What's New in Gurobi 9.0

Webinar

GUROBI OPTIMIZATION

The World's Fastest Solver

Tobias Achterberg
17/18 December 2019

Highlights

Performance

- New cuts, new solution improvement heuristics, AVX-512 support, etc.

Major features

- Non-convex MIQCP (bilinear)
- Piecewise-linear (PWL) constraints
- Function constraints with automatic PWL translation
- MIP scenario analysis (what if, MIP sensitivity)
- New matrix friendly API for gurobipy (support for SciPy sparse matrices)
- New Compute Server capabilities
- Batch optimization in Compute Server

Other improvements

- Intermediate solution files
- Support for lazy constraint callback for Compute Server
- Indices for variables and constraints in OO APIs
- Model attribute file
- Interactive shell updated from Python 2.7 to 3.7

Highlights

Performance

- New cuts, new solution improvement heuristics, AVX-512 support, etc.

Major features

- Non-convex MIQCP (bilinear)
- Piecewise-linear (PWL) constraints
- Function constraints with automatic PWL translation
- MIP scenario analysis (what if, MIP sensitivity)
- New matrix friendly API for gurobipy (support for SciPy sparse matrices)
- New Compute Server capabilities
- Batch optimization in Compute Server

Other improvements

- Intermediate solution files
- Support for lazy constraint callback for Compute Server
- Indices for variables and constraints in OO APIs
- Model attribute file
- Interactive shell updated from Python 2.7 to 3.7

LP Performance

Default: 7% faster

- Concurrent LP on 4 threads

Simplex: 5% faster (primal), no performance change (dual)

- Improved linear system solving (Ftran/Btran)
- Improved numerics for warm-start and corner cases

Barrier: 7% faster

- Crossover: improved numerics
- Support for AVX-512, 40% boost on models with expensive factorization
  - yields another 4% overall improvement on AVX-512 systems

MIP Performance

MILP: 18% faster (26% faster on models that take >100 seconds)

- Cuts:
  - new RLT cuts
  - new BQP cuts
  - new RelaxLift cuts
  - second type of "SubMIP" cuts
  - use LP to find another aggregation for MIR cut separator
  - randomize aggregation order in MIR cut separator
  - try more scaling factors in MIR cut separator
  - more aggressive cover cut separation
  - occasionally separate "close cuts"
  - tuned cut loop abort criteria for main and parallel cut loops
  - improved dual bound updates from parallel root cut loops
  - improved cut selection
  - improved performance of zero-half and mod-k cut separators
  - limit effort in GUB cover and network cut separation procedures
  - limit effort in some very expensive cut separation procedure
  - fixed a performance issue in symmetry cuts
- Heuristics:
  - new solution improvement heuristic
  - new "lurking bounds" heuristic
  - extended some heuristics to work for models with SOS constraints
- Presolve:
  - implied product detection
  - detect implicit piece-wise linear functions
  - sparsify objective function
  - substitute sub-expressions in presolve to sparsify constraints
  - better work limits in constraint sparsification
  - improved parallel column/row presolve reduction
  - activate SOS2 to big-M translation in some cases
- Additional improvements:
  - improved disconnected components
  - extended disconnected components detection to work for models with SOS constraints
  - reduced wait time in parallel synchronization by more flexible work load distribution
  - propagate objective function in node probing

New Solution Improvement Heuristics

Heuristics have improved significantly

Still opportunities to do better

- Particularly when the relaxation isn’t a good guide

Better improvement heuristics

- ImproveStartGap, ImproveStartNodes, ImproveStartTime
- Comparison against old version on a set of difficult models
    - New scheme finds better solution on 83% of models

Convex MIQP and MIQCP Performance

MIQP: 24% faster

- Most MILP improvements apply
- Conversion of variables with concave objective into binaries for pure "box QPs"

MIQCP: 6% faster

- Most MILP improvements apply
- Improved presolve and node presolve propagation for quadratic constraints
- Propagate ≥ direction of quadratic equality constraints in presolve
- Extended disconnected components detection to work for MIQCPs
- Extended a number of primal heuristics to work for MIQCPs
    - including the zero-objective heuristic
- Improved fix-and-dive heuristics for MIQCPs

Major Features

Highlights

Performance

- New cuts, new solution improvement heuristics, AVX-512 support, etc.

Major features

- Non-convex MIQCP (bilinear)
- Piecewise-linear (PWL) constraints
- Function constraints with automatic PWL translation

non-linear optimization

- MIP scenario analysis (what if, MIP sensitivity)
- New matrix friendly API for gurobipy (support for SciPy sparse matrices)
- New Compute Server capabilities
- Batch optimization in Compute Server

Other improvements

- Intermediate solution files
- Support for lazy constraint callback for Compute Server
- Indices for variables and constraints in OO APIs
- Model attribute file
- Interactive shell updated from Python 2.7 to 3.7

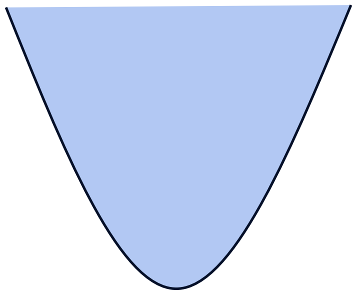
Non-Convex QP, QCP, MIQP, and MIQCP

A lot of applications

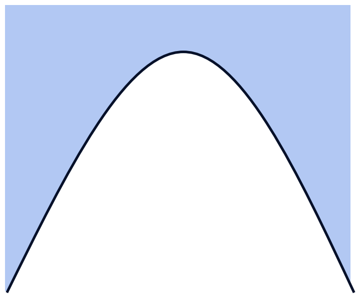
- Pooling problem (blending problem is LP, pooling introduces intermediate pools $\rightarrow$ bilinear)
- Petrochemical industry (oil refinery: constraints on ratio of components in tanks)
- Wastewater treatment
- Emissions regulation
- Agricultural / food industry (blending based on pre-mix products)
- Etc.

