A Brief History of Computational Linear Programming

Geraldo Veiga

$\mathcal{R}^n$ Ciência e Tecnologia

1st Brazilian Workshop on Interior Point Methods
April 27-28, 2015 - Campinas, Brazil

Outline

Pioneers I

- Fourier [1826] studies the properties of system of linear inequalities, more complex than system of equations
- De la Vallée-Poussin [1911] develops an iterative procedure for linear minimax estimation which can be adjusted to solve linear optimization problems [Farebrother, 2006]
- As early as 1930, A.N. Tolstoı described a number of solution approaches for transportation problems [Schrijver, 2012]
- Kantorovich [1939] proposes rudimentary algorithm for linear programming applied to production planning

Pioneers II

- These contributions only come to attention after independent development of linear programming theory and the Simplex Method
- Other contributions to optimization by mathematicians in the USSR also went unrecognized elsewhere victim of government personal and ideological obstacles imposed to international scientific interchange [Polyak, 2014]

Pioneers III

- A quote from Kantorovich betrays an attempt of making *dual prices* more palatable to Marxist orthodoxy [Todd, 2002]

  > *"I want to emphasize again that the greater part of the problems of which I shall speak, relating to the organization and planning of production, are connected specifically with the Soviet system of economy and in the majority of cases do not arise in the economy of a capitalist society."*
  > *[Kantorovich, 1939]*

The Simplex Method I

- George Dantzig proposes the Simplex Method in 1947 [Dantzig, 2002]
- Early works by Leontief, von Neumann and Koopsman directly influenced the theoretical development of linear programming [Dantzig, 2002]
- From Dantzig's point of view: Not just a qualitative tool in the analysis of economic phenomena, but a method to compute actual answers [Bixby, 2012]
- Unfortunately, not all economists are keen of numbers, the 1975 Nobel Prize in Economics was awarded to Kantorovich and Koopsman, ignoring Dantzig's contribution [Nobelprize.org, 2015]

The Simplex Method II

- First application to the solution of a non-trivial LP: 21x17 instance of Stigler Diet Problem (computation time was 120 man-days!) [Dantzig, 1963]
- Orchard-Hays (1954) produces first successful LP software
- Sparse matrix representation and product-form of the inverse
- Largest problem solved: 26 x 71 solved in 8 hours [Bixby, 2002]

The Simplex Method I

- Designed "to be computable", developed side-by-side with digital computers [Dantzig, 2002]
- Large scale methods: special matrix, structures Dantzig-Wolfe and Benders decomposition[Dantzig, 1955, Dantzig and Wolfe, 1961, Benders, 1962]

The Simplex Method II

- Familiar examples extracted from the original articles:

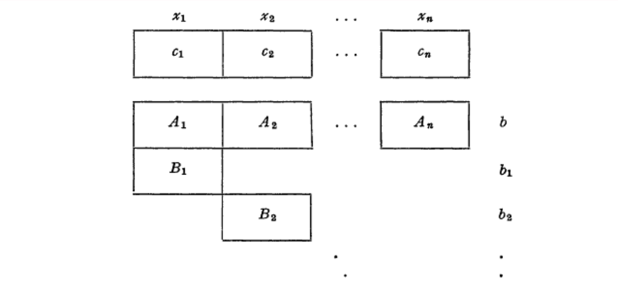
2. BLOCK TRIANGULARITY (GENERAL CASE)

By "block" triangular[2] we mean that if one partitions the matrix of coefficients of the technology matrix into submatrices, the submatrices (or blocks) considered as elements form a *triangular system*,

$$\begin{bmatrix} A_{11} & & & \\ A_{21} & A_{22} & & \\ \cdots & \cdots & \cdots & \\ A_{T1} & A_{T2} & \cdots & A_{TT} \end{bmatrix}. \tag{4}$$

[2] A term suggested by Walter Jacobs.

The Simplex Method III



The Simplex Method IV

For example, von Neumann, [**9**], in considering a constantly expanding economy, developed a linear dynamic model whose matrix of coefficients may be written in the form (5)

$$
(5)\qquad \begin{bmatrix}
A & & & & & \\
-B & A & & & & \\
 & -B & A & & & \\
 & & -B & \ddots & & \\
 & & & \ddots & \ddots & \\
 & & & & -B & A
\end{bmatrix},
$$

where A is the submatrix of coefficients of activities initiated in period t, and B is the submatrix of *output* coefficients of these activities in the following period.

- Extensions to quadratic programming and linear complementarity.
- *Large scale* not in today's sense. Then, 1000*x*2000 would be the limit of tractability with methods that used special matrix structures. Still, the methods survive, updated to advances in computer technology, especially parallel architectures

The Simplex Method V

- The common belief is that the Simplex Method is Exponential behavior in theory and almost linear in practice
- Sparse LU representation of the basis with Bartel-Golub/Forrest-Tomlin/Fletcher-Matthews update.[Forrest and Tomlin, 1972]

Doubting the Simplex Method I

- Even Dantzig had his doubts [Todd, 2002]:

  > *"Luckily the particular geometry used in my thesis was the one associated with the columns of the matrix instead of its rows. This column geometry gave me the insight which led me to believe that the simplex method would be an efficient solution technique. I earlier had rejected the method when I viewed it in the row geometry because running around the outside edges seemed so unpromising."* *[Dantzig, 1991]*

- Finite behavior was enough in early analysis of the algorithm
- However ... It was not even finite!

