



**GUROBI**  
OPTIMIZATION

What's New in  
**Gurobi 12.0**

Tobias Achterberg  
Fernando Orozco



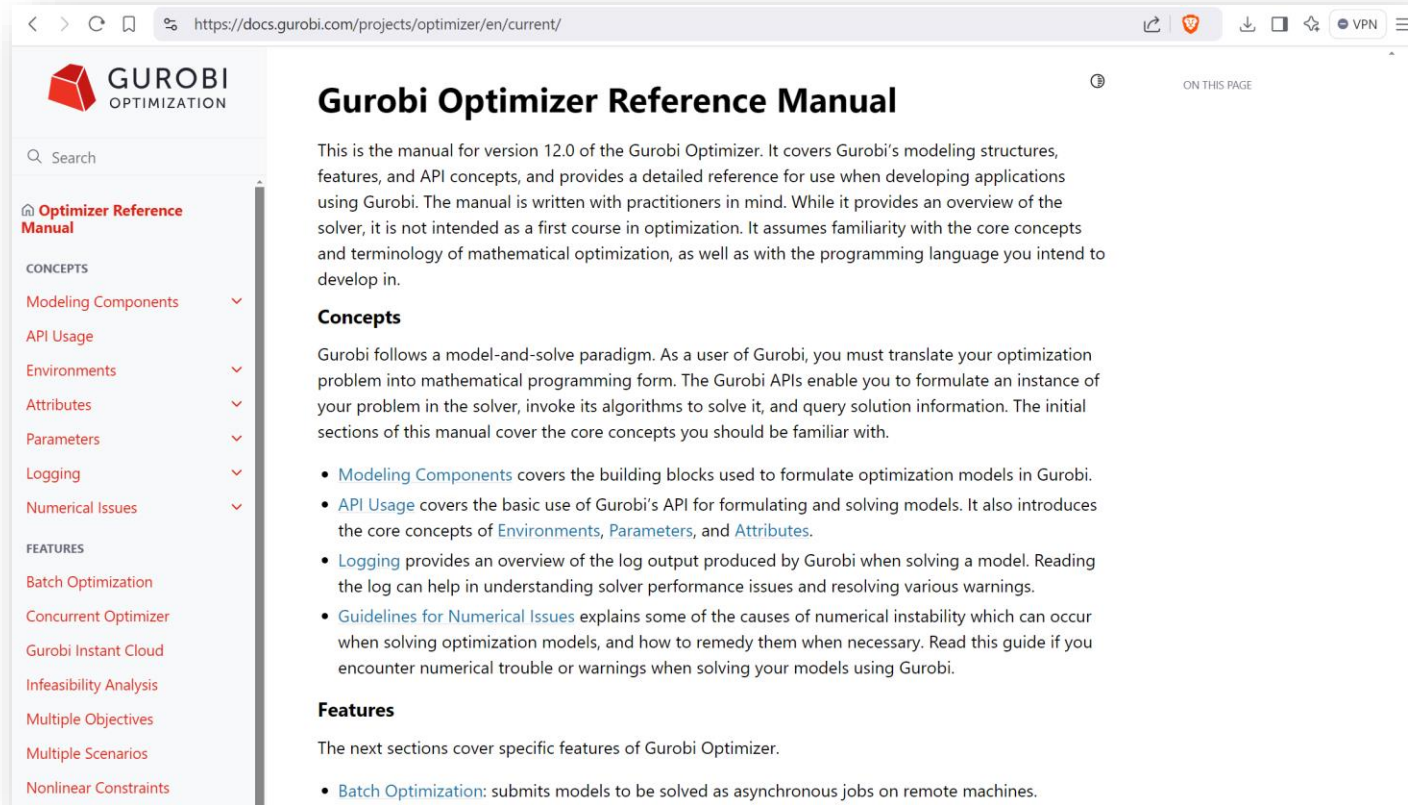
# Documentation





# New Documentation

- Completely revamped documentation
- Based on Sphinx
- Hosted on readthedocs
- More modern design
- Better structure, more cross-references
- Example code in tabs for all languages
- Much easier to maintain
  - Continuous integration of fixes and improvements



The screenshot shows the Gurobi Optimizer Reference Manual page. The URL is <https://docs.gurobi.com/projects/optimizer/en/current/>. The page features a navigation sidebar on the left with sections like "Optimizer Reference Manual", "CONCEPTS", and "FEATURES". The main content area is titled "Gurobi Optimizer Reference Manual" and includes an introductory paragraph, a "Concepts" section, and a "Features" section. The "Concepts" section states: "This is the manual for version 12.0 of the Gurobi Optimizer. It covers Gurobi's modeling structures, features, and API concepts, and provides a detailed reference for use when developing applications using Gurobi. The manual is written with practitioners in mind. While it provides an overview of the solver, it is not intended as a first course in optimization. It assumes familiarity with the core concepts and terminology of mathematical optimization, as well as with the programming language you intend to develop in." The "Features" section lists: "The next sections cover specific features of Gurobi Optimizer." and "Batch Optimization: submits models to be solved as asynchronous jobs on remote machines."



The screenshot shows the Gurobi Workforce Examples page. The URL is <https://docs.gurobi.com/projects/examples/en/current/examples/workforce.html>. The page features a navigation sidebar on the left with sections like "Gurobi Examples", "Example Tour", and "Example Source Code". The main content area is titled "Workforce Examples" and includes an introductory paragraph: "This section includes source code for all of the Gurobi workforce examples. The same source code can be found in the examples directory of the Gurobi distribution." Below this, there is a tabbed interface for "workforce1" with language options: C, C++, C#, Java, Matlab, Python, R, and Visual Basic. The C# tab is selected, showing a code snippet: 

```
/* Copyright 2024, Gurobi Optimization, LLC */  
  
/* Assign workers to shifts; each worker may or may not be available on a  
particular day. If the problem cannot be solved, use IIS to find a set of  
conflicting constraints. Note that there may be additional conflicts  
besides what is reported via IIS. */  
  
using System;
```



# Performance



# Gurobi 12 Performance

problem class	overall (>1 sec)	hard models (>100 sec)
LP (default)	2.6%	0.9%
LP (barrier)	2.2%	4.8%
LP (dual simplex)	4.4%	3.6%
LP (primal simplex)	2.6%	2.0%
QP	9.1%	—*
SOCP	37.3%	—*
MIP	13.1%	18.9%
MIQP	13.0%	38.3%
MIQCP	4.1%	3.3%
nonconvex MIQCP	27.7%	68.5%
IIS	22.7%	37.3%

\* too few “hard” models

# Gurobi 12 Performance

problem class	overall (>1 sec)	hard models (>100 sec)
LP (default)	2.6%	0.9%
LP (barrier)	2.2%	4.8%
LP (dual simplex)	4.4%	3.6%
LP (primal simplex)	2.6%	2.0%
QP	9.1%	—*
SOCP	37.3%	—*
MIP	13.1%	18.9%
MIQP	13.0%	38.3%
MIQCP	4.1%	3.3%
nonconvex MIQCP	27.7%	68.5%
IIS	22.7%	37.3%

\* too few “hard” models

# LP Presolve Improvements

- Derived variables presolve reduction 0.3%
  - Consider (3.0% on affected models)

$$\min\{c^T x \mid a_i^T x + d_i y \leq b_i, i = 1, \dots, m; l \leq x \leq u; y_l \leq y \leq y_u\}$$

with single variable  $y$  and variable vector  $x$

- If

$$\forall x \in [l, u] \exists y \in [y_l, y_u]: a_i^T x + d_i y \leq b_i \text{ for all } i = 1, \dots, m$$

then

- $y$  and all these constraints can be removed and
    - the value of  $y$  can be calculated after an optimal solution for  $x$  has been found.
  - Also applicable to MIP, but need to verify integrality restrictions
- 
- Stopping aggregator earlier for LP presolve 0.3%
    - Wait for next presolve pass to continue (1.1% on affected models)

