Tulip.jl: an interior-point solver with abstract linear algebra

Miguel Anjos [a,b] Andrea Lodi [a,b,c] **Mathieu Tanneau** [a,b,c]

(a) École polytechnique de Montréal

(b) GERAD

(c) CERC in Data Science for Real-time decision making

March 13, 2019

GERAD GROUP FOR RESEARCH IN DECISION ANALYSIS

CANADA EXCELLENCE RESEARCH CHAIR DATA SCIENCE FOR REAL-TIME DECISION-MAKING

POLYTECHNIQUE MONTRÉAL

Linear programming (Primal-Dual standard form)

$$(P) \quad \min_{\mathbf{x}} \quad c^T\mathbf{x} \quad s.t. \quad A\mathbf{x} = b, \quad \mathbf{x} \geq 0$$

$$(D) \quad \max_{\mathbf{y},\mathbf{s}} \quad b^T\mathbf{y} \quad s.t. \quad A^T\mathbf{y} + \mathbf{s} = c, \quad \mathbf{s} \geq 0$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^m$, $\mathbf{s} \in \mathbb{R}^n$

- Solved with Simplex or Interior-Point
- Workhorse of
  - MILP
  - Decomposition (Dantzig-Wolfe & Benders)
  - Polyhedral Outer approximations
  - Cutting plane methods
  - ...

Geometric view

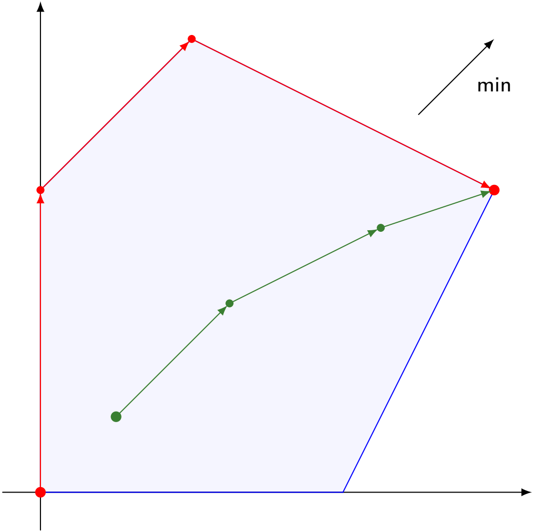
$$(P) \quad \min_{\mathbf{x}} \quad c^T \mathbf{x}$$
$$s.t. \quad A\mathbf{x} = b$$
$$\mathbf{x} \geq 0$$

Simplex

- Many cheap iterations
- Extreme vertices (basic points)

Interior-Point

- Few expensive iterations
- Interior points ($\mathbf{x} > 0$)



$$(P) \quad \min_{\mathbf{x}} \quad c^T\mathbf{x} \quad s.t. \quad A\mathbf{x} = b, \quad \mathbf{x} \geq 0$$

$$(D) \quad \max_{\mathbf{y},\mathbf{s}} \quad b^T\mathbf{y} \quad s.t. \quad A^T\mathbf{y} + \mathbf{s} = c, \quad \mathbf{s} \geq 0$$

KKT optimality conditions:

$$A\mathbf{x} = b \quad \text{[primal feas.]} \tag{1}$$
$$A^T\mathbf{y} + \mathbf{s} = c \quad \text{[dual feas.]} \tag{2}$$
$$\forall i, \ \mathbf{x}_i \cdot \mathbf{s}_i = 0 \quad \text{[slackness]} \tag{3}$$
$$\mathbf{x}, \mathbf{s} \geq 0 \tag{4}$$

Algorithm	(1)	(2)	(3)	$\mathbf{x} \geq 0$	$\mathbf{s} \geq 0$
(primal) Simplex	✓	✓	✓	✓	*
Interior-point	✓	✓	*	$\mathbf{x} > 0$	$\mathbf{s} > 0$

✓: at each iteration; *: at optimality only

Short history of IPMs:

- The seminal paper [Karmarkar, 1984]
- [Mehrotra, 1992]: predictor-corrector algorithm (implemented in most IPM codes)
- Multiple centrality corrections [Gondzio, 1996]
- Reference textbook [Wright, 1997]
- [Gondzio, 2012]: more recent survey of IPMs