Doubting the Simplex Method II

- Cycling is possible in degenerate LPs.
- But it got fixed with Bland's pivoting Rule [Bland, 1977]
- Theory of computational complexity gets developed
- Is linear programming in P?
- Klee and Minty [1972] shows the simplex method with a common pivot rule is of exponential complexity

The Simplex method comes back I

- Theory counters with bounds on the expected number of pivot steps [Borgwardt, 1982]
- The work of Karmarkar had stimulated a rebirth of interest in LP, both on the theoretical and computation sides [Bixby, 2012]
- Computational studies on problems with numbers of variables ranging up to the millions also reaffirm confidence
- More recent linear algebra improvements such as Markowitz threshold and sparse partial pivoting [Bixby, 2002]
- Modern implementation (XMP, OSL, CPLEX, Gurobi, Mosek, Xpress) with power preprocessors,

The Simplex method comes back II

- The Simplex method is also naturally suited for mixed integer problem in Branch-and-bound and Branch-and-cut algorithms [Bixby, 2002]
- Remains a primary computational tool in linear and mixed-integer programming (MIP)
- Parallel implementations of the simplex method usually must exploit special structures
- A general approach hindered by the changing sparse pattern of the basic matrix

The Ellipsoid Method I

- The Ellipsoid Method [Khachiyan, 1979]
- Revolutionary for complexity theory without computational impact
- However brings back the idea of solving linear programs with traditionally non-linear techniques

Interior Point Methods I

- Karmarkar's algorithm [Karmarkar, 1984]
  - Projective algorithm with a potential function sets a lower complexity for linear programming: $\mathcal{O}(n^{3.5}L)$
  - Claims of great performance gains for a dual-affine scaling variant [Adler et al., 1989a]
  - Similar algorithm had gone unnoticed by LP researchers [Dikin, 1967]
- Primal-Dual/Path Following methods
  - New wave of interest in linear programming reintroduces path-following methods developed in the nonlinear context: Logarithm Barrier Function [Fiacco and McCormick, 1990] and Method of Centers [Huard, 1967]
  - Central trajectory methods with lower complexity $\mathcal{O}(n^3L)$
  - Primal/Dual infeasible methods become standard for implementation, included in leading LP software.
  - [Shanno, 2012]

A Personal Reminiscence I

- Initial contact with Karmarkar article
- Narendra Karmarkar was a recent PHd graduate from UC Berkeley, Computer Science department, working under Richard Karp
- Talk by Narendra Karmarkar at Evans Hall where he claimed a modified algorithm was "hundreds of times faster" than the Simplex Method
- Even featured on the first page of Sunday NY Times in November 18, 1984! [Gleick, 1984]
- "Breakthrough in Problems Solving" was the headline (N.B. AT&T was known for its ability to place Bell Labs stories in the science section of the NY Times)
- But making it first page news was unheard

A Personal Reminiscence II

- In Berkeley's IEOR computer lab, I built a simple implementation in APL of the original projective algorithm confirmed the small number of iterations
- Linear algebra infrastructure under APL did not allow for a serious performance analysis
- Ilan Adler arranged with Narendra Karmakar to collaborate in a serious implementation to vouch for the speed claims
- Mauricio Resende and myself ended up leading the effort along with other students in Ilan Adler's graduate seminar [Adler et al., 1989b]

A Personal Reminiscence III

- Contrary to other recollections [Gill et al., 2008], we were never told or felt that any of the ideas in the algorithms were proprietary by AT&T. There was however a self imposed restriction to avoid any possible copyright infringement. Then, we never saw nor asked for code that Karmarkar might have written himself.
- In retrospect, I feel that his comments in the early talks on the speed of the algorithms were based on simple prototypes that were not ready for sharing.
- The ill feelings in initial presentations by Karmarkar [Shanno, 2012] were caused more from problem of personal style than company policy.