# Simplex Improvements

- Primal simplex
  - Improved **numerics** in ratio test 1.1%
- Dual simplex
  - Performance improvement in Harris ratio test 1.9%
  - Major rework of **numerical aspects** 1.3%
  - Less objective shifting 0.7%
  - Better crash basis for free variables 0.6%
  - Improved **numerics** in ratio test 0.4%
  - Improved **numerics** in feasibility check for basic variables 0.4%



# Barrier Improvements

- Simplex: factorize more often in crossover 1.0%
- Simplify handling of dense blocks in A 0.7%
  - No need for dense blocks to use complex data structures to exploit sparsity
- Use iterative linear system solves in first barrier iterations 0.4%
  - Faster iterations than with Cholesky factorization (9.7% on affected models)
  - Less accurate

# Gurobi 12 Performance

problem class	overall (>1 sec)	hard models (>100 sec)
LP (default)	2.6%	0.9%
LP (barrier)	2.2%	4.8%
LP (dual simplex)	4.4%	3.6%
LP (primal simplex)	2.6%	2.0%
QP	9.1%	—*
<b>SOCP</b>	<b>37.3%</b>	<b>—*</b>
MIP	13.1%	18.9%
MIQP	13.0%	38.3%
MIQCP	4.1%	3.3%
nonconvex MIQCP	27.7%	68.5%
IIS	22.7%	37.3%

\* too few “hard” models

# SOCP Improvements

- Doubleton presolve aggregation inside cones 7.8%
  - Linear equality  $y = ax$  can be used for aggregation even if  $y$  appears in a cone
- Improved dense column handling for SOCP 4.7%
- Implicit handling of cone variable upper bounds 2.7%
  - See Andersen, Roos, Terlaky (2000), Sturm (2002), Goldfarb, Scheinberg (2005)

# Gurobi 12 Performance

problem class	overall (>1 sec)	hard models (>100 sec)
LP (default)	2.6%	0.9%
LP (barrier)	2.2%	4.8%
LP (dual simplex)	4.4%	3.6%
LP (primal simplex)	2.6%	2.0%
QP	9.1%	—*
SOCP	37.3%	—*
MIP	13.1%	18.9%
MIQP	13.0%	38.3%
MIQCP	4.1%	3.3%
nonconvex MIQCP	27.7%	68.5%
IIS	22.7%	37.3%

\* too few “hard” models

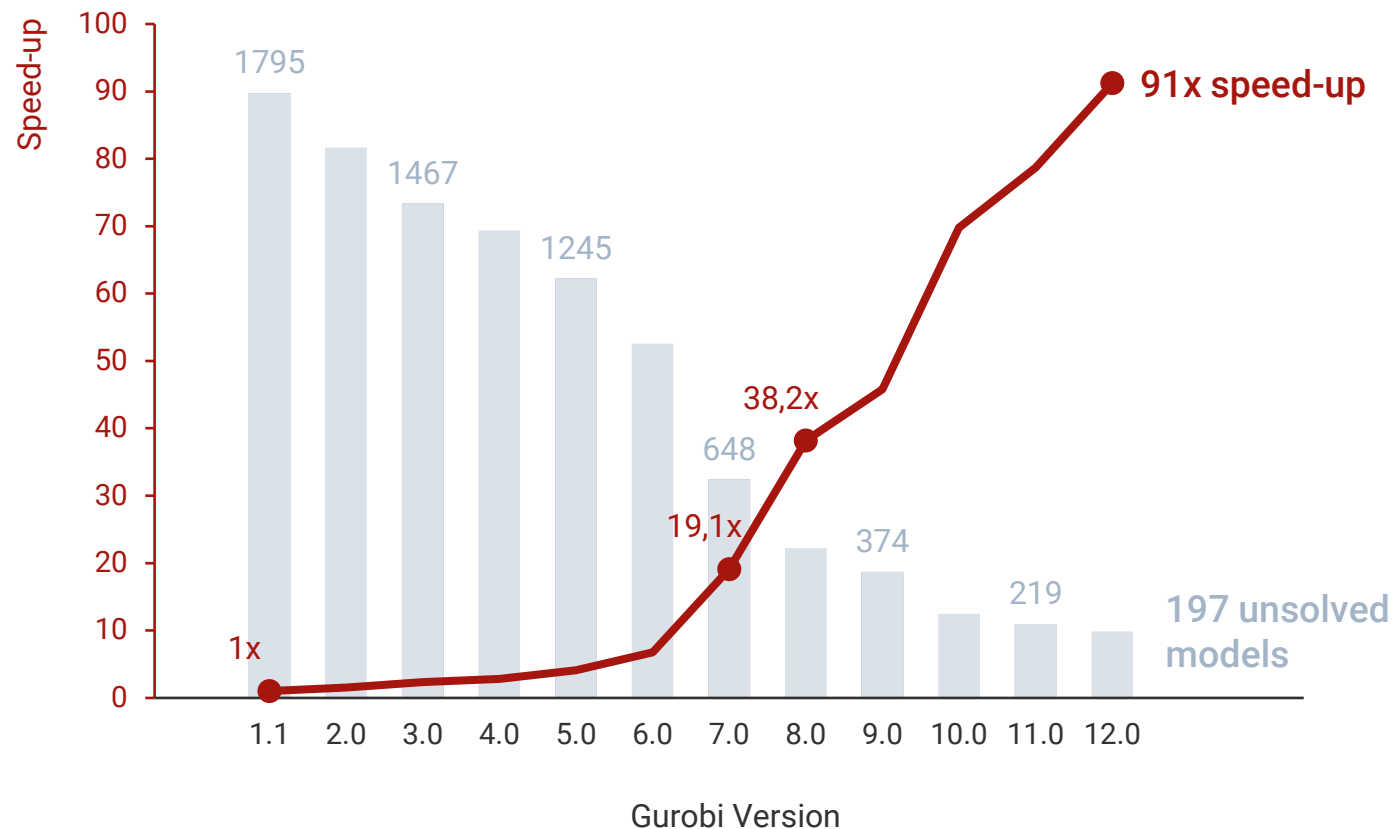
# MIP Improvements

• Simplex	7.6%
• 7 individual improvements (including Harris ratio test and numerical improvements)	
• Presolve	4.3%
• 22 individual improvements (including derived variables reduction)	
• Cutting planes	2.0%
• 11 individual improvements (including dual implied bound cuts)	
• Node presolve	1.6%
• 5 individual improvements	
• Parallelism	1.5%
• Preempt sub-MIPs and node LP solves for parallel synchronization	
• Improved usage of hyper-threads	
• Primal heuristics	1.2%
• 4 individual improvements	
• Branching	0.9%
• Use Driebeek penalties more often instead of strong branching	
• Conflict analysis	0.3%
• Do not ignore Farkas proofs with flipped basic variables	
• Memory management	0.2%
• Reduced number of memory allocations	



## Gurobi Version Comparison: Speed and Solvability (PAR-10)

Gurobi MILP Benchmark Suite



# MILP

## Performance Evolution

In Gurobi's MILP benchmark suite, the latest version delivers:

- A **91x** speed-up over version 1.1 in geometric mean (PAR-10) of runtimes
- Only **197** models remain **unsolved** after 10,000 seconds with the latest version. The test set consists of all models that can be solved by at least one version.

