Interior-Point Solver

Jacek Gondzio

ISMP, Atlanta, August 2000

Department of Mathematics & Statistics
The University of Edinburgh
EH9 3JZ Edinburgh, U.K.
Email: gondzio@maths.ed.ac.uk
URL: http://www.maths.ed.ac.uk/~gondzio

In collaboration with:
Andreas Grothey, Edinburgh
Robert Sarkissian, Cambridge

Interior-Point Methods reached maturity

- well understood theory
- fast commercial and free IPM solvers
- IPMs are suitable for large-scale problems

Large-scale problems are usually structured
(dynamics, uncertainty, spatial distribution, etc)

Affordable parallelism: Beowulf Project
Becker et al., Harnessing the Power of Parallelism in a Pile-of-PCs.

Develop Interior-Point Solver that

- solves **any** structured problem
- is fast
- runs in parallel

Object-Oriented Design

- define **Abstract Algebra**
  dedicated to IPM's
- implement **different algebras**
  for **block-structured matrices**

Structures and IPMs

- Todd, Math. Prog., 1988;
- Birge & Qi, Mang. Sci., 1988;
- Schultz & Meyer, SIAM Opt, 1991;
- Hurd & Murphy, ORSA JoC, 1992;
- Choi & Goldfarb, Math. Prog., 1993;
- Jessup, Yang & Zenios, SIAM Opt, 1994;
- Grigoriadis & Khachiyan, SIAM Opt, 1996;
- Gondzio, Sarkissian & Vial, EJOR, 1997;

Block-Structured Matrices

Staircase Structure

$$A = \begin{pmatrix} A_1 & & & & \\ B_1 & A_2 & & & \\ & B_2 & & & \\ & & \ddots & & \\ & & & B_{n-1} & A_n \end{pmatrix}.$$

Primal Block-Angular Structure

$$A = \begin{pmatrix} A_1 & & & & \\ & A_2 & & & \\ & & \ddots & & \\ & & & A_n & \\ B_1 & B_2 & \cdots & B_n & B_{n+1} \end{pmatrix}.$$

Dual Block-Angular Structure

$$A = \begin{pmatrix} A_1 & & & C_1 \\ & A_2 & & C_2 \\ & & \ddots & \vdots \\ & & & A_n \quad C_n \end{pmatrix}.$$

Row and Column Bordered Structure

$$A = \begin{pmatrix} A_1 & & & C_1 \\ & A_2 & & C_2 \\ & & \ddots & \vdots \\ & & & A_n \quad C_n \\ B_1 & B_2 & \cdots & B_n \quad B_{n+1} \end{pmatrix}.$$

Augmented system (symmetric but indefinite)

$$\begin{bmatrix} -\Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r \\ h \end{bmatrix}.$$

Eliminate

$$\Delta x \;=\; \Theta A^T \Delta y - \Theta r,$$

to get **Normal equations** (symmetric, positive definite)

$$(A\Theta A^T)\Delta y \;=\; g = A\Theta r + h.$$

Two step solution method:

- factorization to LDL^T form,
- backsolve to compute direction Δy.

$$A = \begin{pmatrix} A_1 & & & & \\ & A_2 & & & \\ & & \ddots & & \\ & & & A_n & \\ B_1 & B_2 & \cdots & B_n & B_{n+1} \end{pmatrix}.$$

Normal-equations matrix

$$AA^T = \begin{pmatrix} A_1A_1^T & & & & A_1B_1^T \\ & A_2A_2^T & & & A_2B_2^T \\ & & \ddots & & \vdots \\ & & & A_nA_n^T & A_nB_n^T \\ B_1A_1^T & B_2A_2^T & \cdots & B_nA_n^T & C \end{pmatrix},$$

where

$$C = \sum_{i=1}^{n+1} B_iB_i^T.$$

Implicit inverse

$$\begin{aligned} A_iA_i^T &= L_iL_i^T \\ \tilde{B}_i &= B_iA_i^TL_i^{-T} \\ S = C - \textstyle\sum_{i=1}^n \tilde{B}_i\tilde{B}_i^T &= L_SL_S^T \end{aligned}$$

Dual Block-Angular Structure

$$A = \begin{pmatrix} A_1 & & & C_1 \\ & A_2 & & C_2 \\ & & \ddots & \vdots \\ & & & A_n \quad C_n \end{pmatrix}.$$

Normal-equations matrix

$$AA^T = \begin{pmatrix} A_1A_1^T & & & \\ & A_2A_2^T & & \\ & & \ddots & \\ & & & A_nA_n^T \end{pmatrix} + CC^T,$$

where $C \in \mathcal{R}^{m\times k}$ defines a rank-k corrector.

