What's New In Gurobi 5.0

Dr. Edward Rothberg
Our Main Goals for Gurobi 5.0

- Expand the reach of the product:
  - New problem types:
    - Quadratic constraints:
      - QCP, SOCP, MIQCP
    - Massive numbers of constraints:
      - Through lazy constraints
  - New types of users:
    - MATLAB
    - R

- More options for working with infeasible models:
  - Barrier homogeneous algorithm
  - Feasibility relaxations
Our Main Goals for Gurobi 5.0

- Improved performance and robustness:
  - Barrier crossover improvements
  - Concurrent optimizer control
  - New simplex warm-start option

- Plus a few other features:
  - Retrieve variable/constraint by name
  - Floating license usage detail
Agenda

- Overview of 5.0
- Performance benchmarks
- Q&A
Quadratic Constraints

- Main focus of Gurobi 5.0
- We'll discuss them at the end
LAZY CONSTRAINTS
Models with Massive Constraint Counts

- Some models have too many constraints to explicitly state

- Gurobi 5.0 supports lazy constraints
  - State core constraints up front
  - Only add non-core constraints if/when they are violated

- Important difference between Gurobi and CPLEX
  - CPLEX lazy constraints force you to turn off important features
  - Parallel, dynamic search, ...
  - You take two steps backward before you even start
  - No need to turn off important features with our approach

- New Traveling Salesman Problem (TSP) example
  - Thorough demonstration of new feature
SUPPORT NEW TYPES OF USERS
Gurobi Interfaces

- In Gurobi 4.6, we support:
  - Native:
    - C, C++, C#, Java, Python, VB
  - Third-party:
    - AIMMS, AMPL, Frontline Solver (EXCEL), GAMS, MPL, Solver Foundation

- What's missing?
  - Two main target areas:
    - Engineering
    - Data analysis
Engineering

- MATLAB extremely widely used in science and engineering

- People already using Gurobi through MATLAB
  - Open-source GurobiMex wrapper [Wotao Yin]

- Advantages of Gurobi wrapper:
  - Exposes more Gurobi features
  - New Gurobi features supported immediately
    - E.g., quadratic constraints
  - We handle integration
    - No need to compile anything
  - We support it
Predictive Analytics Platforms

- One measure of platform popularity: kaggle.com
  - "Kaggle is a platform for data prediction competitions that allows organizations to post their data and have it scrutinized by the world's best data scientists"...
What is R?

- "R is an open source programming language and software environment for statistical computing and graphics. The R language is widely used among statisticians for developing statistical software and data analysis." [wikipedia.org]
MATLAB and R Interfaces

- **Design goal:**
  - Seamless integration with languages

- **Key point:**
  - Languages are both matrix-oriented

- **KISS (Keep It Short and Simple)**
  - User builds constraint matrices using native language features
  - Gurobi function accepts matrices ($A, Q$) and vectors ($\text{obj, rhs, lb, ub, ...}$)
Gurobi MATLAB and R Interfaces

MATLAB:

model.A = sparse([1 1 0; 0 1 1])
model.obj = [1 2 3]
model.rhs = [1 1]
model.sense = ['<', '<']

result = gurobi(model)

disp(result.objval)

R:

model$A <- matrix(c(1,1,0,0,1,1), nrow=2)
model$obj <- c(1,2,3)
model$rhs <- c(1,1)
model$sense <- c('<=', '<=')

result <- gurobi(model)

print(result$objval)
INFEASIBLE MODELS
BARRIER HOMOGENEOUS ALGORITHM
Barrier Homogeneous Algorithm

- Standard barrier algorithm does not converge for infeasible/unbounded models

- Homogeneous algorithm [Xu, Hung, and Ye, 1996]
  - Converges:
    - To optimal solution for feasible & bounded models
    - To primal unboundedness proof for unbounded models
    - To dual unboundedness proof for infeasible models
  - Downside:
    - One extra linear solve per iteration – performance cost
    - Slightly less numerically robust
Barrier Homogeneous Algorithm

- Early termination
  - No need to wait for convergence
  - Stop as soon as you have an unboundedness proof