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

Test set has 8273 models:  
 - 788 discarded due to inconsistent answers  
 - 2286 discarded that none of the versions can solve  
 - speed-up measured on >100s bracket: 3076 models

# Gurobi 12 Performance

problem class	overall (>1 sec)	hard models (>100 sec)
LP (default)	2.6%	0.9%
LP (barrier)	2.2%	4.8%
LP (dual simplex)	4.4%	3.6%
LP (primal simplex)	2.6%	2.0%
QP	9.1%	—*
SOCP	37.3%	—*
MIP	13.1%	18.9%
MIQP	13.0%	38.3%
MIQCP	4.1%	3.3%
nonconvex MIQCP	27.7%	68.5%
IIS	22.7%	37.3%

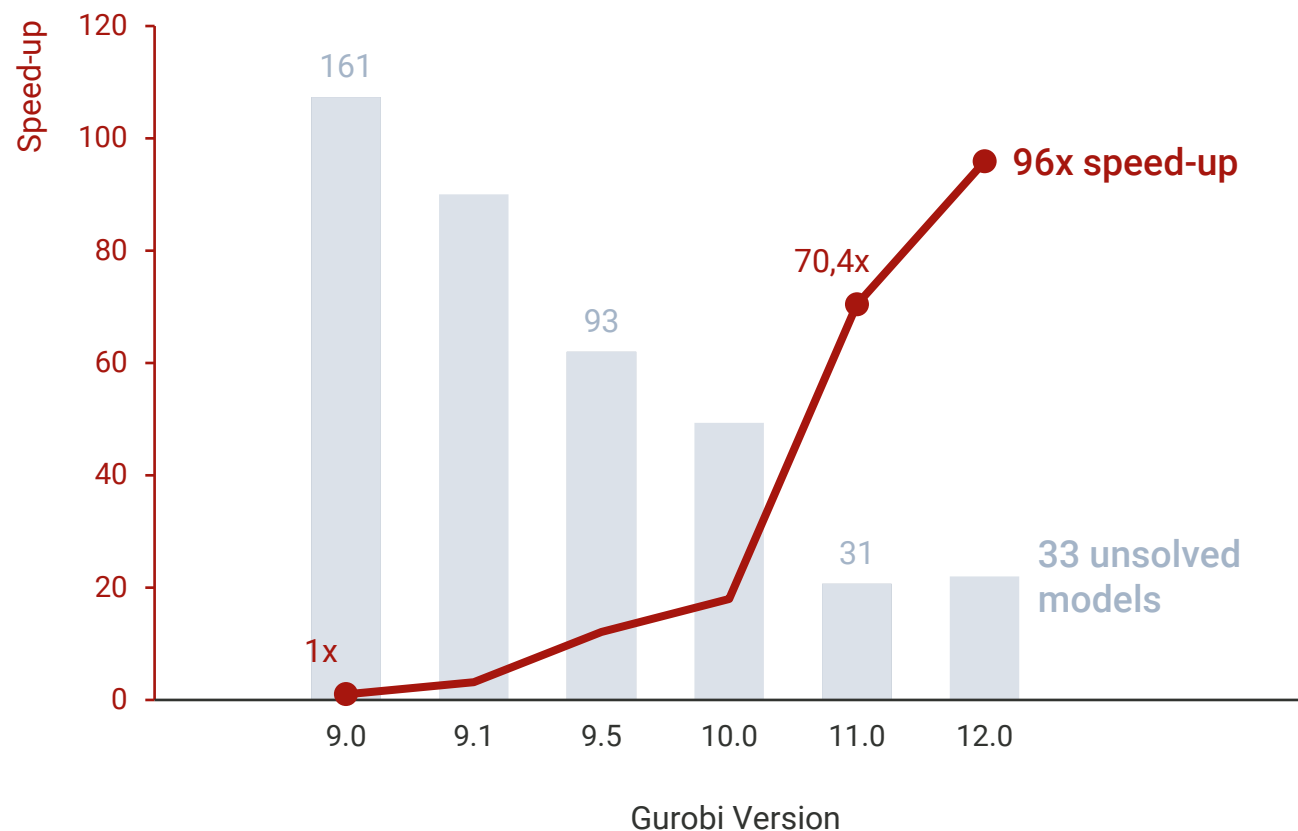
\* too few “hard” models

# Nonconvex MIQCP Improvements

• Node presolve <ul style="list-style-type: none"><li>• Improved reduced cost fixing</li></ul>	10.2%
• Presolve <ul style="list-style-type: none"><li>• Improved quadratic constraint to indicator translation</li><li>• Quadratic aggregation</li></ul>	7.1%
• Branching <ul style="list-style-type: none"><li>• Improved spatial branching value selection</li><li>• Use more strong branching in spatial branching</li></ul>	4.2%
• Restarts <ul style="list-style-type: none"><li>• More aggressive restarts for quadratic and nonlinear models</li></ul>	2.6%
• Primal heuristics <ul style="list-style-type: none"><li>• Improved fixing strategy in RINS-like heuristic</li></ul>	1.7%
• Simplex improvements for nonconvex MIQCP <ul style="list-style-type: none"><li>• Deactivate numerically problematic McCormick constraints</li></ul>	0.7%
• MIP and simplex improvements often carry over	(not explicitly measured)

## Gurobi Version Comparison: Speed and Solvability (PAR-10)

Gurobi Nonconvex MIQCP Benchmark Suite



# Nonconvex MIQCP

## Performance Evolution

In Gurobi's nonconvex MIQCP benchmark suite, the latest version delivers:

- A **96x** speed-up over version 9.0 in geometric mean (PAR-10) of runtimes
- Only **33** models remain **unsolved** after 10,000 seconds with the latest version. The test set consists of all models that can be solved by at least one version.

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:  
 - 51 discarded due to inconsistent answers  
 - 332 discarded that none of the versions can solve  
 - speed-up measured on >100s bracket: 286 models

# Gurobi 12 Performance

problem class	overall (>1 sec)	hard models (>100 sec)
LP (default)	2.6%	0.9%
LP (barrier)	2.2%	4.8%
LP (dual simplex)	4.4%	3.6%
LP (primal simplex)	2.6%	2.0%
QP	9.1%	—*
SOCP	37.3%	—*
MIP	13.1%	18.9%
MIQP	13.0%	38.3%
MIQCP	4.1%	3.3%
nonconvex MIQCP	27.7%	68.5%
IIS	22.7%	37.3%

\* too few “hard” models



# IIS Improvements

## Disconnected Components

- MIP IIS computation fundamental operation
  - Remove a set of constraints, solve the resulting sub MIP
    - If infeasible, discard the removed constraints; remaining problem is also infeasible
    - Otherwise restore the removed constraints, remove another set of constraints
  - Numerous possible IIS search strategies for choosing constraints to remove
    - Disconnected components can help identify good sets of constraints to remove
- Model

$$\begin{pmatrix} A_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_k \end{pmatrix} \begin{pmatrix} x^1 \\ \vdots \\ x^k \end{pmatrix} \leq \begin{pmatrix} b^1 \\ \vdots \\ b^k \end{pmatrix}$$

is infeasible if and only if at least one of

$$A_i x^i \leq b^i$$

is infeasible

# IIS Improvements

## Disconnected Components

- Exploit disconnected components 8.1%
  - For models with disconnected components, find one infeasible component
  - Try smallest component first
  - Apply node limit for component solves
    - Bad case: difficult feasible small component, trivially infeasible larger component
  - Discard other components from infeasible subsystem
- Produce disconnected components earlier 9.5%
  - Change IIS search strategy to produce disconnected components earlier
  - Find separator in row intersection graph
  - Try removing all rows in separator at once
    - If this is infeasible: we got a disconnected infeasible subsystem

$$\begin{pmatrix} A_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_k \end{pmatrix} \begin{pmatrix} x^1 \\ \vdots \\ x^k \end{pmatrix} \leq \begin{pmatrix} b^1 \\ \vdots \\ b^k \end{pmatrix}$$

# IIS Improvements

## Exploiting Neighborhoods

- Grow neighborhood of “definite” IIS rows
  - IIS algorithm identifies “definite” IIS rows:
    - Rows that need to be member of infeasible subsystem
  - Grow neighborhood in row intersection graph until certain size
  - Try removing everything else
    - If this is infeasible: we got a smaller infeasible subsystem
  - Will often automatically zoom in on small component