Implicit inverse (Sherman-Morrison-Woodbury)

$$\begin{aligned} A_iA_i^T &= L_iL_i^T \\ \text{diag}(A_iA_i^T) &= \text{diag}(L_iL_i^T) = LL^T \\ \tilde{C}_i &= L_i^{-1}C_i \\ S = I_k + \textstyle\sum_{i=1}^n \tilde{C}_i^T\tilde{C}_i &= L_SL_S^T \end{aligned}$$

$$\begin{aligned} (AA^T)^{-1} &= (LL^T + CC^T)^{-1} \\ &= (LL^T)^{-1} - (LL^T)^{-1}CS^{-1}C^T(LL^T)^{-1} \end{aligned}$$

Data Structure

Linear Algebra Module :

- Given A, x and Θ, compute Ax, A^Tx, $A\Theta A^T$.
- Given $A\Theta A^T$, compute the Cholesky factorization $A\Theta A^T = LL^T$
- Given L and r, solve $Lx = r$ and $L^Tx = r$

Common choice: A *single* data structure to compute the general sparse operations

How can we deal with block-structures?
Blocks may be

- general sparse matrices
- dense matrices
- very particular matrices (identity, projection, GUB)
- block-structured

It is worth considering *many* data structures.

Important observation for primal and dual block-angular structures:

The linear-algebra module can be implemented using the linear algebra modules of blocks.

Solutions

- Exploit block structures using essentially the same operations.
- Use data abstraction to achieve generality.
- How the matrix is *stored* defines how computations are done.
- Add many particular data structures rather than modify previous one for new applications

We define a **Polymorphic Algebra Class**:
an Algebra for IPMs.

The Algebra contains the following functions:

- `NewAlgebra`
- `FreeAlgebra`
- `PrintAlgebra`

- `MatrixTimesVector`
- `MatrixTransTimesVector`
- `ComputeAThetaATrans`
- `ComputeCholesky`
- `SolveL`
- `SolveLTrans`

- `GetColumn`
- `GetSparseColumn`
- `SolveL_Ej`
- `SolveLTrans_Ej`

Vector

We define a **Vector Class**
for primal and dual vectors in IPMs.

Vector is a friend of **Algebra**.

The Vector contains the following functions:

- `NewVector`
- `FreeVector`
- `PrintVector`

- `ddotVector`
- `copyVector`
- `daxpyVector`
- `normofVector`

The key program construct that describes the structure of the LP is a **Tree**.
Tree has a recursive definition.

Both **Vector** and **Algebra** are friends of **Tree**.

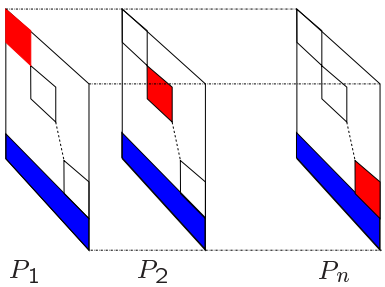
Parallelism

Two distinct parts:

- The Linear Algebra Module
- The Primal-dual method

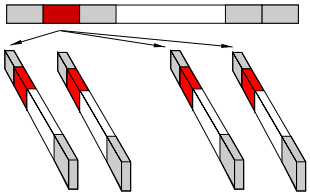
Linear Algebra Module:

- Memory Layout for the Matrix



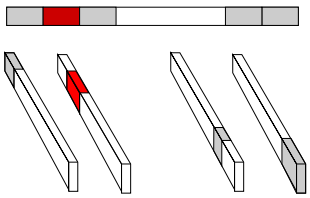
- Very good speed-ups
- Reduces peak memory