(Some) software for LP/QP:

- All commercial solvers (CPLEX, GRB, Mosek, Xpress, etc.)
- Open source: CLP, GLPK, OOQP, (PCx), (HOPDM)

Compute initial point (see [Mehrotra, 1992])

$$(\mathbf{x}^0, \mathbf{y}^0, \mathbf{s}^0) \text{ with } \mathbf{x}^0 > 0, \mathbf{s}^0 > 0$$

Compute search direction

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{bmatrix} \cdot \begin{bmatrix} \Delta x^{aff} \\ \Delta y^{aff} \\ \Delta s^{aff} \end{bmatrix} = \begin{bmatrix} b - A\mathbf{x} \\ c - A^T\mathbf{y} - \mathbf{s} \\ -XSe \end{bmatrix} \quad \text{[predictor]}$$

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{bmatrix} \cdot \begin{bmatrix} \Delta x^{cc} \\ \Delta y^{cc} \\ \Delta s^{cc} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sigma\mu e - \Delta X^{aff} \Delta S^{aff} \end{bmatrix} \quad \text{[corrector]}$$

Update current solution

$$(\mathbf{x}^+, \mathbf{y}^+, \mathbf{s}^+) = (\mathbf{x}, \mathbf{y}, \mathbf{s}) + \alpha(\Delta^{aff} + \Delta^{cc})$$

Repeat until convergence

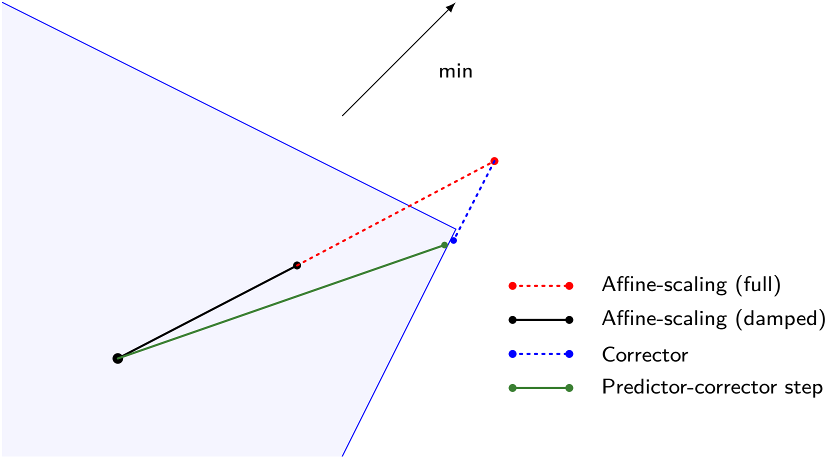


Figure: Mehrotra's Predictor-Corrector, in **x** space

- LP in standard Primal-Dual form

$$
(P) \quad \min_{\mathbf{x}} \quad c^T \mathbf{x} \quad s.t. \quad A\mathbf{x} = b, \quad \mathbf{x} \geq 0
\qquad\qquad
(D) \quad \max_{\mathbf{y},\mathbf{s}} \quad b^T \mathbf{y} \quad s.t. \quad A^T \mathbf{y} + \mathbf{s} = c, \quad \mathbf{s} \geq 0
$$

- At each iteration, solve (several) Newton systems of the form

$$
\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \end{bmatrix} = \begin{bmatrix} \xi_d \\ \xi_p \\ \xi_{\mathbf{xs}} \end{bmatrix}
$$

- Polynomial-time algorithm (see [Wright, 1997])
- Very efficient on large-scale problems

Newton systems of the form

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \end{bmatrix} = \begin{bmatrix} \xi_d \\ \xi_p \\ \xi_{\mathbf{xs}} \end{bmatrix}$$

solved multiple times in each iteration, with various right-hand side.