3.8%



# Global MINLP



# Solving MINLPs to Global Optimality

## Required

- Outer approximation
  - To construct LP relaxation
- Adaptive constraints
  - To tighten LP relaxation based on local bounds of search tree nodes
- Spatial branch-and-bound
  - To resolve nonlinear constraint violations

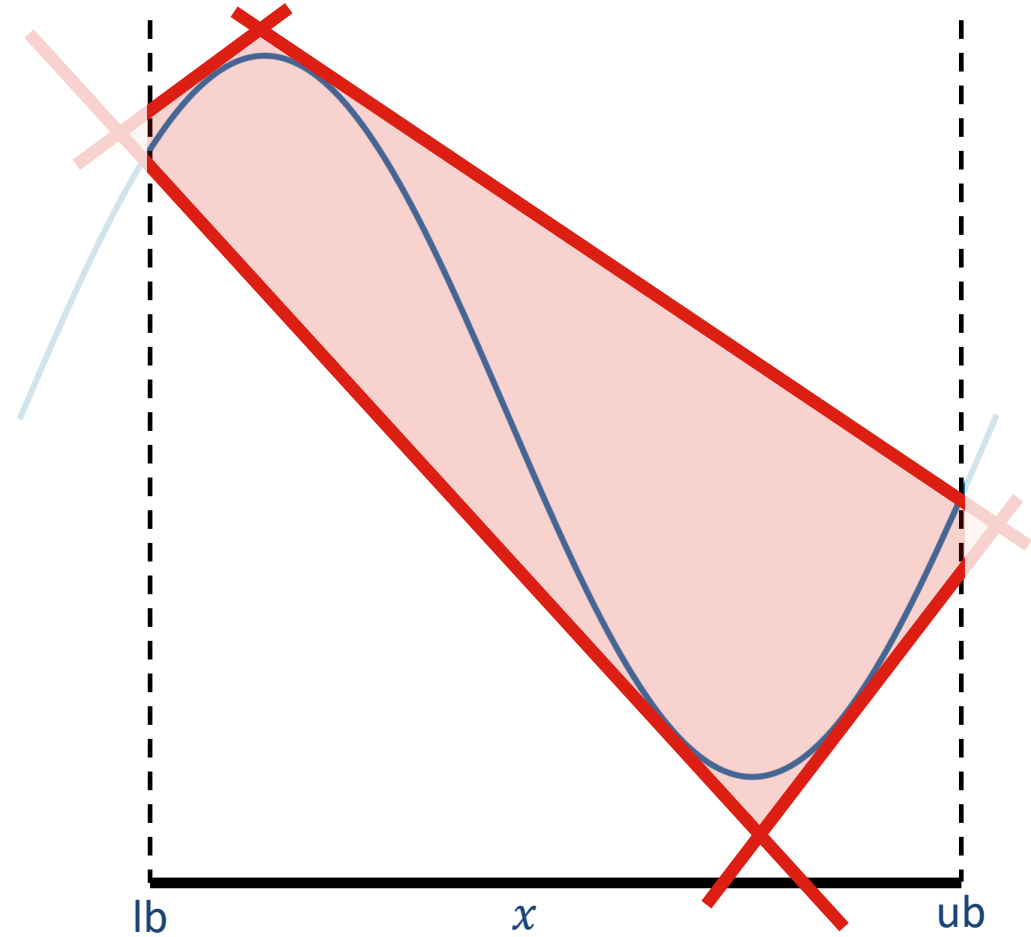
## Optional

- Presolve and OBBT
  - To tighten domains of variables and improve problem formulation
- Cutting planes
  - To tighten the LP relaxation
- Node presolve
  - To tighten local bounds of variables at search tree nodes
- Primal heuristics
  - To help finding feasible solutions
  - In particular: interior point algorithm to get locally optimal solutions for continuous NLPs



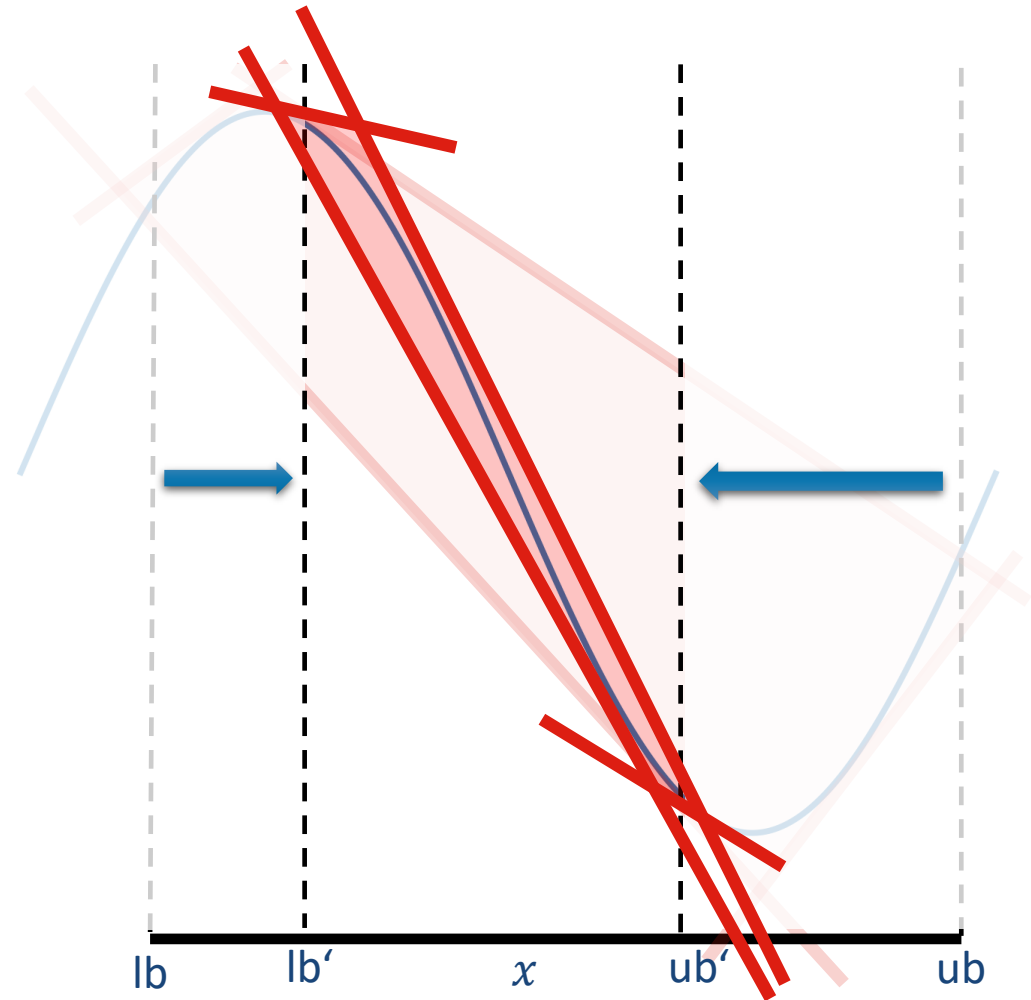
# Dynamic Outer Approximation

- Linear relaxation of convex hull of nonlinear constraint
- Based on global bounds of variables
- Use small number of tangents and secants



# Dynamic Outer Approximation

- Tighter bounds (at local search tree nodes)
- Yield tighter outer approximation
- Automatically calculated using “adaptive constraints”
  - Change coefficients and rhs values in LP relaxation on the fly



# Nonlinear Constraints

- Gurobi 9.0 and later provide API to define univariate general function constraints

