What's New in Gurobi 11.0

Dr. Jue Xue
Technical Account Manager
April 9, 2024

GUROBI OPTIMIZATION

Agenda

Performance and New Features

- **LP**
- **MILP**
- **MIQCP**

Global MINLP

- **Adaptive Constraints**

Other New Features

Cluster and Compute Server Improvements

Java & Gurobipy Enhancements

Gurobi solves the broadest range of problems – regardless of size or type.

LP	**QP**	**QCP**
MILP	**MIQP**	**MIQCP** Convex & nonConvex
SOCP	**Bi-linear**	**MINLP** New in 11

Performance Improvements Highlights

11.0 vs. 10.0

MODEL TYPE / ALGORITHM	OVERALL SPEED-UP (>1s)	HARD MODELS (>100s)
• MILP	8.6 %	12.4%
• MIQP	12.8 %	22.8%
• Convex MIQCP	9.2 %	18.2%
• Non-Convex MIQCP	133.4% (2.3x)	480.2% (5.8x)

Mean runtime improvements for different algorithms across all models of a particular type

Source: R&D - Internal model database

GUROBI OPTIMIZATION

Linear Programming

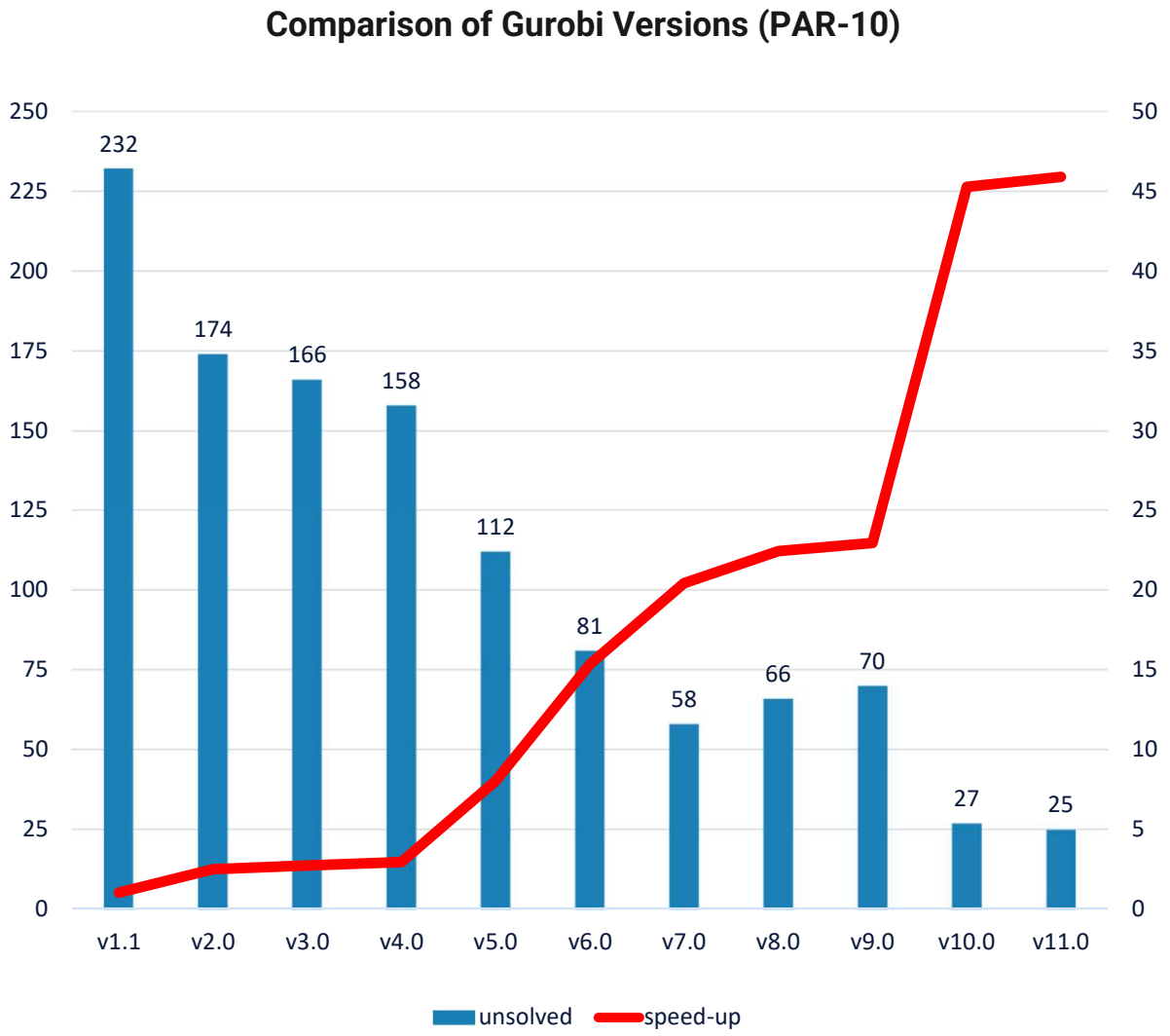
GUROBI OPTIMIZATION

LP

Performance Evolution

Default settings:

- Gurobi 1 – 4: dual simplex
- Gurobi 5+: concurrent LP



Time limit: 10000 sec.
Intel Xeon CPU E3-1240 v5 @ 3.50GHz
4 cores, 8 hyper-threads
32 GB RAM

Test set has 2574 models:
- 225 discarded due to inconsistent answers
- 77 discarded that none of the versions can solve
- speed-up measured on >100s bracket: 596 models

Concurrent LP Algorithm Parameters

GUROBI OPTIMIZATION

Gurobi 10.0

- `Method`
  - `-1`: automatic
  - `0`: primal simplex
  - `1`: dual simplex
  - `2`: barrier
  - `3`: non-deterministic concurrent LP
  - `4`: deterministic concurrent LP
  - `5`: deterministic concurrent simplex

Gurobi 11.0

- `Method`
  - `-1`: automatic
  - `0`: primal simplex
  - `1`: dual simplex
  - `2`: barrier
  - `3`: non-deterministic concurrent LP
  - `4`: deterministic concurrent LP
  - `5`: deterministic concurrent simplex (deprecated)
- ConcurrentMethod
  - `-1`: automatic
  - `0`: barrier/dual/primal
  - `1`: barrier/dual
  - `2`: barrier/primal
  - `3`: dual/primal

GUROBI OPTIMIZATION

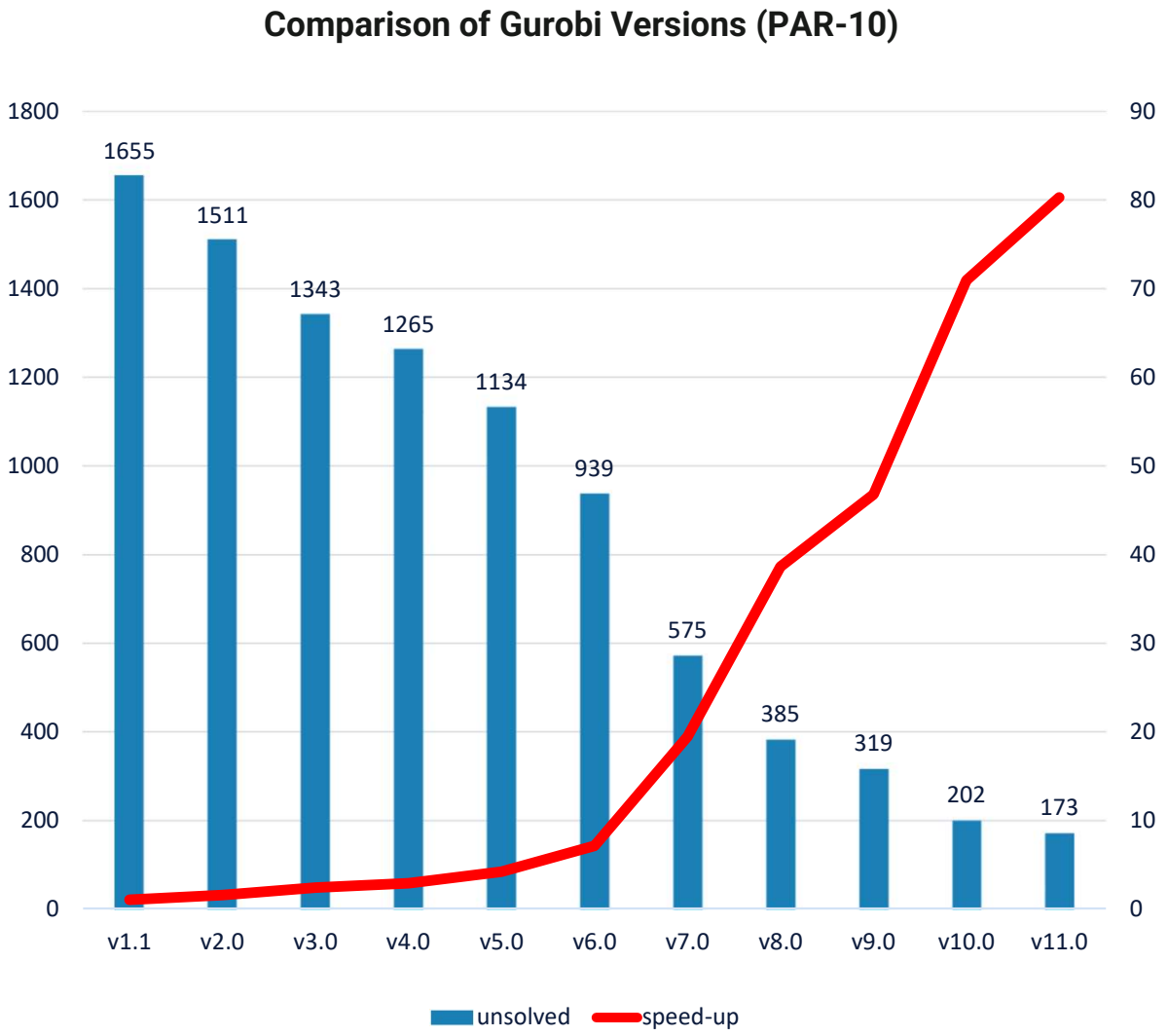
Mixed Integer Linear Programming

GUROBI OPTIMIZATION

MILP

Performance Evolution

~80x faster
since Release 1



Time limit: 10000 sec.
Intel Xeon CPU E3-1240 v5 @ 3.50GHz
4 cores, 8 hyper-threads
32 GB RAM

Test set has 7766 models:
- 714 discarded due to inconsistent answers
- 2124 discarded that none of the versions can solve
- speed-up measured on >100s bracket: 2892 models

Mixing Path Cuts

GUROBI OPTIMIZATION

- New procedure for finding constraints
  - … to which mixed integer aggregation rounding can be applied
- Using “U-cut procedure” from Christophel
  - Integrated in our MIR aggregation procedure

- Parameter `MixingCuts` with values -1, 0, 1, 2
- Performance impact: 0.5% overall, 1.2% on >100sec models

- References:
  - O. Gunluk, Y. Pochet: Mixing mixed-integer inequalities.
    Math. Program. 90, 429–457 (2001). https://doi.org/10.1007/PL00011430
  - P. Christophel: Separation algorithms for cutting planes based on mixed integer row relaxations:
    implementation and evaluation in the context of mixed integer programming solver software. (PhD thesis)
    University of Paderborn, 2009, pp. 1-222