Bad memory layout for vectors:

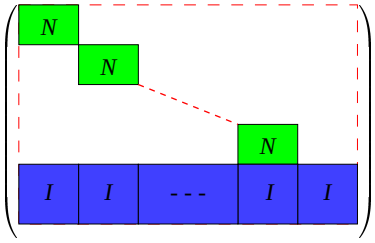


- excessive communications
- bad balance between computations and communications

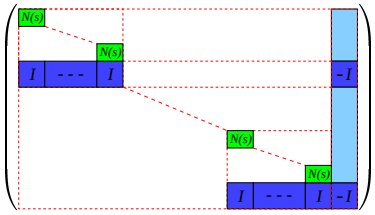
Good memory layout for vectors:



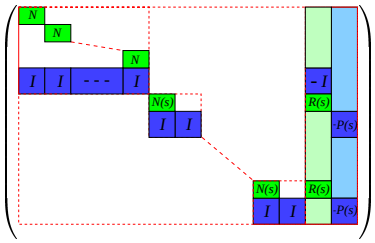
- Multicommodity flow problem



- Survivable network design problem



- Network capacity investment problem



Multicommodity Flow Problem

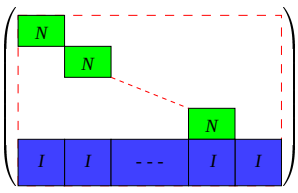
A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is given. $\mathcal{V}$ are its **nodes** and $\mathcal{E} \subset \{(i,j) : i \in \mathcal{V}, j \in \mathcal{V}, i \neq j\}$ are its **arcs**.

We associate **cost** $c_{ij} \geq 0$ with arc (i,j).
We associate **capacity** $K_{ij} \geq 0$ with arc (i,j).
A set of **demands** $k \in \mathcal{D}$:
ship a flow from a **source** to a **target**.

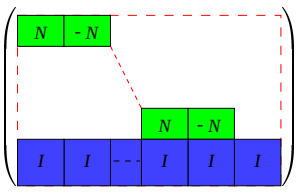
Minimum Cost Network Flow Problem:

$$\begin{array}{lll} \min & \sum_{(i,j)\in\mathcal{E}} c_{ij} \sum_{k\in\mathcal{D}} x_{ij}^{(k)} & \\ \text{s. t.} & \sum_{k\in\mathcal{D}} x_{ij}^{(k)} \leq K_{ij}, & \forall (i,j) \in \mathcal{E}, \\ & N x^{(k)} = d^{(k)}, & \forall k \in \mathcal{D}. \end{array}$$

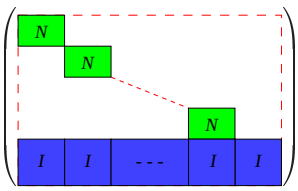
directed network



undirected network



Minimum Cost Network Flow Problem



Prob	Network data Nodes	Arcs	Demands
RealNet	119	308	7021
Random6	100	300	200
Random12	300	600	1000
Random16	300	1000	1000
Random20	400	1000	5000

Pentium Pro 300 MHz, 384 MB:

Problem	Rows	Columns	Iters	Time
RealNet	14232	72996	31	98
Random6	8715	51300	20	35
Random12	88506	353400	40	1183
Random16	87710	581000	39	2958
Random20	160201	799000	46	5823

SUN Enterprise 450

4 processors 400MHz UltraSparc-II with 4MB built-in cache. Each processor has 512MB RAM.