- $y = e^x, y = a^x$

```
addGenConstrExp(), addGenConstrExpA()
```

- $y = \ln(x), y = \log_a(x)$

```
addGenConstrLog(), addGenConstrLogA()
```

- $y = \sin(x), y = \cos(x), y = \tan(x)$

```
addGenConstrSin(), addGenConstrCos(), addGenConstrTan()
```

- $y = x^a$

```
addGenConstrPow()
```

- $y = ax^3 + bx^2 + cx + d$

```
addGenConstrPoly()
```

- Gurobi 9.0 – 10.0:

- Function constraints are replaced during presolve by a piecewise-linear **approximation**

- Gurobi 11.0:

- Can choose to treat function constraints **exactly** by a dynamic outer approximation

- Gurobi 12.0:

- Supports **general nonlinear constraints** by a dynamic outer approximation
    - Multivariate compound nonlinear constraints

# General Nonlinear Constraints

- Gurobi supports selected univariate function constraints  $y = f(x)$ 
  - Trigonometric, power functions, logarithms, exponentials, etc.
  - Can be used as building blocks for general (multivariate) nonlinear constraints
- Example: Suppose we want to model

$$\begin{array}{c}
 z = \ln w \\
 \underbrace{\hspace{10em}} \\
 v = \sqrt{u} \qquad \qquad \qquad w = x + v \\
 \underbrace{\hspace{10em}} \\
 f(x) = \underbrace{\sqrt{1 + x^2}}_u + \ln(\underbrace{x + \sqrt{1 + x^2}}_w) \leq 2, \quad x \geq 0
 \end{array}$$

- We introduce auxiliary variables  $u, v, w, z \geq 0$  and constraints as follows:
  - $u = 1 + x^2, u = v^2, w = x + v, z = \ln w$
  - Then  $f(x) \leq 2$  can be represented as  $v + z \leq 2$

# Decomposition and Feasibility Tolerance

$$\begin{aligned}u - x^2 &= 1 \\v - \sqrt{u} &= 0 \\w - x - v &= 0 \\z - \ln w &= 0 \\v + z &\leq 2\end{aligned}$$



$$\sqrt{1 + x^2} + \ln(x + \sqrt{1 + x^2}) \leq 2$$



# Decomposition and Feasibility Tolerance

- Each individual constraint of decomposition is subject to feasibility tolerance
- Result could be that general nonlinear constraint is violated much more

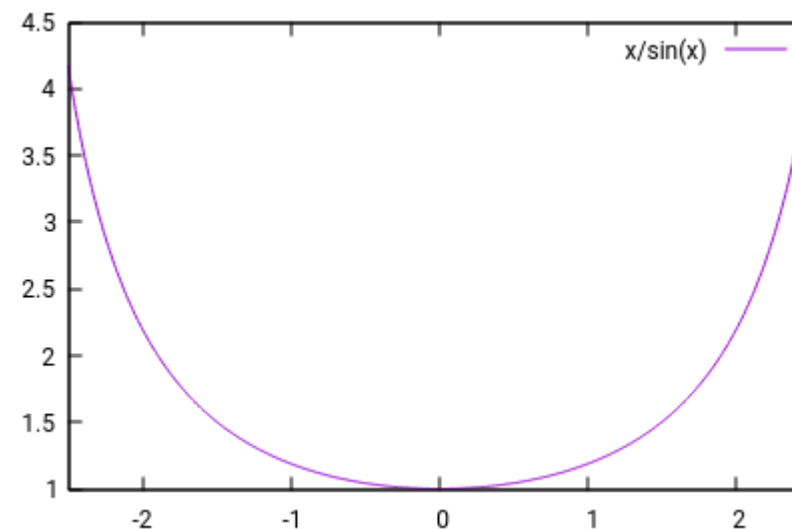
$$\begin{aligned}
 u - x^2 &= 1 \quad (\pm\varepsilon) \\
 v - \sqrt{u} &= 0 \quad (\pm\varepsilon) \\
 w - x - v &= 0 \quad (\pm\varepsilon) \\
 z - \ln w &= 0 \quad (\pm\varepsilon) \\
 v + z &\leq 2 \quad (+\varepsilon)
 \end{aligned}$$



$$\sqrt{1 + x^2} + \ln(x + \sqrt{1 + x^2}) \leq 2 \quad (+\varepsilon)$$

# Decomposition and Feasibility Tolerance

- Example:  $y = \frac{x}{\sin x}$
- One solution:
  - $x' = 0.0001$
  - $y' = 1.0000000016666666$
- Decomposition:  $u = \sin x, v = u^{-1}, y = x \cdot v$
- One solution:
  - $x' = 0.0001$
  - $u' = 0.000099999999833333343$
  - $v' = 10000.000016666666$
  - $y' = 1.0000000016666666$



- A solution with a violation within a tolerance of  $10^{-6}$ :
  - $x'' = 0.0001$
  - $u'' = 0.000098999999833333343 \quad u'' = \sin x'' - 10^{-6}$
  - $v'' = 10101.010118015167$
  - $y'' = 1.0101010118015167$

Violation of  $10^{-6}$  in auxiliary constraint leads to violation of  $10^{-2}$  in composite constraint

Compound constraint behaves very nicely for  $x \in [-2.5, 2.5]$ : no numerical issues expected

# Decomposition and Feasibility Tolerance

- Each individual constraint of decomposition is subject to feasibility tolerance
- Result could be that general nonlinear constraint is violated much more

**GUROBI 11**

$$\begin{aligned}
 u - x^2 &= 1 \quad (\pm\varepsilon) \\
 v - \sqrt{u} &= 0 \quad (\pm\varepsilon) \\
 w - x - v &= 0 \quad (\pm\varepsilon) \\
 z - \ln w &= 0 \quad (\pm\varepsilon) \\
 v + z &\leq 2 \quad (+\varepsilon)
 \end{aligned}$$



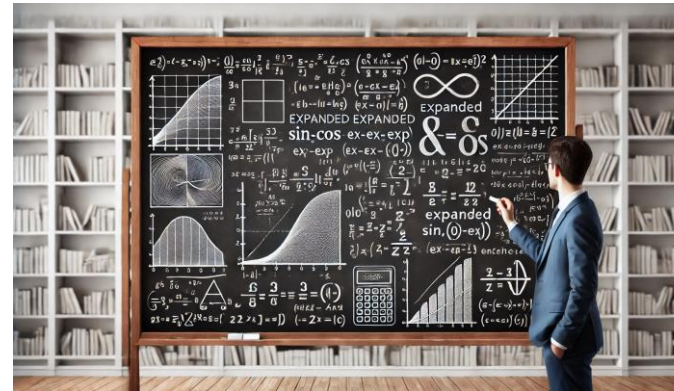
**GUROBI 12**

$$\sqrt{1 + x^2} + \ln(x + \sqrt{1 + x^2}) \leq 2 \quad (+\varepsilon)$$

- If user performed the decomposition manually (required in **Gurobi 11**)
  - No way for Gurobi to know that an underlying compound constraint exists
  - Solution will be feasible (within tolerances) only for decomposed model
- If user specifies general nonlinear constraint directly (available in **Gurobi 12**)
  - Gurobi can check the violation of the compound constraint and reject violated solutions

# Nonlinear Constraints APIs for Gurobi 12

- Preferred Gurobi API: gurobipy
  - Matrix-friendly algebraic modeling
- Third-party modeling tools
  - AMPL
  - GAMS
  - JuMP
  - Frontline
  - Additional vendors may add support for Gurobi 12 nonlinear constraints later
- Low-level array-based Gurobi APIs
  - C
  - C++
  - Java
  - .NET
  - LP and MPS files



