Gurobi LP Algorithms

GUROBI
OPTIMIZATION

Design goals for Gurobi LP and MIP Algorithms

- Leading optimization performance
  - Particularly fast for large or difficult models
- Make the most of the latest CPUs
  - All licenses use all processor cores
  - Built with latest CPU instruction sets
- Interfaces that are lightweight
  - Model initialization uses little memory & CPU
  - Unified interface to access model elements

What's Inside Gurobi Optimizer

- Automatic presolve
  - For both LP and MIP
- Algorithms for continuous optimization
  - Simplex
  - Barrier
- Algorithms for discrete optimization
  - Branch-and-bound
- Programming interfaces
  - C, C++, Java, .NET, Python, MATLAB, R
- Full-featured interactive shell

Linear Programming

A *linear program* (LP) is an optimization problem of the form

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & \sum_{j=1}^{n} c_j x_j \\
\text{subject to} \quad & \sum_{j=1}^{n} A_{ij} x_j = b_i, \quad i = 1, \ldots, m, \\
& \ell_j \le x_j \le u_j, \quad j = 1, \ldots, n,
\end{aligned}
$$

Presolve

LP Presolve

- Goal
  - Reduce the problem size
- Example

$x + y + z \le 5$ (1)

$u - x - z = 0$ (2)

……….

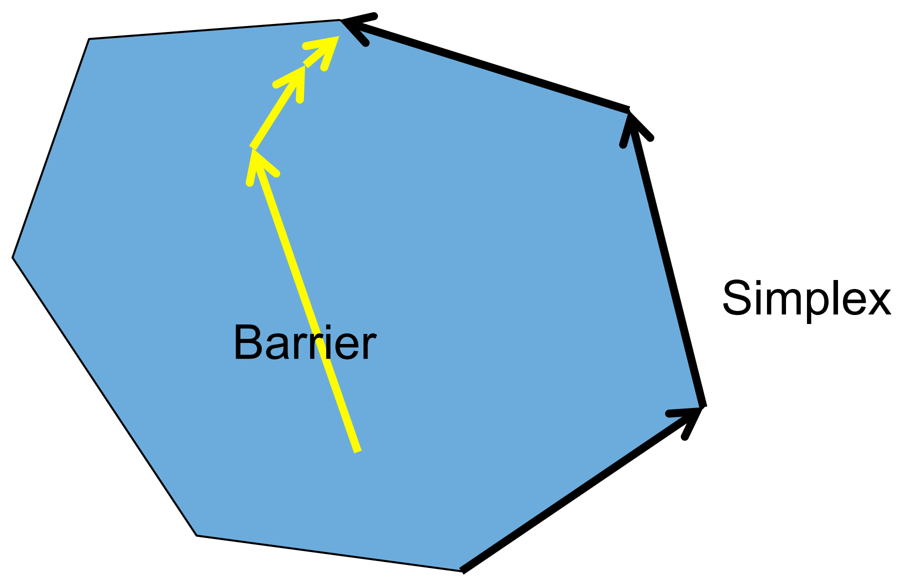
$0 \le x, y, z \le 1$ (3)

u is free (4)

- Reductions
  - Redundant constraint
    - (3) -> $x + y + z \le 3$, so (1) is redundant
  - Substitution
    - (2) and (4) -> u can be substituted with $x + z$

Simplex and Barrier

Simplex and Barrier



Simplex and Barrier

- Primal & dual simplex method
  - Numerically stable (most challenging part)
- Parallel barrier method with crossover
  - Can effectively exploit multiple cores
- Concurrent optimization
  - Run both simplex and barrier simultaneously
  - Solution is reported by first one to finish
  - Great use of multiple CPU cores
  - Best mix of speed and robustness
  - Distributed concurrent:
    - Run simplex and barrier on different machines

Karush-Kuhn-Tucker Conditions (LP)

- Conditions for LP optimality:
  - Primal feasibility: $Ax = b$ $(x \geq 0)$
  - Dual feasibility: $A'y + z = c$ $(z \geq 0)$
  - Complementarity: $x'z = 0$

	Primal feas	Dual feas	Complementarity
Primal simplex	Maintain	Goal	Maintain
Dual simplex	Goal	Maintain	Maintain
Barrier	Goal	Goal	Goal