- Interesting observation
  - Standard barrier algorithm produces unboundedness proof quite reliably
    - Nearly as reliably as homogeneous algorithm

- Changes in Gurobi 5.0 barrier
  - Standard algorithm now detects infeasibility and unboundedness
  - Homogeneous algorithm for missed cases
Feasibility Relaxation

- For infeasible models, often useful to relax constraints

- We included a *feasopt* example with Gurobi 4.6
  - Users wanted this as a feature

- New *feasibility relaxation* routine *feasRelax()*
  - *feasRelax*(relaxobjtype, minrelax, relax_bounds, relax_constrs)
    - relaxobjtype: linear, quadratic, or cardinality penalty
    - minrelax: just find minimum relaxation, or optimize original objective while minimizing relaxation
  - Example:
    ```python
    model.feasRelax(GRB.FEASRELAX_LINEAR, true, true, true)
    model.optimize()
    ```
  - More complex version allows you to specify penalties for individual bounds/constraints
OTHER NEW FEATURES
Crossover Robustness

- Significant improvement in barrier crossover robustness
- Example: cont1_l (very difficult model)

  - Version 4.6:
    
    | Iteration | Objective  | Primal Inf. | Dual Inf. | Time        |
    |------------|------------|-------------|-----------|-------------|
    | 173731     | 8.7407451e-03 | 1.065363e+03 | 0.000000e+00 | 43650s     |
    | 173833     | 8.7463111e-03 | 1.054566e+03 | 0.000000e+00 | 43688s     |
    | 173935     | 8.7527310e-03 | 1.053641e+03 | 0.000000e+00 | 43729s     |
    | 174037     | 8.7574529e-03 | 1.059092e+03 | 0.000000e+00 | 43770s     |
    | 174139     | 8.7623337e-03 | 1.053525e+03 | 0.000000e+00 | 43809s     |
    | 174241     | 8.7673530e-03 | 1.063367e+03 | 0.000000e+00 | 43851s     |
    | 174343     | 8.7729649e-03 | 1.052670e+03 | 0.000000e+00 | 43890s     |
    
    - Version 5.0:
      
      | Iteration | Objective  | Primal Inf. | Dual Inf. | Time        |
      |------------|------------|-------------|-----------|-------------|
      | 3241       | 2.7703753e-03 | 0.000000e+00 | 0.000000e+00 | 7910s     |
      | 3241       | 2.7703753e-03 | 0.000000e+00 | 0.000000e+00 | 9040s     |

Solved in 3241 iterations and 9040.65 seconds
Optimal objective 2.770375295e-03
Concurrent Optimizer Control

- Concurrent allows you to apply multiple algorithms to an LP model
  - First one that finishes returns result
  - Useful for performance robustness
  - Algorithm choices hard-coded in Gurobi 4.6

- New detailed control:
  ```python
  env0 = model.getConcurrentEnv(0)
  env0.setParam("Method", 0)
  env1 = model.getConcurrentEnv(1)
  env1.setParam("Method", 1)
  model.optimize()
  ```
Floating License Accounting

- New 'gurobi_cl --tokens' command:

  Checking status of Gurobi token server 'server0'...

  Token server functioning normally.
  Maximum allowed uses: 10, current: 2
  Tokens currently in use...

<table>
<thead>
<tr>
<th>Client</th>
<th>HostName</th>
<th>Client IP Address</th>
<th>UserName</th>
</tr>
</thead>
<tbody>
<tr>
<td>mars</td>
<td></td>
<td>192.168.1.34</td>
<td>rothberg</td>
</tr>
<tr>
<td>i7</td>
<td></td>
<td>192.168.1.23</td>
<td>bixby</td>
</tr>
</tbody>
</table>
Other Features

- Retrieve by name:
  - New methods:
    - $v = \text{getVarByName}(\text{"varname"})$;
    - $c = \text{getConstrByName}(\text{"constrname"})$;
  - Implemented using a hash table
    - Constant time retrieval

- Simplex warm-start from primal/dual vectors
  - Previously only from a basis
QUADRATIC CONSTRAINTS
Problem Statement