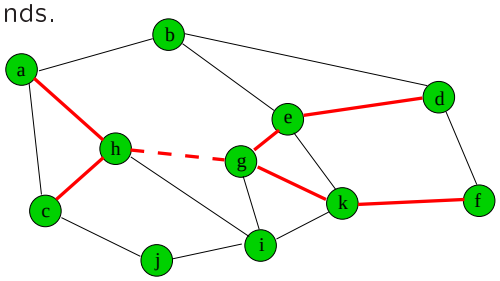
Minimum Cost Network Flow Problem:

Prob	Sizes	
	Rows	Cols
RealNet	14232	72996
Random6	8715	51300
Random12	88506	353400
Random16	87710	581000
Random20	160201	799000

Prob	1 Proc		2 Procs		3 Procs		4 Procs	
	time	S-up	time	S-up	time	S-up	time	S-up
RN	75	1.0	40	1.88	28	2.68	22	3.41
R6	68	1.0	39	1.74	28	2.43	23	2.96
R12	1243	1.0	709	1.75	504	2.47	403	3.08
R16	3401	1.0	1987	1.71	1367	2.49	1097	3.10
R20	5041	1.0	2826	1.78	1942	2.60	1593	3.16

Speed-ups:
about 1.8 on two processors;
about 2.5 on three processors;
about 3.1 on four processors.

A network is *survivable* if, following an elementary failure, there is a way, using some capacity, to rearrange the traffic assignment to meet all demands.



Two routings:
from node a to node f, 3 units pass through the edges (a,h), (h,g), (g,k) and (k,f);
from node c to node d, 5 units pass through the edges (c,h), (h,g), (g,e) and (e,d).
Common edge (h,g) breaks down.

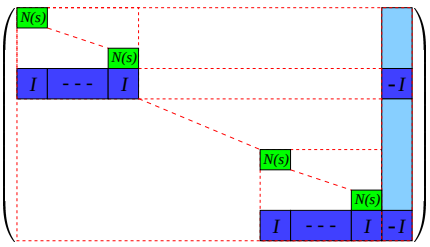
Local rerouting: one demand: send $5+3=8$ units between the endpoints h and g of the broken edge.
Global rerouting: two demands: send 3 units between a and f, and 5 units between c and d.

Survivable Network Design Problem

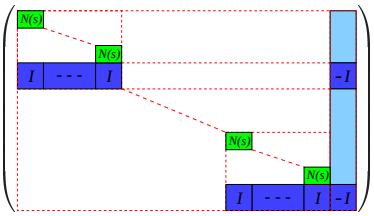
A variable $x^{(k)} = (x_{i,j}^{(k)})_{(i,j)\in\mathcal{E}(s)}$ represents the feasible flow of $\mathcal{G}(s)$ between the source and the target node, for a demand $d^{(k)}, k \in \mathcal{R}_s$ in state $s \in \mathcal{S}$.

Survivable Network Design Problem:

$$\begin{aligned}
\min \quad & c^T y \\
\text{s. t.} \quad & \sum_{k\in\mathcal{R}_s} x_{ij}^{(k)} \le K_{ij}^{(s)} + y_{ij}, && \forall (i,j)\in\mathcal{E}(s), \forall s\in\mathcal{S}, \\
& N(s)\, x^{(k)} = d^{(k)}, && \forall k \in \mathcal{R}_s, \forall s \in \mathcal{S}, \\
& x^{(k)} \ge 0, && \forall k \in \mathcal{R}_s, \forall s \in \mathcal{S}, \\
& 0 \le y \le \bar{y},
\end{aligned}$$



Survivable Network Design Problem



Prob	Basic data			Failure data	
	nodes	arcs	Routes	Fails	CondDems
T1	16	23	84	38	300
T2	31	68	961	99	2877
P56	35	56	1000	85	5832
PB1	30	57	150	81	1110
PB2	37	71	300	102	3266
PB3	40	77	200	109	1942
PB4	45	87	300	123	3658
PB5	65	127	400	179	7234
pb14	111	25	500	135	1804

Prob	Size		New IPM		Cplex 6.0	
	Rows	Cols	Iters	Time	Iters	Time
T1	3100	7466	16	6	12	8.8
T2	31542	117468	32	396	38	Failed
P56	55630	168161	35	363	26	605
PB1	22213	72514	25	122	23	143
PB2	59021	207901	34	518	35	730
PB3	54657	188266	29	407	25	518
PB4	83561	294735	33	735	31	1131
PB5	242570	886178	48	3956	34	Failed
pb14	34847	197868	94	2661	57	1793

SUN Enterprise 450

4 processors 400MHz UltraSparc-II with 4MB built-in cache. Each processor has 512MB RAM.