Non-Convex QP, QCP, MIQP, and MIQCP

Prior Gurobi versions: remaining Q constraints and objective after presolve needed to be convex



$x^T Q x \leq b$



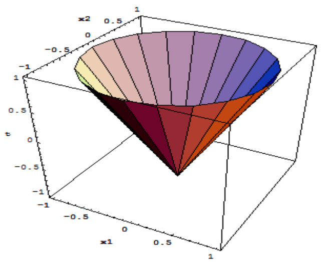
convex
$-y + x^2 \leq 0$

non-convex
$-y - x^2 \leq 0$

If Q is positive semi-definite (PSD) then $x^T Q x \leq b$ is convex

- Q is PSD if and only if $x^T Q x \geq 0$ for all x

But $x^T Q x \leq b$ can also be convex in certain other cases, e.g., second order cones (SOCs)



SOC: $x_1^2 + \cdots + x_n^2 - z^2 \leq 0$

$x^2 + y^2 - z^2 \leq 0, z \geq 0$: at level z, (x, y) is a disc with radius z

Non-Convex QP, QCP, MIQP, and MIQCP

What about non-convex quadratic constraints or objectives?

- Presolve might be able to convexify or to linearize them
- If this fails: `GRB_ERROR_Q_NOT_PSD` or `GRB_ERROR_QCP_EQUALITY_CONSTRAINT`

Gurobi 9.0 can solve any quadratic problem to global optimality

- No longer returns errors, just solves it (if the "NonConvex" parameter is set to 2)
- Automatically transforms arbitrary non-convex quadratic constraints into bilinear constraints

$$3x_1^2 - 7x_1x_2 + 2x_1x_3 - x_2^2 + 3x_2x_3 - 5x_3^2 = 12$$ (non-convex Q constraint)

$$p_{11} \coloneqq x_1^2, p_{12} \coloneqq x_1x_2, p_{13} \coloneqq x_1x_3, p_{22} \coloneqq x_2^2, p_{23} \coloneqq x_2x_3, p_{33} \coloneqq x_3^2$$ (6 bilinear constraints)

$$3p_{11} - 7p_{12} + 2p_{13} - p_{22} + 3p_{23} - 5p_{33} = 12$$ (linear constraint)

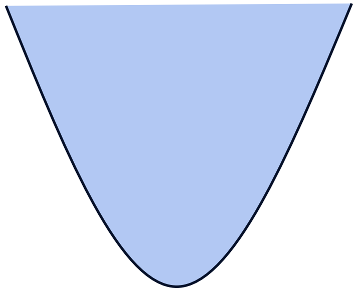
- Solver engine is able to deal with bilinear constraints
  - cutting planes
  - spatial branching

Non-Convex QP, QCP, MIQP, and MIQCP

Algorithmic treatment of bilinear constraints

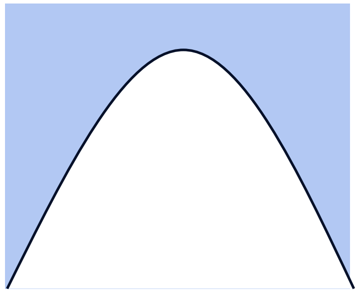
- General form: $a^T z + dxy \leqq b$ (linear sum plus single product term, inequality or equation)

Consider square case ($x = y$):



convex

$-z + x^2 \leq 0$



non-convex

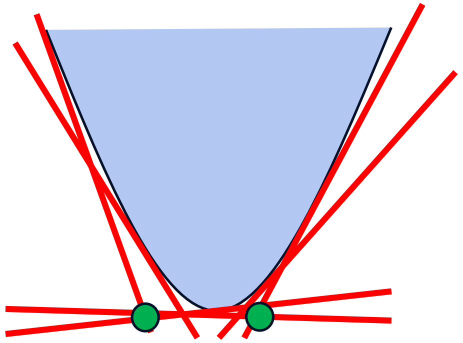
$-z - x^2 \leq 0$

Non-Convex QP, QCP, MIQP, and MIQCP

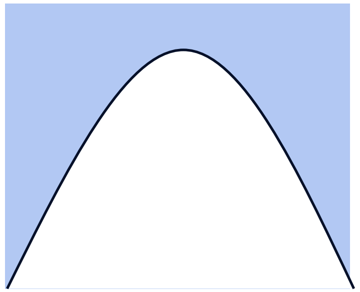
Algorithmic treatment of bilinear constraints

- General form: $a^T z + dxy \leqq b$ (linear sum plus single product term, inequality or equation)

Consider square case ($x = y$):



convex
$-z + x^2 \leq 0$



non-convex
$-z - x^2 \leq 0$

easy: add tangent cuts

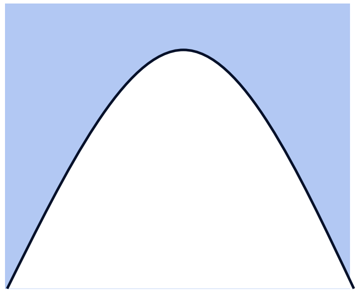
Non-Convex QP, QCP, MIQP, and MIQCP

Algorithmic treatment of bilinear constraints

- General form: $a^T z + dxy \leqq b$ (linear sum plus single product term, inequality or equation)

Consider square case ($x = y$):

non-convex
$-z - x^2 \leq 0$

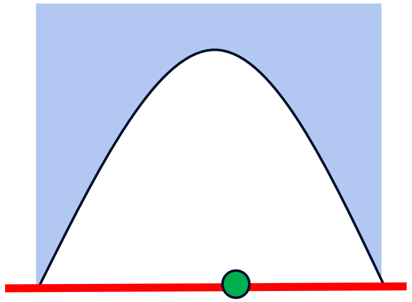


Non-Convex QP, QCP, MIQP, and MIQCP

Algorithmic treatment of bilinear constraints

- General form: $a^T z + dxy \leqq b$ (linear sum plus single product term, inequality or equation)

Consider square case ($x = y$):

