
GUROBI
OPTIMIZATION
What's New in
Gurobi 13.0
Where Speed Meets Innovation

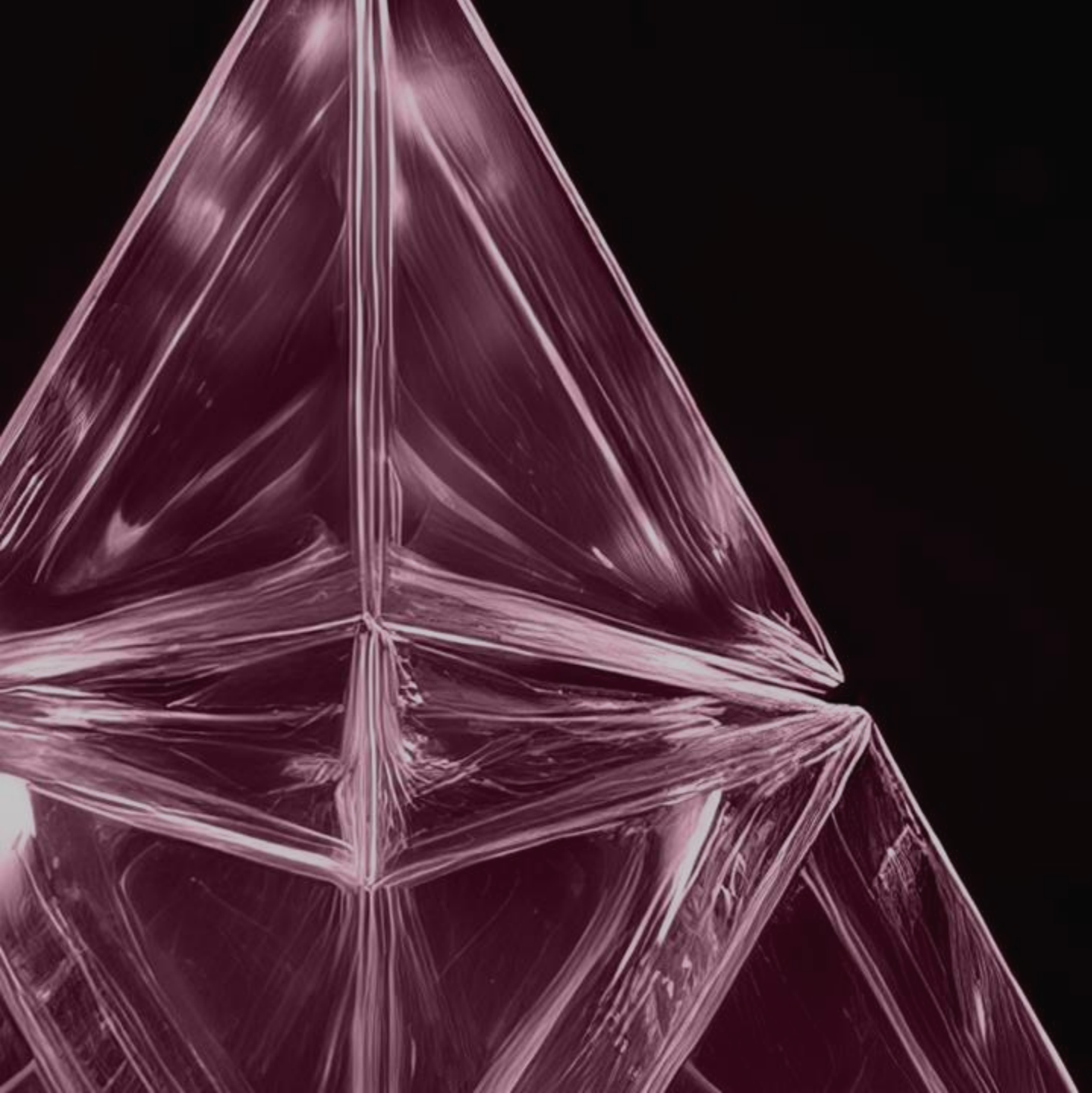
Agenda

Performance Improvements

Huge LPs, PDHG & GPU

Nonlinear Optimization

Platform Improvements



Gurobi 13: Performance

GUROBI OPTIMIZATION

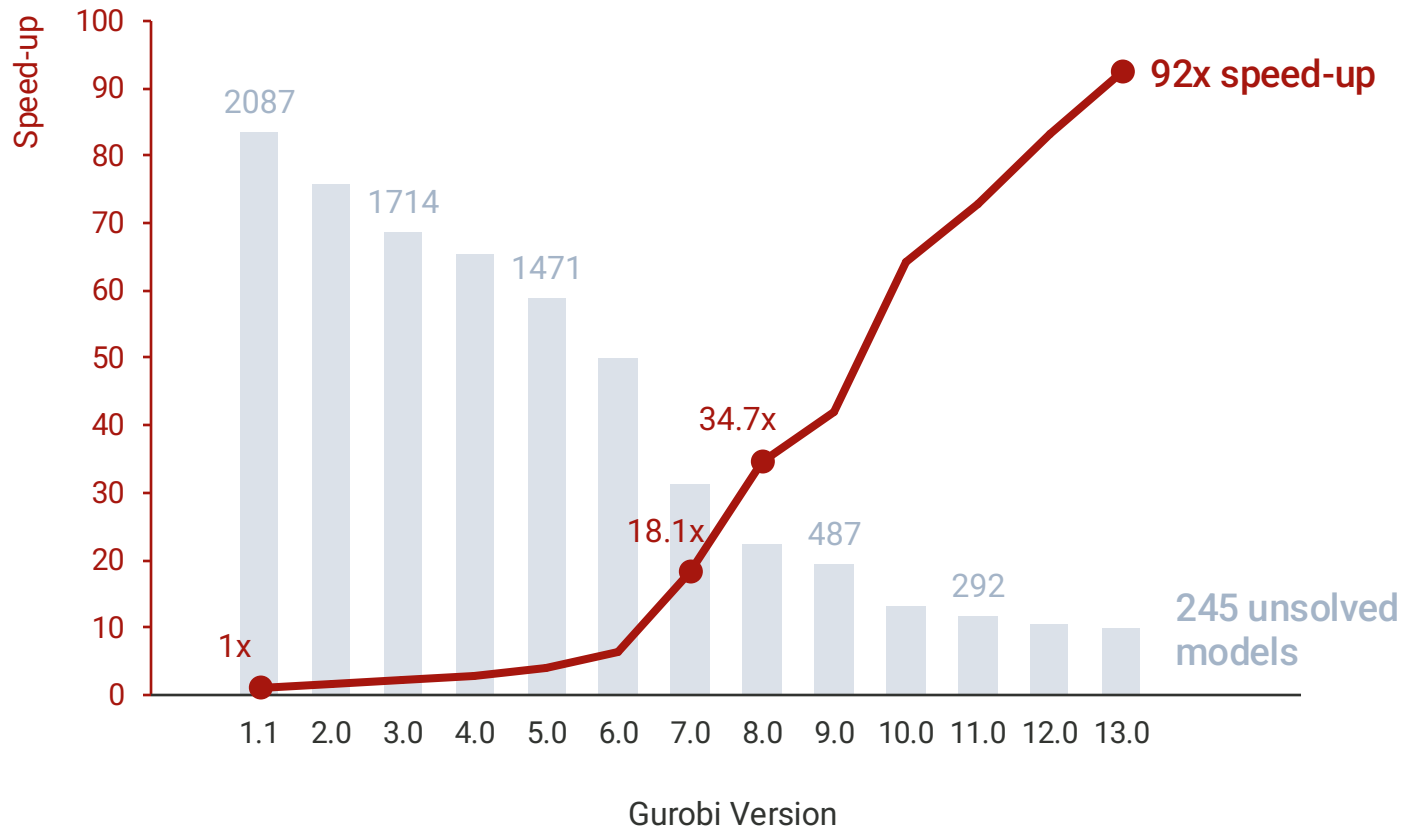
Gurobi 12 vs. 13 Comparison

ALGORITHM	SPEED-UP *Overall (>1 sec)*	SPEED-UP *On Hard Models (>100 sec)*
LP	0.6%	3.9%
MIP	8.2%	15.9%
MIQP	5.1%	7.3%
MIQCP	13.2%	25.8%
Nonconvex MIQCP	54.7%	2.68x
MINLP	2.52x	6.34x*
IIS	67.8%	2.01x*

* too few “hard” models

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 **92x** speed-up over version 1.1 in geometric mean (PAR-10) of runtimes
- Only **245** 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 9423 models:
- 960 discarded due to inconsistent answers
- 2641 discarded that none of the versions can solve
- speed-up measured on >100s bracket: 3517 models

MIP Improvements

- Presolve
  - Aggregator
    - Numerical improvements
    - Different approach based on LU factorization
    - Aggressive equality aggregation
  - Implied free variable based non-zeros removal
- Cutting planes
  - Master knapsack cuts added
  - More aggressive Gomory cuts
  - Improved MIR cuts
- Disconnected component improvement
  - Restructure code to reduce memory footprint
  - Work with partial or no solution
  - Enable presolve for component solves
- Propagation improvements
  - Avoid prop. of long non-binary parts
- Use infeasible vector as starting point for improvement heuristics
- NoRel heuristic improvements
- On-Off constraint detection
  - New heuristic
  - OBBT to improve bounds
- Inactive constraint detection
- Simplex
  - Primal ratio test bugfix
  - Dual ratio test improvements
  - Better bound propagation in presolve
- General MIP Framework improvements
  - Improved degenerate moves code
  - Improved parallelization

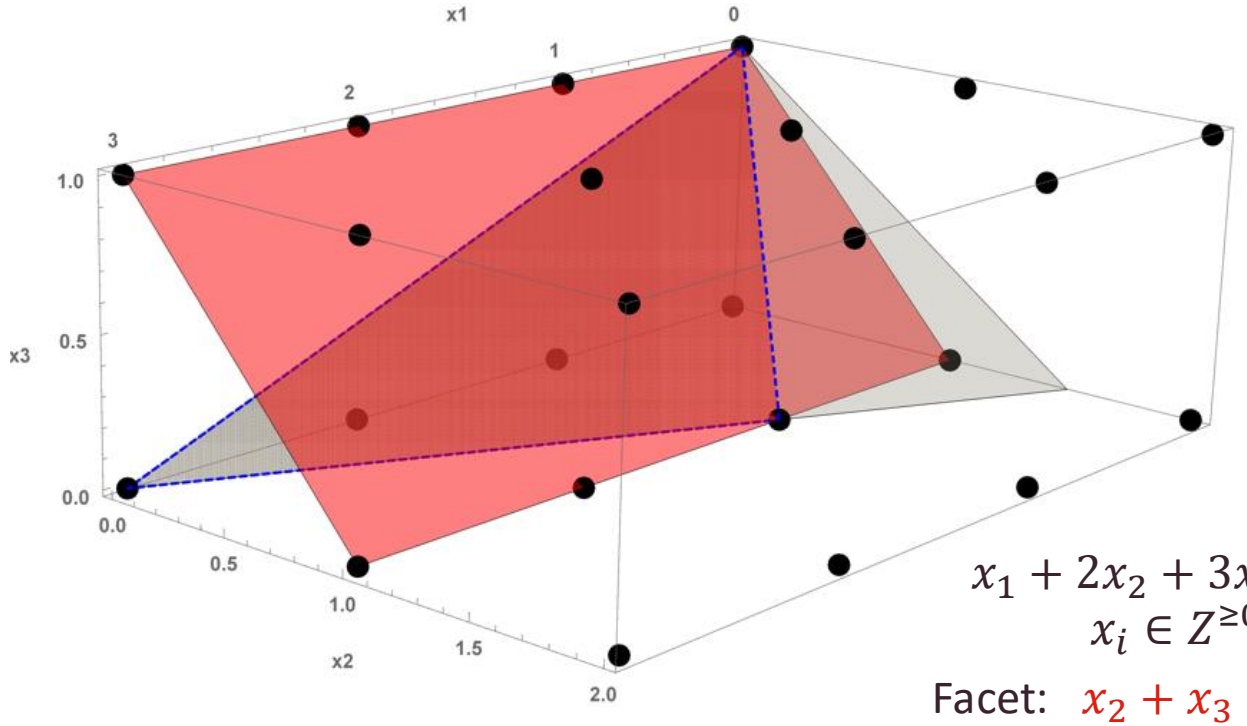
GUROBI OPTIMIZATION

Master Knapsack Cuts

- A class of strong facets of the master knapsack polytope
- Separated from single-row relaxations of the form

$$\sum_{i=1}^{n} ix_i = n\,, x_i \in Z, x_i \geq 0$$

- The cuts have a simple structure, with low-denominator rational coefficients
- Improvement of 2.5% on affected models (~10% on >100 sec models)



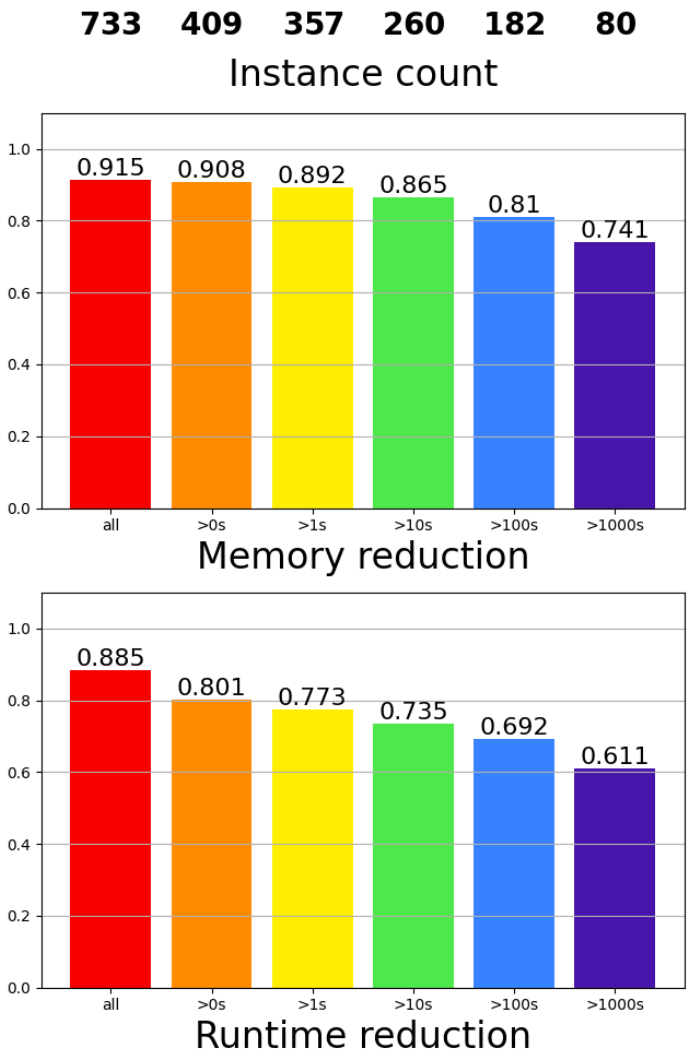
$$x_1 + 2x_2 + 3x_3 = 3$$
$$x_i \in Z^{\geq 0}$$

Facet: $x_2 + x_3 \leq 1$

Example in higher dimension

$$x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 + 7x_7 + 8x_8 = 8, \quad x_i \in Z^{\geq 0}$$

Facet: $\frac{1}{4}x_2 + \frac{1}{4}x_3 + \frac{1}{2}x_4 + \frac{3}{4}x_5 + \frac{3}{4}x_6 + x_7 + x_8 \leq 1$