GUROBI OPTIMIZATION

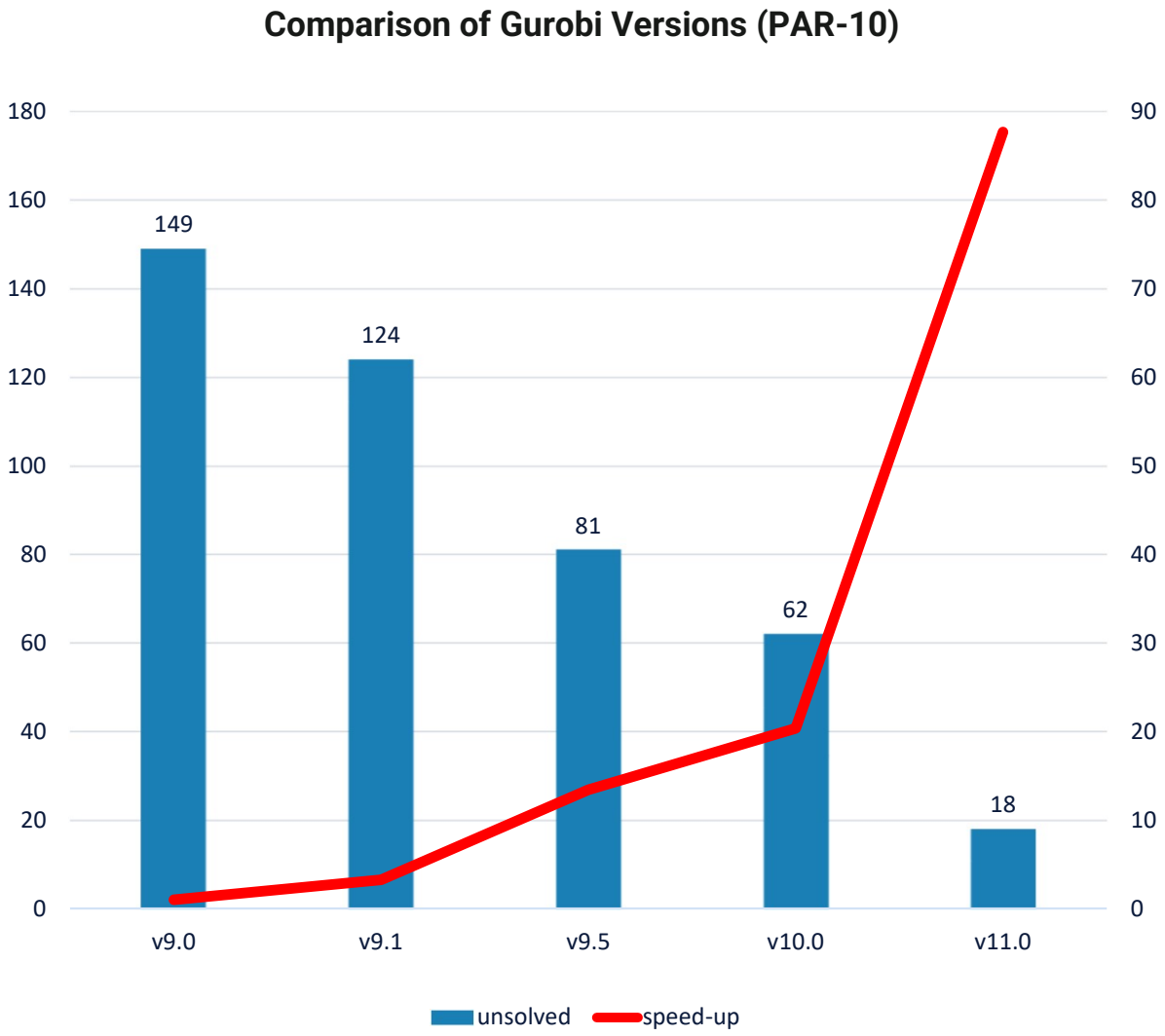
Non-convex MIQCP

GUROBI OPTIMIZATION

Non-convex MIQCP

Performance Evolution

~88x faster
since Release 9



Time limit: 10000 sec.
Intel Xeon CPU E3-1240 v5 @ 3.50GHz
4 cores, 8 hyper-threads
32 GB RAM

Test set has 1064 models:
- 50 discarded due to inconsistent answers
- 344 discarded that none of the versions can solve
- speed-up measured on >100s bracket: 275 models

Quadratic Objective and Constraints

NonConvex Parameter

GUROBI OPTIMIZATION

- Nonconvexity could be a sign of an error in model or data
- NonConvex:
  - `-1`: automatic
  - `0`: return error if original model has nonconvex Q objective or constraints
  - `1`: return error if presolved model has nonconvex Q that cannot be linearized
  - `2`: accept nonconvex Q by using a bilinear transformation
- NonConvex default value change – may surprise or break user code!