Simplex

- Primal feasibility constraints

$$Ax = b$$

- Partition into basic and nonbasic variables

$$Bx_B + Nx_N = b$$

- Solve for basic variables

$$x_B = B^{-1}\left(b - Nx_N\right)$$

Solved by maintaining B=LU

Pricing

- Dual variables

$$y = B^{-T} c_B$$

- Reduced costs

$$z = c - A^T y$$

- Reduced costs give pricing information
  - Change in objective per unit change in variable value

Pivot

$$x_B = B^{-1}\left(b - Nx_N\right)$$

- Simplex pivot:
  - Choose a non-basic variable to enter the basis
    - Pick one with a negative reduced cost
  - Push one variable out of the basis
  - Update primal variables, dual variables, reduced costs, basis, basis factors, etc.
  - Continue

Simplex Example

- Min $z = 9 - 2x_3 - x_4 + 3s_1 + 4s_2$
  $x_1 = 3 - 8x_3 + x_4 + 2s_1 + 3s_2$
  $x_2 = 2 + 6x_3 + 2x_4 + 3s_1 + 4s_2$
  Where x_1, x_2 are basic, $x_i, s_i \geq 0$

- Pricing strategies to select entering variable
  - Both x_3 and x_4 have negative reduced costs
  - $rc(x_3) = -2 < -1 = rc(x_4)$, x_3 seems to be better, but
  - $norm(x_3) = sqrt(8*8+6*6) = 10$
  - $rc(x_3) / norm(x_3) > rc(x_4) / norm(x_4)$
  - x_4 is better based on the steepest edge algorithm

- Steepest edge algorithm (scaling by norm)
  - Crucial for simplex performance and robustness
  - Gurobi parameters: SimplexPricing, NormAdjust

Interior-Point Method

- Basic algorithm (Karmarkar, Fiacco & McCormick, Dikin):
  - Modify KKT conditions:
    - $Ax = b$
    - $A'y + z = c$
    - $Xz = \mu\, e$
- Linearize complementarity condition
- Iterate, reducing μ in each iteration
- Provable convergence

Reduces to a linear solve

- Linearize and simplify:

$$\begin{pmatrix} -\theta & A' \\ A & 0 \end{pmatrix} \begin{pmatrix} dx \\ dy \end{pmatrix} = \begin{pmatrix} r2 \\ r1 \end{pmatrix} \quad \text{(augmented system)}$$

  - $\theta_j = z_j / x_j$
  - $x_j * z_j = 0$ at optimality, so $\theta_j \rightarrow 0$ or ∞

- Further simplification:

$$A\,\theta^{-1} A'\; dy = b \qquad \text{(normal equations)}$$

Computational Steps

- In each iteration:
  - Form $A\,\theta^{-1}\,A'$
  - Factor $A\,\theta^{-1}\,A' = L\,D\,L'$ *(Cholesky fact.)*
  - Solve $L\,D\,L'\,x = b$
  - A few Ax and $A'x$ computations
  - A bunch of vector operations