Disconnected Components

Reworked disconnected component approach

- To reduce memory footprint
- Be more numerical robust
- Only 2.7% of our MIP instances are affected

Performance for a selected customer

- Memory footprint: 19% reduction
- Runtime: 30% reduction

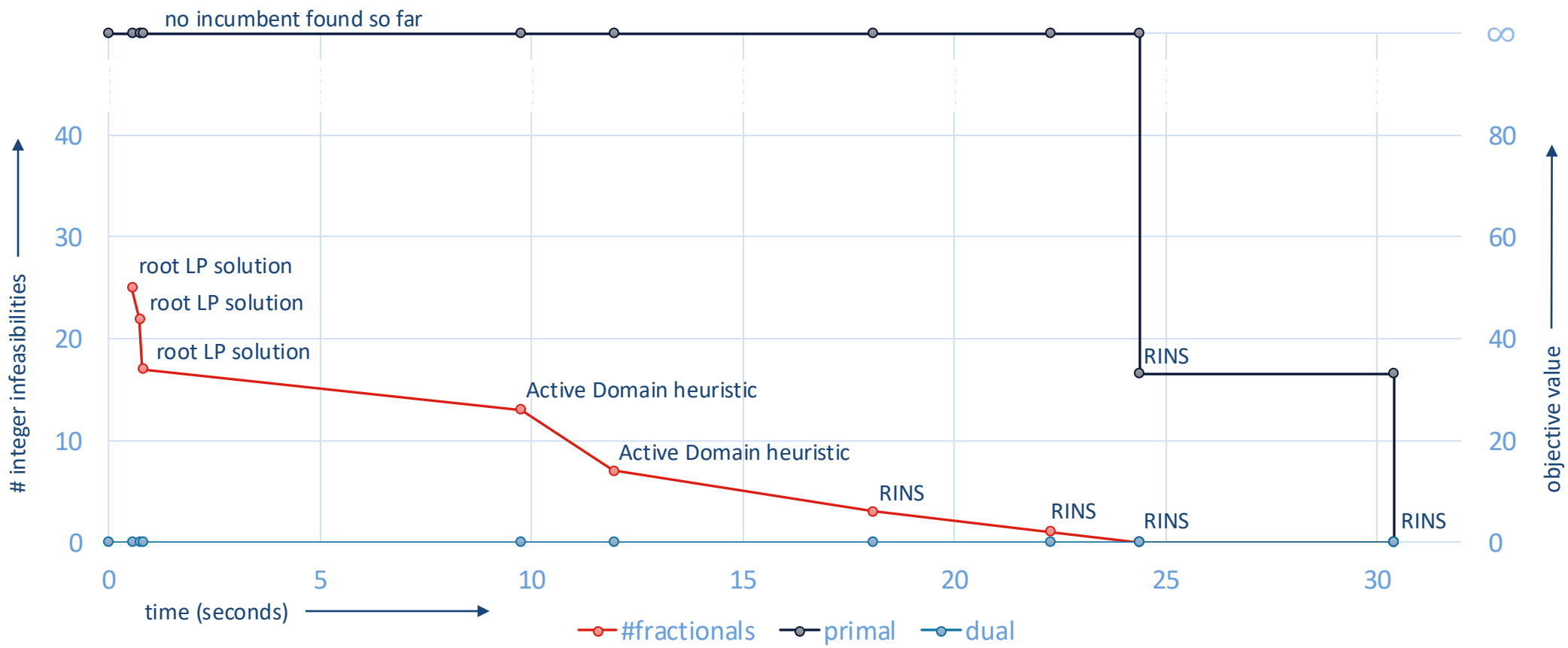
GUROBI SUMMIT

Infeasible Solutions for Improvement Heuristics

GUROBI OPTIMIZATION

- Improvement heuristics like RINS need incumbent solution to work on
- Idea: as long as no incumbent is available, use infeasible integer solution
  - rounded root cut loop LP solution
  - heuristic solutions that were rejected due to numerical issues
  - original model solutions (e.g., MIP start) that were cut off by dual presolve
  - any such infeasible solution in a sub-MIP solve of a heuristic
  - infeasible MIP starts provided by user
- Use infeasible solutions for
  - improvement heuristics like RINS
  - NoRel heuristic
  - Partition heuristic
- Performance (on models that take at least 100 seconds to solve)
  - 5% speed-up for time to proven optimality
  - 11% speed-up for time to first feasible solution

neos-5074751-dziwna



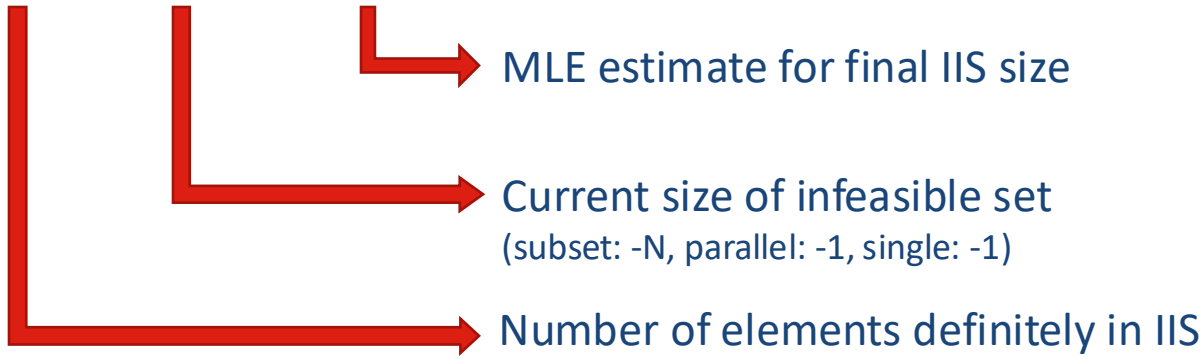
© 2025 Gurobi Optimization, LLC. Confidential. All Rights Reserved | 10

GUROBI OPTIMIZATION

IIS Performance

- IIS size guess since Gurobi 9.1
  - Initially, with simple estimate
- Using maximum likelihood estimate (MLE) since Gurobi 10.0
- New in Gurobi 13: use MLE to decide which IIS primitive to apply next
  - subset delete
    - fast, if likelihood is high that elements can be removed from IIS
  - parallel delete
    - fast, if likelihood is small that elements can be removed from IIS
  - single delete
    - slow, but always successful
- Performance v12 vs. v13: +67.8% overall

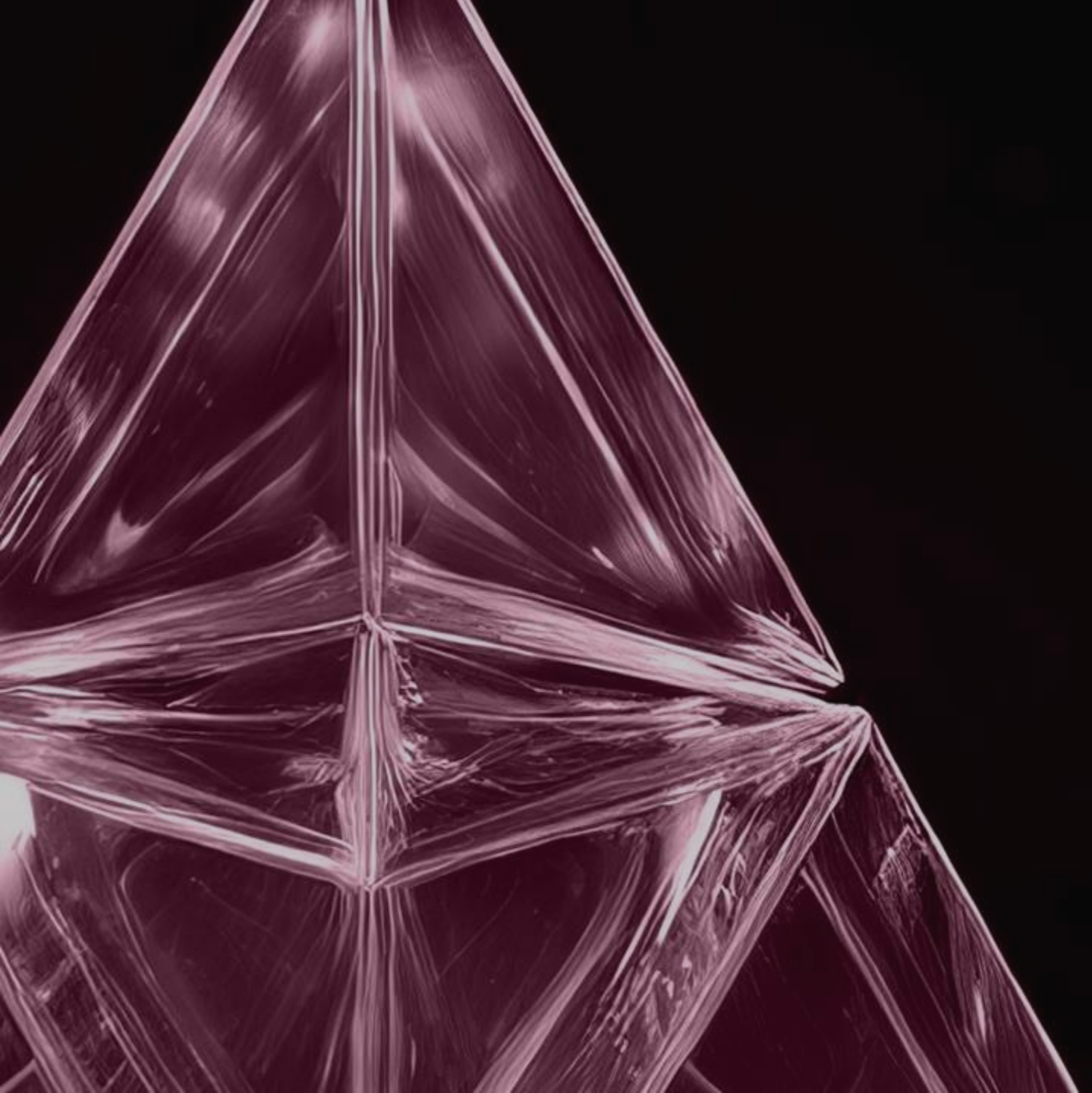
```
Computing Irreducible Inconsistent Subsystem (IIS)...

     Constraints     |      Bounds       | Runtime
 Min     Max   Guess | Min   Max   Guess |
-------------------------------------------------
   0   37698       -     0     0       0          0s
   0   18626       -     0     0       0         23s
   0   18606       -     0     0       0         25s
   0   15405      40     0     0       0         30s
   0   12458      49     0     0       0         35s
   0    8056      41     0     0       0         40s
   0    4014      46     0     0       0         45s
   2     168      44     0     0       0         50s
   5     145      45     0     0       0         55s
  14      67      43     0     0       0         60s
  30      43      43     0     0       0         65s
  43      43      43     0     0       0         67s
```



MLE estimate for final IIS size

Current size of infeasible set
(subset: -N, parallel: -1, single: -1)

Number of elements definitely in IIS
(subset: +0, parallel: +N, single: +1)



Huge LPs, PDHG & GPU

GUROBI OPTIMIZATION

GUROBI OPTIMIZATION

A new horse in the race

- Established LP algorithms
  - Primal Simplex, dual Simplex
  - Barrier with various extensions
- First order methods
  - Have gotten traction in past years
  - Boost in visibilty and attraction due to GPUs

PDHG - A First Order Method

A quick intro

Primal

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Dual

$$\begin{aligned} \max \quad & b^T y \\ \text{s.t.} \quad & A^T y \leq c \end{aligned}$$

These problems can be solved by solving the so-called Saddle-point problem

$$\min_x \max_y L(x, y) = c^T x - y^T A x + b^T y$$

- Chambolle, A. and Pock, T., 2011. A first-order primal-dual algorithm for convex problems with applications to imaging. Journal of mathematical imaging and vision, 40, pp.120-145.

PDHG - A First Order Method

A quick intro

Saddle-point problem

$$\min_x \max_y L(x,y) = c^T x - y^T A x + b^T y$$

Can be solved using Primal-Dual Hybrid Gradient (PDHG) by iteratively computing x_k and y_k (provided $\tau\sigma < 1/\|A\|_2^2$)

$$\nabla_x L(x,y) = c - A^T y$$

$$x_{k+1} = proj_{R_+^n}(x_k - \tau(c - A^T y_k))$$

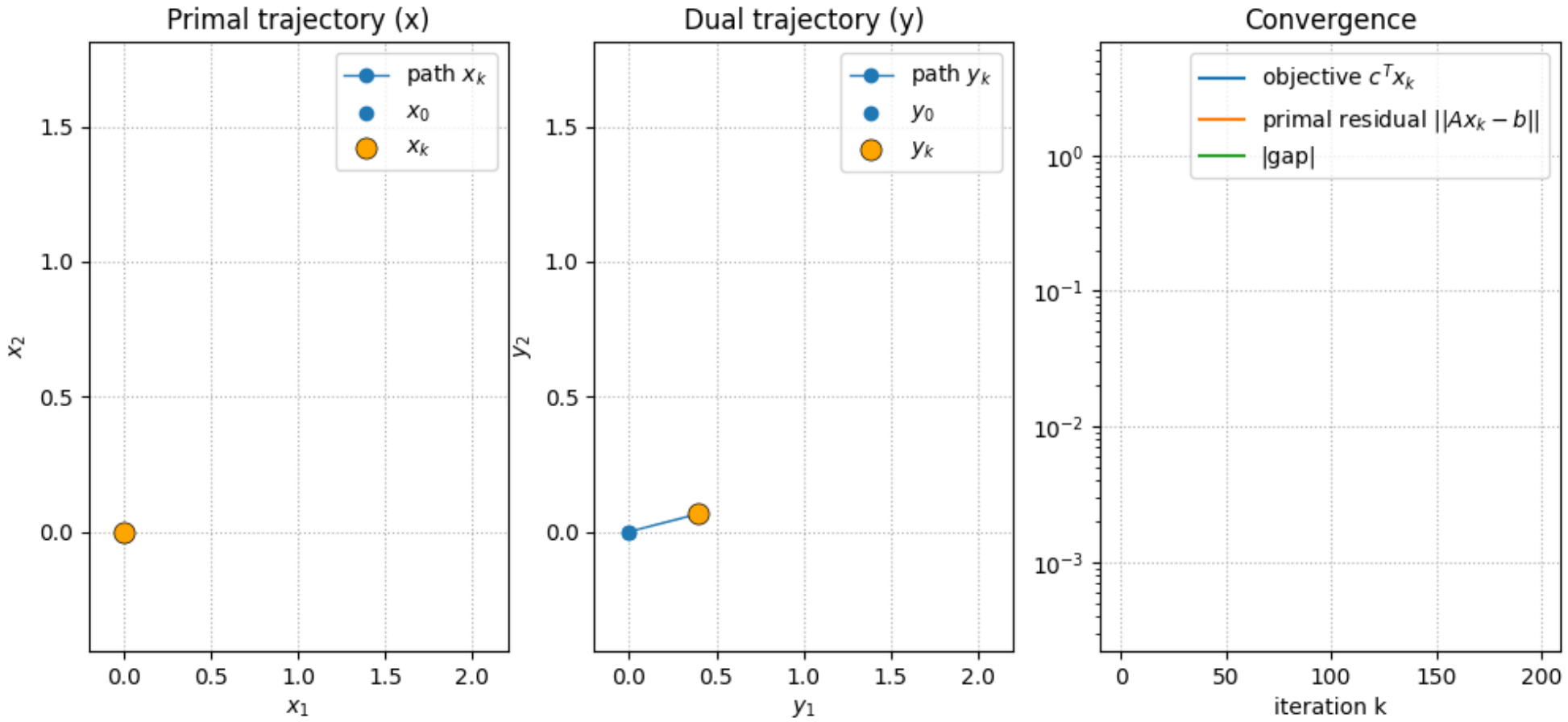
$$\nabla_y L(x,y) = b - Ax$$

$$y_{k+1} = y_k + \sigma(b - A(2x_{k+1} - x_k))$$

- The most expensive operations here are the two matrix-vector products $A^T y_k$ and Ax
- Matrix-vector products are memory-bound, scale very well on GPUs
- PDHG is therefore a good fit on the GPU

Typical behavior of PDHG

$$\begin{aligned} \min \quad & x_1 + 0.5x_2 \\ \text{s.t.} \quad & x_1 + 2x_2 = 3 \\ & x_1 - x_2 = 0.5 \end{aligned}$$



Improving PDHG for LPs

- Algorithmic components that allowed the PDHG algorithm to solve practical LPs (PDLP)
- These include:
  - Adaptive step size
  - Restarts (based on different criteria)
  - Primal weight update

- Applegate, D., Díaz, M., Hinder, O., Lu, H., Lubin, M., O'Donoghue, B. and Schudy, W., 2021. Practical large-scale linear programming using primal-dual hybrid gradient. Advances in Neural Information Processing Systems, 34, pp.20243-20257.
- Lu, H., Peng, Z. and Yang, J., 2025. cuPDLPx: A further enhanced GPU-based first-order solver for linear programming. arXiv preprint arXiv:2507.14051.