  - Gurobi 11.0 default (-1): essentially equivalent to 2
  - Gurobi 10.0 default (-1): equivalent to 1
    - Users now may want to set NonConvex=1 explicitly

Nonlinear Constraints

- Gurobi 9.0 and later provide API to define nonlinear functions
  - e^x, a^x
  - $\ln(x), \log_a(x)$
  - $\sin(x), \cos(x), \tan(x)$
  - x^a
  - $ax^3 + bx^2 + cx + d$

```
addGenConstrExp(), addGenConstrExpA()
addGenConstrLog(), addGenConstrLogA()
addGenConstrSin(), addGenConstrCos(), addGenConstrTan()
addGenConstrPow()
addGenConstrPoly()
```

- Gurobi 9.0 – 10.0:
  - Nonlinear functions are replaced during presolve by a piecewise-linear approximation
- Gurobi 11.0:
  - Can choose to treat nonlinear constraints exactly

FuncNonlinear Parameter and Attribute

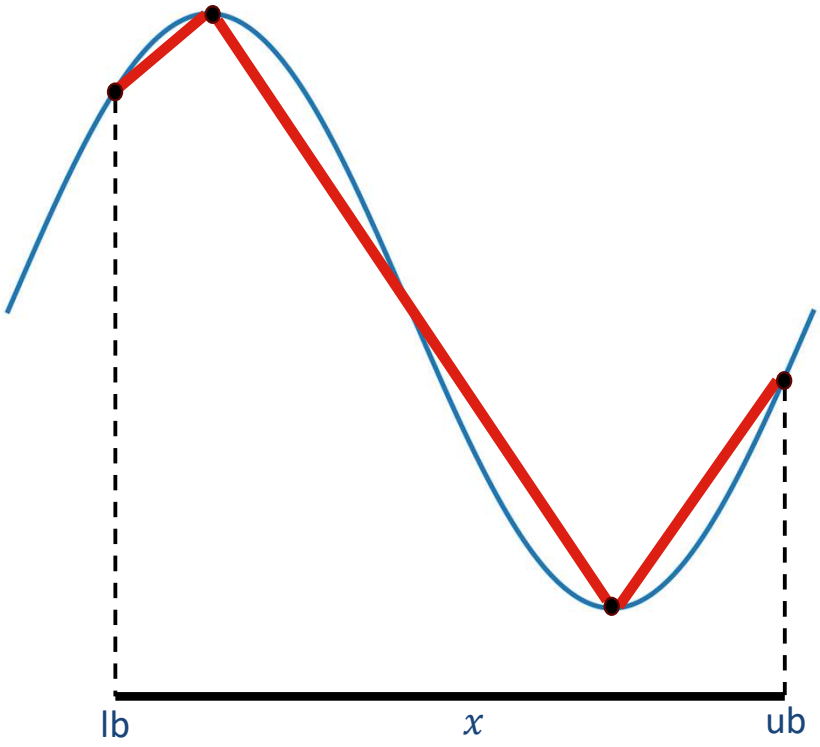
- Existing PWL approximation of general function constraints are controlled by
  - `FuncPieces`
  - `FuncPieceLength`
  - `FuncPieceError`
  - `FuncPieceRatio`

