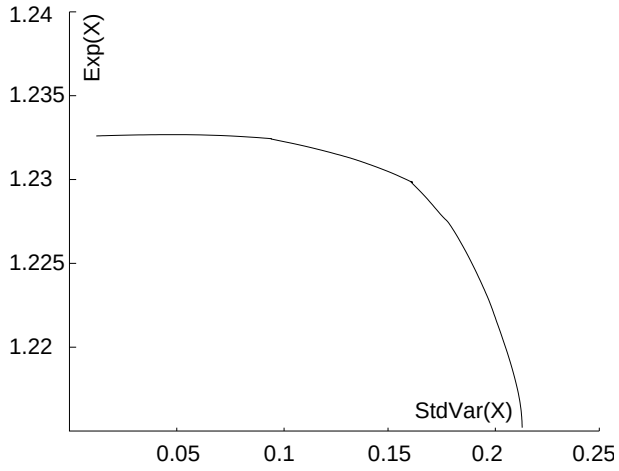
Possibilities for warmstarting are rarely used in Stochastic Programmming

- Solution approaches for SP (Benders Decomposition/IPM) don't warmstart well
- "Interior Point Methods cannot be warmstarted"

Results (Efficient Frontier)

Mean Variance formulation:

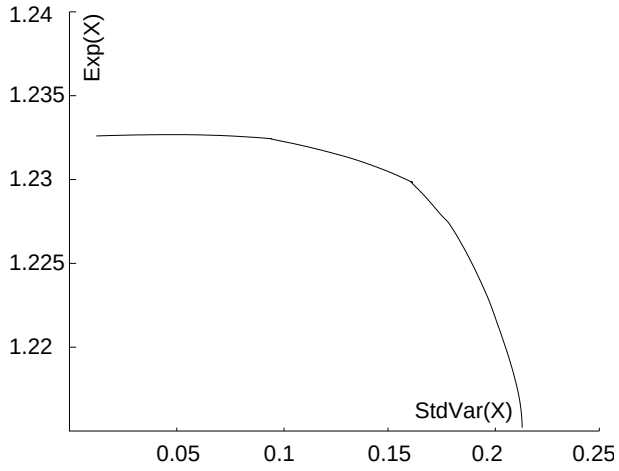
$$\max \mathbb{E}(X) - \rho \mathrm{Var}(X)$$



Results (Efficient Frontier)

Mean Variance formulation:

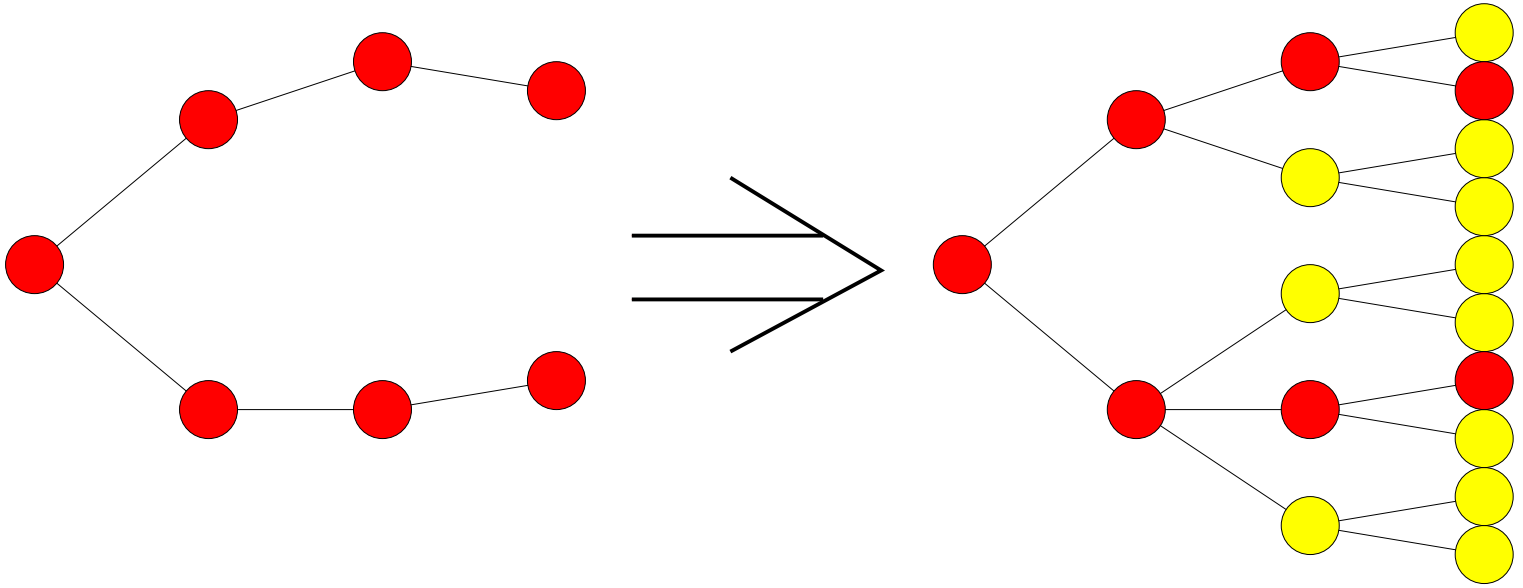
$$\max \mathbb{E}(X) - \rho \mathrm{Var}(X)$$