PDHG – Termination Criteria

Relative Termination

- Primal residual (controlled with new parameter `PDHGRelTol`)

$$\|b - Ax\|_2 \leq \epsilon_{rel}(1 + \|b\|_2)$$

- Dual residual (controlled with new parameter `PDHGRelTol`)

$$\|c - A^T y - \lambda\|_2 \leq \epsilon_{rel}\,(1 + \|c\|_2)$$

- Objective gap (controlled with new parameter `PDHGConvTol`)

$$|b^T y - c^T x| \leq \epsilon_{conv}\,(1 + |b^T y| + |c^T x|)$$

- **Relative** to problem data! This can lead to large violations. ⚠
  - Most PDHG solvers only use Relative Termination Criteria
  - Beware of solution quality when benchmarking PDLP solvers. ⚠

Absolute Termination

- Gurobi uses absolute tolerances for other algorithms
- Gurobi has `PDHGAbsTol` to control this criteria
- Typically reached through crossover, which also obtains basic solution.

Example Model: rwth-timetable

Defaults on GH200

```
Gurobi Optimizer version 13.0.0 build v13.0.0beta1 (armlinux64gpu)
Copyright (c) 2025, Gurobi Optimization, LLC

CPU model: ARM64
Thread count: 72 physical cores, 72 logical processors, using up to 32 threads

Optimize a model with 440134 rows, 923564 columns and 4510786 nonzeros (Min)
Model fingerprint: 0xf4eab31e
Model has 834949 linear objective coefficients
Coefficient statistics:
  Matrix range     [1e+00, 2e+00]
  Objective range  [1e+00, 4e+05]
  Bounds range     [1e+00, 1e+01]
  RHS range        [1e+00, 3e+00]
Presolve removed 206440 rows and 549304 columns
Presolve time: 3.14s
Presolved: 233694 rows, 374260 columns, 3675892 nonzeros

Concurrent LP optimizer: primal simplex, dual simplex, and barrier
Showing barrier log only...

...
Ordering time: 69.98s


Barrier statistics:
Dense cols : 385
AA' NZ     : 2.565e+07
Factor NZ  : 1.273e+09 (roughly 10.0 GB of memory)
Factor Ops : 1.885e+13 (roughly 35 seconds per iteration)
Threads    : 30
...
```

```
...

63   7.15260973e+05  7.15260972e+05  4.57e-09 7.28e-11  1.59e-09   855s

Barrier solved model in 63 iterations and 855.23 seconds
Optimal objective 7.15260973e+05

Crossover log...

   50685 DPushes remaining with DInf 9.9961285e-02               857s
   11303 DPushes remaining with DInf 4.4423273e-02               860s
       0 DPushes remaining with DInf 0.0000000e+00               864s

   32095 PPushes remaining with PInf 1.5185350e-06               864s
    4650 PPushes remaining with PInf 0.0000000e+00               865s
       0 PPushes remaining with PInf 1.5402888e+00               866s

  Push phase complete: Pinf 1.5402888e+00, Dinf 1.9161667e+01    866s

Iteration    Objective       Primal Inf.    Dual Inf.      Time
   74367    7.1526097e+05   0.000000e+00   1.916167e+01    866s
Crossover time: 11.23 seconds (18.50 work units)

Solved with barrier

Extra simplex iterations after uncrush: 19
   74434    7.1526097e+05   0.000000e+00   0.000000e+00    867s

Solved in 74434 iterations and 866.95 seconds (3307.94 work units)
Optimal objective  7.152609726e+05
```


Example Model: rwth-timetable

PDHG on CPU

```
Non-default parameters:
Method  6
...

Start PDHG using 32 threads

                     Objective                Residual
     Iter       Primal          Dual         Primal    Dual     Compl     Time
        0   2.43298400e+07 -1.18242230e+07  2.00e+00 0.00e+00  6.45e+01     3s
     2603   7.18700010e+05  7.16453294e+05  2.06e-02 6.16e+01  1.06e-02     5s
    10303   7.15956714e+05  7.15075108e+05  6.23e-03 6.31e-02  1.06e-03    10s
    17903   7.15652265e+05  7.15076646e+05  2.07e-03 1.26e-01  7.39e-04    15s
    25603   7.15111796e+05  7.15081045e+05  2.00e-03 1.60e-01  1.02e-04    20s
    33303   7.15052264e+05  7.15090353e+05  2.00e-03 4.06e-01  2.19e-05    25s
    41003   7.15052801e+05  7.15102002e+05  2.00e-03 4.06e-01  1.37e-05    30s
    48703   7.15094111e+05  7.15201781e+05  2.00e-03 3.23e+00  2.84e-05    35s
    56303   7.15261014e+05  7.15248962e+05  2.62e-07 1.19e+01  9.63e-04    40s
    64003   7.15261011e+05  7.15258972e+05  3.41e-08 2.81e+00  2.14e-04    45s
    70403   7.15261010e+05  7.15261265e+05  1.33e-07 5.20e-01  6.28e-05    49s

PDHG solved model in 70403 iterations and 49.21 seconds (60.59 work units)
Optimal objective 7.15261010e+05
...
```

```
Crossover log...

   60385 DPushes remaining with DInf 7.8870150e-02                49s
    3665 DPushes remaining with DInf 3.9477411e+01                50s
       0 DPushes remaining with DInf 1.2871262e+02                55s

   26153 PPushes remaining with PInf 1.5976922e-02                55s
   19432 PPushes remaining with PInf 1.6360125e-02                55s
       0 PPushes remaining with PInf 0.0000000e+00                57s

  Push phase complete: Pinf 0.0000000e+00, Dinf 1.9323679e+03     57s

Iteration    Objective       Primal Inf.    Dual Inf.      Time
   77774    7.1526099e+05   0.000000e+00   1.932368e+03     57s
   78514    7.1526097e+05   0.000000e+00   6.154455e+01     60s
Crossover time: 12.80 seconds (17.84 work units)

Extra simplex iterations after uncrush: 17
   79005    7.1526097e+05   0.000000e+00   0.000000e+00     62s

Solved in 79005 iterations and 62.46 seconds (78.92 work units)
Optimal objective  7.152609726e+05
```


Example Model: rwth-timetable

PDHG on GPU

```
Non-default parameters:
Method  6
PDHGGPU  1

Optimize a model with 440134 rows, 923564 columns and 4510786 nonzeros (Min)
Model fingerprint: 0xf4eab31e
Model has 834949 linear objective coefficients
Coefficient statistics:
  Matrix range     [1e+00, 2e+00]
  Objective range  [1e+00, 4e+05]
  Bounds range     [1e+00, 1e+01]
  RHS range        [1e+00, 3e+00]
Presolve removed 206440 rows and 549304 columns
Presolve time: 3.06s
Presolved: 233694 rows, 374260 columns, 3675892 nonzeros


Start PDHG on GPU

                     Objective                Residual
     Iter       Primal          Dual         Primal    Dual     Compl     Time
        0   2.43298400e+07 -1.18242230e+07  2.00e+00 0.00e+00  6.45e+01    4s
    13104   7.15852435e+05  7.15074867e+05  3.44e-03 3.59e-02  9.59e-04    5s
    63304   7.15261009e+05  7.15260533e+05  9.93e-08 5.12e-01  9.59e-05   10s
    65504   7.15261011e+05  7.15261180e+05  8.11e-08 3.10e-01  6.43e-05   10s

PDHG solved model in 65504 iterations and 10.22 seconds (246.44 work units)
Optimal objective 7.15261011e+05
...
```

```
Crossover log...

   60163 DPushes remaining with DInf 8.9044933e-02                10s
       0 DPushes remaining with DInf 2.1340349e+01                15s

   26143 PPushes remaining with PInf 8.0291180e-03                15s
       0 PPushes remaining with PInf 0.0000000e+00                17s

  Push phase complete: Pinf 0.0000000e+00, Dinf 9.1626111e+02     17s

Iteration    Objective       Primal Inf.    Dual Inf.      Time
   77513    7.1526100e+05   0.000000e+00   9.162611e+02     17s
   78213    7.1526098e+05   0.000000e+00   7.924577e+01     20s
Crossover time: 12.68 seconds (18.17 work units)

Extra simplex iterations after uncrush: 17
   78771    7.1526097e+05   0.000000e+00   0.000000e+00     23s

Solved in 78771 iterations and 23.36 seconds (265.10 work units)
Optimal objective  7.152609726e+05
```

Speed ups: 867s -> 62.5s (**14X** with Method=6) -> 23.4s (**37X** with Method=6 PDHGGPU=1) 😲

Test Set and Results

- Our benchmark set comprises >2000 instances, mostly real-world, customer models
- It is heavily biased towards models that *we can solve today* using barrier or simplex
  - Benchmarking PDHG vs. Gurobi LP default on that set will invariably show a loss
- Filter test set down to models that neither of Simplex or Barrier can solve in 10000sec on a legacy cluster (“difficult” LPs)
- Among these 475 instances, 93 are solved faster by PDHG/GPU on GH200 than by any other LP method on a MacBook Pro M4
- PDHG complements our algorithmic portfolio
- PDHG can enable solving LPs that are intractable otherwise

What's in version 13?

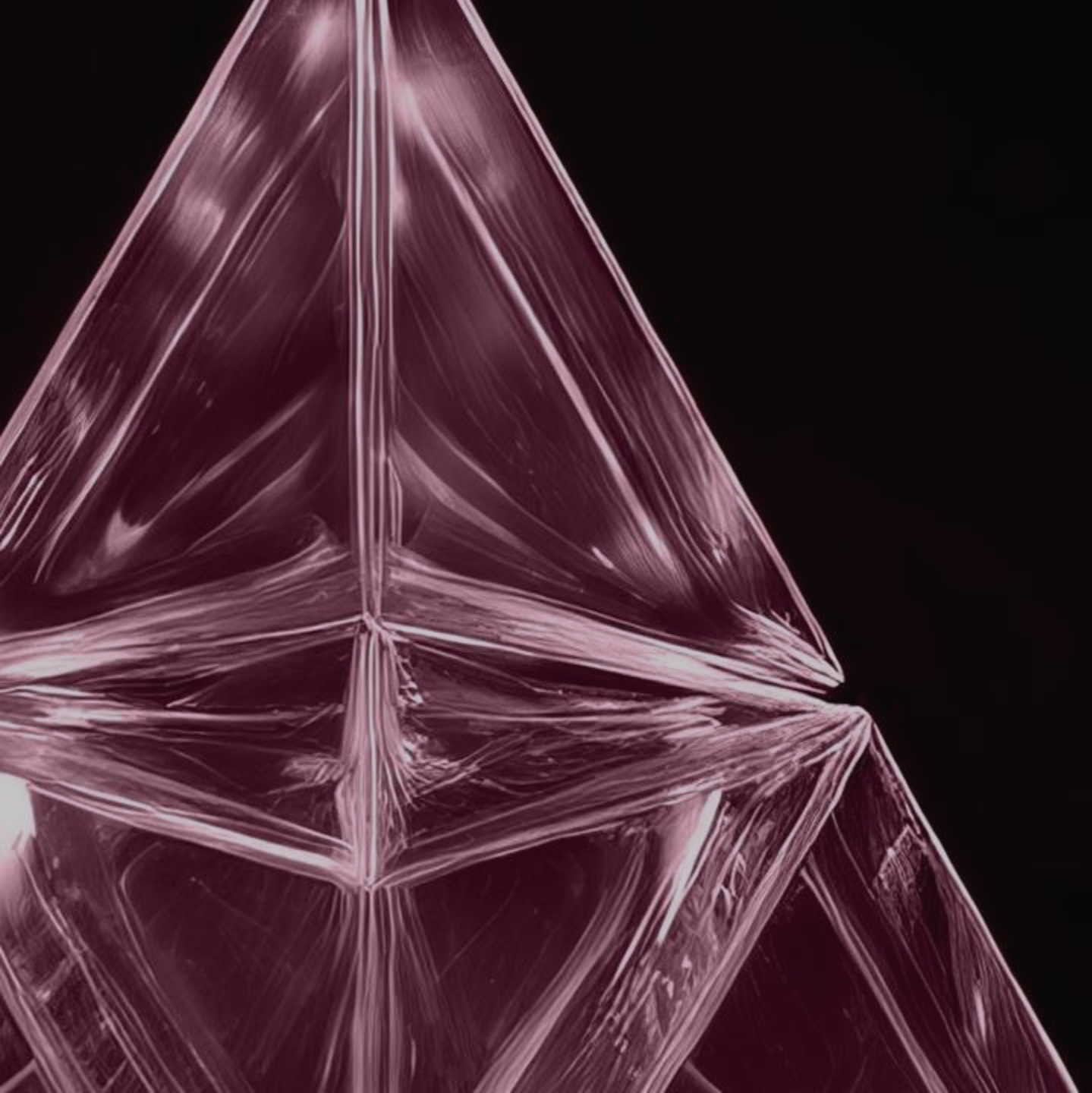
- CPU based PDHG implementation
  - Set via `Method=6`
- GPU based PDHG build via a separate download
  - Additionally set `PDHGGPU=1`
- Integrated with crossover for basic optimal solutions
- LP warmstarts also supported

- PDHG/GPU is “beta” in 13.0.0
- Expected to become “preview” in 13.0.1

Should you try it?

- Do you have a large* model?
- LP or MIP where root relaxation struggles?
- Do you typically see Barrier struggling?
- Do you typically use Barrier with Crossover=0?
- Have access to GPU machines?