- The default behavior of `FuncPieces` is now to use a relative error approach
  - Was mainly restricting the total number of pieces in Gurobi 10.0
- New `FuncNonlinear` attribute to switch between PWL and outer approximation:
  - `-1`: behavior defined by FuncNonlinear parameter
  - `0`: use static PWL approximation
  - `1`: use dynamic outer approximation
- New `FuncNonlinear` parameter to control default (-1) of attributes:
  - `0`: use static PWL approximation
  - `1`: use dynamic outer approximation

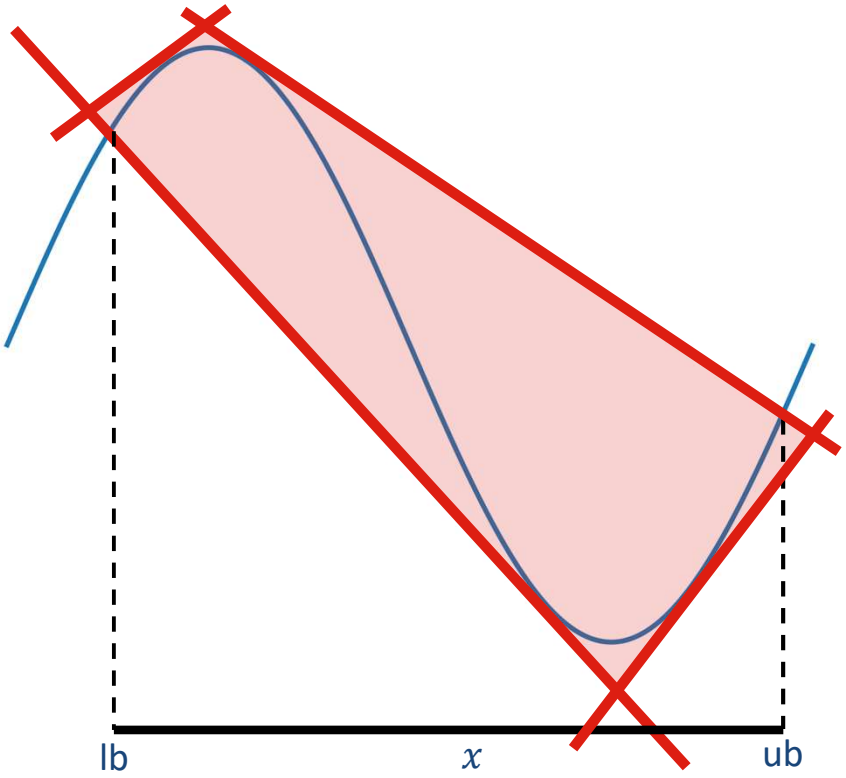
GUROBI OPTIMIZATION

Global MINLP with Adaptive Constraints

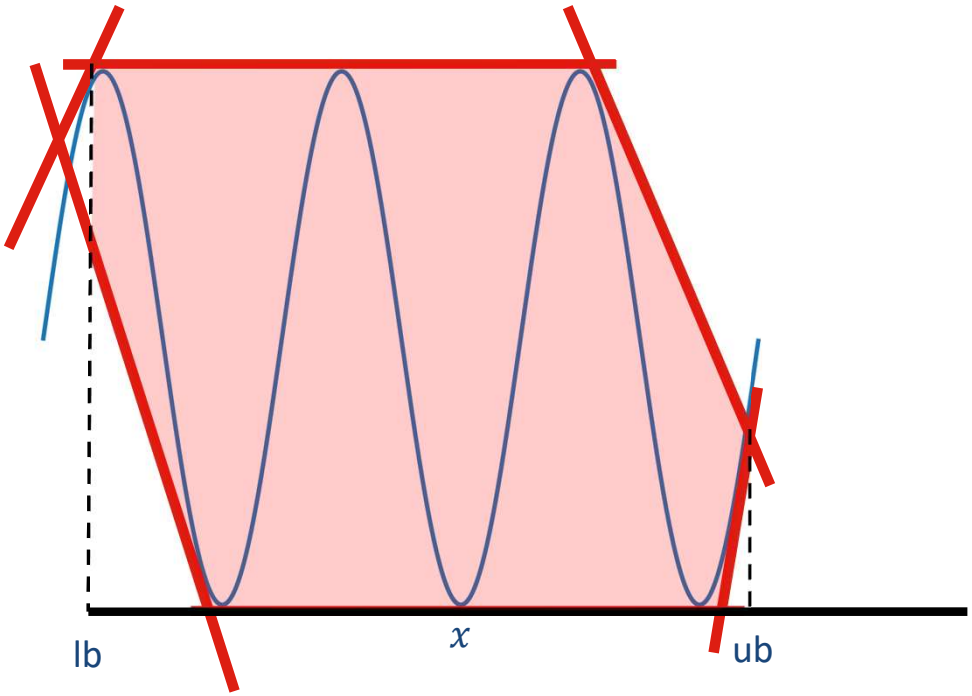
PWL Approximation vs. Outer Approximation