# Nonlinear Expressions in gurobipy

- There is a new NLEExpr object in gurobipy

```
x = model.addVar(name="x")
y = model.addVar(name="y")
z = model.addVar(name="z")
```

```
expr1 = 2.0 * x      # LinExpr
expr2 = x * y       # QuadExpr
expr3 = x * y * z   # NLEExpr
expr4 = x / y       # NLEExpr
```

```
model.addGenConstrNL(z, x / y)      # Constraint z = x / y
model.addConstr(z == x / y)         # Constraint z = x / y
model.addConstr(z <= x / y)         # Not possible
model.addConstr(2.0 * z == x ** 5)  # Not possible
```

Use  $z = x/y - s, s \geq 0$   
instead

Use  $z = (x**5)/2$   
instead

# gurobipy Example: AC Optimal Power Flow

## Input Data and Vector Variables



```
import numpy as np

# Number of Buses (Nodes)
N = 4

# Conductance/susceptance components
G = np.array([[ 1.7647, -0.5882,  0.      , -1.1765 ],
              [-0.5882,  1.5611, -0.3846, -0.5882 ],
              [ 0.      , -0.3846,  1.5611, -1.1765 ],
              [-1.1765, -0.5882, -1.1765,  2.9412 ]])
B = np.array([[ -7.0588,  2.3529,  0.      ,  4.7059 ],
              [  2.3529, -6.629 ,  1.9231,  2.3529 ],
              [  0.      ,  1.9231, -6.629 ,  4.7059 ],
              [  4.7059,  2.3529,  4.7059, -11.7647 ]])

# Assign bounds where fixings are needed
v_lb = np.array([1.0, 0.0, 1.0, 0.0])
v_ub = np.array([1.0, 1.5, 1.0, 1.5])
P_lb = np.array([-3.0, -0.3, 0.3, -0.2])
P_ub = np.array([3.0, -0.3, 0.3, -0.2])
Q_lb = np.array([-3.0, -0.2, -3.0, -0.15])
Q_ub = np.array([3.0, -0.2, 3.0, -0.15])
theta_lb = np.array([0.0, -np.pi/2, -np.pi/2, -np.pi/2])
theta_ub = np.array([0.0, np.pi/2, np.pi/2, np.pi/2])
```

```
import gurobipy as gp
from gurobipy import GRB
from gurobipy import nlfunc

env = gp.Env()
model = gp.Model("OptimalPowerFlow", env=env)

# real power for buses
P = model.addMVar(N, name="P", lb=P_lb, ub=P_ub)

# reactive for buses
Q = model.addMVar(N, name="Q", lb=Q_lb, ub=Q_ub)

# voltage magnitude at buses
v = model.addMVar(N, name="v", lb=v_lb, ub=v_ub)

# voltage angle at buses
theta = model.addMVar(N, name="theta", lb=theta_lb,
ub=theta_ub).reshape(N, 1)

# Minimize Reactive Power at buses 1 and 3
model.setObjective(Q[[0, 2]].sum(), GRB.MINIMIZE)
```

# gurobipy Example: AC Optimal Power Flow

Nonlinear Matrix Constraints using Numpy Broadcasting

```
# Real power balance
#  $P_i = V_i \sum_{j=1}^N V_j (G_{ij} \cos(\theta_i - \theta_j) + B_{ij} \sin(\theta_i - \theta_j))$ 
constr_P = model.addGenConstrNL(
    P,
    v * (v @ (G * nlfunc.cos(theta - theta.T) + B * nlfunc.sin(theta - theta.T))),
    name="constr_P",
)

# Reactive power balance
#  $Q_i = V_i \sum_{j=1}^N V_j (G_{ij} \sin(\theta_i - \theta_j) - B_{ij} \cos(\theta_i - \theta_j))$ 
constr_Q = model.addGenConstrNL(
    Q,
    v * (v @ (G * nlfunc.sin(theta - theta.T) - B * nlfunc.cos(theta - theta.T))),
    name="constr_Q",
)

model.optimize()
```

```
G = np.array([[ 1.7647, -0.5882,  0.      , -1.1765 ],
              [-0.5882,  1.5611, -0.3846, -0.5882 ],
              [ 0.      , -0.3846,  1.5611, -1.1765 ],
              [-1.1765, -0.5882, -1.1765,  2.9412 ]])
B = np.array([[ -7.0588,  2.3529,  0.      ,  4.7059 ],
              [  2.3529, -6.629 ,  1.9231,  2.3529 ],
              [  0.      ,  1.9231, -6.629 ,  4.7059 ],
              [  4.7059,  2.3529,  4.7059, -11.7647 ]])
```



# gurobipy Example: AC Optimal Power Flow

Matrix Constraints using Numpy Broadcasting

```
# Real power balance
# P_i = V_i \sum_{j=1}^{N} V_j (G_{ij} \cos(\theta_i - \theta_j) + B_{ij} \sin(\theta_i - \theta_j))
constr_P = model.addGenConstrNL(
    P,
    v * (v @ (G * nlfunc.cos(theta - theta.T) + B * nlfunc.sin(theta - theta.T))),
    name="constr_P",
)

# Reactive power balance
# Q_i = V_i \sum_{j=1}^{N} V_j (B_{ij} \sin(\theta_i - \theta_j) - G_{ij} \cos(\theta_i - \theta_j))
constr_Q = model.addGenConstrNL(
    Q,
    v * (v @ (G * nlfunc.sin(theta - theta.T) - B * nlfunc.cos(theta - theta.T))),
    name="constr_Q",
)

model.optimize()
```

$$\begin{pmatrix} \cos(\theta[0] - \theta[0]) & \cdots & \cos(\theta[0] - \theta[3]) \\ \vdots & \ddots & \vdots \\ \cos(\theta[3] - \theta[0]) & \cdots & \cos(\theta[3] - \theta[3]) \end{pmatrix}$$

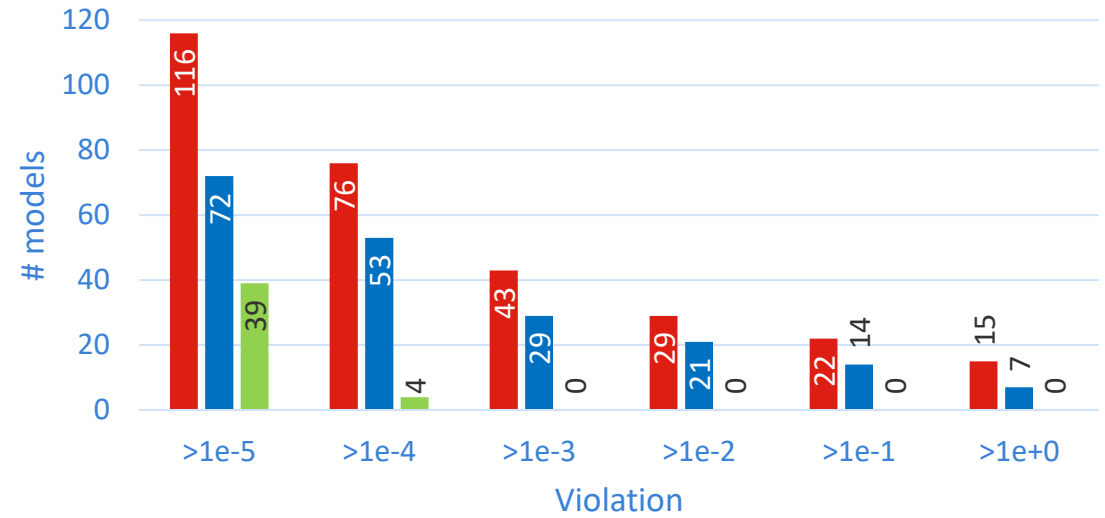
```
G = np.array([[ 1.7647, -0.5882, 0. , -1.1765 ],
              [-0.5882, 1.5611, -0.3846, -0.5882 ],
              [ 0. , -0.3846, 1.5611, -1.1765 ],
              [-1.1765, -0.5882, -1.1765, 2.9412 ]])
B = np.array([[ -7.0588, 2.3529, 0. , 4.7059 ],
              [ 2.3529, -6.629 , 1.9231, 2.3529 ],
              [ 0. , 1.9231, -6.629 , 4.7059 ],
              [ 4.7059, 2.3529, 4.7059, -11.7647 ]])
```