*large=more than 10^5 nonzeros after presolve

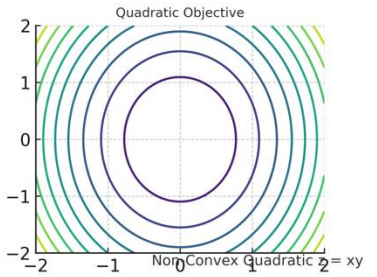


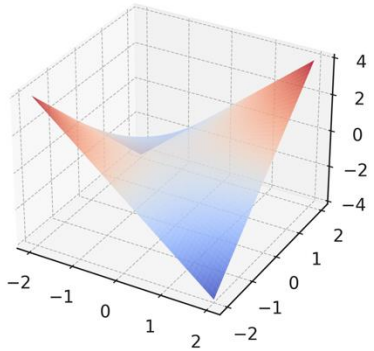
Nonlinear Optimization

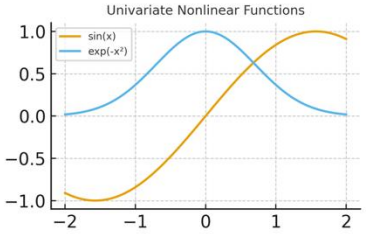
GUROBI OPTIMIZATION

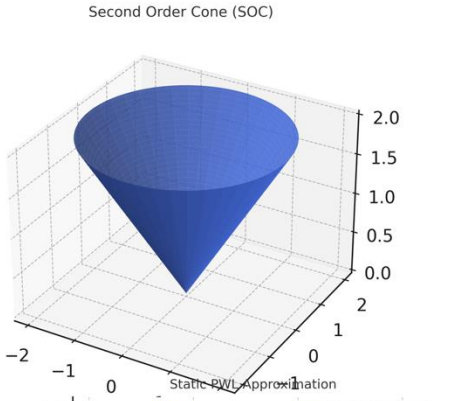
A Nonlinear History of Gurobi

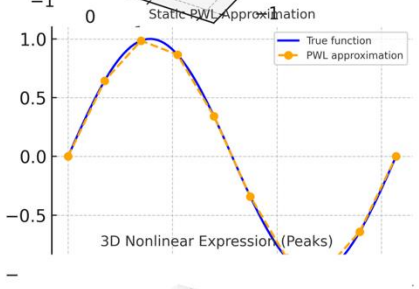
Version	Nonlinear object	Class	Algorithm(s)
4.0	Quadratic Objective	QP, MIQP	Simplex, Barrier, B&B
5.0	Convex Quadratic Constraints (SOC)	QCP, SOCP, MIQCP	Barrier, B&B, OA
9.0	Non-Convex Quadratic	NCQCP, MINCQCP	Spatial B&B
9.0	Static 1-D PWL	MILP	LP based B&B
11.0	Univariate Functions	MINLP	Spatial B&B
12.0	Nonlinear Expressions	MINLP	Spatial B&B

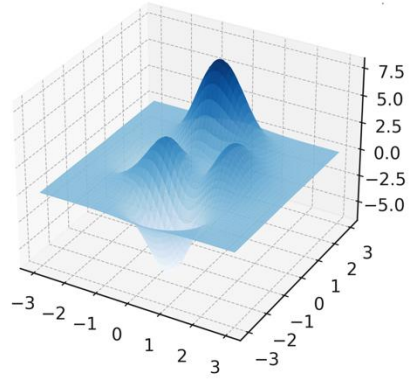




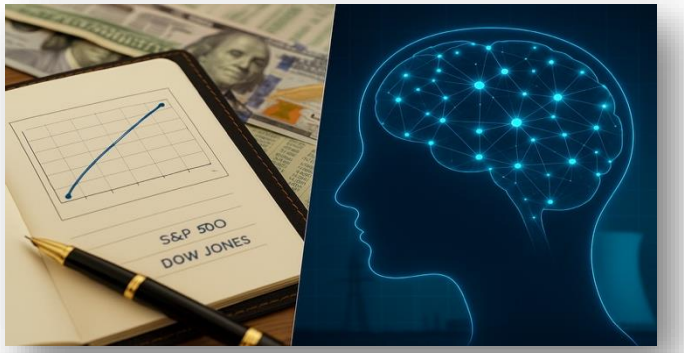








Why Nonlinear?



GUROBI OPTIMIZATION

Domain	Application	Phenomenon	Nonlinearity
Finance	Portfolio Optimization	Risk/Variability	Convex Quadratic
Physics	Truss Topology Optimization	Physical forces	Convex Quadratic
Petro-Chemical	Pooling	Mixing of products	Non-Convex Quadratic
Power Generation	ACOPF	Alternating Current	Cos/Sin or products
Gas/Water Distribution	Network optimization	Flow in pipes	Signpower: $\text{sign}(x) \cdot \lvert x \rvert^a$
Machine Learning	--	Activation	Logistic, tanh

Nonlinear Constraints API (Gurobi ≥12)

Nonlinear constraints of the form

$$y = f(\boldsymbol{x}), \boldsymbol{x} \in \mathbb{R}^n$$

f a composed of

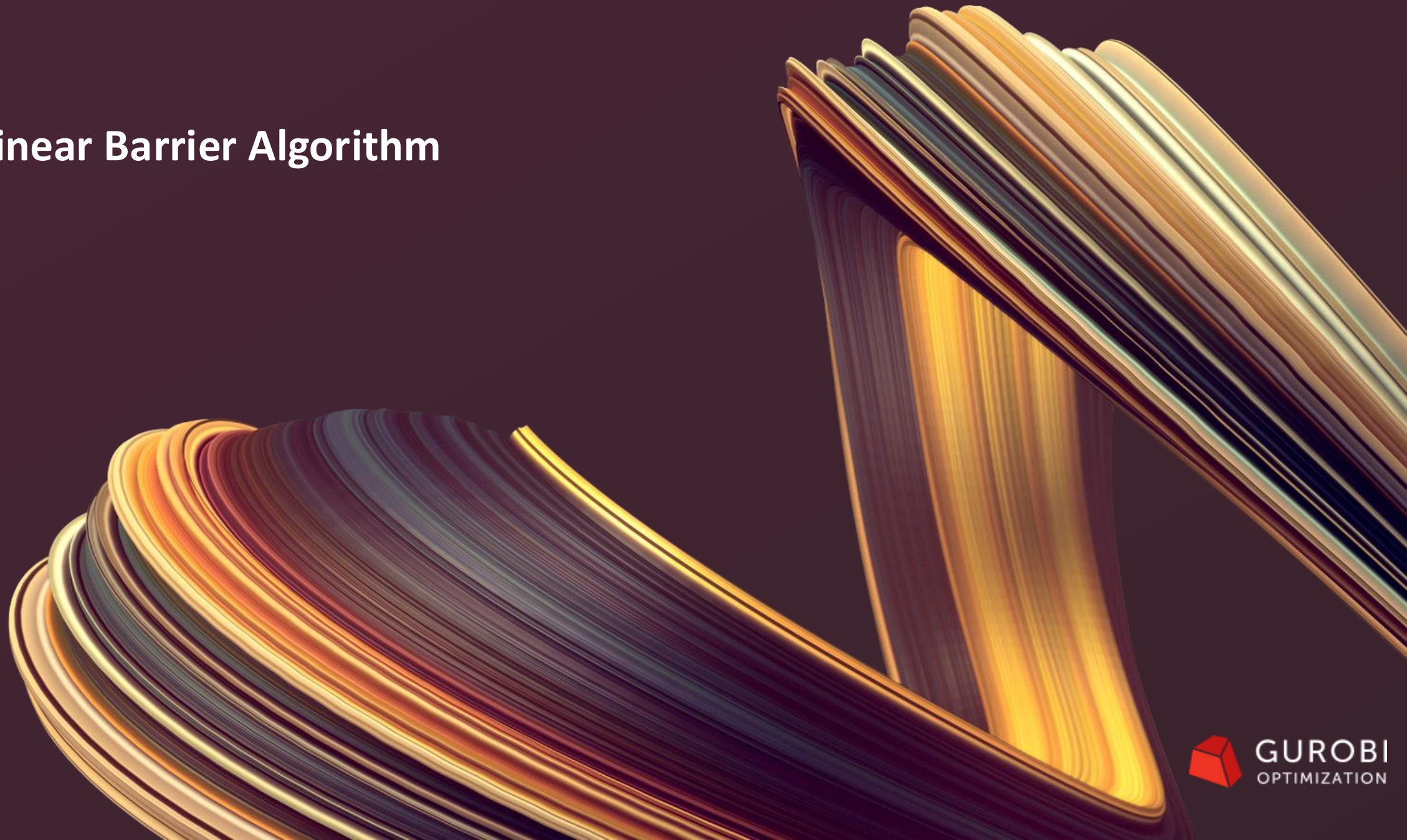
- $+, *, -, /$
- $x^a, a^x, \exp x$
- $\log(x)\,, \log_2(x)\,, \log_{10}(x)$
- $\sin(x)\,, \cos(x)\,, \tan(x)$
- $\text{Logistic}(x)$
- $\tanh(x)$
- $\text{signpow}(x, a) = \text{sign}(x) \cdot |x|^a, \text{a} \in \mathbb{R}_{\geq 1}$

Added in Gurobi 13

$$y = \sin x_1 \cdot x_2 + \frac{\sin x_1}{\exp x_2}$$

```
from gurobipy.nlfunc import sin, exp

model.addConstr(
    y == sin(x1) * x2 + sin(x1)/exp(x2)
)
```


Nonlinear Barrier Algorithm



GUROBI OPTIMIZATION

Non-Convex Optimization

Non-Convex Region

- Finding the highest point
  - Start at some point
  - Take improving steps inside feasible region
  - Converges only to locally optimal solution
- Need divide-and-conquer search algorithm to find globally optimal solution
  - Combinatorial explosion.

Local vs Global

- Finding a local optima is “easier” than proving global optimality
- But that doesn’t mean it’s “easy”:
  - Formally NP-hard
  - Problems are large, very large
  - Problems are nonlinear, very nonlinear
- Algorithms to find local optima generally fall into the domain of nonlinear optimization
- Essential to get solutions with high accuracy (i.e. small infeasibility)

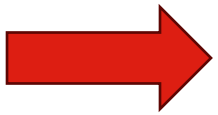
Nonlinear Barrier in Gurobi

Version	Nonlinear object	Class	NL Barrier
4.0	Quadratic Objective	QP, MIQP	
5.0	Convex Quadratic constraints (SOC)	QCP, SOCP, MIQCP	
9.0	Non-Convex Quadratic	NCQCP, MINCQCP	Introduced in 9.5 in NLPheur *heuristic*
9.0	Static PWL approximations	MILP	
11.0	Univariate function	MINLP	Works on univariate nl
12.0	Nonlinear Expressions	MINLP	
13.0	Nonlinear Expressions	NLP	**Directly callable**

Dr. Andreas Wächter

Barrier Algorithm

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0 \\ & x \geq 0 \end{aligned}$$



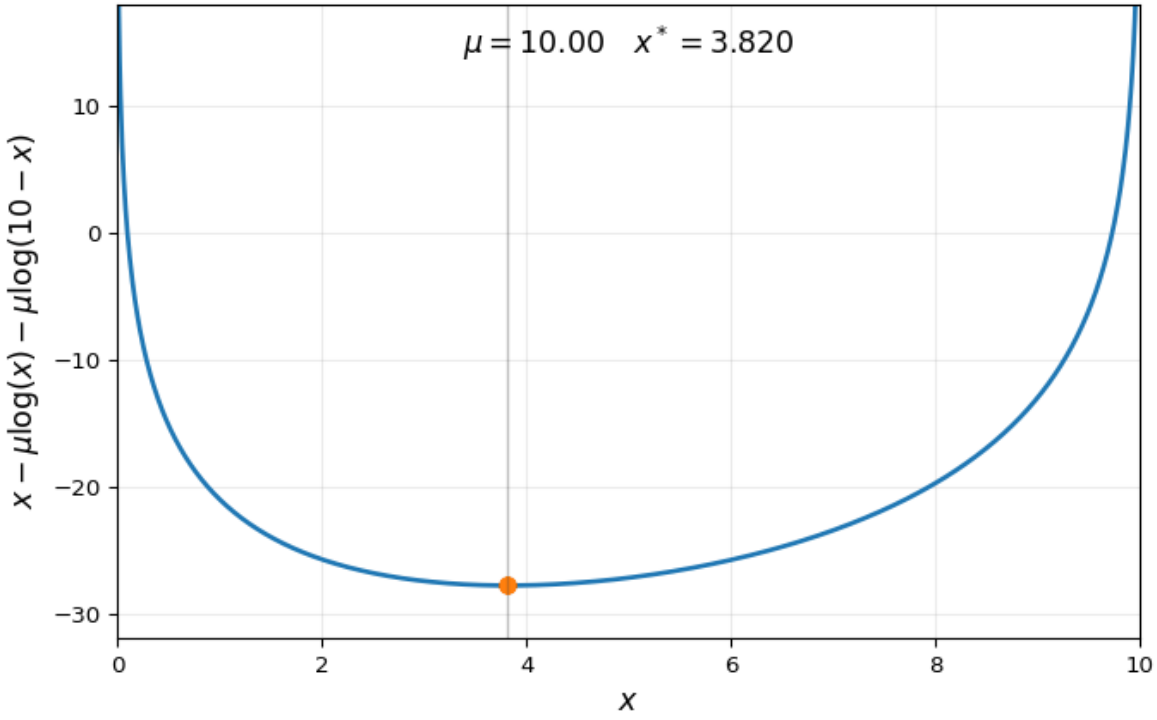
$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) - \mu \log x \\ \text{s.t.} \quad & c(x) = 0 \end{aligned} \qquad (\mathrm{BP}_\mu)$$

Basic Algorithm:

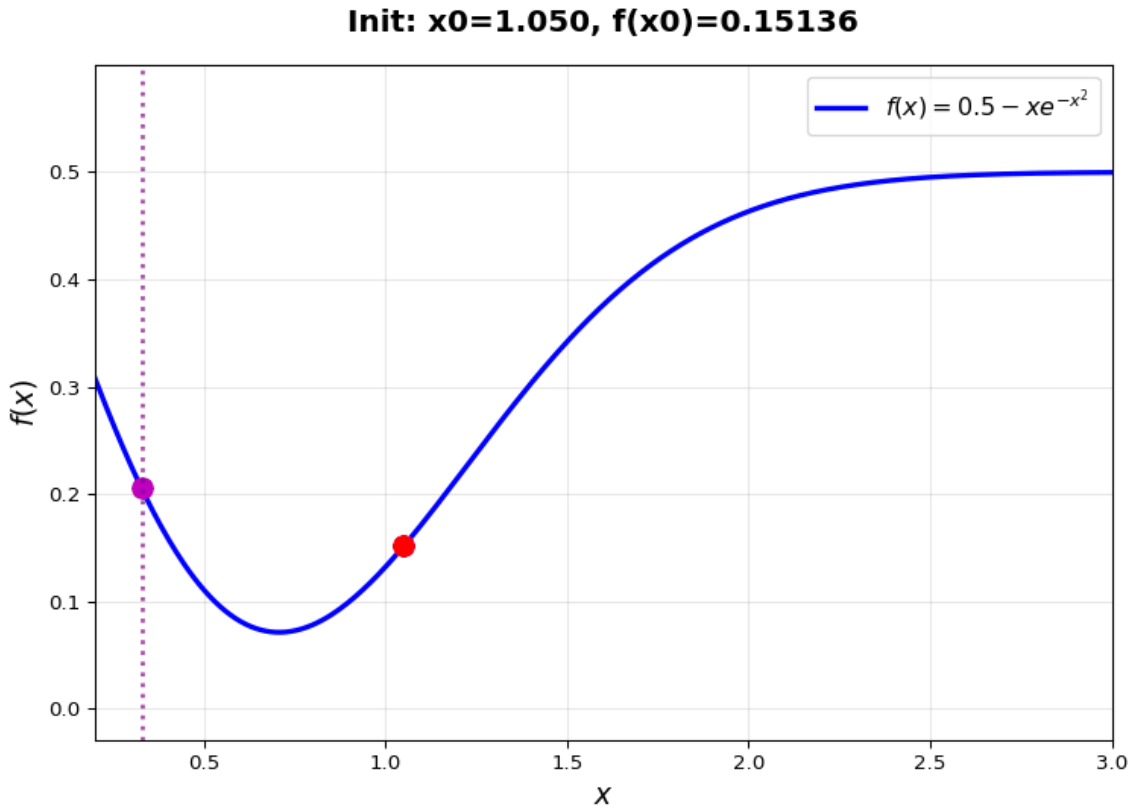
1. Choose $\mathrm{x}_0 \in \mathbb{R}^n$,$\mu_0 > 0$ .Set $\mathrm{k} \leftarrow 0$
2. Compute a step d_k for (BP_μ)
3. Take a step to get new iterate $\mathrm{x}_{\mathrm{k}+1} = \mathrm{x}_\mathrm{k} + \alpha_\mathrm{k}\mathrm{d}_\mathrm{k}$
4. If (BP_μ) has been solved to some accuracy, decrease μ
5. Set $\mathrm{k} \leftarrow \mathrm{k} + 1$; go to 2