- Quadratically Constrained Programming (QCP)
  
  Minimize: \( c'x + \frac{1}{2} x'Qx \)
  Subject To: \( Ax = b \)
  \( x'Q_j x + q_j'x \leq b_i \)
  \( l \leq x \leq u \)
  
  (MIQCP): Some \( x_j \) must be integral

- Different types of convex quadratic constraints:
  - \( x'Qx + q'x \leq b \), Q Positive Semi–Definite (PSD)
  - \( \sum x_j^2 \leq y^2, y \geq 0 \) (Second–Order Cone)
  - \( \sum x_j^2 \leq yz, y, z \geq 0 \) (Rotated Second–Order Cone)
Naming Confusion

- Lots of different names:
  - Second–Order Cone Programming (SOCP)
  - Quadratically Constrained Programming (QCP)
  - Quadratically Constrained Quadratic Programming (QCQP)

- Gurobi 5.0 accepts any combination of:
  - Second–order cone constraints
  - Rotated second–order cone constraints
  - General quadratic constraints with PSD Q

- Gurobi 5.0 is an RSOQCQP solver (?)
  - Concise labels don't work so well to capture QCP capabilities
APPLICATIONS
Applications

- Canonical example: portfolio optimization (Markowitz mean–variance model)
  - Return is linear; risk is quadratic
  - QP: minimize risk, lower bound on return
  - QCP: maximize return, upper bound on risk

- Surprisingly, canonical example rarely arises outside of Wall Street
  - Most applications now in other domains…
Applications

- Other domains:
  - Robust linear programming
  - Antenna array design
  - FIR filter design
  - Truss design
  - Sparse signal recovery (L1-norm methods)

- Can reformulate certain classes of convex non-linear programming problems as SOCP

- For more information:
  - Applications of second-order cone programming, Lobo, Vandenberghe, Boyd, and Lebret, 1998
  - CVX (www.cvxr.com)
INTERFACES
Interfaces

- Three different interfaces, depending on language capabilities:
  - Overloaded object-oriented (C++, .NET, Python):
    ```
    model.addQConstr(x*x + y*y <= 1, "qc0");
    ```
  - Object-oriented, no overloading (Java):
    ```
    GRBQuadExpr qexpr = new GRBQuadExpr();
    qexpr.addTerm(1.0, x, x);
    qexpr.addTerm(1.0, y, y);
    qexpr.addQConstr(qexpr, GRB.LESS_EQUAL, 1.0, "qc0");
    ```
  - C:
    ```
    qrow[0] = 0; qcol[0] = 0; qval[0] = 1.0;
    
    GRBaddqconstr(model, 0, NULL, NULL, 2, qrow, qcol, qval,
                 GRB.LESS_EQUAL, 1.0, "qc0");
    ```
Convex Quadratic Constraint Forms

- Gurobi accepts three different (convex) quadratic constraint forms:
  - **PSD Q:**
    - `model.addQConstr(x*x + x*y + y*y <= 2);`
  - **Second-Order Cone:**
    - `model.addQConstr(x0*x0 + x1*x1 <= y*y);`
  - **Rotated Second-Order Cone:**
    - `model.addQConstr(x0*x0 + x1*x1 <= y*z);`

- Gurobi automatically recognizes type
PERFORMANCE BENCHMARKS
Benchmark Overview

- Two kinds of benchmarks
  - Internal
    - Robustness testing
    - Compare version–over–version improvements
  - Competitive benchmarks
    - Extensive set of benchmarks maintained by Hans Mittelmann
      - [http://plato.la.asu.edu/bench.html](http://plato.la.asu.edu/bench.html)
      - Recently restructured, based upon MIPLIB 2010
LP Competitive Benchmarks
Mittelmann: LP Solve Times