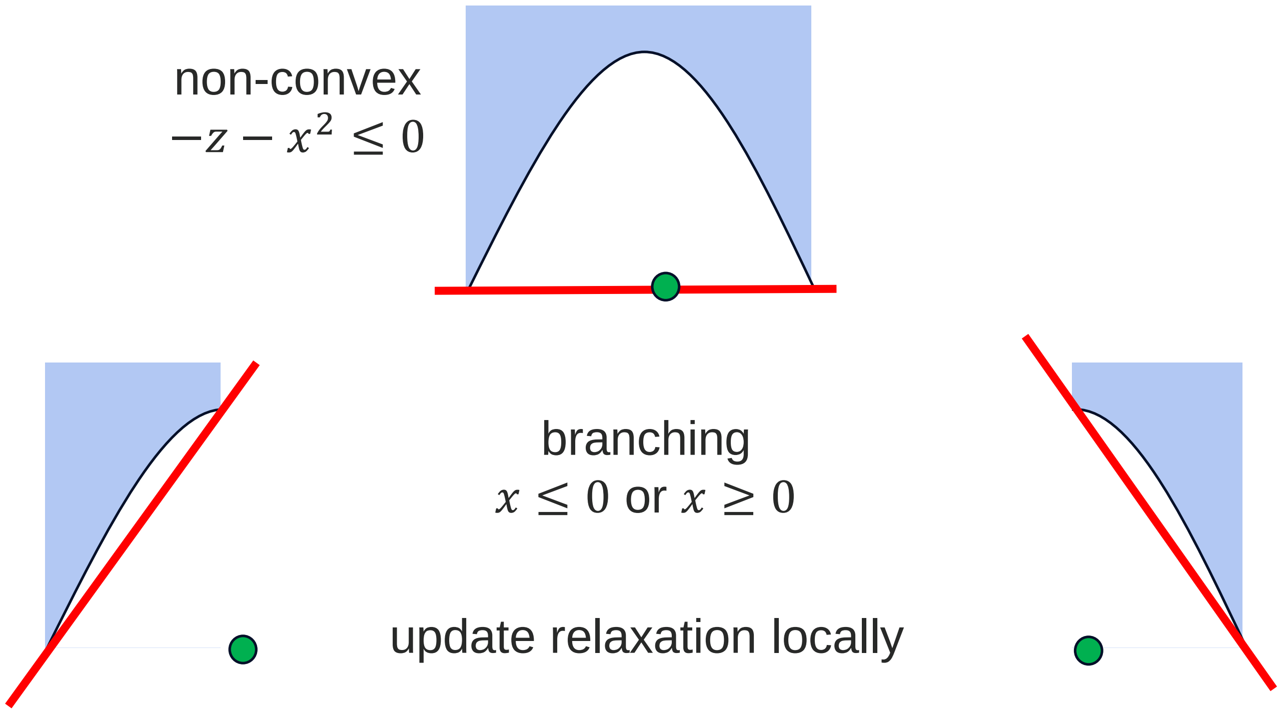


Non-Convex QP, QCP, MIQP, and MIQCP

Algorithmic treatment of bilinear constraints

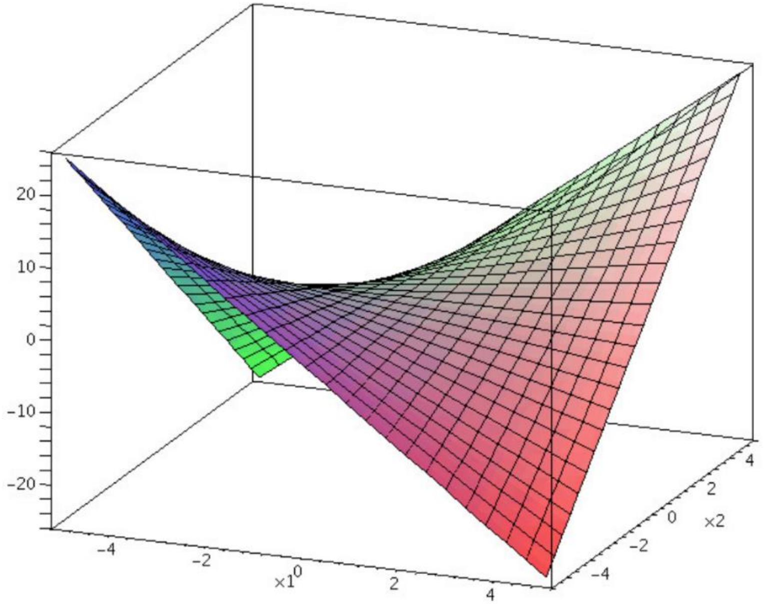
- General form: $a^T z + dxy \leqq b$ (linear sum plus single product term, inequality or equation)

Consider square case ($x = y$):



Non-Convex QP, QCP, MIQP, and MIQCP

Mixed product case: $-z + xy = 0$

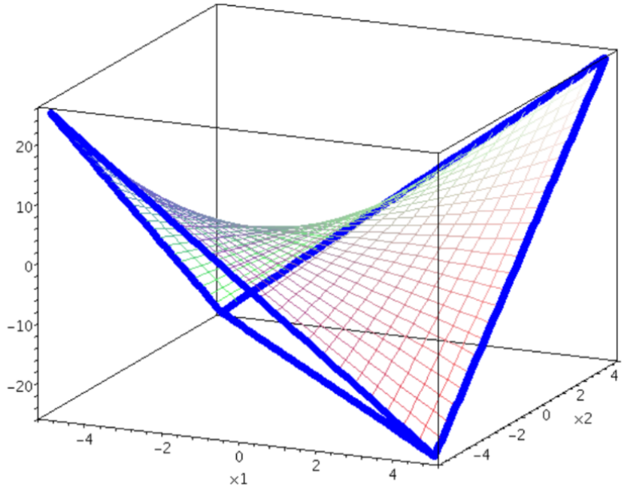


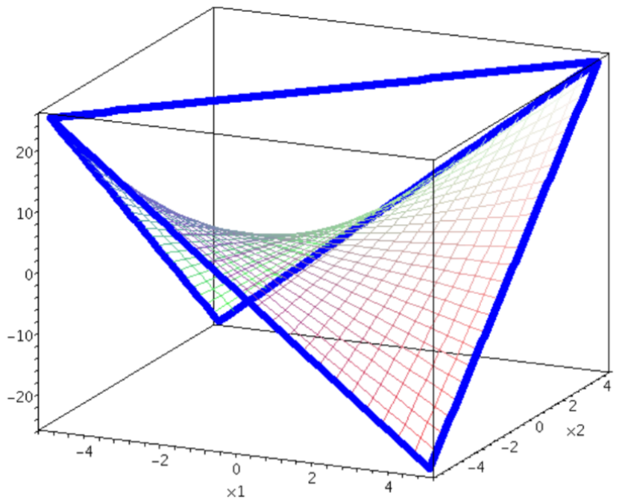
McCormick lower and upper envelopes:

$$-z + l_x y + l_y x \leq l_x l_y$$
$$-z + u_x y + u_y x \leq u_x u_y$$

$$-z + u_x y + l_y x \geq u_x l_y$$
$$-z + l_x y + u_y x \geq l_x u_y$$

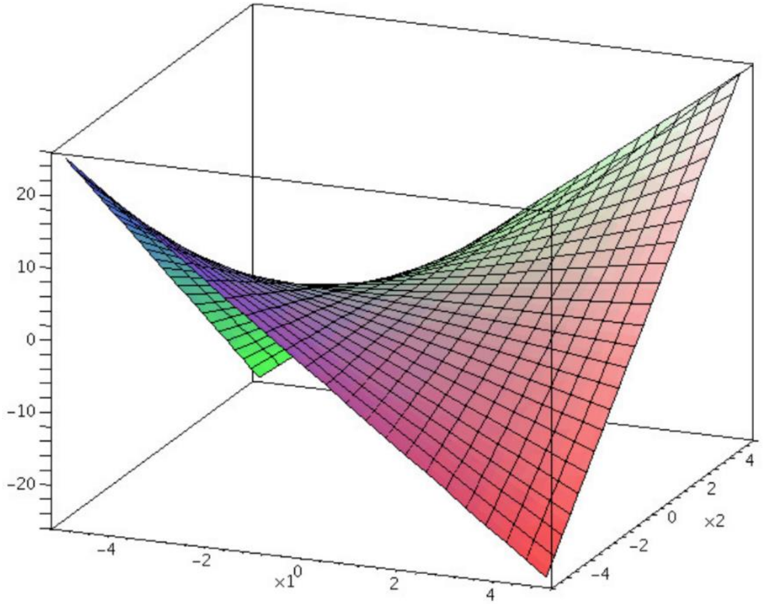
pictures from Costa and Liberti: "Relaxations of multilinear convex envelopes: dual is better than primal"





Non-Convex QP, QCP, MIQP, and MIQCP

Mixed product case: $-z + xy = 0$



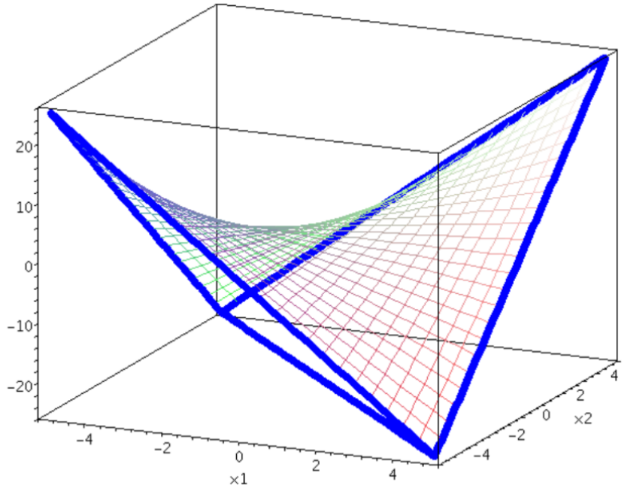
McCormick lower and upper envelopes:

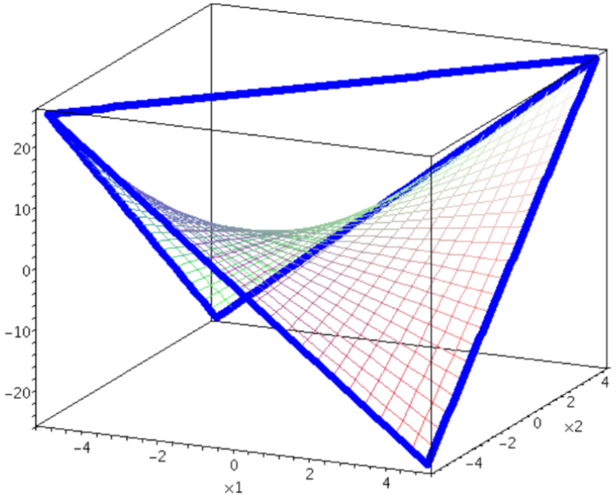
pictures from Costa and Liberti: "Relaxations of multilinear convex envelopes: dual is better than primal"

$$-z + l_x y + l_y x \leq l_x l_y$$
$$-z + u_x y + u_y x \leq u_x u_y$$

$$-z + u_x y + l_y x \geq u_x l_y$$
$$-z + l_x y + u_y x \geq l_x u_y$$

coefficients depend on local bounds





Non-Convex QP, QCP, MIQP, and MIQCP

Algorithmic ingredients

- Presolve translation of general non-convex Q constraints into bilinear constraints
- McCormick relaxation
- Spatial branching (branching on continuous variables)
- "Adaptive constraints"
  - automatically modify coefficients in McCormick relaxation after local bound change
  - alternative to adding more and more locally valid cuts
- Cutting planes
  - RLT cuts (RLT = Reformulation Linearization Technique)
  - BQP cuts (BQP = Boolean Quadric Polytope)
  - not yet in Gurobi: SDP cuts (SDP = Semi-Definite Program)
  - ...

Can also use techniques for MILP

- Detection of linearization of products with a binary variable
- Results in performance improvement on MILP: RLT cuts and BQP cuts

Non-Convex QP, QCP, MIQP, and MIQCP

Preliminary results vs. existing non-convex MIQCP solvers

- QPLIB benchmarks of Prof. Hans Mittelmann (Arizona State Univ.)
  - http://plato.asu.edu/bench.html
- Gurobi is faster and solves more models within time limit
- But: other solvers usually solve general MINLP, not specialized to non-convex MIQCP

Test set		Mosek	Knitro	Bonmin	CBC	Couenne	OcterAct	Baron	SCIP	F-SCIP	Antigone	Minotaur	Gurobi
non-convex binary	ratio						64.7x	16.2x	64.8x	46.7x	63.0x	85.8x	**1.0x**
	solved (80)						17	41	19	24	23	7	**80**
non-convex discrete	ratio					25.5x	30.6x	11.9x	18.3x	5.1x	12.6x	28.8x	**1.0x**
	solved (88)					8	1	24	15	41	29	4	**66**
non-convex continuous	ratio					5.1x	4.9x	2.2x	4.2x	2.7x	1.6x	5.4x	**1.0x**
	solved (49)					8	8	22	7	14	**29**	6	27
convex discrete	ratio	7.0x	11.4x	10.8x	31.1x			7.3x	13.5x	20.3x	28.6x	17.9x	**1.0x**
	solved (31)	12	9	10	2			11	11	8	2	11	**21**

results from December 16, 2019

Highlights

GUROBI OPTIMIZATION

Performance

- New cuts, new solution improvement heuristics, AVX-512 support, etc.