# Global MINLP

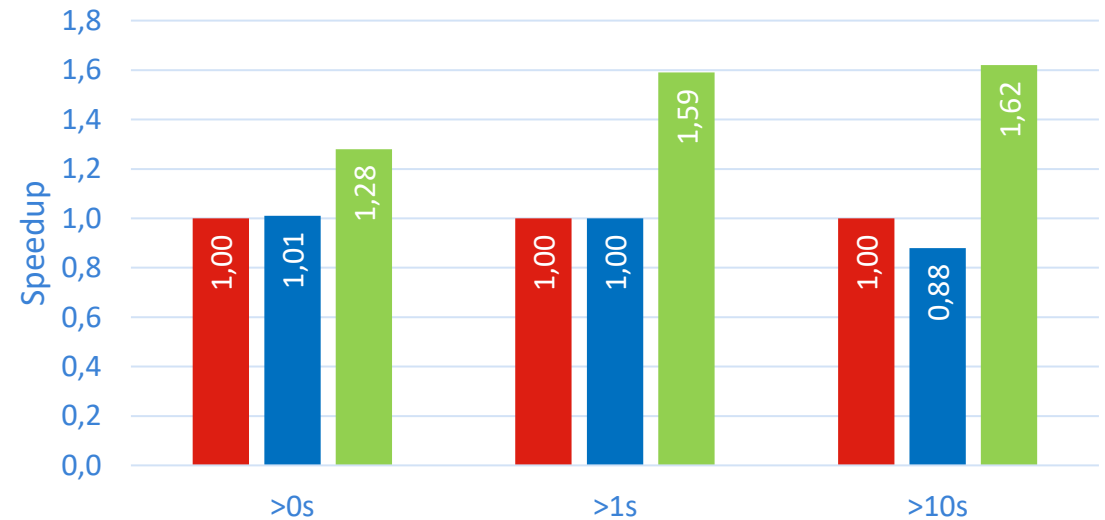
## Solution Quality and Performance

- Gurobi 11:
  - function constraints (univariate) models
- Gurobi 12:
  - function constraints (univariate) models
  - nonlinear constraints (multivariate) models

- Better solution quality: **v11** < **v12 univariate** < **v12 multivariate**



- Equally fast for univariate, faster for multivariate NL constraints



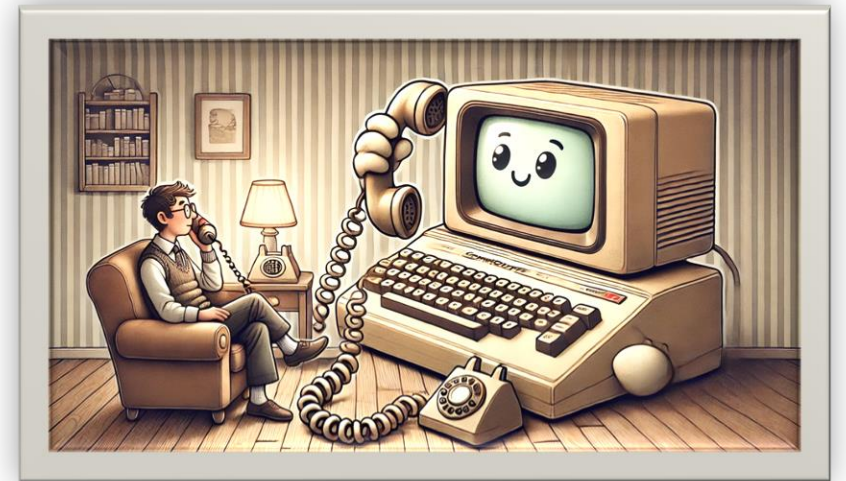


# Minor Features



# Callbacks

- Support for SetSolution from callbacks for Compute Server runs
  - Pass user heuristic solutions to solving process
  - E.g., to repair solution that violates lazy constraint
- Additional information in multi-objective callback
  - Solving status
  - runtime
  - work
  - iteration count
  - node count
  - number of nodes left
  - incumbent value
  - dual bound
  - MIP gap
  - Provides information about the sub-problem solve for the most recent objective function
- Callback-settable parameters
  - TimeLimit
  - WorkLimit
  - NodeLimit
  - BarIterLimit
  - PumpPasses
  - Simplify custom termination criteria



# Tuning Tool

- Restrict tuning to a set of parameters
  - E.g., only tune cutting planes or heuristics
- Tuning multi-objective models





## Additional Features

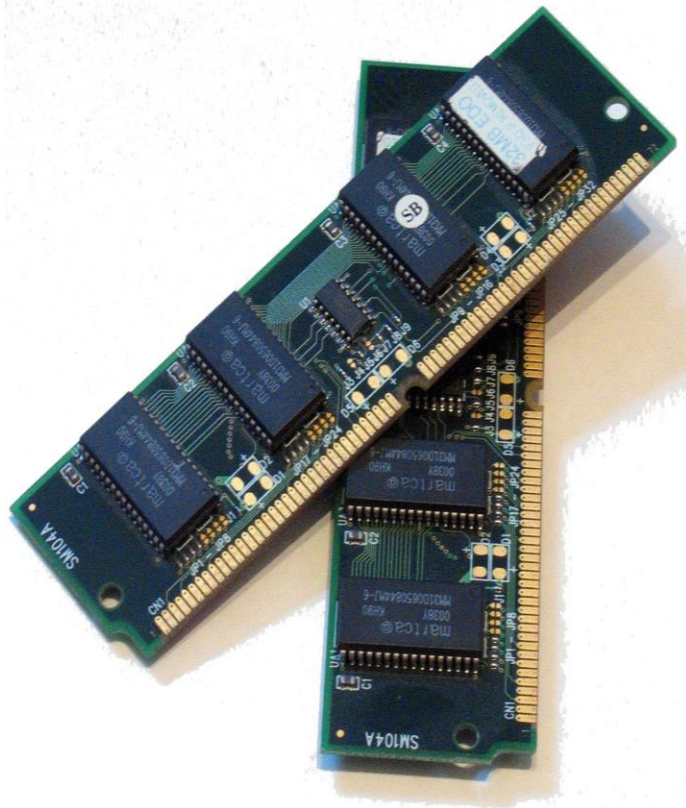
- “ObjBound” attribute for LP models
- “BarPi” attribute for LP models
- In-place conversion of models to fixed model
- Anonymizing data files with “IgnoreNames=1” parameter
- Support for XZ compressed files
- Logging improvements
  - Display path to Gurobi shared library
  - Display non-default parameter settings
- Asynchronous optimization in gurobipy



# Hardware Resource Management



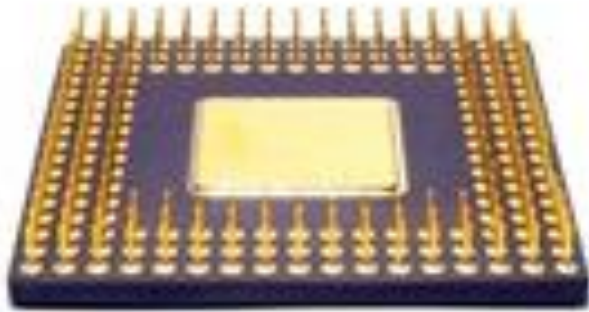




# Memory Management

- Fewer memory allocations
  - Avoids out-of-memory due to memory fragmentation
  - 0.5% performance improvement on hard MIP models
- Less memory consumption for large solution pools
  - Due to sparse storage of solution pool vectors
  - 1.4% performance improvement on hard MIP models
    - With PoolSearchMode=2 and PoolSolutions=1000
    - 46 mem limit hits → 35 time limit hits
- Attributes to query memory consumption
  - Current and maximum memory used
  - During (in callback) and after optimization