A Personal Reminiscence IV

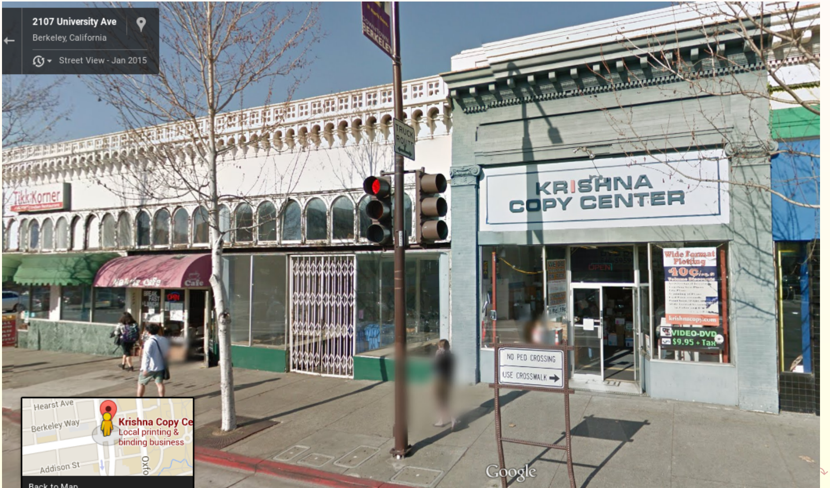
- However, a very legitimate complain comes later with AT&T trying to enforce patents on the Karmarkar's algorithm and the affine scaling interior point method [Karmarkar, 1988, Vanderbei, 1988, 1989]
- Eventualy AT&T's Korbx system, an attempt to commercialize interior point methods, failed, making this discussion mute.

A Personal Reminiscence

- Library of Alexandria - Interior Point Wing

A Personal Reminiscence

- Christiano Lyra knocks at my door selects a pile of papers and runs to Krishna Copy Center



A Personal Reminiscence

- Back at Unicamp brilliant student Aurélio Oliveira reads the whole
  lot, writes dissertation, articles etc ... Ans begets this workshop.

Dual Affine Algorithm

- c, x n-vectors; A $m \times n$ matrix; b, y m-vectors

$$\max \{ b^\top y \mid A^\top y \leq c \}$$

- Add slack variables

$$\max \{ b^\top y \mid A^\top y + v = c, \; v \geq 0 \}$$

- Scaling transformation

$$\hat{v} = D_v^{-1} v \;\; \text{where} \;\; D_v = \mathrm{diag}(v_1^k, \ldots, v_m^k)$$

- Projected gradient as search direction

$$h_y = (A D_v^{-2} A^\top)^{-1} b \;\; \text{and} \;\; h_v = -A^\top h_y$$

Affine-Dual Algorithm II

procedure dualAffine (A, b, c, y^0, *stopping criterion*, γ)
$k := 0$;
do *stopping criterion* not satisfied $\rightarrow$
  $v^k := c - A^\top y^k$;
  $D_v := \mathrm{diag}(v_1^k, \ldots, v_m^k)$;
  $h_y := (AD_v^{-2}A^\top)^{-1} b$;
  $h_v := -A^\top h_y$;
  if $h_v \geq 0 \rightarrow$ **return fi**;
  $\alpha := \gamma \times \min\{-v_i^k/(h_v)_i \mid (h_v)_i < 0, i = 1, \ldots, m\}$;
  $y^{k+1} := y^k + \alpha h_y$;
  $k := k + 1$;
od
end dualAffine

Primal-Dual Algorithm with infeasibilities I

- Formulation:
  - Upper bounds for a subset of variables
  - c, x, s, z are n-vectors
  - u_b, x_b, s_b, w_b n_b-vectors – x_n, s_n n_n-vectors
  - A $m \times n$ matrix – b, y m-vectors
- Add slack variables

$$\min \{c^\top x \mid Ax = b, x_b + s_b = u_b,\ x \geq 0, s_b \geq 0\}$$

$$\max \{b^\top y - u_b^\top w_b \mid A_b^\top y - w_b + z_b = c_b,$$
$$A_n^\top y + z_n = c_n,\ w_b \geq 0,\ z \geq 0\}$$

Primal-Dual Algorithm with infeasibilities II

- $X = \mathrm{diag}(x)$, $S = \mathrm{diag}(s)$, $W = \mathrm{diag}(w)$, $Z = \mathrm{diag}(z)$
- μ Central trajectory parameter
- Karush-Kuhn-Tucker conditions:

$$
\begin{aligned}
Ax &= b \\
x_b + s_b &= u_b \\
A_b^\top y - w_b + z_b &= c_b \\
A_n^\top y + z_n &= c_n \\
XZe &= \mu e \\
S_b W_b e &= \mu e \\
x, s_b, w_b, z &> 0
\end{aligned}
$$

Primal-Dual Algorithm with infeasibilities III

- System of equations with primal and dual infeasibilities

$$
\begin{aligned}
A\Delta x^k &= -(Ax^k - b) = r_p^k \\
\Delta x_b^k + \Delta s_b^k &= -(x_b^k + s_b^k - u_b) = r_u^k \\
A_b^\top \Delta y^k - \Delta w_b^k + \Delta z_b^k &= -(A_b^\top y^k + z_b^k - w_b^k - c_b) = (r_d^k)_b = 0 \\
A_n^\top \Delta y^k + \Delta z_n^k &= -(A_n^\top y^k + z_n^k - c_n) = (r_d^k)_n \\
Z^k \Delta x^k + X^k \Delta z^k &= -(X^k Z^k e - \mu_k e) = r_{xz}^k \\
W_b^k \Delta s_b^k + S_b^k \Delta w_b^k &= -(W_b^k S_b^k e - \mu_k e) = (r_{sw}^k)_b
\end{aligned}
$$