Major features

- Non-convex MIQCP (bilinear)
- Piecewise-linear (PWL) constraints
- Function constraints with automatic PWL translation

non-linear optimization

- MIP scenario analysis (what if, MIP sensitivity)
- New matrix friendly API for gurobipy (support for SciPy sparse matrices)
- New Compute Server capabilities
- Batch optimization in Compute Server

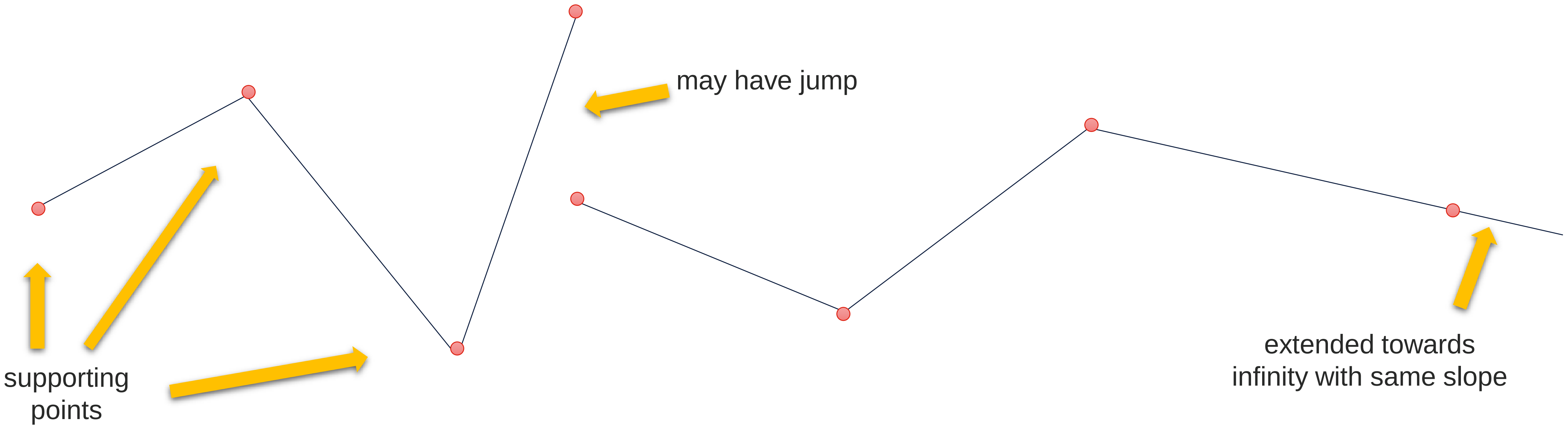
Other improvements

- Intermediate solution files
- Support for lazy constraint callback for Compute Server
- Indices for variables and constraints in OO APIs
- Model attribute file
- Interactive shell updated from Python 2.7 to 3.7

Piecewise-Linear (PWL) Constraints

A new type of general constraint

Users specify the supporting points as a list of (x,y) tuples



$$y = PWL\left(x; (x_1, y_1), \dots, (x_n, y_n)\right)$$

Piecewise-Linear (PWL) Constraints

A new type of general constraint

Users specify the supporting points as a list of (x,y) tuples

Example

- $y = e^x, \; 0 \leq x \leq 1$
- To approximate this as PWL constraint with 100 pieces, generate 101 points (xpts, ypts):

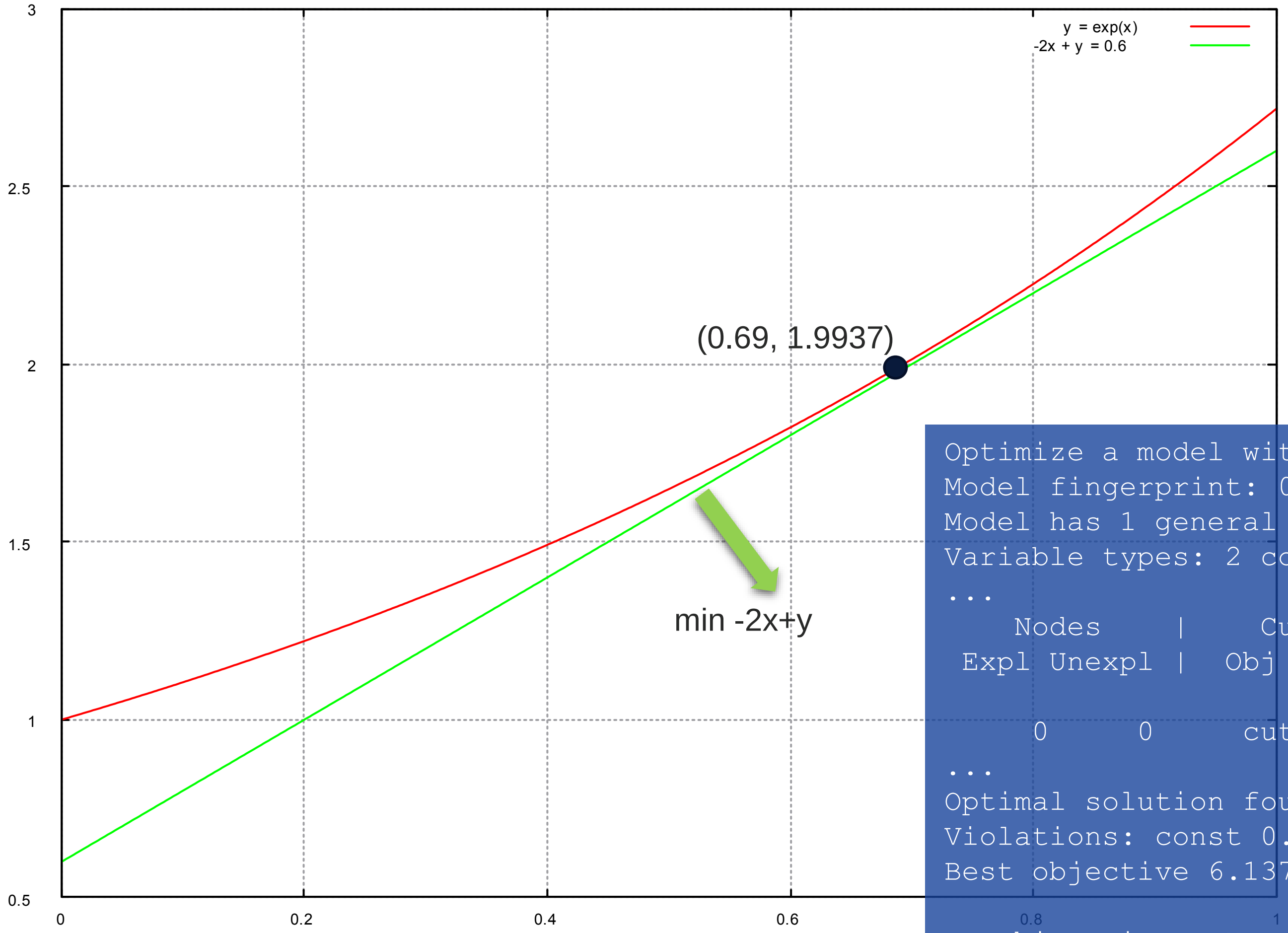
$$(0, e^0), (0.01, e^{0.01}), \ldots, (1, e^1)$$

