What's New in Gurobi 7.0
What's New?

- New employees
- New features in 7.0
  - Major and minor
- Performance improvements
- New Gurobi Instant Cloud
The newest members of the Gurobi team...

Daniel Espinoza
Senior Developer

Frank Häger
Managing Director – EMEA Sales

Michel Jaczynski
Senior Architect

Melita Romero
Marketing Manager

Amal de Silva
Senior Support Engineer
Major New Features in Gurobi 7.0

- Python modeling enhancements
  - Much more direct mapping from mathematical notation to code
- General constraints
  - Shorthand for modeling common logical constraints
- Multi-objective optimization
  - Algorithms for trading off competing objectives
- Solution pool
  - Returns the $n$ best solutions instead of just the one best
- New default lazy update semantics
  - Simplifies modeling
Minor Enhancements

• API
  • Enhanced .NET interface
    • More use of .NET properties
    • Simplified parameter setting routines in other OO APIs
  • Parameter tuning tool enhancement
    • New tuning criterion
    • MIP start support
• Platforms
  • Support for Python 3.5 on Mac
    • And associated Anaconda Python package
• Other
  • New termination criteria:
    • *BestObjStop, BestBdStop*: stop when obj/bound reaches specified value
  • New cutting planes:
    • *InfProofCuts, StrongCGCuts*
  • Added control over heuristics:
    • *DegenMoves*
Python Modeling Enhancements
Python Modeling Enhancements

• Significant improvement in the expressiveness of our Python API

• New facilities
  • Model.addVars: add a set of variables (indexed over given indices)
  • Model.addConstrs: add a set of constraints (using a Python generator expression)
  • tupledict: a collection of variables that makes it easy to build linear expressions on subsets

• Python models become more concise and easier to read
Python Modeling Enhancements – addVars

• Old:

```python
transport = {}
for w in warehouses:
    for p in plants:
        transport[w, p] = model.addVar(obj=transportCosts[w, p])
```

• New:

```python
transport = model.addVars(warehouses, plants, obj=transportCosts)
```

• First arguments provide indices
  • Two dimensions in this example
    • First iterates over members of list of warehouses, second over list of plants
    • Arbitrary number of dimensions, each indexed over members of a list or an integer
Python Modeling Enhancements – tupledict

• New *tupledict* class, returned by `model.addVars()`

```python
transport = model.addVars(warehouses, plants, obj=transportCosts)
```

• Makes it easier to build linear expressions on sets of variables
  • `transport.sum()`: linear expression that captures the sum of the variables in the *tupledict*
  • `transport.sum(w, '*')`: sum over a subset of the variables
  • `transport.prod(coeffs)`: weighted sum
  • `transport.prod(coeffs, '*', p)`: weighted sum

• Useful in many different situations
Python Modeling Enhancements – addConstrs

• Old:

```python
for w in warehouses:
    model.addConstr(sum(transport[w,p] for p in plants) == demand[w])
```

• New:

```python
model.addConstrs(transport.sum(w, '*') == demand[w] for w in warehouses)
```
Putting It Together

• Old:

```python
transport = {}
for w in warehouses:
    for p in plants:
        transport[w,p] = model.addVar(obj=transportCosts[w,p])

for w in warehouses:
    model.addConstr(sum(transport[w,p] for p in plants) == demand[w])
```

• New:

```python
transport = model.addVars(warehouses, plants, obj=transportCosts)
model.addConstrs(transport.sum(w, '*') == demand[w] for w in warehouses)
```
The Power of Python

- Lives within a powerful and versatile programming language
- Thousands of modules available for downloading
- Example:
  - Add 6 lines to our TSP example (using bokeh package)…

```python
ptseq = [points[k] for k in tour+[tour[0]]]
x, y = zip(*ptseq)
p = figure(title="TSP tour", x_range=[0,100], y_range=[0,100])
p.circle(x, y, size=8)
p.line(x, y)
show(p)
```
General Constraints
General Constraints