Primal-Dual Algorithm with infeasibilities IV

- Normal Equations

$$A\Theta^k A^\top \Delta y^k = \bar{b}$$

where

$$\Theta^k = \begin{bmatrix} (Z_b^k(X_b^k)^{-1} + W_b^k(S_b^k)^{-1})^{-1} & 0 \\ 0 & (Z_n^k)^{-1}X_n^k \end{bmatrix}$$

$$\begin{aligned} \bar{b} = r_p^k &+ A_b\Theta_b^k((r_d^k)_b + (S_b^k)^{-1}(r_{sw}^k - W_b r_u^k) - (X_b^k)^{-1} r_{xz}^k) \\ &+ A_n\Theta_n^k((r_d^k)_n - (X_n^k)^{-1} r_{xz}^k) \end{aligned}$$

Primal-Dual Algorithm with infeasibilities V

- Other search direction computed without substantial computational effort

$$
\begin{aligned}
\Delta x_b^k =& \Theta_b^k A_b^\top \Delta y^k - \Theta_b^k((r_d^k)_b + \\
& (S_b^k)^{-1}(r_{sw}^k - W_b r_u^k) - (X_b^k)^{-1}(r_{xz}^k)_b) \\
\Delta x_n^k =& \Theta_n^k A_n^\top \Delta y^k - \Theta_n^k((r_d^k)_n - (X_n^k)^{-1}(r_{xz}^k)_n) \\
\Delta s_b^k =& r_u^k - \Delta x_b^k \\
\Delta z^k =& (X^k)^{-1}(r_{xz} - Z^k \Delta x^k) \\
\Delta w_b^k =& A_b^\top \Delta y^k + \Delta z_b^k
\end{aligned}
$$

Parallelization opportunities in Interior Point Direct Factorization I

- Main computational step common to all variants is the solutions of a system of normal equations

$$A\Theta^k A^\top \Delta y^k = \bar{b}$$

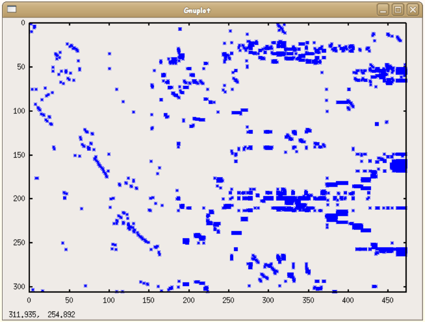
- Examining an implementation in Matlab/Octave, potentially computationally expensive steps:
- Computing system matrix

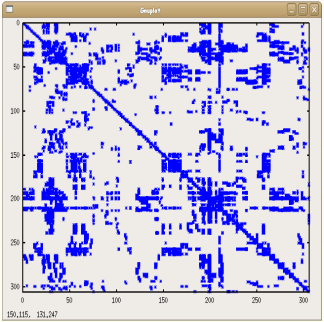
```
B = A*sparse(diag(d))*A';
```

- Custom parallel sparse linear algebra

Parallelization opportunities in Interior Point Direct Factorization II

- Example: BandM from the Netlib collection



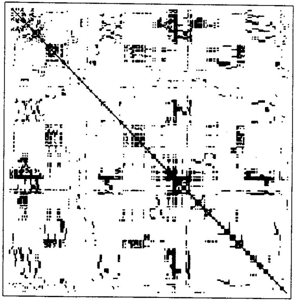


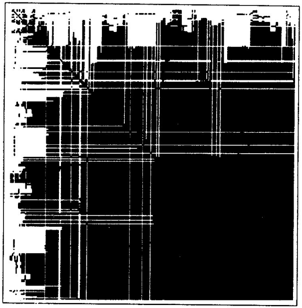
Parallelization opportunities in Interior Point Direct Factorization III

- Order for sparsity

```
ordering = symamd(B);
```

- Reordering for sparsity: Matrix $AA^\top$ and Cholesky factors without ordering [Adler et al., 1989a]





Parallelization opportunities in Interior Point Direct Factorization IV

- Matrix $AA^\top$ and Cholesky factors after minimum *degree ordering*

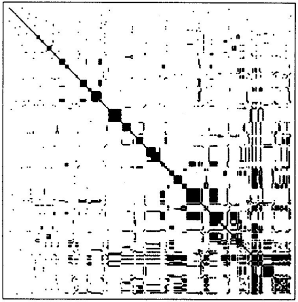


Figure 6. Nonzero pattern of AA^T after ordering (*minimum degree* ordering heuristic).

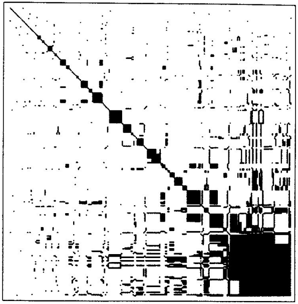


Figure 7. Nonzero pattern of *LU* factors after ordering (*minimum degree* ordering heuristic).