Barrier Function Illustration

$$\begin{aligned} \min \quad & x \\ \text{s.t.} \quad & 0 \le x \le 10 \end{aligned} \quad \Rightarrow \quad \min x \; - \mu \log x \; - \mu \log(10 - x)$$



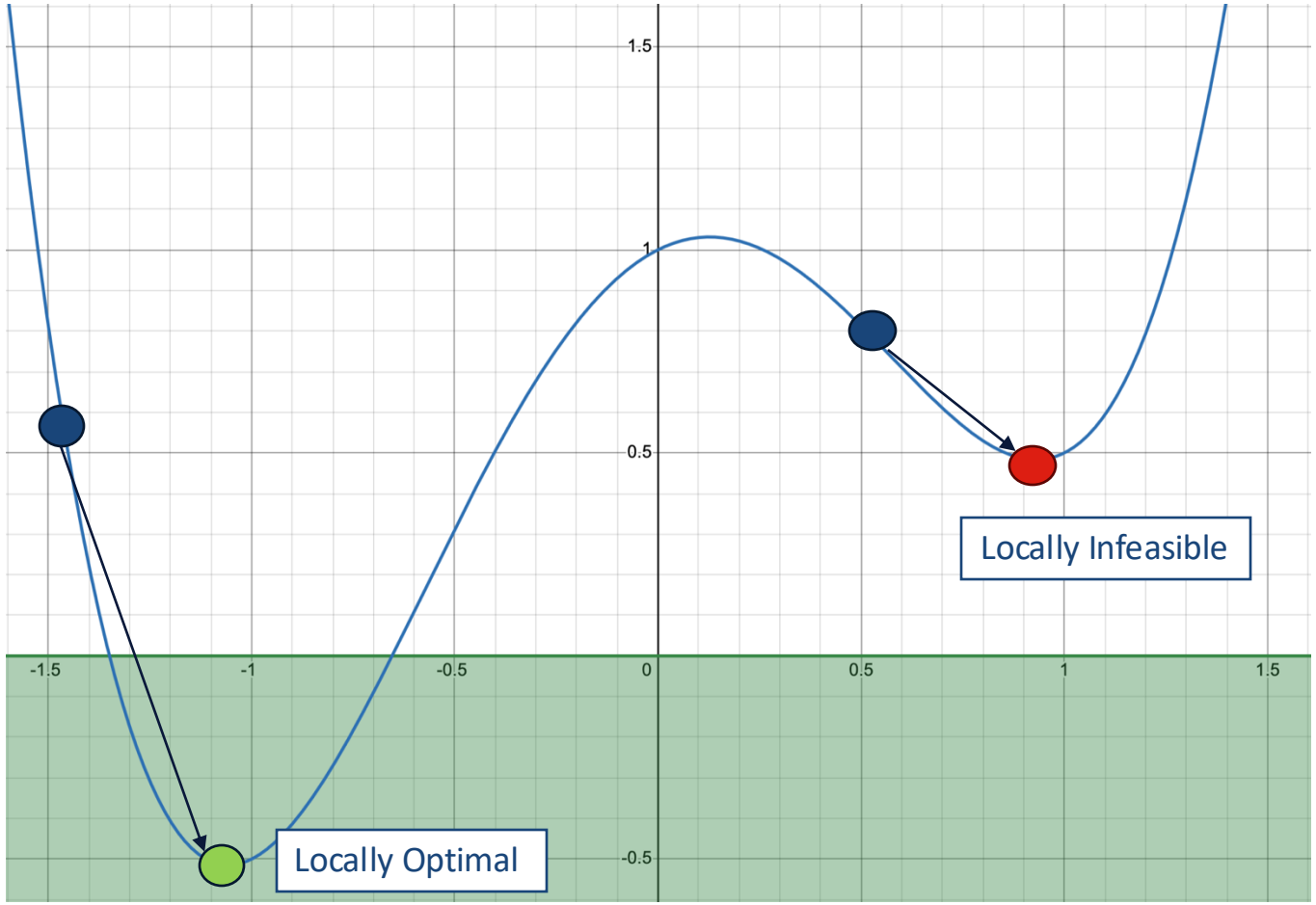
Newton's Method Illustration



Nonlinear Barrier

- Interior point algorithm for nonlinear programming
- Convergence to 1st order optimality conditions
  - Superlinear rate: *very fast local* convergence
  - Accurate solutions
- Regularization for non-convex problem
- Relies on indefinite factorization of KKT system
- Line search, Feasibility restoration, …
- The approach has proven to be very robust in practice
  - IPOPT, Knitro, …

Locally Optimal / Locally Infeasible



$$\begin{aligned} &min \;\; y \\ &\text{s.t.}\; y = \; x^4 - 2x^2 + 0.5x + 1 \\ &\quad y \leq 0 \end{aligned}$$

- Starting point $x = -1.5$
  → Locally Optimal
- Starting point $x = 0.5$
  → Detect locally infeasible
  → Changes to a feasibility recovery algorithm, minimizes violation.
  → Locally Infeasible

Using NL Barrier in Gurobi 13.0

GUROBI OPTIMIZATION

`OptimalityTarget=1`

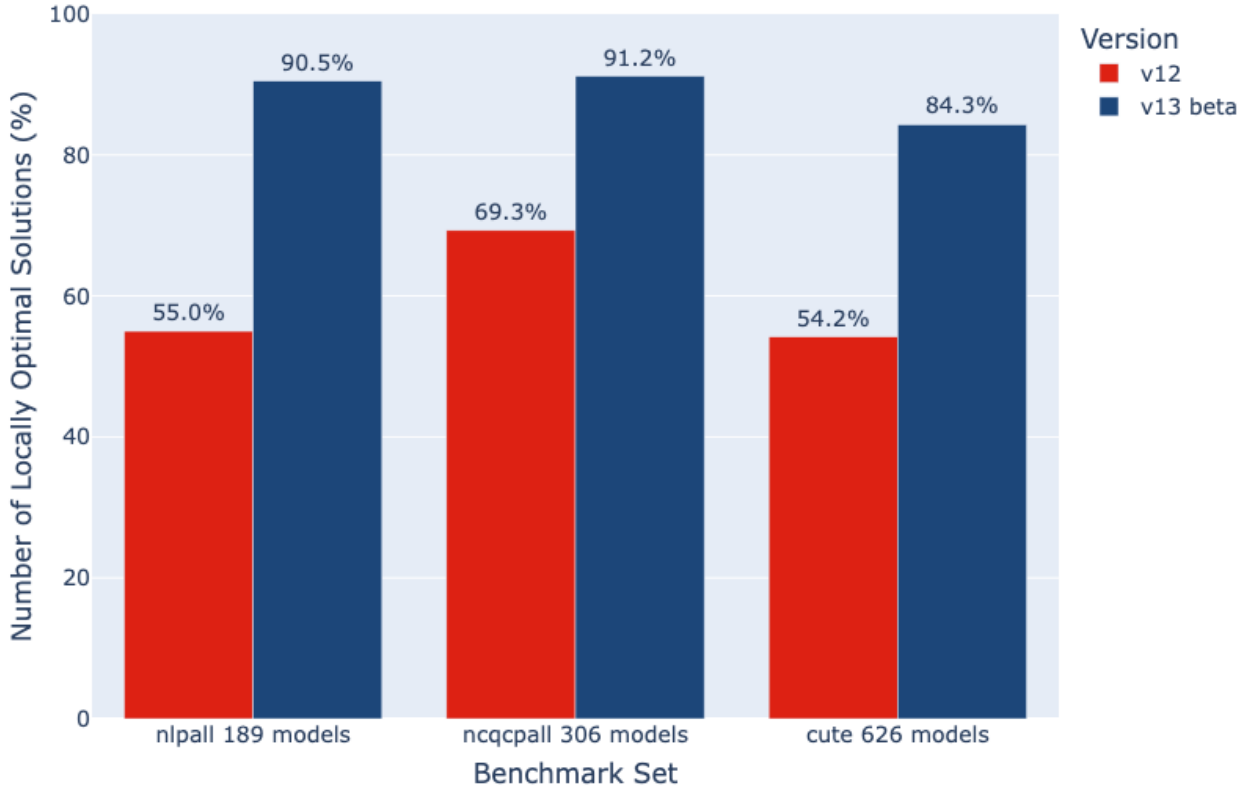
- Other parameter values:
  - -1: default (set to 0)
  - 0: use global MINLP solver
  - 1: use NL barrier (local)
- Only suitable for nonlinear continuous problems
  - Ignored if problem is LP or convex QP, QCP
  - Error if there are discrete variables or SOS constraints
- *"Preview"* feature in Gurobi 13
  - May change API behavior in future
  - No dual information returned
  - Not all quality attributes available

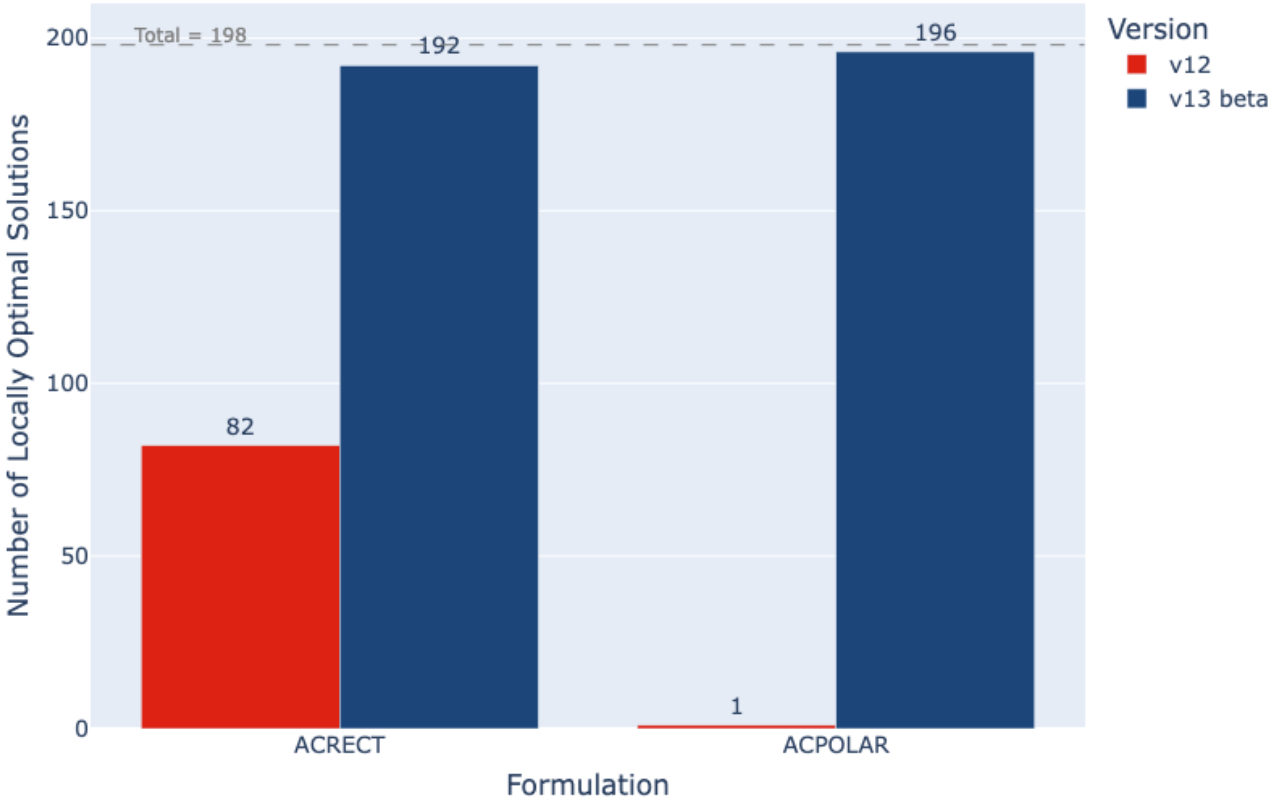
GUROBI OPTIMIZATION

NL Barrier – v13 Improvements

- Feasibility restoration (feas relax)
- Simple line search (without filter)
- Iterative refinement
- Algorithmic differentiation
- Handling numerical difficulties to achieve more accurate solutions

Solve rate comparison





ACOPF Performance

Key Ingredients:

- Better handling of starting points
- Feasibility restoration
- Algorithmic differentiation

Instances from https://github.com/power-grid-lib/pglib-opf
generated using https://github.com/Gurobi/gurobi-optimods


```
CPU model: AMD EPYC 7402P 24-Core Processor, instruction set [SSE2|AVX|AVX2]
Thread count: 12 physical cores, 12 logical processors, using up to 12
threads

Non-default parameters:
OptimalityTarget  1

Optimize a model with 4025 rows, 32604 columns and 5167 nonzeros (Min)
Model fingerprint: 0x1d675a52
Model has 2 linear objective coefficients
Model has 177 quadratic objective terms
Model has 37064 quadratic constraints
Coefficient statistics:
  Matrix range     [1e+00, 8e+03]
  QMatrix range    [4e-07, 1e+03]
  QLMatrix range   [1e+00, 1e+00]
  Objective range  [1e+00, 2e+03]
  QObjective range [2e+01, 2e+03]
  Bounds range     [5e-04, 1e+03]
  RHS range        [5e-04, 1e+00]
  QRHS range       [5e-04, 1e+06]
Presolve removed 4007 rows and 14193 columns

Continuous model is non-convex -- solving with NL barrier
```

Running ACOPF case 6468_rte

- Accurately represents the French very high voltage and high voltage transmission network from 2013:
  - 399 generators
  - 18000 branches
  - 6468 nodes


```
Presolve removed 5857 rows and 45606 columns
Presolve time: 0.59s
Presolved: 7080 rows, 43326 columns, 28005 nonzeros
Presolved model has 77652 quadratic constraint(s)
Warning, incomplete warm-start solution
Ordering time: 0.09s

NL barrier statistics:
  Hessian NZ  : 7.194e+04
  Jacobian NZ : 3.180e+05
  Factor NZ   : 1.679e+06
  Factor Ops  : 2.426e+07
  Threads     : 1

                          Residuals
Iter     Objective      Primal     Dual      Compl     Step     Time
   0   4.25101256e+05  3.11e+01  1.73e+02  1.84e+04  0.00e+00     1s
   1   4.25101259e+05  3.11e+01  1.73e+02  1.84e+04  4.75e-10     1s
   2   4.49822250e+05  3.09e+01  1.72e+02  1.83e+04  6.04e-03     1s
   3   4.85968510e+05  3.06e+01  1.72e+02  1.83e+04  9.64e-03     1s
   4   5.49358735e+05  3.00e+01  1.70e+02  1.82e+04  1.83e-02     2s
   5   6.21857031e+05  2.93e+01  1.68e+02  1.79e+04  2.30e-02     2s
….
```