• Constraint types:
  • In 6.5:
    • Linear constraints
    • Quadratic constraints
    • SOS constraints
  • In 7.0: new general constraints
    • Min, Max, Abs, And, Or, Indicator

• General constraints easier to read, write, and maintain
  • Example: \( r = \max(x_1, x_2, x_3) \)
  • Linearization:
    • \( r = x_1 + s_1; r = x_2 + s_2; r = x_3 + s_3 \) (s_j non-negative)
    • \( z_1 + z_2 + z_3 = 1 \) (z_j binary)
    • \( \text{SOS1}(s_1,z_1), \text{SOS1}(s_2,z_2), \text{SOS1}(s_3,z_3) \)
  • Which would you rather read/write/maintain?
General Constraints

• Examples:
  • Model.addGenConstrAnd(x0, [x1,x2,x3,x4])
    • Sets binary variable $x_0 = x_1 \land x_2 \land x_3 \land x_4$ (logical AND)
  • Model.addGenConstrIndicator(x0, 1, 2*x1 + 3*x2 + x3 + 2*x4 <= 1)
    • If $x_0 = 1$, then $2x_1 + 3x_2 + x_3 + 2x_4 \leq 1$ must be satisfied

• Available from all of our programming language APIs
• Mostly a modeling convenience feature, but…
  • Presolve can sometimes create a tighter formulation
Multi-Objective Optimization
Multi-Objective Optimization

• User can specify multiple objective functions

• Two options for combining them:
  • Blended
    • User provides weights
    • Weights used to combine objectives
  • Hierarchical
    • User provides priorities
    • Optimize highest priority objective first
    • Then optimize next highest, but without degrading highest priority objective (too much)
    • Repeat for each objective, in order of decreasing priority
  • Can combine the two

• Two parameters to control the optimization
  • MultiObjPre: presolve level on the whole multi-objective model
  • MultiObjMethod: select barrier, dual and primal to optimize for subsequent objectives, only for continuous models
Solution Pool
Solution Pool

- In 6.5, we find *an* optimal solution
- In 7.0, you can ask for:
  - The $k$ best solutions
  - $k$ solutions within a specified gap from optimal
- New parameters
  - **PoolSolutions**: number of solutions to keep
    - Default is now 10
    - Previously kept as many as we found
  - **PoolSearchMode**: strategy for finding solutions
    - 0: default (one optimal solution)
    - 1: more solutions (but not systematic)
    - 2: $k$ best solutions (systematic)
  - **PoolGap**: maximum gap between best and worst solution
    - Useful for limiting search time
    - Why search for solutions with 50% gap if you don't want them anyway?
Solution Pool

• Runtime penalty usually small, but can be substantial
  • Disables dual reductions in presolve
    • Dual reduction: can discard a solution when you can prove that an equivalent or better solution always exists
  • Dual reductions sometimes quite powerful
Lazy Updates
New Default Lazy Update Semantics

• In Gurobi 6.5 default settings, model changes are put in a queue
  • Must call update() to apply queued changes
  • Plusses and minuses:
    • Update is expensive – good to have control over when it happens
    • Adding appropriate update calls can be tedious

• Parameter **UpdateMode** set to 1 by default in Gurobi 7.0
  • Model changes are still put in a queue, but…
  • Can refer to newly created variables and constraints without first calling update()
New Default Lazy Update Semantics

• Typical usage pattern in Gurobi 6.5:
  • Add new variables to model
  • Call update() to clear queue and actually add variables
  • Add constraints on those variables

• New update semantics avoid one update() call

• Less common usage pattern in Gurobi 6.5:
  • Repeat:
    • Add a few variables to model
    • Call update()
    • Add a few constraints on those variables
  • Frequent update() calls can be expensive
    • No longer needed
New Default Lazy Update Semantics