Additional Steps

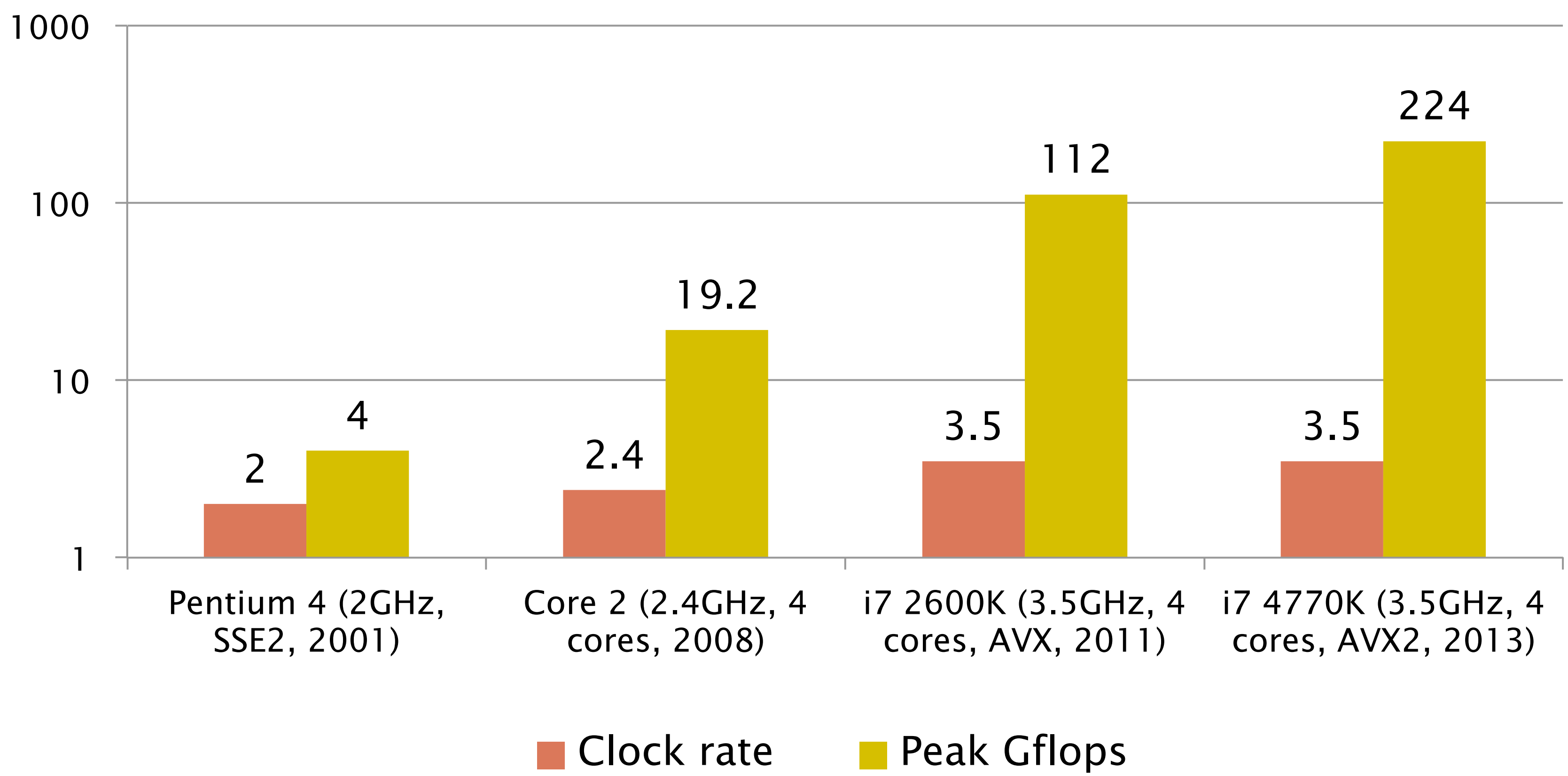
- Setup steps:
  - Presolve (same for simplex)
  - Compute fill-reducing ordering
- Post-processing steps:
  - Perform crossover to a basic solution

Essential Differences

- Simplex:
  - Thousand/millions of iterations on extremely sparse matrices
  - Each iteration extremely cheap
  - Few opportunities to exploit parallelism
- Barrier:
  - Dozens of expensive iterations
  - Much denser matrices
  - Lots of opportunities to exploit parallelism

Performance Trends

- CPU performance from 2001 to today:



LP Performance

- Performance results:
  - Gurobi 6.0, quad-core Xeon E3-1240
  - Dual simplex on 1 core, barrier on 4 cores
  - Models that take >1s

	GeoMean
Dual simplex	2.50
Primal simplex	5.27
Barrier	1.28
Concurrent	1.00
Det. concurrent	1.10

QP Performance

- For QP models, choice is much clearer:

	GeoMean
Dual simplex	50
Barrier	1

QCP Performance

- No simplex method for QCP
- Barrier is the only option

Performance on LP with Piecewise Objective

- Primal and dual simplex in Gurobi 6.0
- No Barrier method, only possible on converted model

Warm Start

- Warm start
  - Solve an LP and get optimal solution and basis
  - Change it slightly
  - Resolve it from previous solution or basis
- Simplex can warm start well
- Barrier cannot
- Warm start is crucial for MIP

Thank You