## CPU Utilization

- More root node parallelization
  - Run sub-MIP heuristics with multiple threads
  - In addition to parallel root cut loops
  - 0.4% performance improvement overall (4 cores)
- Better MIP search tree parallelization
  - Preempt long running heuristics and node LP solves
  - Continue heuristics and node LP solves after MIP synchronization
  - 1.4% performance improvement overall (4 cores)



picture by [Torkild Retvedt](https://www.flickr.com/photos/torkildr/3462607995/in/photostream/)  
<https://www.flickr.com/photos/torkildr/3462607995/in/photostream/>




# Compute Server Threads

- Thread-based load balancing
  - Compute Server
    - Limit total number of threads running at same time
  - Cluster Manager
    - Assign jobs to servers based on thread load

# Thread-based Load Balancing

- Improves the distribution of threads across cluster nodes to ensure that concurrent jobs use available computational resources efficiently without overloading any node.
- Remote server configuration properties
  - NODE\_THREADLIMIT: A limit on the total number of threads allowed to run concurrently on the server simultaneously.
  - FIXED\_NODE\_THREADLIMIT: Indicates whether the node thread limit can be changed once a node has started.
- ThreadLimit gives upper bound on Threads parameter

## Cluster Manager

ADDRESS	JOB LIMIT	GROUP	IDLE TIME	%MEMORY	%CPU	THREAD LIMIT	THREAD RESERVATION
 192.168.10.181:61000	2	grp2	0millis	3.1%	100%	16	16
 192.168.31.163:61000	1	grp1	138millis	2.5%	0.3%	N/A	N/A
 192.168.56.175:61000	2	grp1	0millis	2.8%	50.9%	16	8

## Batch

```
→ data grbcluster batch solve glass4.mps --app batchapp --thread-limit 16
info : Batch 90d3f7a2-8a9a-43cb-8750-570f8880c586 created
```

## Client

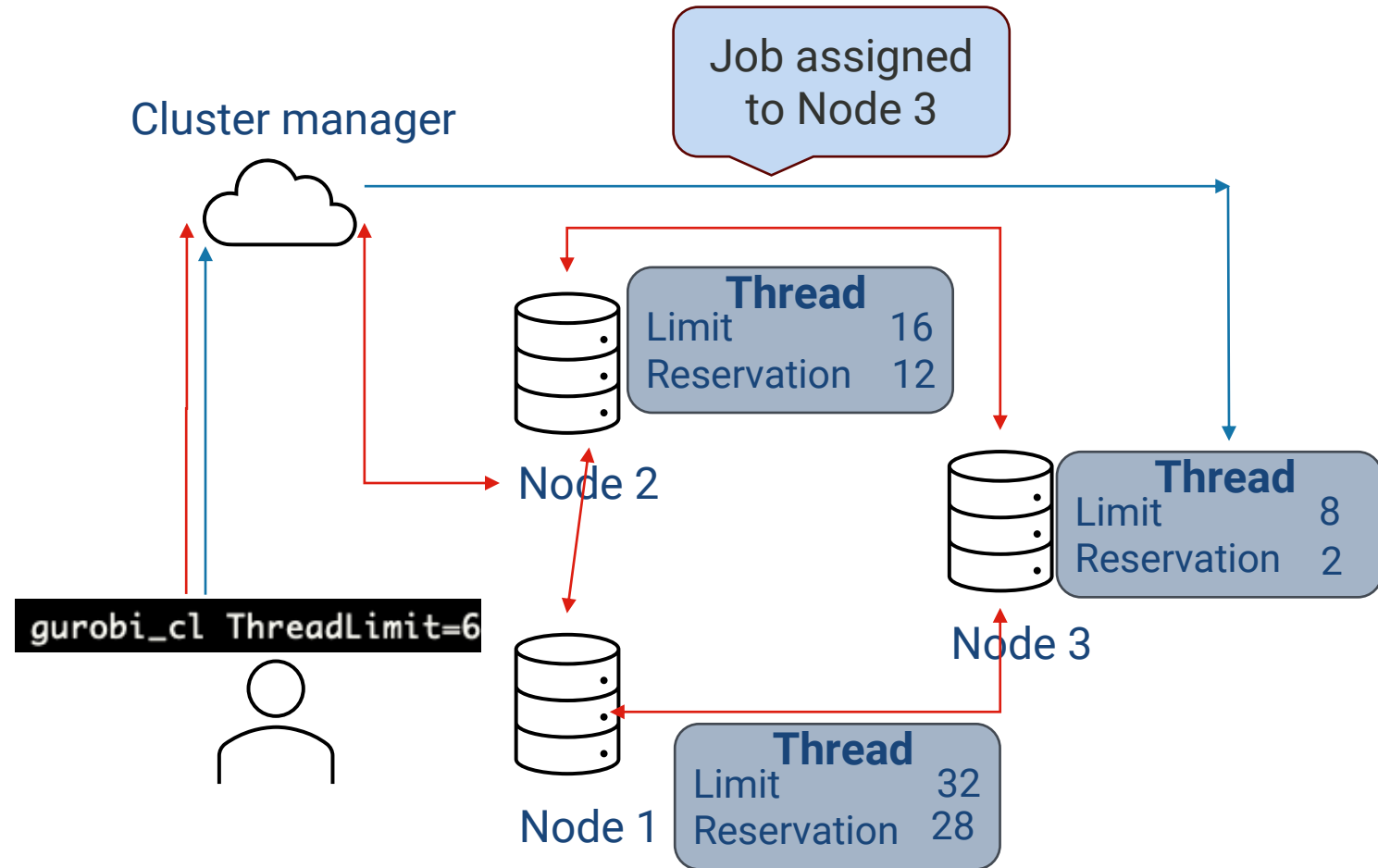
```
→ data gurobi_cl ThreadLimit=3 glass4.mps
Set parameter CSManger to value "https://cluster9.gurobi.com"
```

## Python

```
2 self.env.setParam("ThreadLimit", 2)
3 self.env.start()
```

# Thread-based Load Balancing

- Load Balancing Algorithm
- Places jobs where threads are most available.
- Prioritizes nodes with fewer running jobs.
- Jobs exceeding all nodes' thread limits are rejected.



# Job Interruption

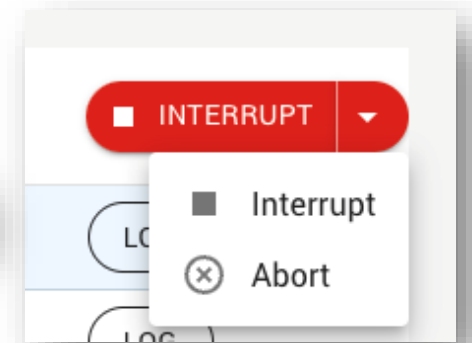
- Administrators can interrupt running jobs via the Web UI or the

```
grbcluster job interrupt <JOBID>
```

command.

- Behavior on Interruption:
  - Optimization stops gracefully, as if a normal stopping condition (e.g., time limit) was reached.
  - The job remains active, awaiting further instructions.
- Compatibility: Available in runtimes 12.0.0+. Ignored in earlier versions.

```
grbcluster job interrupt f53430ad
```



## Cluster jobs (19)

<input type="checkbox"/>	STARTED AT ↓ (CST)	USERNAME	OPTIMIZATION STATUS	VERSION
<input type="checkbox"/>	Dec 6, 2024 10:45 AM	sysadmin	INTERRUPTED	12.0.0
<input type="checkbox"/>	Dec 6, 2024 10:44 AM	sysadmin	INTERRUPTED	12.0.0



**GUROBI**  
OPTIMIZATION

Questions?