- Gurobi examples have been modified for new update semantics
  - Every single `update()` call went away
- Uncommon usage pattern:
  - "Here are a bunch of variables and constraints"
  - "Can you remind me of what I just added?"
  - This still requires an `update()` call with new semantics
Other Enhancements
Improved .NET Property Support

• Better support for properties in .NET
  • Old: `model.GetEnv().Set(GRB.IntParam.MIPFocus, 2)`
  • New: `model.Parameters.MIPFocus = 2`

• Also leads to better *ToolTips* support in Visual Studio
Simplified Parameter Setting

• Parameters also easier to set in other OO APIs
  • Set them on the model rather than on the environment
  • Java example
    • Old: `model.getEnv().set(GRB.IntParam.MIPFocus, 2)`
    • New: `model.set(GRB.IntParam.MIPFocus, 2)`
Parameter Tuning Tool Enhancement

• In 6.5, tuning tries to minimize time to a proven optimal solution
  • Failing that, secondary criterion objective is to minimize the optimality gap

• In 7.0, new **TuneCriterion** parameter chooses secondary objective:
  • -1: auto
  • 0: minimize runtime
  • 1: minimize the optimality gap
  • 2: find the best feasible solution
  • 3: find the best objective bound

• Primary goal is always to minimize time to find a proven optimal solution

• Also added support for MIP starts
Platforms

• Added support for Python 3.5 on Mac
  • Plus a Mac Anaconda 3.5 package
• Added support for R 3.3
Performance Improvements
MILP Improvements

- Branching
  - Improved shadow costs, especially for SOS models
  - Pseudo cost improvements
- Cuts
  - New strong-CG cuts
  - New infeasibility proof cuts
  - Better sub-additive functions for Gomory cuts
  - Improvements on variety of cuts, aggregation, separation, lifting, numerics, etc
- Conflict analysis
- Presolve
  - Improvements using GUB and VUB for reductions
  - Improved and more stable aggregation
  - More coefficient strengthening
  - More efficient implementations of some reductions
  - Improved node presolve
- Symmetry
  - Improve speed and increase aggressiveness
- Heuristics
  - New and improved heuristics
We refer to the literature for details on the implementation and performance of these improvements. A comprehensive list of the most significant changes follows:

- We have included a new option to improve the performance of the primal simplex method when solving dense problems.
- Our default, non-deterministic concurrent optimization has been significantly sped up. A new parameter allows you to control the degree of concurrency.
- We have improved the performance of the dual simplex method, especially for problems with a large number of variables.
- The barrier method has been optimized to handle problems with a large number of constraints.
- We have implemented a new algorithm for solving problems with a large number of variables and constraints.
SOCP/QCP Improvement

• Barrier centering improvements
• Better dense column handling
MIQCP/MISOCP Improvements

- Stronger cuts on Q constraints using integrality
- Improved and extended presolve and node presolve reductions
- SOCP/QCP improvements
- General MIP improvements
MIQP Improvements

• Mainly due to general MIP improvements
Gurobi 7.0 Performance Benchmarks
Two Kinds of Benchmarks

• Internal benchmarks
  • Most important: compare Gurobi version-over-version
  • Based on internal library of 4131 models

• External, competitive benchmarks
  • Conducted by Hans Mittelmann, Arizona State University
    • [http://plato.asu.edu/bench.html](http://plato.asu.edu/bench.html)
  • For MIP largely based upon MIPLIB 2010
Internal Benchmarks
Gurobi MIP Library