Running ACOPF case 6468_rte

- Accurately represents the French very high voltage and high voltage transmission network from 2013:
  - 399 generators
  - 18000 branches
  - 6468 nodes


```
…
  41    2.83887656e+06  9.67e+00  1.89e+05  7.51e+02  1.52e-03     6s
  42*   2.83887656e+06  9.67e+00  3.10e+04  1.42e+04  0.00e+00     6s
  43*   2.83302648e+06  7.78e+00  2.10e+04  1.02e+04  1.95e-01     7s
  44*   2.83009023e+06  6.82e+00  1.14e+04  6.90e+03  1.23e-01     7s
  45*   2.82881891e+06  6.32e+00  1.01e+04  6.24e+03  7.45e-02     7s
  46*   2.82844843e+06  5.14e+00  8.90e+03  5.16e+03  2.23e-01     7s
  47*   2.82843805e+06  5.12e+00  8.60e+03  5.07e+03  2.85e-03     8s
  48*   2.82736002e+06  4.54e+00  6.83e+03  4.26e+03  1.13e-01     8s
…
  63*   2.97552582e+06  1.80e+00  3.41e+02  1.06e+02  4.59e-01    11s
  64*   3.06728379e+06  3.52e+00  1.48e+02  4.62e+01  6.76e-01    11s
  65*   3.17581783e+06  7.05e+00  6.50e+00  1.48e+00  9.66e-01    11s
  66*   3.20837525e+06  3.52e+00  1.32e+00  1.00e+00  1.00e+00    11s
  67*   3.21003568e+06  3.11e-02  1.77e-02  1.00e+00  1.00e+00    11s
  68    3.21721689e+06  2.79e-01  6.98e+02  1.84e+04  6.00e-02    11s
  69    3.21721689e+06  2.79e-01  5.12e+02  1.35e+04  4.75e-10    11s
…
```

Running ACOPF case 6468_rte

- Accurately represents the French very high voltage and high voltage transmission network from 2013:
  - 399 generators
  - 18000 branches
  - 6468 nodes


```
 120   2.45283950e+06  3.79e-01  2.34e+00  3.51e-04  3.52e-01    17s
 121   2.45280236e+06  2.19e-01  1.12e+00  3.48e-04  5.59e-01    17s
 122   2.45278005e+06  2.44e-02  1.01e-03  3.46e-04  1.00e+00    17s
 123   2.45278173e+06  3.40e-04  1.38e-05  3.46e-04  1.00e+00    17s
 124   2.45278169e+06  1.50e-05  9.54e-07  3.46e-04  1.00e+00    17s
 125   2.45276480e+06  1.51e-02  4.93e-01  1.32e-04  4.00e-01    17s
 126   2.45275827e+06  1.40e-02  1.18e+00  2.09e-05  3.05e-01    17s
 127   2.45275017e+06  1.28e-02  6.30e-01  1.51e-05  6.14e-01    17s
 128   2.45274731e+06  5.30e-03  2.71e-01  1.44e-05  6.96e-01    18s
 129   2.45274624e+06  4.37e-04  3.87e-05  1.43e-05  1.00e+00    18s
 130   2.45274627e+06  1.65e-05  8.74e-06  1.43e-05  1.00e+00    18s
 131   2.45274627e+06  3.83e-07  5.96e-08  1.43e-05  1.00e+00    18s
 132   2.45274549e+06  1.71e-04  7.43e-02  1.52e-06  6.72e-01    18s
 133   2.45274523e+06  9.70e-05  3.80e-02  1.70e-07  7.43e-01    18s
 134   2.45274515e+06  3.81e-05  6.63e-03  1.66e-07  8.78e-01    18s
 135   2.45274514e+06  1.50e-06  3.02e-06  1.65e-07  1.00e+00    18s
 136   2.45274514e+06  1.86e-09  1.94e-09  1.65e-07  1.00e+00    18s
 137   2.45274513e+06  2.88e-07  8.60e-04  1.11e-09  8.50e-01    19s
 138   2.45274513e+06  1.26e-08  5.51e-09  1.00e-09  1.00e+00    19s
 139   2.45274513e+06  1.86e-09  2.09e-10  1.00e-09  1.00e+00    19s

NL barrier solved model in 139 iterations and 18.85 seconds (13.08 work
units)
First-order optimal solution
Solution objective 2.452745130706e+06
```

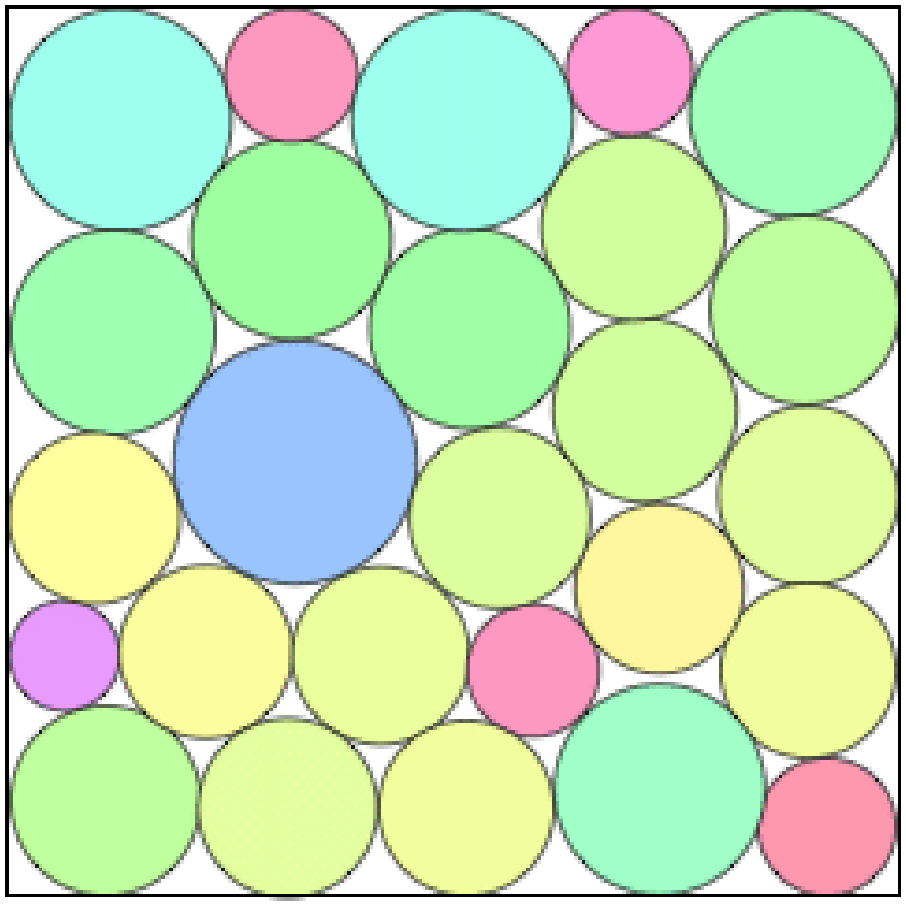
GUROBI OPTIMIZATION

Running ACOPF case 6468_rte

- Accurately represents the French very high voltage and high voltage transmission network from 2013:
  - 399 generators
  - 18000 branches
  - 6468 nodes
- Would take FOREVER seconds to solve to global optimality

Integration of Barrier in the Global Solver

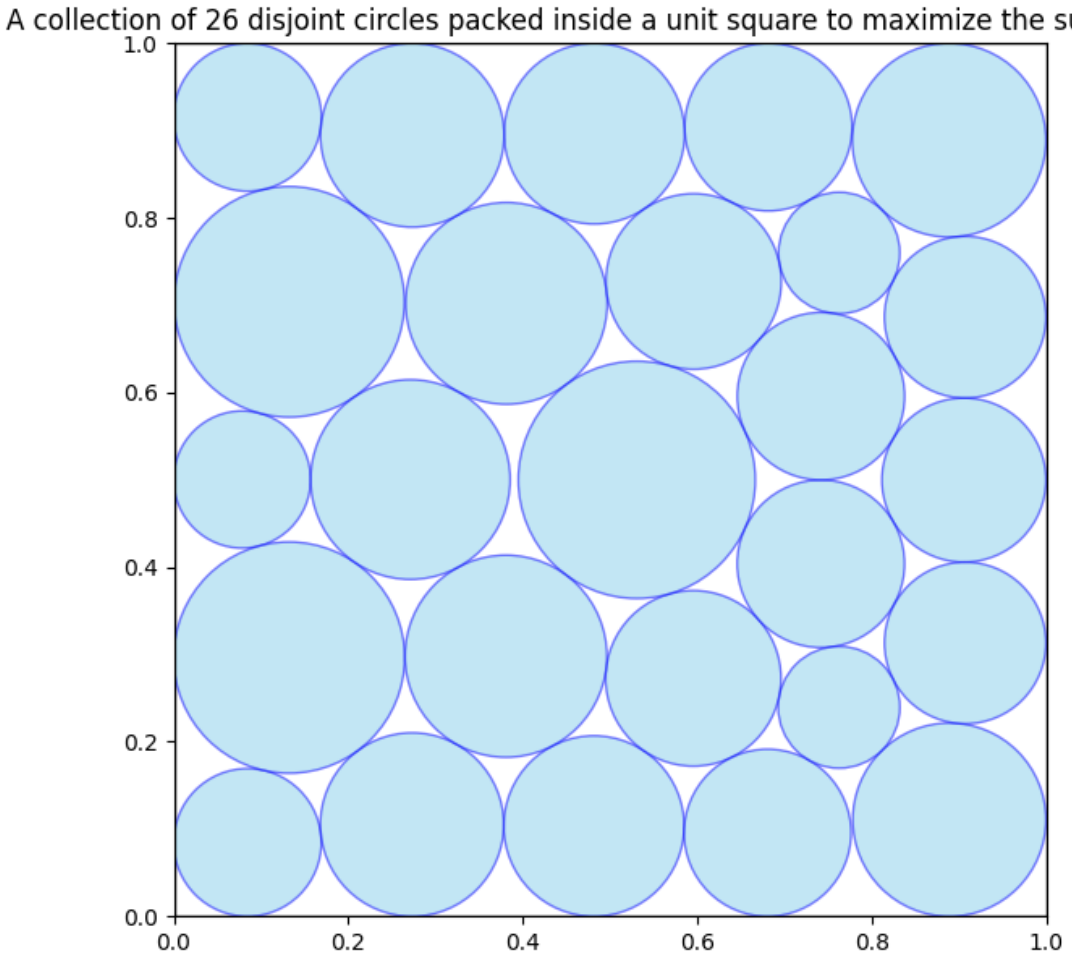
- NL Barrier used as *"heuristic"* in global solver
- Goal: finding good locally optimal solution
- Controlled by parameter `NLPheur`, **new values:**
  - -1: default
  - 0: off
  - 1: mild (essentially run once or twice at the root)
  - 2: moderate (models with integers, run for every *assignment* encountered)
  - 3: aggressive (run at node with node local bounds)