- Gurobi 5.0 vs. Competition: Solve times (> 1.0 means Gurobi faster)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>CPLEX</th>
<th></th>
<th>XPRESS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P=1</td>
<td>P=4</td>
<td>P=1</td>
<td>P=4</td>
</tr>
<tr>
<td>LP Dual Simplex</td>
<td>1.5X</td>
<td>–</td>
<td>1.2X</td>
<td>–</td>
</tr>
<tr>
<td>LP Barrier (plus Crossover)</td>
<td>–</td>
<td>1.1X</td>
<td>–</td>
<td>1.3X</td>
</tr>
<tr>
<td>LP Concurrent</td>
<td>–</td>
<td>1.8X</td>
<td>–</td>
<td>3.6X</td>
</tr>
</tbody>
</table>

© 2012 Gurobi Optimization
MILP Competitive Benchmarks
Mittelmann (MIPLIB 2010): MILP Solve Times

- Gurobi 5.0 vs. Competition: Solve times (> 1.0 means Gurobi faster)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>CPLEX</th>
<th>XPRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P=1</td>
<td>P=4</td>
</tr>
<tr>
<td>Optimality</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feasibility</td>
<td>–</td>
<td>8.4X</td>
</tr>
<tr>
<td>Infeasibility</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
### Mittelmann: MILP Solvability

**Gurobi 5.0 vs. Competition: Hit time limit**

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Gurobi</th>
<th>CPLEX</th>
<th>XPRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIPLIB 2010</td>
<td>12</td>
<td>20</td>
<td>23</td>
</tr>
<tr>
<td>Feasibility</td>
<td>2</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Infeasibility</td>
<td>1*</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

*Out of memory*
QP Competitive Benchmarks
**Mittelmann: QP Solve Times**

- Gurobi 5.0 vs. Competition: Solve times (> 1.0 means Gurobi is faster)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>CPLEX</th>
<th>XPRESS</th>
<th>MOSEK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P=4</td>
<td>P=4</td>
<td>P=4</td>
</tr>
<tr>
<td>QP (continuous)</td>
<td>1.4X</td>
<td>1.0X</td>
<td>9.2X</td>
</tr>
<tr>
<td>MIQP</td>
<td>2.7X</td>
<td>1.7X</td>
<td>–</td>
</tr>
<tr>
<td>SOCP (continuous)</td>
<td>2.9X</td>
<td>–</td>
<td>1.4X</td>
</tr>
<tr>
<td>MIQCP</td>
<td>13.6X</td>
<td>4.1X</td>
<td>–</td>
</tr>
</tbody>
</table>
Gurobi gets Better as the Problems get Tougher
Mittelmann LP Benchmark

Dual Simplex Gurobi Vs. CPLEX

- Both solvers <100s: 0.97X
- Full test set: 1.49X
- At least one solver >100s: 2.78X

Gurobi 5.0 vs. CPLEX 12.4
Gurobi Solves 11 previously unsolved models

- Tested against the MIPLIB 2010 Challenge Set in January 2012
  - Models previously unsolved by any commercial or academic code
  - More at: http://miplib.zib.de

<table>
<thead>
<tr>
<th>b2c1s1</th>
<th>maxgasflow</th>
<th>opm2-z10-s2</th>
</tr>
</thead>
<tbody>
<tr>
<td>opm2-z11-s8</td>
<td>opm2-z12-s14</td>
<td>opm2-z12-s7</td>
</tr>
<tr>
<td>rmatr200-p10</td>
<td>satellites3-40-fs</td>
<td>satellites3-40</td>
</tr>
<tr>
<td>wnq-n100-mw99-14</td>
<td>transportmoment</td>
<td>(only 36s!)</td>
</tr>
</tbody>
</table>

- Application domains
  - Lot sizing, Gas network transport, Mining, Satellite scheduling, p-Median problem, Weighted n-Queens

Make sure to not just test us on easy models. Our best performance will likely be on your hardest models.
For More Information…

- All Gurobi documentation available at [www.gurobi.com](http://www.gurobi.com)
  - Quick Start Guide
  - Reference Manual
  - Example Tour

- Download the product at [www.gurobi.com](http://www.gurobi.com)
  - Free for academics
  - Free size-limited trial for commercial users

- Contact Gurobi:
  - info@gurobi.com
Thanks For Coming

Questions?
Type Them Into the 'Questions' Pane