Parallelization opportunities in Interior Point Direct Factorization V

- Reordering rows of A to avoid *fill-in*
- Optimal ordering is *NP-Complete* [Yannakakis, 1981]
- Linear solvers compute the ordering during the *Analyse* step, based solely on the matrix sparsity pattern
- Performed only once in interior point algorithms, sparsity pattern are identical for all iterations
- Parallel/Distributed MPI based implementations available: ParMETIS

Parallelization opportunities in Interior Point Direct Factorization VI

- Direct Cholesky factorization

```
R = chol(B(ordering,ordering));
```

- Repeated at every iteration, consumes most of the computational effort
- For larger problems: Main parallelization target
- Chart displaying the portion of the algorithm running time for Netlib problems, suggesting a increase with size
- Available Parallel/Distributed implementations: MUMPS [Amestoy et al., 2000] for distributed memory architectures and PARDISO for shared memory

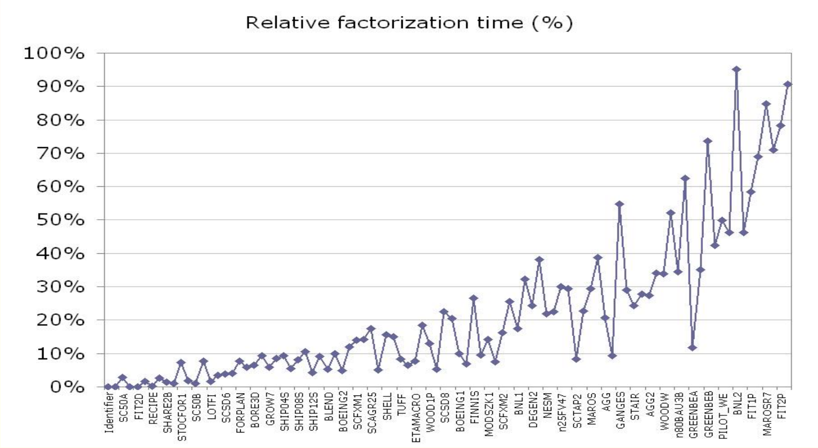
Parallelization opportunities in Interior Point Direct Factorization VII

- Triangular Solution for rhs

```
dy(ordering) = R\(R'\bbar(ordering));
```

- General sparse linear algebra parallelization
- In distributed implementations, parallelization implied by *Factorization* step

Parallelization opportunities in Interior Point Direct Factorization VIII



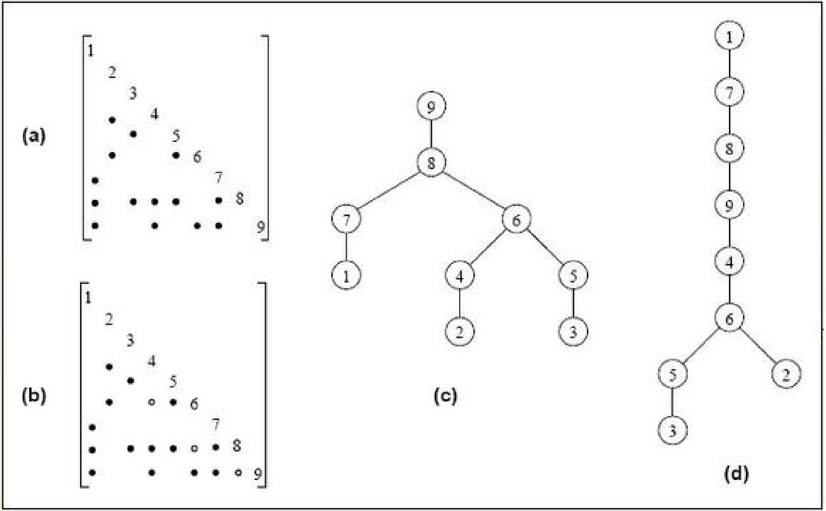
Parallelization strategies

- MPI: Message Passing/Distributed Memory
  - Standard for high-performance computing
  - Processors operate with private memory spaces, sharing results of only through point-to-point or collective communication
  - Goals are high performance, scalability and portability
  - Bindings for Fortran and C/C++
  - Target architectures are both high performance computer clusters tightly linked with fast switched interconnects and grids of loosely-coupled systems
- Shared memory multiprocessing
  - Multiple computing threads operate in shared memory space
  - Programming standards: OpenMP and Pthreads (Posix threads)
  - Suited for multi-core processor architectures
- Hybrid model of parallel programming use multi-core MPI nodes executing shared memory threads