Survivable Network Design Problem:

Prob	Sizes	
	Rows	Cols
PB1	22213	72514
PB2	59021	207901
PB3	54657	188266
PB4	83561	294735
PB5	242570	886178

Prob	1 Proc		2 Procs		3 Procs		4 Procs	
	time	S-up	time	S-up	time	S-up	time	S-up
PB1	93	1.0	50	1.86	38	2.45	30	3.10
PB2	385	1.0	201	1.91	146	2.64	119	3.24
PB3	310	1.0	171	1.81	118	2.63	94	3.30
PB4	601	1.0	375	1.60	239	2.52	208	2.89
PB5	3033	1.0	1733	1.75	1259	2.41	1086	2.80

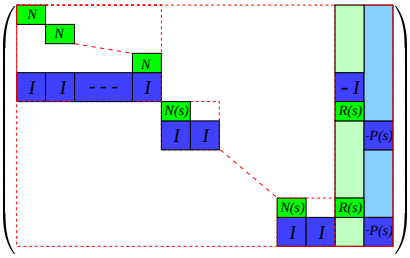
Speed-ups:

about 1.8 on two processors;
about 2.5 on three processors;
about 3.1 on four processors.

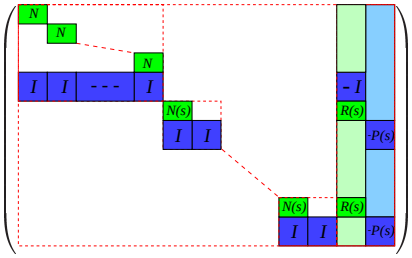
Base capacity $y = (y_{(i,j)})_{(i,j)\in\mathcal{E}}$.
Spare capacity $z = (z_{(i,j)})_{(i,j)\in\mathcal{E}}$.

Network Capacity Investment Problem:

$$
\begin{array}{lll}
\min & c^T y + d^T z & \\
\text{s. t.} & \sum_{k\in\mathcal{D}} x_{ij}^{(k)} \le y_{ij}, & \forall (i,j)\in\mathcal{E}, \\
& N\, x_0^{(k)} = d^{(k)}, & \forall k\in\mathcal{D}, \\
& \sum_{k\in\mathcal{R}_s} x_{ij}^{(k)} \le y_{ij} + z_{ij}, & \forall (i,j)\in\mathcal{E}(s), \forall s\in\mathcal{S}, \\
& N(s)\, x^{(k)} = d^{(k)}, & \forall k\in\mathcal{R}_s, \forall s\in\mathcal{S}, \\
& x_0^{(k)} \ge 0, & \forall k\in\mathcal{D}, \\
& x^{(k)} \ge 0, & \forall k\in\mathcal{R}_s, \forall s\in\mathcal{S}, \\
& 0 \le y \le \bar{y}, & \\
& z \ge 0. &
\end{array}
$$



Network Capacity Investment Problem



Problem	nodes	edges	demands
T1	12	25	66
T2	26	42	264
T3	53	79	1378
P1	25	41	300
P2	35	58	595
P3	45	91	990

Pr	Size		NewIPM		Cplex 6.0 Barrier		Cplex 6.0 Simplex	
	Rows	Cols	It	Time	It	Time	It	Time
T1	1021	2400	15	1.7	20	1.65	1577	2.0
T2	3414	7266	23	10.6	21	Failed	2852	6.8
T3	13053	26860	25	49.7	22	69.1	9112	68
P1	3241	6970	28	10.8	25	7.2	2474	5.2
P2	6492	13978	28	26.2	23	22.1	7829	46.3
P3	14221	32760	49	138.2	54	226.9	42520	867

Conclusions

It is **advantageous** to exploit structure.
It is **easy** to exploit block structure.

Flexibility:

- **tree** representation of the matrix
- common **layer** (interface) to every structure
- new data structures may be easily handled

Portability: (C and MPI):

- SUN Enterprise
- IBM SP1/2
- Cluster of Linux PC's

Efficiency:
achieved in sequential and parallel code.