PWL approximation



outer approximation



GUROBI OPTIMIZATION

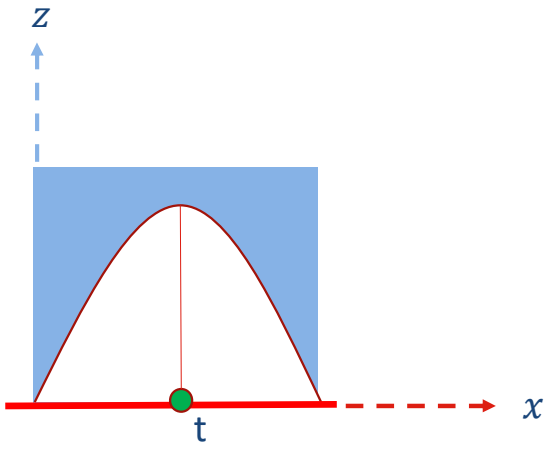
"Large" Domains

- Not much to get from the relaxation if domain of x is large
- Branching on x tightens the relaxation quickly!
- Tighter initial bounds will speed up performance

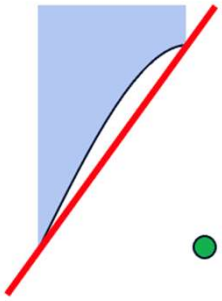
Branching

- After solving the convex relaxation, how do we branch on the violated nonconvexities?

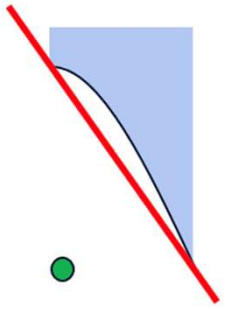
non-convex
$-z - x^2 \leq 0$



$$\begin{array}{lllllll} \min & c^T x & + & d^T z & & & \\ \text{s.t.} & Ax & + & Dz & \leq & b & \\ & -x_i x_j & + & z_{ij} & = & 0 & \text{for all } (i,j) \in S \\ & l & \leq & x & \leq & u & \\ & & & x_j & \in & \mathbb{Z} & \text{for all } j \in I \end{array}$$



branching
$x \leq t$ or $x \geq t$



update relaxation bounds and the associated McCormick envelopes locally

Adaptive Constraints

- Adaptive constraints change dynamically as the algorithm proceeds
  - They involve values that change as the algorithm proceeds
- Coefficients and right-hand sides of an envelope constraints depend on the local bounds of variables
  - Whenever local bounds change, coefficients, and right-hand sides are updated
    - This may lead to a singular or ill-conditioned basis
    - Products of bounds in the McCormick constraints can lead to very large or small right-hand side values
- By contrast, MILP has almost no adaptive constraints
  - Local bound strengthening can occur, but it mostly involves tighter bounds and doesn't affect any constraint matrix values
  - (Optional) Implied bound cuts/coefficient reduction can modify the matrix coefficients

GUROBI OPTIMIZATION

Other New Features

Copying Models From Local Environment

- Copy model from one environment to another

```
c = m.copy()     # regular copy: c in same environment as m
c = m.copy(env)  # new: c is created in environment env
```

- Use case: parallel execution of two optimization runs

- Caveat:
  - Can copy to a remote (Compute Server) model but not from a remote model

Interrupt and Resume with Change of Threads

- Interrupting a solve and then calling optimize() again:
  - Gurobi 10: changes to `Threads` parameter in between are ignored
  - Gurobi 11: changes to `Threads` parameter will be obeyed when resuming

- Example use case:

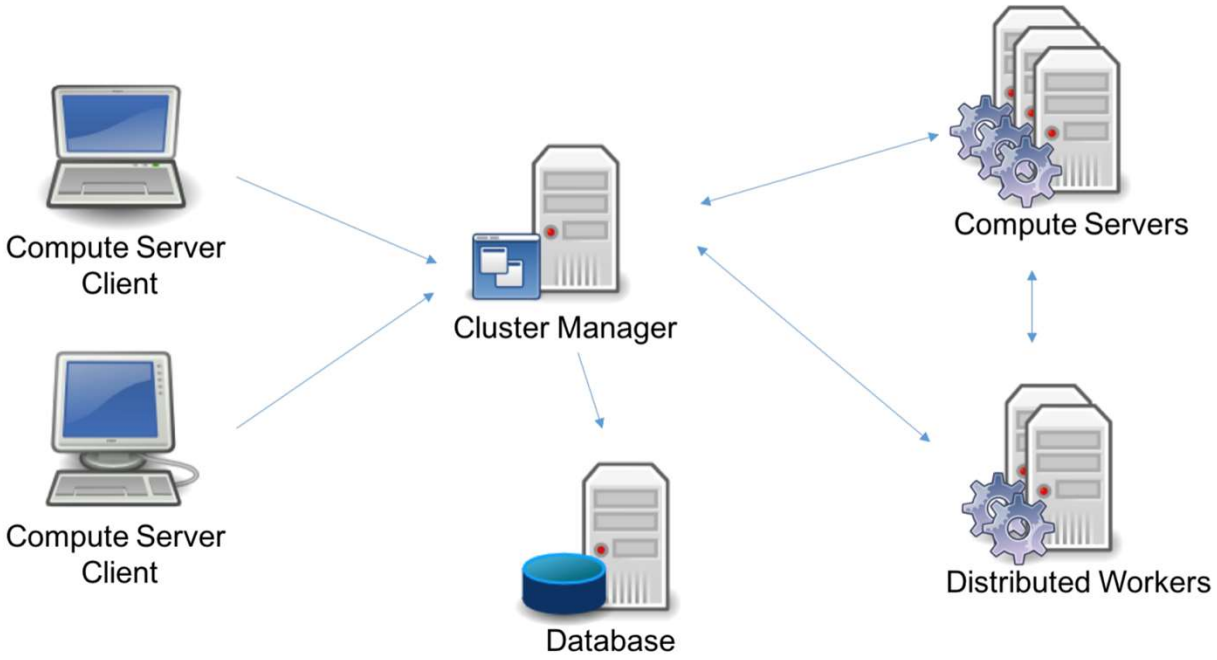
```
m.Params.Threads = 8
m.Params.SoftMemLimit = 4
m.optimize()
if m.status == gp.GRB.MEM_LIMIT:
  m.Params.Threads = 1
  m.optimize()
```


GUROBI OPTIMIZATION

Cluster and Compute Server Enhancements

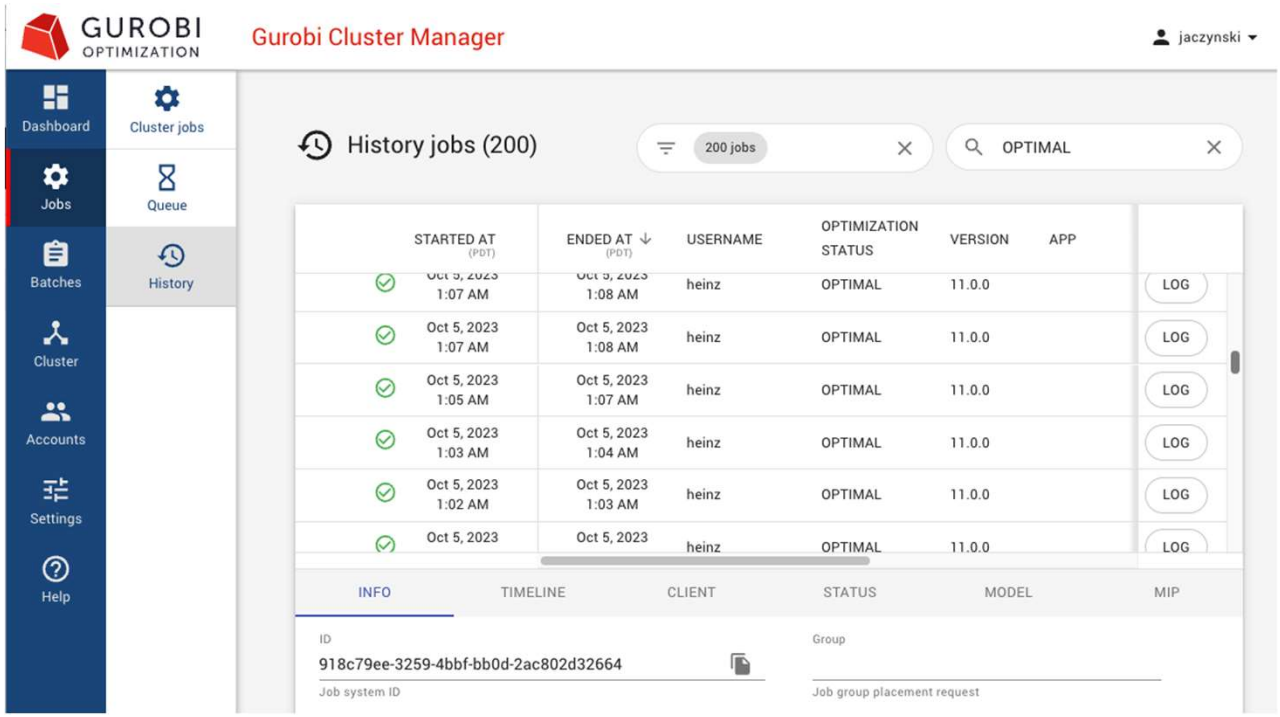
Increased Flexibility for Compute Cluster

- Compute Cluster Overview



Improved Cluster Manager/Compute Server

Compute Server/Cluster Manager facilitates the deployment and use of optimization services on-premises or on private cloud.