- Python code:

```
n = 100
xpts = [1.0*k/n for k in range(n+1)]
ypts = [math.exp(xpts[k]) for k in range(n+1)]

model = Model("pwltest")
x = model.addVar(lb=0, ub=1, name="x")
y = model.addVar(name="y")
gc = model.addGenConstrPWL(x, y, xpts, ypts, "gc")
model.setObjective(-2*x + y)
model.optimize()
```

Piecewise-Linear (PWL) Constraints




```
Optimize a model with 0 rows, 2 columns and 0 nonzeros
Model fingerprint: 0xe289cdc2
Model has 1 general constraint
Variable types: 2 continuous, 0 integer (0 binary)
...
    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0     cutoff    0          0.61372    0.61372  0.00%     -    0s
...
Optimal solution found (tolerance 1.00e-04)
Violations: const 0.0000e+00, bound 0.0000e+00, int 0.0000e+00, genconstr 0.0000e+00
Best objective 6.137155332431e-01, best bound 6.137155332431e-01, gap 0.0000%

gurobi> print x.X
0.69
gurobi> print y.X
1.99371553324
```

Function Constraints with Automatic PWL Translation

$y = f(x)$

Support: polynomial, $\log(x)$, $\log_a(x)$, e^x, a^x, x^a, $\sin(x)$, $\cos(x)$, $\tan(x)$

Another new type of general constraint

Example

- $y = e^x, \; 0 \leq x \leq 1$
- Python code: `gc = model.addGenConstrExp(x, y, name="gc")`

Gurobi will automatically compute breakpoints and perform PWL translation

- Smart translation for periodic functions sin(), cos(), and tan()
- Use actual functions during presolve
- Bound strengthening in presolve may lead to more efficient PWL translation

Options for Automatic PWL Translation

Options

- FuncPieces, FuncPieceLength, FuncPieceError, FuncPieceRatio
- Attributes: specific for a function constraint
- Parameters: for all function constraints

Speed-versus-accuracy

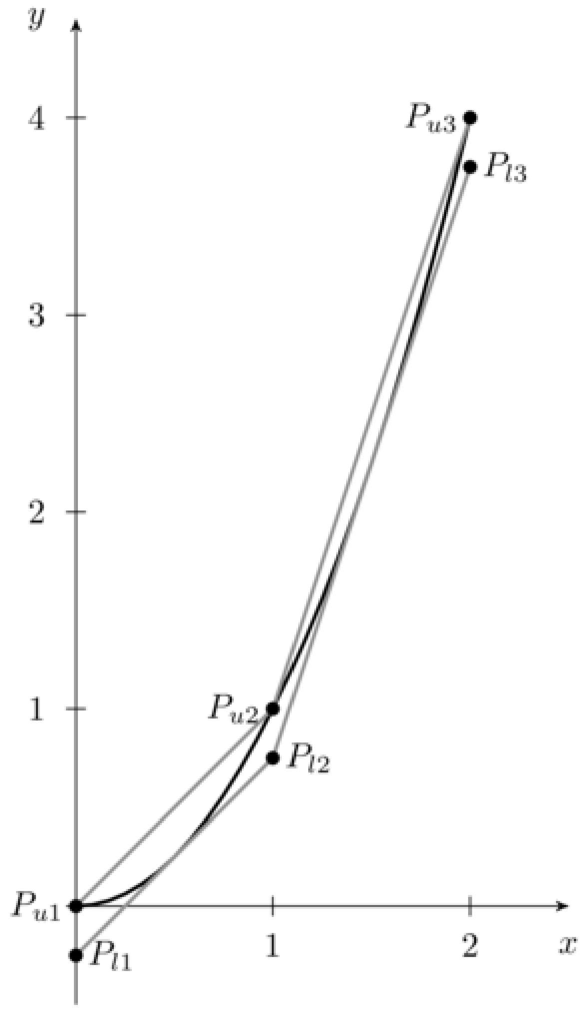
- FuncPieces, FuncPieceLength, FuncPieceError
- Choices for using piece length, number of pieces and maximum allowed error

Underestimate and overestimate

- FuncPieceRatio

Example, $y = x^2$

- Underestimation: (P_{l1}, P_{l2}, P_{l3})
- Overestimation: (P_{u1}, P_{u2}, P_{u3})
- Error: 0.25 for both pieces [0,1] and [1,2]



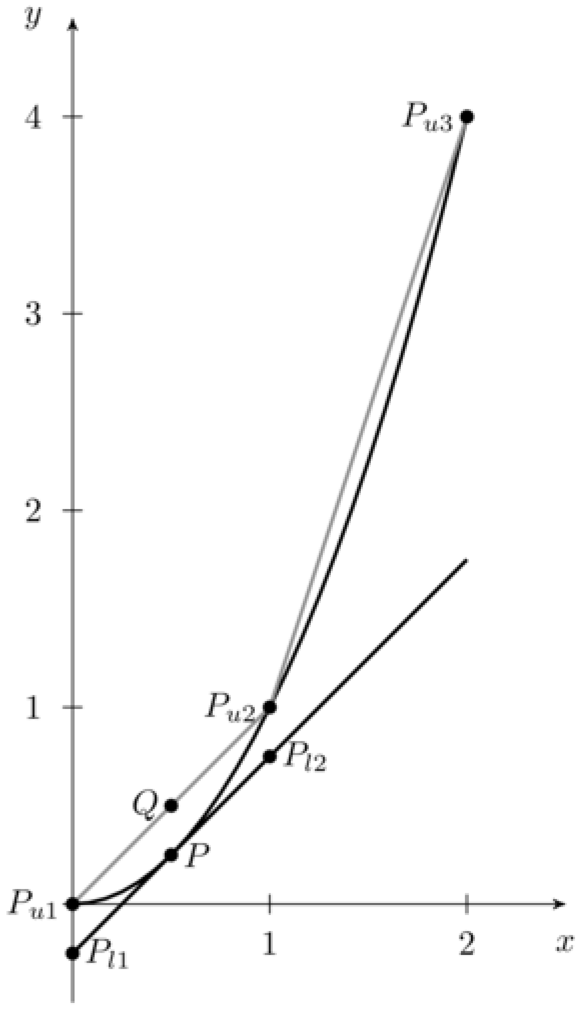
Non-linear Capability

Theory

- Multivariate polynomials can be decomposed into bilinear functions
  - $z = x^4 y^2$
  - Let $u = x^2$, $v = u^2$, $w = y^2$ (bilinear)
  - Then $z = vw$ (bilinear)
- PWL constraints allow to approximate single-variable non-linear functions
- Solve wide range of non-linear programming problems to global optimality
  - e.g., $z = sin(x^2 y) \cdot e^{x+y^2}$