Two ways to make an Interior-Point faster:

- Reduce the number of iterations (better algorithm)
- Reduce the time per iteration (better linear algebra)

Initial Newton system:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \end{bmatrix} = \begin{bmatrix} \xi_d \\ \xi_p \\ \xi_{xs} \end{bmatrix}$$

Substitute Δs to obtain the **Augmented system**

$$\begin{bmatrix} -\Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \xi_d - X^{-1}\xi_{xs} \\ \xi_p \end{bmatrix}$$

$$\Delta s = X^{-1}(\xi_{xs} - S\Delta x)$$

where $\Theta := XS^{-1}$

- :(Left-hand matrix is indefinite (though regularization can be used)
- :(Still costly to solve
- :) More handy if free variables and/or non-linear terms

Substitute Δx to obtain the **Normal equations**

$$
\begin{aligned}
(A\Theta A^T)\Delta y &= \xi_p + A\Theta(\xi_d - X^{-1}\xi_{xs}) \\
\Delta x &= \Theta(A^T\Delta y - \xi_d + X^{-1}\xi_{xs}) \\
\Delta s &= X^{-1}(\xi_{xs} - S\Delta x)
\end{aligned}
$$

:) $A\Theta A^T$ is positive-definite

:) Cholesky factorization $A\Theta A^T = LL^T$

$\implies$ specialized Cholesky based on A

Unit block-angular matrix

$$A = \begin{bmatrix} e^T & & \\ & \ddots & \\ & & e^T \\ A_1 & \cdots & A_N \end{bmatrix}$$

Found in Dantzig-Wolfe decomposition + column-generation

$$A\Theta A^T = \begin{bmatrix} e^T\theta_1 & & & (A_1\theta_1)^T \\ & \ddots & & \vdots \\ & & e^T\theta_R & (A_R\theta_R)^T \\ A_1\theta_1 & \cdots & A_R\theta_R & \Phi \end{bmatrix}$$

$\Longrightarrow$ exploit structure to accelerate Cholesky factorization

https://github.com/ds4dm/Tulip.jl

Main features

- Homogeneous self-dual algorithm + multiple corrections
- Upper-bounds handled explicitly
- Algorithm uses abstract linear algebra (`A::AbstractMatrix`)
- Generic sparse Cholesky + specialized for Unit block-angular
- MathProgBase interface

WIP

- MOI interface
- Improved stability & general sparse linear algebra

Netlib benchmark (`https://www.netlib.org/lp/`)

- Small LP instances, some problematic
- Only consider feasible instances with no free variables
- No presolve, no crossover, single thread
- Most solved in < 1s

Results

- Tulip runs into numerical issues numerical issues, but...
- ...faster than CLP, GLPK, IpOpt on “hard" instances
  (hard $=$ solved in > 0.1s by all solvers)

Instances:

- $m = 24, 48, 96$ linking constraints
- $N = 2^{10}$ to 2^{15} sub-problems
  Each sub-problem solved with Gurobi
- Same column-generation code
- Master problem statistics:
  - $N + m$ constraints
  - up to $\simeq 4{-}10 \times N$ variables
  - $\simeq 4{-}10 \times N \times m$ non-zeros

Solver setup:

- Barrier algorithm, no cross-over
- No presolve
- Single thread
- Tulip: Generic IPM + specialized linear algebra

Computational results:

- Barrier (almost always) faster than Simplex
- Computing times (for Restricted Master Problem)
  - vs Mosek: -33% (total time); -50% (per-iteration time)
  - vs Gurobi: -60% (total time); -70% (per-iteration time)
  - vs CPLEX: -55% (total time); -45% (per-iteration time)

Takeaway:

- IPM solver for linear programming
- Generic algorithm + specialized linear algebra
- Possible to beat SOTA solvers

Roadmap:

- MOI interface
- Numerical stability
- Extension to QP

Open JuMP-related questions:

- Passing structure information to solver
- Problem modification

Thank you!
https://github.com/ds4dm/Tulip.jl

Questions?

mathieu.tanneau@polymtl.ca

Gondzio, J. (1996).
Multiple centrality corrections in a primal-dual method for linear programming.
Computational Optimization and Applications, 6(2):137–156.

Gondzio, J. (2012).
Interior point methods 25 years later.
European Journal of Operational Research, 218(3):587 – 601.

Karmarkar, N. (1984).
A new polynomial-time algorithm for linear programming.
Combinatorica, 4(4):373–395.

Mehrotra, S. (1992).
On the implementation of a primal-dual interior point method.
SIAM Journal on Optimization, 2(4):575–601.

Wright, S. (1997).
Primal-Dual Interior-Point Methods.
Society for Industrial and Applied Mathematics.