constraints	variables	procs	ρ =0.001	0.005	0.01	0.05	0.1	0.5	1	5	10
223,321	76,881	1	14	14	14	14	14	13	17	16	17
			14	5	5	5	4	5	5	8	8
533,725	198,525	1	14	14	14	14	14	15	18	18	17
			14	5	5	5	6	5	5	9	10
5,982,604	16,316,191	32	24	23	24	23	25	22	24	23	24
			24	8	11	13	11	13	12	12	14
70,575,308	192,478,111	512	52	53	45	43	44	42	44	46	46
			52	13	13	15	15	16	16	23	25

$\Rightarrow$ 50%-60% of iterations saved on average

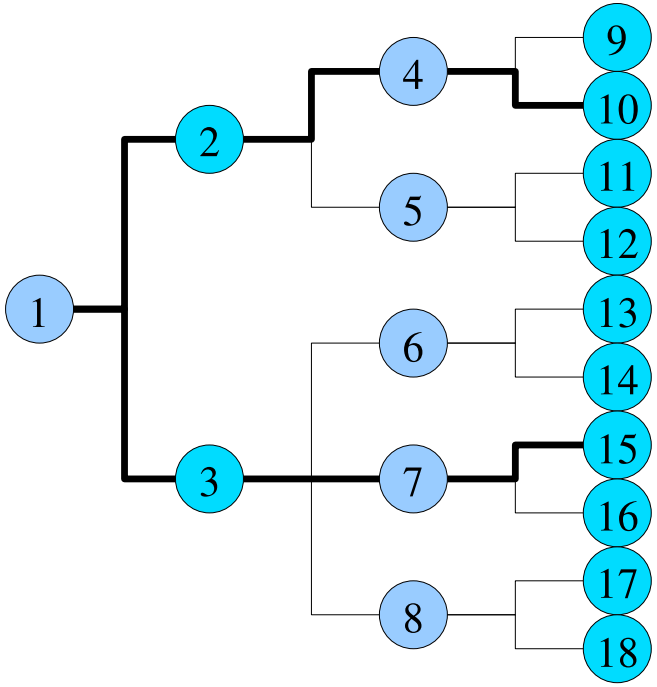
Idea: Use Warmstart to speed up solution of Stochastic Program



- Solve problem on a **reduced** scenario tree first
- Copy solution vectors from **nearby** scenarios to construct starting point for **full** problem.
- Solve full problem (using this starting point)

$\Rightarrow$ Saves up to 60% of iterations on problem up to 4096 scenarios.