Reality

- Errors will amplify for decomposing/combining
- Errors for PWL approximation can be big
- Example:
  - $y = x - 0.25,\ y = x^2$
  - $y = x - 0.25$ is for line (P_{l1}, P, P_{l2})
  - $P = (x, y) = (0.5, 0.25)$ is feasible
  - PWL approximation of $y = x^2$ with line segments (P_{u1}, P_{u2}, P_{u3})
  - $y = 0.5$ at $x = 0.5$, Q(0.5, 0.5) is quite far from $P(0.5, 0.25)$
  - Infeasible
  - FuncPieceLength=1 is too big: largest error = 0.25 is too big to be feasible



Highlights

Performance

- New cuts, new solution improvement heuristics, AVX-512 support, etc.

Major features

- Non-convex MIQCP (bilinear)
- Piecewise-linear (PWL) constraints
- Function constraints with automatic PWL translation
- MIP scenario analysis (what if, MIP sensitivity) dealing with uncertainty
- New matrix friendly API for gurobipy (support for SciPy sparse matrices)
- New Compute Server capabilities
- Batch optimization in Compute Server

Other improvements

- Intermediate solution files
- Support for lazy constraint callback for Compute Server
- Indices for variables and constraints in OO APIs
- Model attribute file
- Interactive shell updated from Python 2.7 to 3.7

MIP Scenario Analysis

Motivation

- Math model is usually an approximation for the real-world application
- Inputs are often approximations e.g. forecasted demands
- Business condition or environment may change
- Important to know the sensitivity of the computed solution to the changes, e.g.
  - What if the demand for this item increases from 10 to 12?

Wrong approach

- Fix all integer variables, solve fixed model as LP, look at reduced costs and duals of fixed LP
- Gives bogus results! Does not make sense from mathematical point of view!

Our approach

- Have a base model
- Use attributes to specify a set of scenarios
  - each scenario is described by changes to the base model
  - currently supported changes: objective coefficients, variable bounds, linear constraint right hand sides
- Compute optimal solutions for all scenarios
- Solutions provide the insight into how the solution would change for different scenarios

MIP Scenario Analysis

GUROBI OPTIMIZATION

Simple approach

- Solve each scenario as an independent MIP
- Old example: sensitivity.py

New approach in Gurobi 9.0

- Convenient to define scenarios
  - use attributes
  - example sensitivity.py is rewritten by using the new feature
- Performance
  - faster
  - may add distributed version with Compute Server in a future version
  - information, such as objective bounds, feasible solutions for each scenario, still available even if you stop early (e.g., due to time limit)

Dealing With Uncertainty

MIP Scenario Analysis

- Find optimal solution for each individual scenario
- Each solution may be infeasible for other scenarios
- Analyze business model: how sensitive are best decisions w.r.t. input data?
- MIP-version of LP sensitivity analysis

Stochastic Optimization

- Find one solution that is optimal w.r.t. expected value over all scenarios
- User needs to specify probabilities for each scenario
  - or: provide probability density functions for random variables

Robust Optimization

- Find one solution that is optimal in worst case scenario
- Model coefficients are not fixed but user specifies ranges
  - more generally: coefficients should be in a specified convex set

Highlights

Performance

- New cuts, new solution improvement heuristics, AVX-512 support, etc.

Major features

- Non-convex MIQCP (bilinear)
- Piecewise-linear (PWL) constraints
- Function constraints with automatic PWL translation
- MIP scenario analysis (what if, MIP sensitivity)
- New matrix friendly API for gurobipy (support for SciPy sparse matrices) data scientists, engineers
- New Compute Server capabilities
- Batch optimization in Compute Server

Other improvements

- Intermediate solution files
- Support for lazy constraint callback for Compute Server
- Indices for variables and constraints in OO APIs
- Model attribute file
- Interactive shell updated from Python 2.7 to 3.7

New Matrix Friendly API for gurobipy

GUROBI OPTIMIZATION

About Python

- A very popular programming language
- Huge library
  - Standard library
  - 3rd party packages with simple package manager
- A top language for data science

Data scientists and engineers are used to work with matrices

- Large fraction of them use Python with NumPy and SciPy

gurobipy accepts NumPy's ndarrays and scipy.sparse matrices as input

- More convenient if the underlying model is naturally expressed with matrices
- Faster because no modeling objects for individual linear expressions are created
- API offers two layers (only linear constraints shown here):
  - Add matrix constraints directly from the data through `Model.addMConstrs(A, x, sense, b)`
  - Use matrix variable modeling objects `x = Model.addMVar(shape)`, add constraints through `Model.addConstr(A @ x <= b)`

Python Matrix API – A Tiny Example

mip1.py: algebraic syntax

```
import gurobipy as gp
from gurobipy import GRB

m = gp.Model("mip1")

x = m.addVar(vtype=GRB.BINARY, name="x")
y = m.addVar(vtype=GRB.BINARY, name="y")
z = m.addVar(vtype=GRB.BINARY, name="z")

m.setObjective(x + y + 2 * z, GRB.MAXIMIZE)

m.addConstr(x + 2 * y + 3 * z <= 4, "c0")
m.addConstr(x + y >= 1, "c1")

m.optimize()
```

matrix1.py: matrix expressions

```
import numpy as np
import scipy.sparse as sp
import gurobipy as gp
from gurobipy import GRB

m = gp.Model("matrix1")

x = m.addMVar(shape=3, vtype=GRB.BINARY, name="x")

obj = np.array([1.0, 1.0, 2.0])
m.setObjective(obj @ x, GRB.MAXIMIZE)

data = np.array([1.0, 2.0, 3.0, -1.0, -1.0])
row = np.array([0, 0, 0, 1, 1])
col = np.array([0, 1, 2, 0, 1])
A = sp.csr_matrix((data, (row, col)), shape=(2, 3))

rhs = np.array([4.0, -1.0])

m.addConstr(A @ x <= rhs, name="c")

m.optimize()
```

Highlights

GUROBI OPTIMIZATION

Performance

- New cuts, new solution improvement heuristics, AVX-512 support, etc.