https://erich-friedman.github.io/packing/cirRsqu/

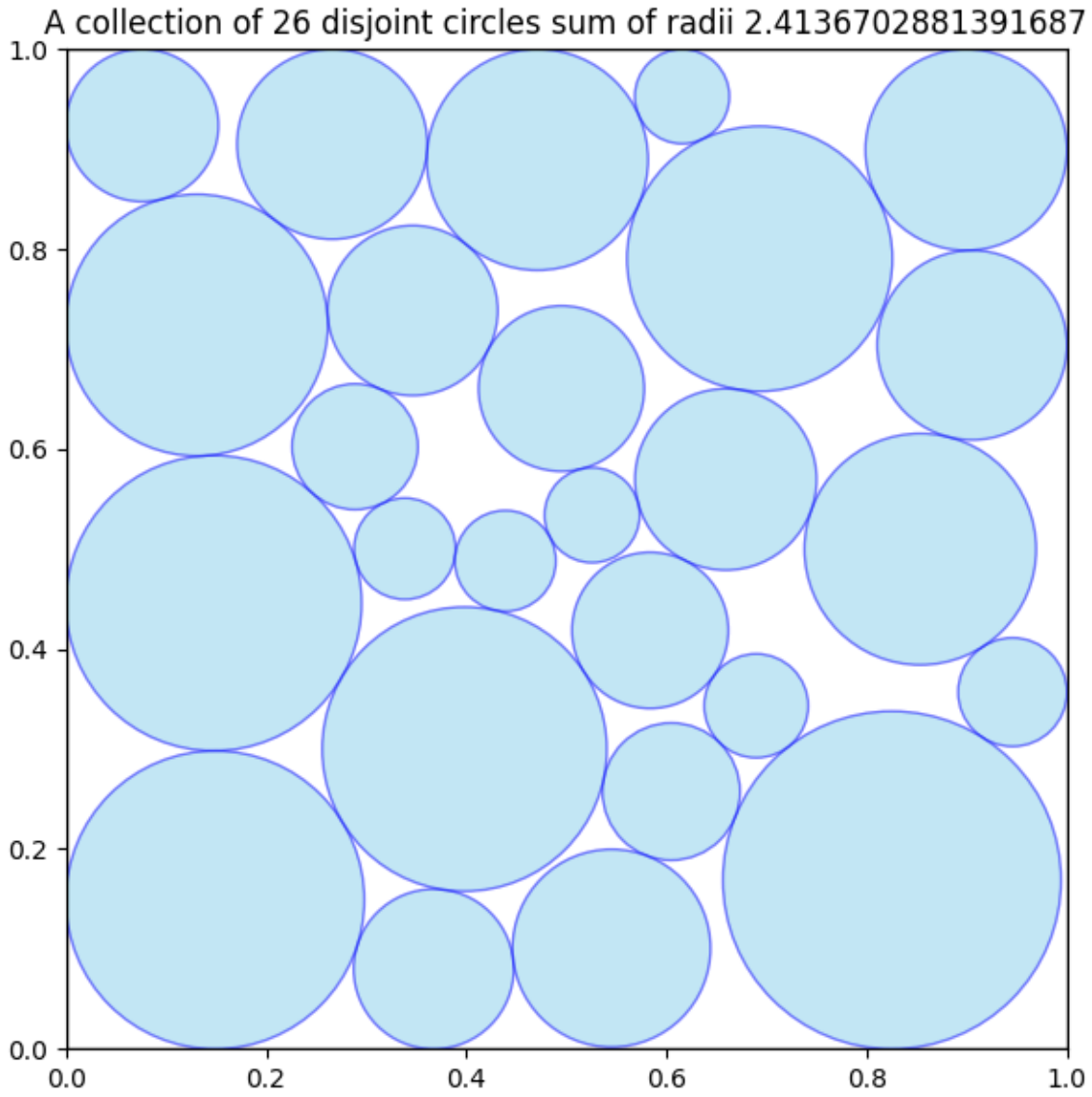
Circle Packing Problem

Previous Best	2.634+



Circle Packing Problem

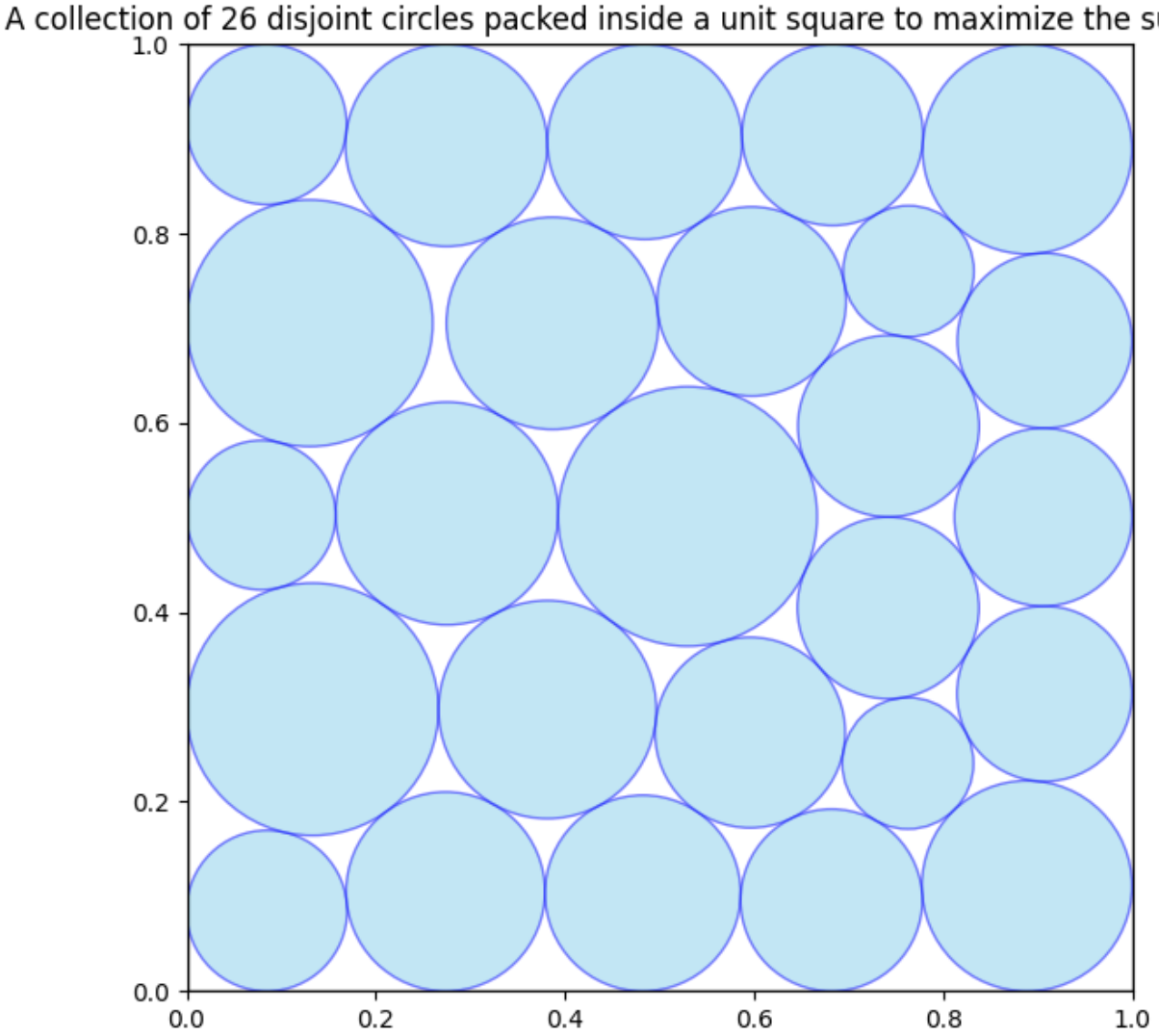
Previous Best	2.634+
AlphaEvolve	2.635**86275**



GUROBI OPTIMIZATION

Circle Packing Problem

Previous Best	2.634+
AlphaEvolve	2.635**86275**
Gurobi 12	2.41367028



GUROBI OPTIMIZATION

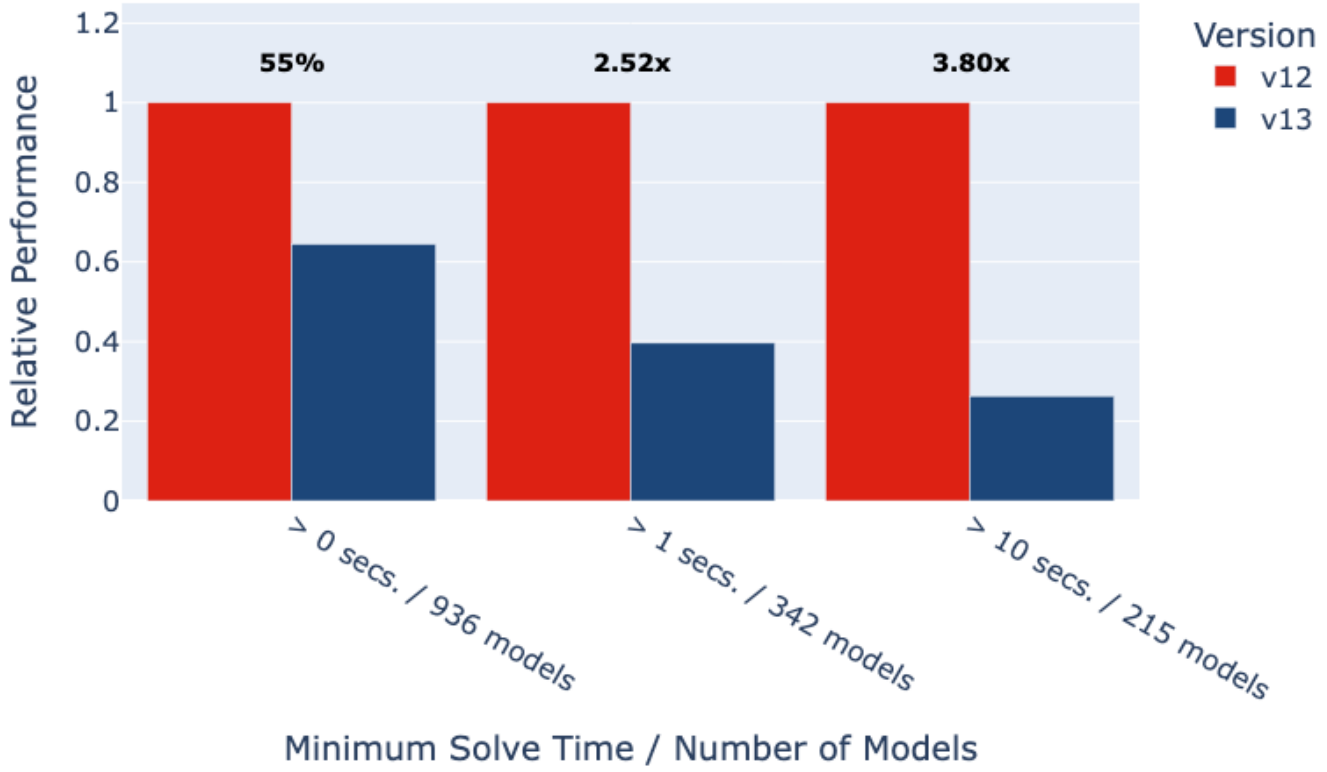
Circle Packing Problem

Previous Best	2.634+
AlphaEvolve	2.635**86275**
Gurobi 12	2.41367028
Gurobi 13	2.635**98301**

GUROBI OPTIMIZATION

MINLP Performance Improvements

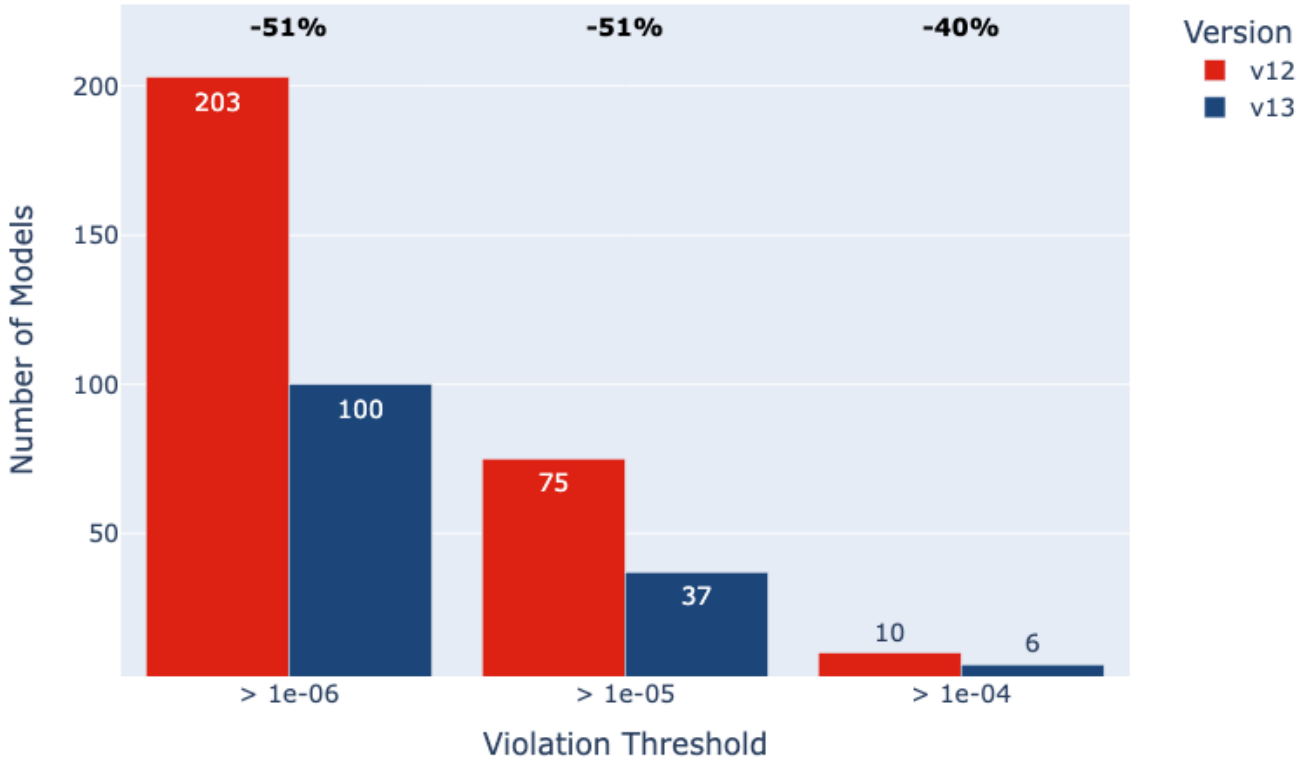
- Performance comparison by solve time threshold
- Statistics on 1394 models in internal set
- Removed 458 models not solved by either versions



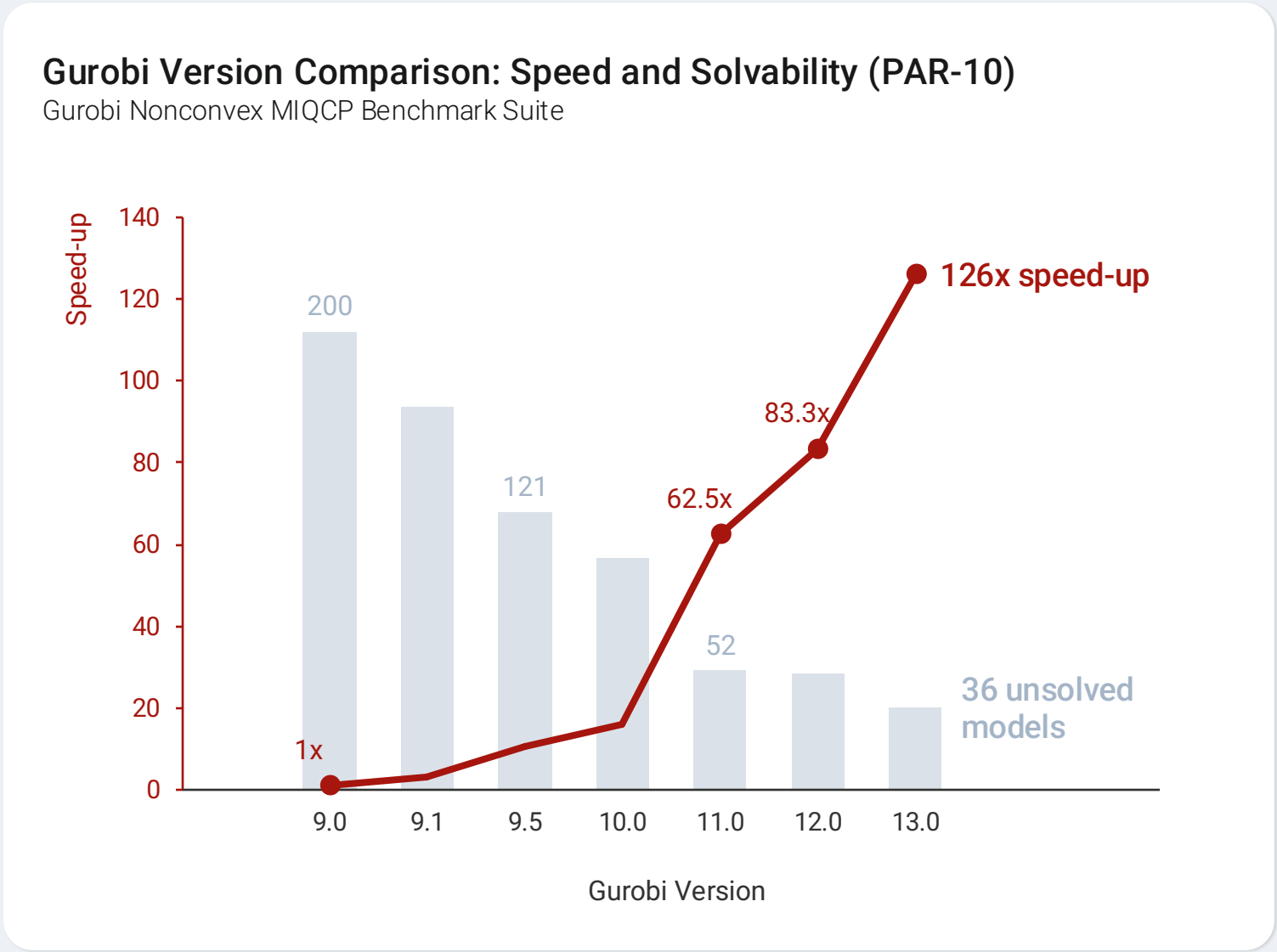
GUROBI OPTIMIZATION

MINLP Solution Quality Improvements

- Cumulative constraint violation by threshold
- Statistics on 1394 models in internal set
- V12: 1219 solutions
- V13: 1309 solutions