- New Look&Feel
- Time zone selection and formatting
- Improved security
- Enforce memory/time limit server side

More Flexible Enterprise Integration

- Case sensitivity settings for usernames
- Support for SAML Authentication Protocol
  - Ex: Microsoft Azure AD, Okta, JumpCloud, Google and others

okta

Google Workspace

jumpcloud

- Support of Microsoft Azure CosmosDB
  - In addition to Amazon DocumentDB and MongoDB

mongoDB.

GUROBI OPTIMIZATION

APIs: Java & Gurobipy Enhancements

Java API

GUROBI OPTIMIZATION

- Java package name is now `com.gurobi.gurobi` instead of `gurobi`
  - To follow Java standard naming scheme
- Java package now distributed on Maven Central
  - Most popular Java package repository
  - Similar to PyPI for Python
  - Requested by multiple customers
  - Helps build and deployment processes for Java users

gurobipy: Installation changes

GUROBI OPTIMIZATION

- Type hinting batteries included
  - No more `gurobipy-stubs`
- `setup.py install` is no more
  - Offline installs are possible with pip
  - Hash verification is possible with pip
- conda and pip play nicely together
  - No more duplicate installs
  - Cleaner install for our open-source packages on conda

```
The conflict is caused by:
    The user requested gurobipy==11.0.0b1
    gurobipy-stubs 2.0.0 depends on gurobipy==10.*
    The user requested gurobipy==11.0.0b1
    gurobipy-stubs 1.0.1.post0 depends on gurobipy==9.5.*
```

```
# Name                    Version           Build  Channel
blas                      1.0                 mkl
bottleneck                1.3.5        py311hb9e55a9_0
bzip2                     1.0.8                h1de35cc_0
ca-certificates           2023.08.22           hecd8cb5_0
gurobi                    10.0.3              py311_0    gurobi
gurobipy                  10.0.3               pypi_0    pypi
gurobipy-pandas           1.0.0                pypi_0    pypi
intel-openmp              2023.1.0        ha357a0b_43547
```

gurobipy: Matrix-friendly API integration

GUROBI OPTIMIZATION

- Callback functions now accept matrix-friendly API objects

```
x_sol = model.cbGetSolution(x)
model.cbLazy(A @ x <= b)
```

- Numpy-style concatenation (hstack, vstack, concatenate)

```
X = model.addMVar((n, m))
Y = model.addMVar((n, k))
XY = gp.concatenate((X, Y), axis=1)  # (n, m+k) MVar
```

- Matrix-friendly indicator constraints (vectorized, broadcastable)

```
z = model.addVar(vtype=GRB.BINARY)
x = model.addMVar(n)
model.addGenConstrIndicator(z, True, A @ x <= b)  # MGenConstr ...
```

gurobipy: Debugging assists

GUROBI OPTIMIZATION

- Disable the default environment (opt-in feature)
  - Set environment variable `GUROBIPY_ALLOW_DEFAULTENV=0`
  - Helps with debugging token & remote job leaks, thread safety

```
>>> import gurobipy as gp
>>> model = gp.Model()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "src/gurobipy/model.pxi", line 80, in gurobipy.Model.__init__
  File "src/gurobipy/gurobi.pxi", line 55, in gurobipy.gurobi._getdefaultenv
gurobipy.GurobiError: Tried to start the default environment, but GUROBIPY_ALLOW_DEFAULTENV=0 is set
```

- Less silent failures
  - `Env.setParam` raises an exception for unknown parameters
  - `Model.getAttr/setAttr` raises an exception for variables not in a model
  - `Select/sum/prod` raises an exception if too many keys are passed

gurobipy: Other notable mentions

GUROBI OPTIMIZATION

- Any callable object can be a callback
  - Makes callable classes an option for callbacks
  - Avoids the `model._attribute` workaround
  - Check out the refreshed tsp.py and callback.py examples
- Performance improvements
  - `addConstr(A @ x == b)` ~2x faster for sparse data
  - ~10-20% faster term-based modelling patterns (credit to the Cython developers for that one!)
- Check out the Detailed Release Notes for a complete list

```
52 class TSPCallback:
53     """Callback class implementing lazy constraint
54     callbacks, solutions are checked for subtours
55     constraints are added if needed."""
56
57     def __init__(self, nodes, x):
58         self.nodes = nodes
59         self.x = x
60
61     def __call__(self, model, where):
62         """Callback entry point: call lazy constra
63         solutions are found. Stop the optimization
64         user code."""
65         if where == GRB.Callback.MIPSOL:
66             try:
67                 self.eliminate_subtours(model)
68             except Exception:
69                 logging.exception("Exception occur
70                 model.terminate()
```

GUROBI
OPTIMIZATION

Questions?