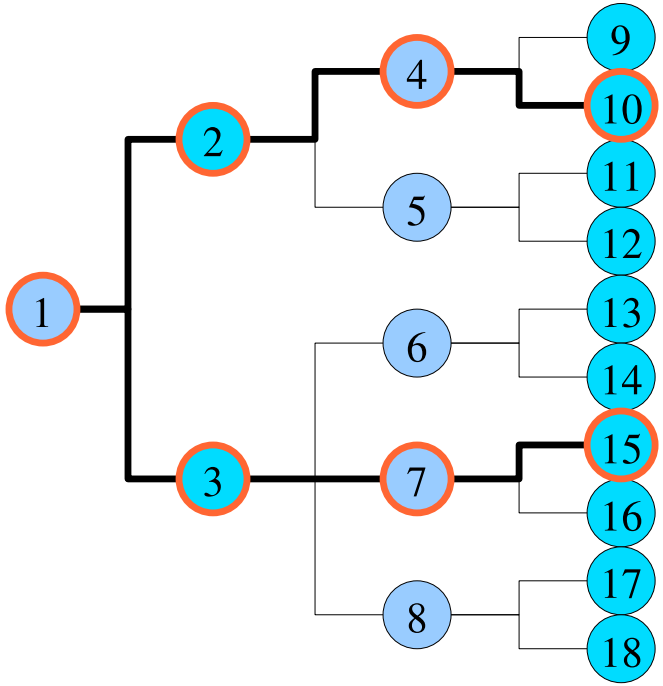
Construction of the warm-start iterate



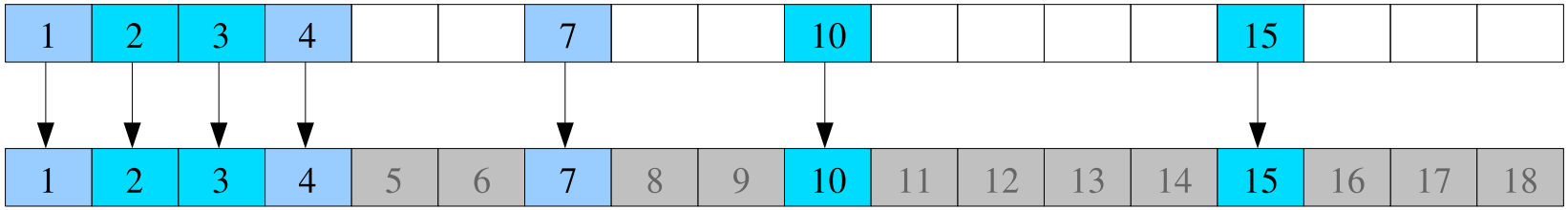
1	2	3	4			7			10					15			

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

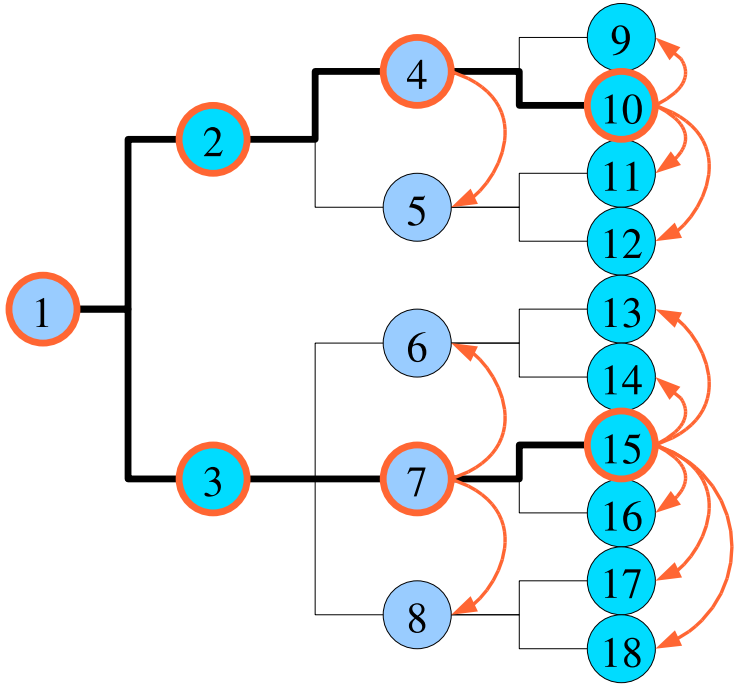
Construction of the warm-start iterate



Nodes in the reduced tree:
the solution is already available

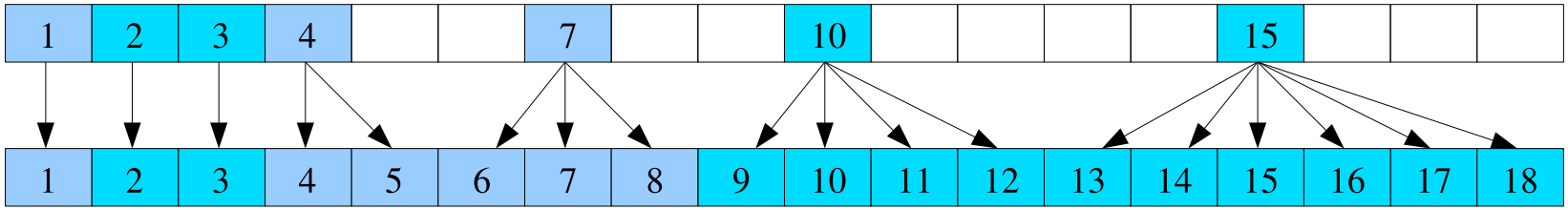


Construction of the warm-start iterate



Nodes in the reduced tree:
the solution is already available

Remaining nodes:
copy the solution from the corresponding reduced-tree node



Conclusions:

- OOPS is a general purpose IPM solver with object-oriented linear algebra
- Will scale up to massively parallel architecture
- Can solve problems with 10^9 variables using direct factorization methods
- Allows the solution of realistic financial planning problems

Future Work on OOPS:

- Extend warmstarting procedure:
  - IPM difficult to warmstart
  - Multi-Step Scheme
  - Complexity of such a scheme
  - Carry over to other structures (PDE constrained optimization)
- Incorporation of iterative solvers (structured pre-conditioners)
- Integrate into structured modelling language

Object-Oriented Parallel Solver (OOPS):

References:

- J. Gondzio and A. Grothey, *Parallel interior point solver for structured quadratic programs: application to financial planning problems*, Annals of OR 152 (2007), Vol 1, pp 319–339.
- J. Gondzio and A. Grothey, *Solving nonlinear portfolio optimization problems with the primal-dual interior point method*, European Journal of Operational Research, 181 (2007), Vol 3, p.1019-1029.
- J. Gondzio and A. Grothey, *Exploiting Structure in Parallel Implementation of Interior Point Methods for Optimization*, Tech. Rep. MS-04-004, School of Maths, University of Edinburgh, Dec 2004.
- J. Gondzio and A. Grothey, *Direct Solution of Linear Systems of Size* 10^9 *Arising in Optimization with Interior Point Methods*,in: Parallel Processing and Applied Mathematics, Lecture Notes in Computer Sciences 3911.