GUROBI OPTIMIZATION



Nonconvex MIQCP

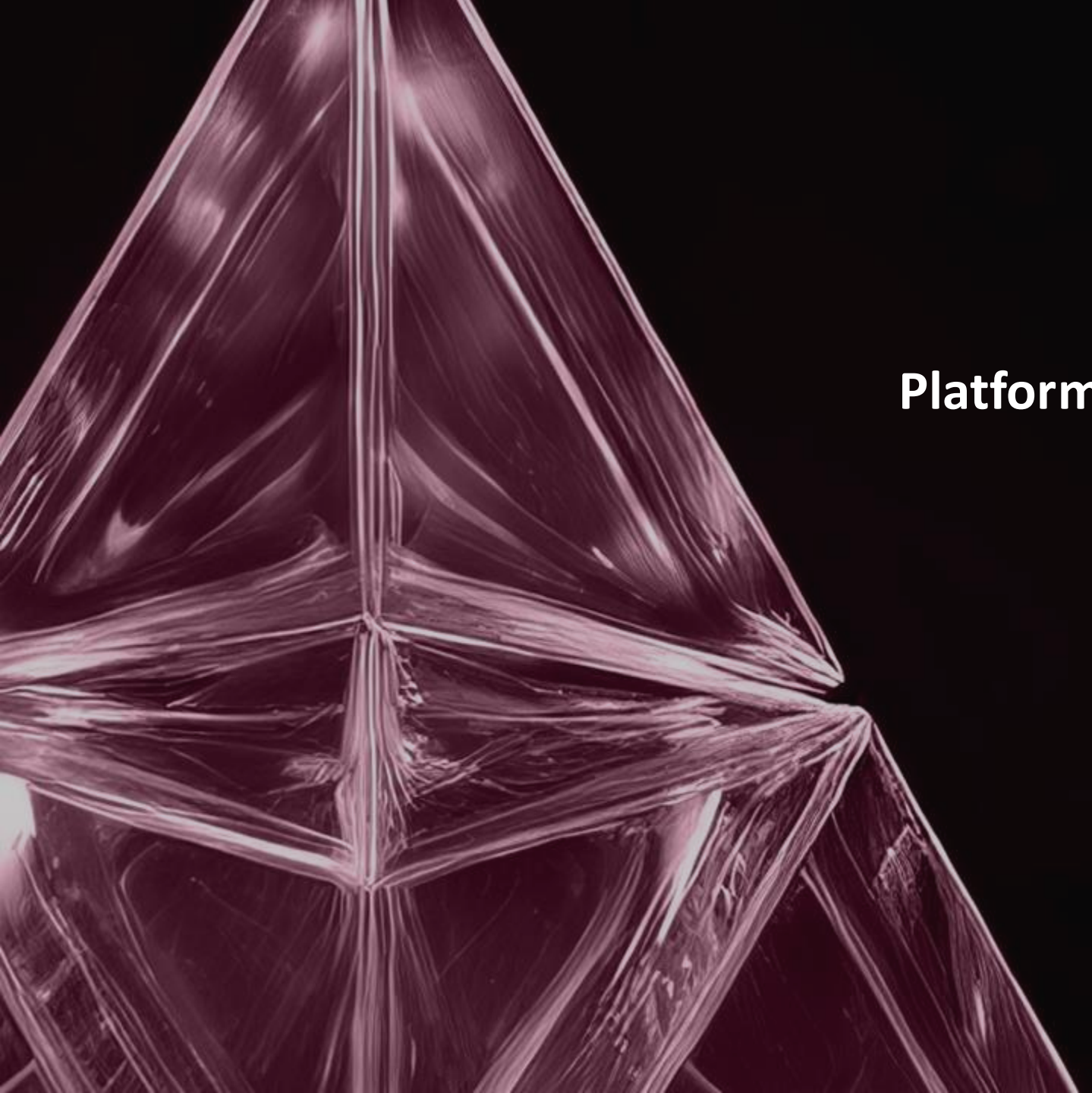
Performance Evolution

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

- A **126x** speed-up over version 9.0 in geometric mean (PAR-10) of runtimes
- Only **36** 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.

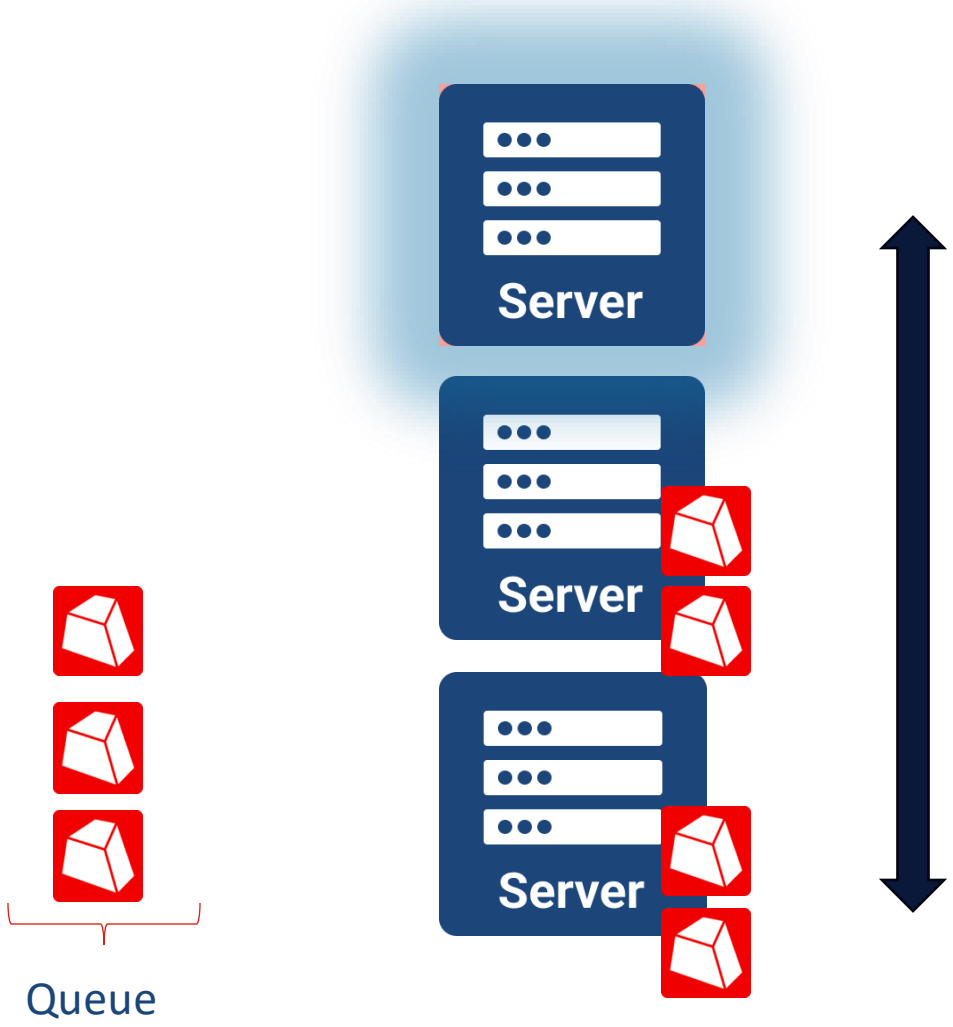
Time limit: 10000 sec.
Intel Xeon CPU E3-1240 v5 @ 3.50GHz
4 cores, 8 hyper-threads
32 GB RAM

Test set has 1226 models:
- 89 discarded due to inconsistent answers
- 375 discarded that none of the versions can solve
- speed-up measured on >100s bracket: 337 models



Platform Improvements & Conclusions

GUROBI OPTIMIZATION



Kubernetes Autoscaler Operator

- Automatically scale your Gurobi Compute Servers through Kubernetes
- Intelligent scaling is configurable through parameters:
  - job queue
  - waiting time
  - idle time
  - number of queued jobs
- Safe scale down logic does not interrupt jobs
- Available through DockerHub

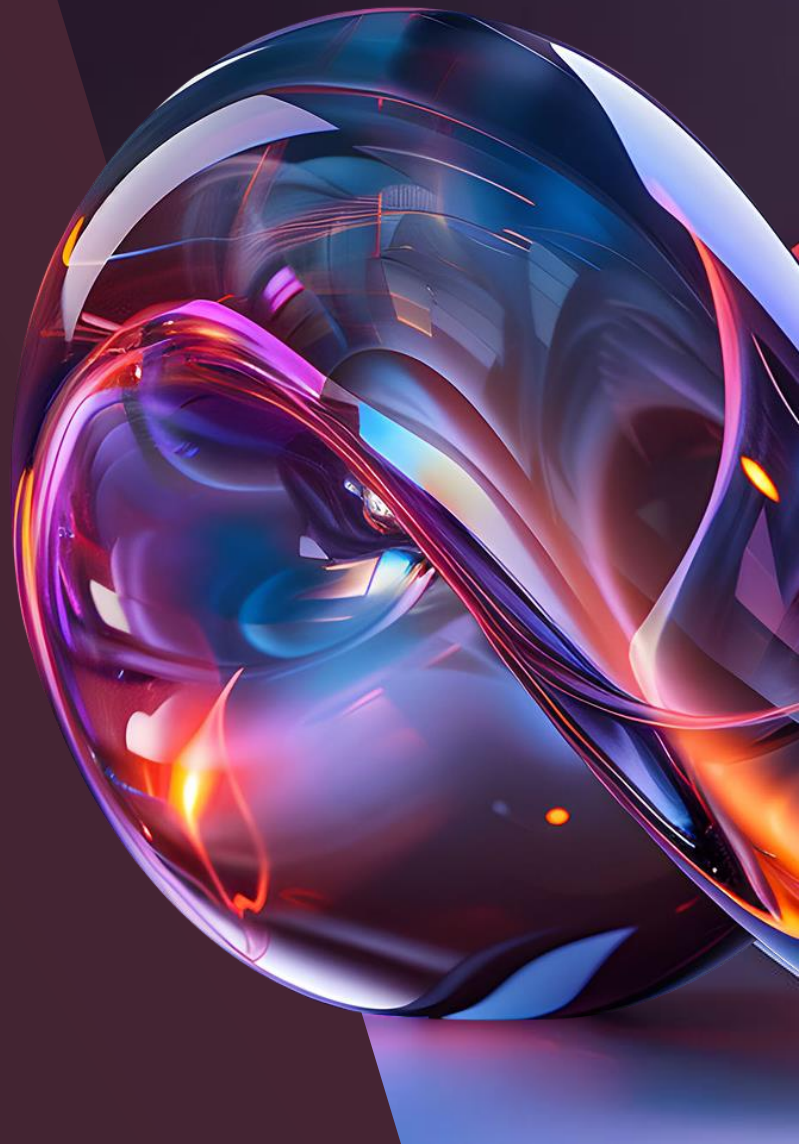
Accessibility

Accessibility Enhancements for Cluster Manager in Version 13

- Full **keyboard navigation** support
- All features now **screen-reader accessible**
- Clearer structures and improved **text contrast**

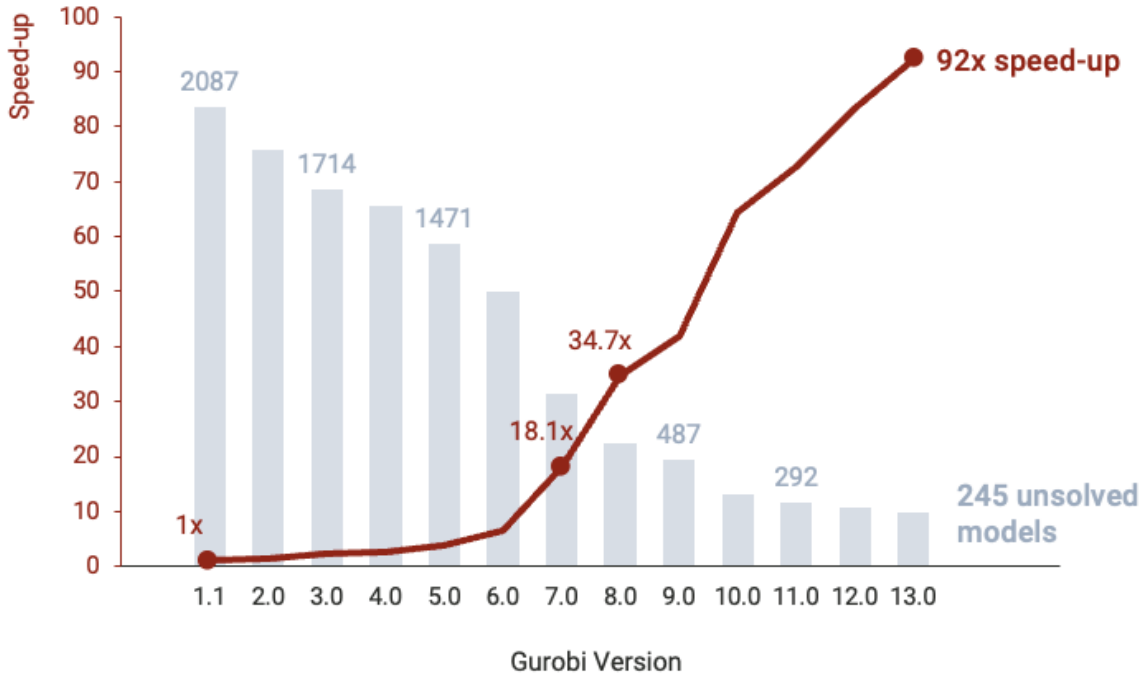
Consistent Accessibility Across Platforms

- Builds on earlier improvements to the **website** and **user portal**
- Delivers a **consistent, inclusive experience** across Gurobi platforms for everyone
- Fully aligned with **WCAG 2.1 AA** and the **European Accessibility Act (EN 301 549)**

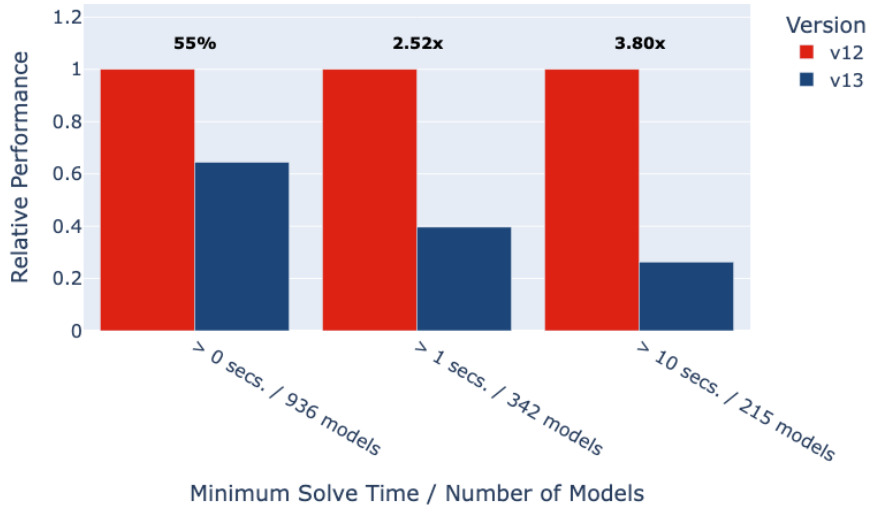


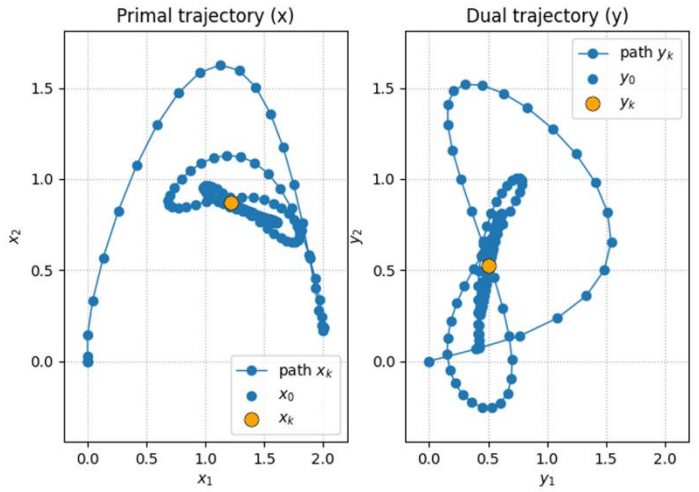
Conclusions

MILP Performance



MINLP (+ NL Barrier)





PDHG + GPU

Webinar Series

2026 -13.0 Webinar Series Continues:

- Local NL Barrier
- Global MINLP
- PDHG on GPU
- Gurobi Intelligence – GenAI Update

Stay tuned at Gurobi.com/events

Q&A