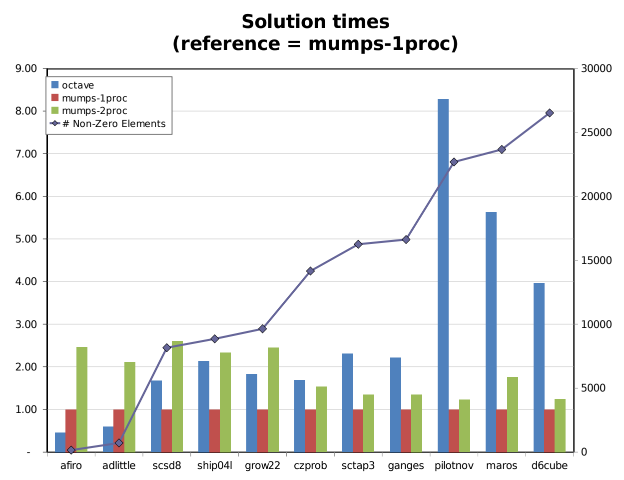
Experimenting with MUMPS

- Multifrontal Massively Parallel Solver MUMPS [Amestoy et al., 2000] for distributed memory architectures
- Multifrontal methods first build an assembly tree
- At each node, a dense submatrix (frontal matrix) is assembled using data from the original matrix and from the children of the node
- Main source of parallelism consists in simultaneously assigning same level frontal matrices onto separate processors
- MUMPS uses standard linear algebra libraries BLAS, BLACS, ScaLAPACK
- BLAS functions can use shared memory parallelism, depending on implementation
- Experiments with Netlib collection unsuccessful due to small size, but suggest better performance as problems grow

Multifrontal assembly trees for two orderings



Experiments with Netlib problems



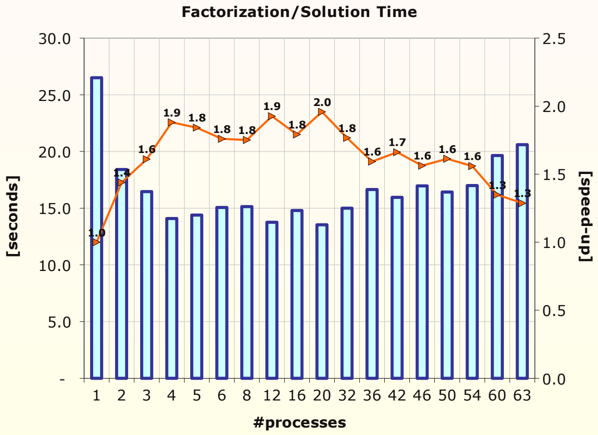
Power system expansion planning model

- Linear relaxation of mixed integer planning model for the expansion of a combined hydro and thermal power system
- Formulated with Optgen© modeling tool, developed by PSR
- Problem instance generated with Brazilian system of 280 hydro and 120 thermal plants
- LP size: 840285 columns, 598066 rows and 2462627 nonzeros entries
- Interior Point linear system: 360455 rows and 27390204 nonzeros

Case Study Experiment

- Experiment solves one typical system from an interior point iteration
- SGI Altix ICE 8200 with 64 quad-Core Intel Xeon CPU and 512 Gbytes of distributed RAM, using a Infiniband interconnect
- Software infrastructure: MUMPS 4.8.3 with BLAS, BLACS, ScaLAPACK provided by Intel MKL 10.1
- MUMPS is successful in low-scale parallelization
- Times for the Analyze stage comparable
- Total computation is dominated by matrix-matrix multiplication
- Shared memory parallelism using OpenMP in the BLAS and Lapack routines has little effect in this architecture

MUMPS Speedup



Conclusion and future work

- This experiment with parallelization was class project at COPPE-UFRJ by Luiz Carlos da Costa, Jr. and Fernanda Thomé [Maculan et al., 2008]
- Large-scale problems using implementations with direct factorization can profit from parallelization, but less than expected
- Parallelization still an art form: No assurance of performance, too dependent on the infrastructure and algorithms
- MUMPS and other MPI-based tools are designed for high performance clusters
- Multi-core workstations are a better suited for shared memory parallelization
- Other sources of parallelism must be addressed
- Experiments with iterative methods for solving interior point linear systems

Some ideas for future development

- Identifying an optimal basis
  - Most modern optimization software packages include an Interior Point implementation. However, they still rely on a cumbersome *crossover* step to produce a Simplex-like optimal solution
  - In network flow problems it it possible to guess and test the optimal basis. More interestingly, the guessed basis is used to build a preconditioner in iterative solutions of the main system of equations Resende and Veiga [1993]
  - Can this be generalized for LP and separating preconditioners?
- Parallel implementations
  - Continued increase in computer processing power depends on multicore and distributed architectures
  - Successful parallelization using direct factorization only for special structure problems [Gondzio and Grothey, 2006]
  - Using pure iterative methods would be instantly parallelizable
  - Design of preconditioners must also take parallel architectures into consideration. Usually, they must yeald systems easy to solve (diogonal, triangular) and parallelize.

References I

Ilan Adler, Narendra Karmarkar, Mauricio G. C. Resende, and Geraldo Veiga. Data Structures and Programming Techniques for the Implementation of Karmarkar's Algorithm. *ORSA Journal on Computing*, 1(2):84–106, May 1989a. ISSN 0899-1499. doi: 10.1287/ijoc.1.2.84. URL http://pubsonline.informs.org/doi/abs/10.1287/ijoc.1.2.84.