(4131 models)
### Performance Improvements in Gurobi 7.0

<table>
<thead>
<tr>
<th>Problem Class</th>
<th>&gt;1s</th>
<th></th>
<th>&gt;100s</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#</td>
<td>Wins</td>
<td>Losses</td>
<td>Speedup</td>
</tr>
<tr>
<td>LP: conc.</td>
<td>387</td>
<td>98</td>
<td>50</td>
<td>1.10x</td>
</tr>
<tr>
<td>QCP</td>
<td>97</td>
<td>87</td>
<td>3</td>
<td>1.46x</td>
</tr>
<tr>
<td>MILP</td>
<td>2048</td>
<td>976</td>
<td>560</td>
<td>1.22x</td>
</tr>
<tr>
<td>MIQP</td>
<td>91</td>
<td>32</td>
<td>25</td>
<td>1.09x</td>
</tr>
<tr>
<td>MIQCP</td>
<td>200</td>
<td>110</td>
<td>47</td>
<td>1.48x</td>
</tr>
</tbody>
</table>

- Gurobi 6.5 vs. 7.0: > 1.00x means that Gurobi 7.0 is faster than Gurobi 6.5
Continual Performance Improvements

Nearly a 2x average improvement per major release

Graph showing cumulative speedup over Gurobı version pairs.

Time limit: 10000 sec.
- Intel Xeon CPU E3-1240 v3 @ 3.40GHz
- 4 cores, 8 hyper-threads
- 32 GB RAM

Test set has 3740 models:
- 229 discarded due to inconsistent answers
- 908 discarded that none of the versions can solve
- speed-up measured on >100s bracket: 1227 models

~43x improvement (seven years)
Continual Performance Improvements

Time limit: 10000 sec.
Intel Xeon CPU E3-1240 v3 @ 3.40GHz
4 cores, 8 hyper-threads
32 GB RAM

Test set has 3740 models:
- 229 discarded due to inconsistent answers
- 908 discarded that none of the versions can solve
- speed-up measured on >100s bracket: 1227 models

Number of unsolved models

<table>
<thead>
<tr>
<th>Version</th>
<th>Number of Unsolved Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1.1</td>
<td>545</td>
</tr>
<tr>
<td>v2.0</td>
<td>459</td>
</tr>
<tr>
<td>v3.0</td>
<td>350</td>
</tr>
<tr>
<td>v4.0</td>
<td>323</td>
</tr>
<tr>
<td>v5.0</td>
<td>259</td>
</tr>
<tr>
<td>v6.0</td>
<td>164</td>
</tr>
<tr>
<td>v7.0</td>
<td>53</td>
</tr>
</tbody>
</table>
External Benchmarks

Hans Mittelmann: http://plato.asu.edu/bench.html
Gurobi 7.0 vs. Competition: Solve times

- > 1.0 means Gurobi faster

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>#</th>
<th>CPLEX 12.7.0</th>
<th></th>
<th>XPRESS 8.0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>P=1</td>
<td>P=4</td>
<td>P=12</td>
<td>P=48</td>
</tr>
<tr>
<td>Optimality</td>
<td>86</td>
<td>1.26x</td>
<td>1.38x</td>
<td>1.07x</td>
<td>-</td>
</tr>
<tr>
<td>Feasibility</td>
<td>33</td>
<td>-</td>
<td>1.41x</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Infeasibility</td>
<td>19</td>
<td>-</td>
<td>1.00x</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>&quot;Solvable&quot; Optim.</td>
<td>213</td>
<td>-</td>
<td>-</td>
<td>1.13x</td>
<td>1.19x</td>
</tr>
</tbody>
</table>

- Number of solved models in “solvable set”
  - P=12: Gurobi 207, Cplex 201, Xpress 178
  - P=48: Gurobi 210, Cplex 206, Xpress 181
- Complete test data available here (data from November 14, 2016):
  - [http://plato.asu.edu/ftp/feas_bench.html](http://plato.asu.edu/ftp/feas_bench.html): "Feasibility", time limit 3600 sec., time to first solution
LP Solve Times