Major features

- Non-convex MIQCP (bilinear)
- Piecewise-linear (PWL) constraints
- Function constraints with automatic PWL translation
- MIP scenario analysis (what if, MIP sensitivity)
- New matrix friendly API for gurobipy (support for SciPy sparse matrices)
- Cluster Manager
- Batch optimization in Compute Server

corporate IT, private cloud, offline solves

Other improvements

- Intermediate solution files
- Support for lazy constraint callback for Compute Server
- Indices for variables and constraints in OO APIs
- Model attribute file
- Interactive shell updated from Python 2.7 to 3.7

Cluster Manager

Administer cluster of Gurobi Compute Servers

- IT department can control and track cluster size and work-load
- Users can monitor and manage their jobs

User management

- Allows to assign users to different roles (user, admin, cluster admin)
- Improved security and API keys

Web UI for on-premise

- User-friendly graphical interface for administrators and users

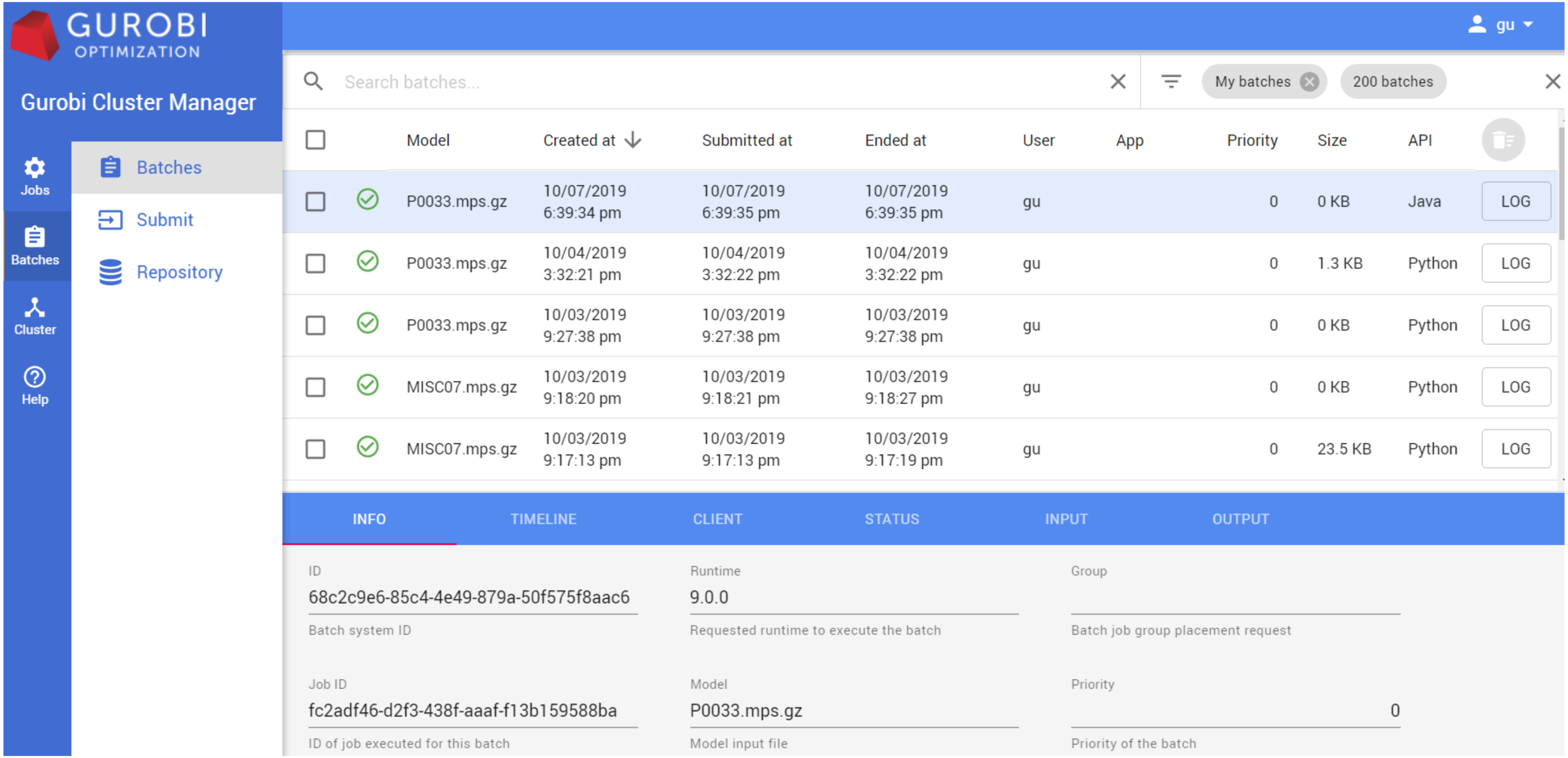
Job history

- Statistics and logs for individual jobs

Batch mode support

- Offline optimization for long running jobs
- Client application may disconnect and retrieve results later

Cluster Manager – Web UI



Batch Optimization in Cluster Manager

Motivation

- Compute Server allows the client code to off-load computing work to a server
- A MIP often takes very long to solve
    - requiring the client code to wait may not be convenient
- Feature request
    - allow client code to disconnect from the Compute Server and to shut down
    - later, client code should be able to reconnect and retrieve results

Technical Problem

- Reconnecting to the server creates a mapping problem
    - variable and constraint objects at client side are gone

Our solution

- Create models locally at client site
- Tag a set of relevant variables and constraints
- Submit the model to Compute Server, get batch ID and disconnect
- Use job ID to query status and solution in JSON format

Highlights

Performance

- New cuts, new solution improvement heuristics, AVX-512 support, etc.

Major features

- Non-convex MIQCP (bilinear)
- Piecewise-linear (PWL) constraints
- Function constraints with automatic PWL translation
- MIP scenario analysis (what if, MIP sensitivity)
- New matrix friendly API for gurobipy (support for SciPy sparse matrices)
- Cluster Manager
- Batch optimization in Compute Server

Other improvements

- Intermediate solution files
- Support for lazy constraint callback for Compute Server
- Indices for variables and constraints in OO APIs
- Model attribute file
- Interactive shell updated from Python 2.7 to 3.7

Thank You – Questions?

GUROBI OPTIMIZATION

The World's Fastest Solver