Ilan Adler, Mauricio G. C. Resende, Geraldo Veiga, and Narendra Karmarkar. An implementation of Karmarkar's algorithm for linear programming. *Mathematical Programming*, 44(1-3):297–335, May 1989b. ISSN 0025-5610, 1436-4646. doi: 10.1007/BF01587095. URL http://link.springer.com/article/10.1007/BF01587095. An errata correcting the description of the power series algorithm was published in Mathematical Programming 50 (1991), 415.

P. R. Amestoy, I. S. Duff, and J.-Y. L'Excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Computer Methods in Applied Mechanics and Engineering*, 184:501–520, 2000. doi: 10.1016/S0045-7825(99)00242-X.

Jacques F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252, 1962. doi: 10.1007/BF01386316. URL http://www.springerlink.com/index/g203830n1gm58w73.pdf.

Robert E. Bixby. Solving Real-World Linear Programs: A Decade and More of Progress. *Operations Research*, 50 (1):3–15, February 2002. ISSN 0030-364X. doi: 10.1287/opre.50.1.3.17780. URL http://pubsonline.informs.org/doi/abs/10.1287/opre.50.1.3.17780.

Robert E. Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, pages 107–121, 2012. URL http://www.emis.ams.org/journals/DMJDMV/vol-ismp/25_bixby-robert.pdf.

References II

Robert G. Bland. New Finite Pivoting Rules for the Simplex Method. *Mathematics of Operations Research*, 2(2): 103–107, May 1977. ISSN 0364-765X. doi: 10.1287/moor.2.2.103. URL http://pubsonline.informs.org/doi/abs/10.1287/moor.2.2.103.

Dr K.-H. Borgwardt. The Average number of pivot steps required by the Simplex-Method is polynomial. *Zeitschrift für Operations Research*, 26(1):157–177, December 1982. ISSN 0340-9422, 1432-5217. doi: 10.1007/BF01917108. URL http://link.springer.com/article/10.1007/BF01917108.

George B. Dantzig. Upper Bounds, Secondary Constraints, and Block Triangularity in Linear Programming. *Econometrica*, 23(2):174–183, April 1955. ISSN 0012-9682. doi: 10.2307/1907876. URL http://www.jstor.org/stable/1907876.

George B. Dantzig. Linear Programming. In Jan Karel Lenstra, Alexander H. G. Rinnoy Kan, and (eds) Alexander Schrijver, editors, *History of Mathematical Programming, a Collection of Personal Reminiscences*. Elsevier Science Publishers B. V., Amsterdam, 1991. ISBN 0-444-88818-7.

George B. Dantzig. Linear Programming. *Operations Research*, 50(1):42–47, February 2002. ISSN 0030-364X. doi: 10.1287/opre.50.1.42.17798. URL http://pubsonline.informs.org/doi/abs/10.1287/opre.50.1.42.17798.

George B. Dantzig and Philip Wolfe. The Decomposition Algorithm for Linear Programs. *Econometrica*, 29(4): 767–778, October 1961. ISSN 0012-9682. doi: 10.2307/1911818. URL http://www.jstor.org/stable/1911818.

George Bernard Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963. ISBN 0691059136.

C.J. de la Vallée-Poussin. Sur la méthode de l'approximation minimum. *Ann. Soc. Sci. Bruxelles*, 35:1–16, 1911.

References III

I. I. Dikin. Iterative Solution of Problems of Linear and Quadratic Programming. *Soviet Mathematics Doklady*, 8: 674–675, 1967.

Richard William Farebrother. A linear programming procedure based on de la Vallée Poussin's minimax estimation procedure. *Computational Statistics & Data Analysis*, 51(2):453–456, November 2006. ISSN 01679473. doi: 10.1016/j.csda.2005.10.005. URL http://linkinghub.elsevier.com/retrieve/pii/S0167947305002574.

Anthony V. Fiacco and Garth P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. SIAM, 1990. ISBN 9780898712544.

J. J. H. Forrest and J. A. Tomlin. Updating triangular factors of the basis to maintain sparsity in the product form simplex method, Mathematical Programming 2, 263{278. *Mathematical Programming*, 2:263–278, 1972.

J. B. J. Fourier. Solution d'une Question Particulière du Calcul des Inégalités. *Nouveau Bulletin des Sciences par la Société Philomatique de Paris*, pages 99–100, 1826. reprinted: G. Olms, Hildesheim, 1970, pp. 317-319.

Philip E. Gill, Walter Murray, Michael A. Saunders, John A. Tomlin, and Margaret H. Wright. George B. Dantzig and systems optimization. *Discrete Optimization*, 5(2):151–158, May 2008. ISSN 1572-5286. doi: 10.1016/j.disopt.2007.01.002. URL http://www.sciencedirect.com/science/article/pii/S1572528607000321.