- Gurobi 7.0 vs. Competition: Solve times
  - > 1.0 means Gurobi faster

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>CPLEX 12.7.0</th>
<th>XPRESS 8.0</th>
<th>Mosek 8.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplex</td>
<td>1.88x</td>
<td>1.01x</td>
<td>2.52x</td>
</tr>
<tr>
<td>Barrier</td>
<td>1.54x</td>
<td>0.99x</td>
<td>1.90x</td>
</tr>
<tr>
<td>Concurrent</td>
<td>1.92x</td>
<td>1.00x</td>
<td>-</td>
</tr>
</tbody>
</table>

- Complete test data available here (data from November 14, 2016):
Gurobi 7.0 vs. Competition: Solve times

- > 1.0 means Gurobi faster

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>CPLEX 12.7.0</th>
<th>XPRESS 8.0</th>
<th>Mosek 8.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOCP</td>
<td>3.25x</td>
<td>1.29x</td>
<td>0.79x</td>
</tr>
<tr>
<td>MISOCPP</td>
<td>3.40x</td>
<td>1.37x</td>
<td>9.66x</td>
</tr>
<tr>
<td>MIQP</td>
<td>1.24x</td>
<td>1.46x</td>
<td>-</td>
</tr>
<tr>
<td>MIQCP</td>
<td>1.41x</td>
<td>1.15x</td>
<td>-</td>
</tr>
</tbody>
</table>

- Complete test data available here (data from November 14, 2016):
Gurobi Instant Cloud
Gurobi Cloud Offering

• Gurobi has had a Cloud offering for over 5 years
• Important features:
  • Many licensing options
    • Install perpetual licenses on cloud machines or pay by the hour
  • Available on Amazon EC2
    • EC2 has ~85% cloud market share
  • Use any of our API's (C, C++, Java, .NET, Python, MATLAB, R)
    • Full API support
    • No code changes required to use cloud
  • Security
    • All communication encrypted with 256-bit AES encryption
New Instant Cloud

• Completely revamped Gurobi Instant Cloud
• New website
• New client launching facilities
  • Launch cloud machine from user program
    • Using gurobi.lic file:
      
      CLOUDACCESSID=79344c54-48af-4d37-8526-cb8e9b2a1743
      CLOUDKEY=p3XNjBlP5piz5huuzisS6w
  
  • Using programming language APIs:
    
    env = Env.CloudEnv("79344c54-48af-4d37-8526-cb8e9b2a1743",
    "p3XNjBlP5piz5huuzisS6w")

  • Launch through REST API

• New pool facility
  • Allow multiple client programs to share…
    • Configuration information
    • A pool of servers
New Instant Cloud – New Website
# New Instant Cloud – Pools

<table>
<thead>
<tr>
<th>ID</th>
<th>Compute Servers</th>
<th>Machine</th>
<th>Region</th>
<th>License</th>
<th>Rate Plan</th>
<th>Distributed Workers</th>
</tr>
</thead>
<tbody>
<tr>
<td>999999-default</td>
<td>1</td>
<td>c4.4xlarge</td>
<td>us-east-1</td>
<td>999999</td>
<td>Gold</td>
<td>0</td>
</tr>
<tr>
<td>999999-distributed</td>
<td>1</td>
<td>c4.2xlarge</td>
<td>us-east-1</td>
<td>999999</td>
<td>Gold</td>
<td>4</td>
</tr>
<tr>
<td>999999-large</td>
<td>1</td>
<td>r3.8xlarge</td>
<td>us-east-1</td>
<td>999999</td>
<td>Gold</td>
<td>0</td>
</tr>
<tr>
<td>999999-production</td>
<td>5</td>
<td>c4.8xlarge</td>
<td>us-east-1</td>
<td>999999</td>
<td>Gold</td>
<td>0</td>
</tr>
</tbody>
</table>

Showing 1 to 4 of 4 pools
Next Steps

- Slides from this webinar will be made available
  - Attendees will receive an email

- 7.0 upgrades for existing Gurobi users
  - Visit the What’s New in Gurobi page and follow instructions in the bottom:
    http://www.gurobi.com/new

- For additional questions, or to request a free trial, contact us at info@gurobi.com