James Gleick. BREAKTHROUGH IN PROBLEM SOLVING. *The New York Times*, November 1984. ISSN 0362-4331. URL http://www.nytimes.com/1984/11/19/us/breakthrough-in-problem-solving.html.

References IV

Jacek Gondzio and Andreas Grothey. Direct Solution of Linear Systems of Size 109 Arising in Optimization with Interior Point Methods. In Roman Wyrzykowski, Jack Dongarra, Norbert Meyer, and Jerzy Waśniewski, editors, *Parallel Processing and Applied Mathematics*, number 3911 in Lecture Notes in Computer Science, pages 513–525. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-34141-3, 978-3-540-34142-0. URL http://link.springer.com/chapter/10.1007/11752578_62.

P. Huard. Resolution of mathematical programming with nonlinear constraints by the method of centers. pages 209–219. North-Holland, Amsterdam, 1967.

L. V. Kantorovich. Mathematical Methods of Organizing and Planning Production. *Management Sciences*, 6: 366–422, 1939. English translation of the Russian original published in 1939 by the Publication House of the Leningrad State University.

N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, December 1984. ISSN 0209-9683, 1439-6912. doi: 10.1007/BF02579150. URL http://link.springer.com/article/10.1007/BF02579150.

Narendra K. Karmarkar. Methods and apparatus for efficient resource allocation, May 1988. URL http://www.google.com.br/patents/US4744028. U.S. Classification 705/7.22, 700/99, 705/7.25, 705/7.38, 705/7.37; International Classification G06Q10/00, G06Q50/00, H04M7/00, G05B19/418, H04Q3/66, G06F19/00, B65G61/00, G06F9/46, H04M3/00, G05B13/02; Cooperative Classification G06Q10/06312, H04Q3/66, G06Q10/06375, G06Q10/0639, G06Q10/06315, G06Q10/04; European Classification G06Q10/04, G06Q10/06375, G06Q10/0639, G06Q10/06315, G06Q10/06312, H04Q3/66.

L. G. Khachiyan. A Polynomial Algorithm in Linear Programming. *Soviet Mathematics Doklady*, 20:1979, 1979.

V. Klee and G. J. Minty. How Good is the Simplex Algorithm? In O. Shisha, editor, *Inequalities III*, pages 159–175. Academic Press Inc., New York, 1972.

References V

Nelson Maculan, Geraldo Veiga, Luiz Carlos da Costa Jr., and Fernanda S. Thomé. Parallélisation des algorithmes de points intérieurs pour la programmation linéaire, November 2008. URL http://www.lamsade.dauphine.fr/~jfro/JourneesPrecedentes/anciennesJournees/jfro20.html#mardi.

Nobelprize.org. The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 1975, April 2015. URL http://www.nobelprize.org/nobel_prizes/economic-sciences/laureates/1975/.

B. T. Polyak. History of mathematical programming in the USSR: analyzing the phenomenon. *Mathematical Programming*, 91(3):401–416, April 2014. ISSN 0025-5610, 1436-4646. doi: 10.1007/s101070100258. URL http://link.springer.com/article/10.1007/s101070100258.

M. Resende and G. Veiga. An Implementation of the Dual Affine Scaling Algorithm for Minimum-Cost Flow on Bipartite Uncapacitated Networks. *SIAM Journal on Optimization*, 3(3):516–537, August 1993. ISSN 1052-6234. doi: 10.1137/0803025. URL http://epubs.siam.org/doi/abs/10.1137/0803025.

Alexander Schrijver. On the history of the transportation and maximum flow problems. *Mathematical Programming*, 91(3):437–445, 2012. URL http://link.springer.com/article/10.1007/s101070100259.

David Shanno. Who Invented the Interior-Point Method? pages 55–64, 2012. ISSN 1431-0643.

Michael J. Todd. The many facets of linear programming. *Mathematical Programming*, 91(3):417–436, February 2002. ISSN 0025-5610, 1436-4646. doi: 10.1007/s101070100261. URL http://link.springer.com/10.1007/s101070100261.

Robert J. Vanderbei. Methods and apparatus for efficient resource allocation, May 1988. URL http://www.google.com/patents/US4744026. U.S. Classification 705/7.22, 379/112.05, 705/7.37, 705/7.12; International Classification G06Q10/00; Cooperative Classification G06Q10/06312, G06Q10/0631, G06Q10/06375, G06Q10/06; European Classification G06Q10/06, G06Q10/0631, G06Q10/06375, G06Q10/06312.

References VI

Robert J. Vanderbei. Methods and apparatus for efficient resource allocation, December 1989. URL http://www.google.com/patents/US4885686. U.S. Classification 700/99, 379/112.05, 340/4.6; International Classification G06Q10/06; Cooperative Classification G06Q10/06; European Classification G06Q10/06.

M. Yannakakis. Computing the Minimum Fill-In is NP-Complete. *SIAM Journal on Algebraic Discrete Methods*, 2 (1):77–79, March 1981. ISSN 0196-5212. doi: 10.1137/0602010. URL http://epubs.siam.org/doi/abs/10.1137/0602010.