What's New in

Gurobi 12.0

Gurobi 12.0 新亮点和技术创新

顾宗浩 博士，Gurobi CTO 和联合创始人

主讲人介绍

- **顾宗浩 博士**
- Gurobi 首席技术官 (CTO) 和联合创始人
- 上海同济大学机械工程学士和工业管理硕士，佐治亚理工学院工业工程博士
- 数学规划理论和实践领域全球最顶尖的专家之一

性能提升

Gurobi 12 性能提升汇总

GUROBI OPTIMIZATION

类型	整体提升 (>1 秒)	>100 秒复杂模型
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 性能提升汇总

类型	整体提升 (>1 秒)	>100 秒复杂模型
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 预优化提升

- 引申变量预优化消除 Derived variables presolve reduction 0.3%
  - 例如 (3.0% on affected models)

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

  其中单变量 y 和变量向量 x
  - 如果

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

  那么
    - y 和这些约束可以从模型中移除, 并且
    - 待x 最优解发现后再计算 y
  - 同样适用于 MIP, 但需要做整数性验证

- 提早终止LP 预优化聚合 Stopping aggregator earlier for LP presolve 0.3%
  - 等到下一个预优化循环再继续 (1.1% on affected models)

单纯形提升

- 原始单纯形 Primal simplex
  - 提升比率检验的数值计算 Improved numerics in ratio test 1.1%
- 对偶单纯形 Dual simplex
  - 提升Harris 比率检验的计算性能 Performance improvement in Harris ratio test 1.9%
  - 数值方面的重新改写 Major rework of numerical aspects 1.3%
  - 降低目标漂移 Less objective shifting 0.7%
  - 对自由变量处理更好的迅速构造的初始基 Better crash basis for free variables 0.6%
  - 提升比率检验的数值计算 Improved numerics in ratio test 0.4%
  - 提升基变量可行性检验方面的数值计算 Improved numerics in feasibility check for basic variables 0.4%

内点法提升

- 单纯形：在交叉中更频繁分解基矩阵 Simplex: factorize more often in crossover 1.0%
- 对于 A 矩阵稠密数据块的简化处理 Simplify handling of dense blocks in A 0.7%
  - 稠密数据块没有必要使用复杂数据结构以获得稀疏性 No need for dense blocks to use complex data structures to exploit sparsity
- 在开始几个内点循环中采用迭代线性系统求解 Use iterative linear system solves in first barrier iterations 0.4%
  - 比 Cholesky 分解更快的迭代 Faster iterations than with Cholesky factorization (9.7% on affected models)
  - 精确性稍逊 Less accurate

Gurobi 12 性能提升汇总

GUROBI OPTIMIZATION

类型	整体提升 (>1 秒)	>100 秒复杂模型
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

SOCP 提升

- 锥内部的双变量预优化聚合 Doubleton presolve aggregation inside cones 7.8%
  - 即便 y 出现在锥内，线性等式 $y = ax$ 也可以用于聚合 Linear equality $y = ax$ can be used for aggregation even if y appears in a cone
- 改进 SOCP 的稠密列处理 Improved dense column handling for SOCP 4.7%
- 锥变量上界的隐式处理 Implicit handling of cone variable upper bounds 2.7%
  - 参考 See Andersen, Roos, Terlaky (2000), Sturm (2002), Goldfarb, Scheinberg (2005)

Gurobi 12 性能提升汇总

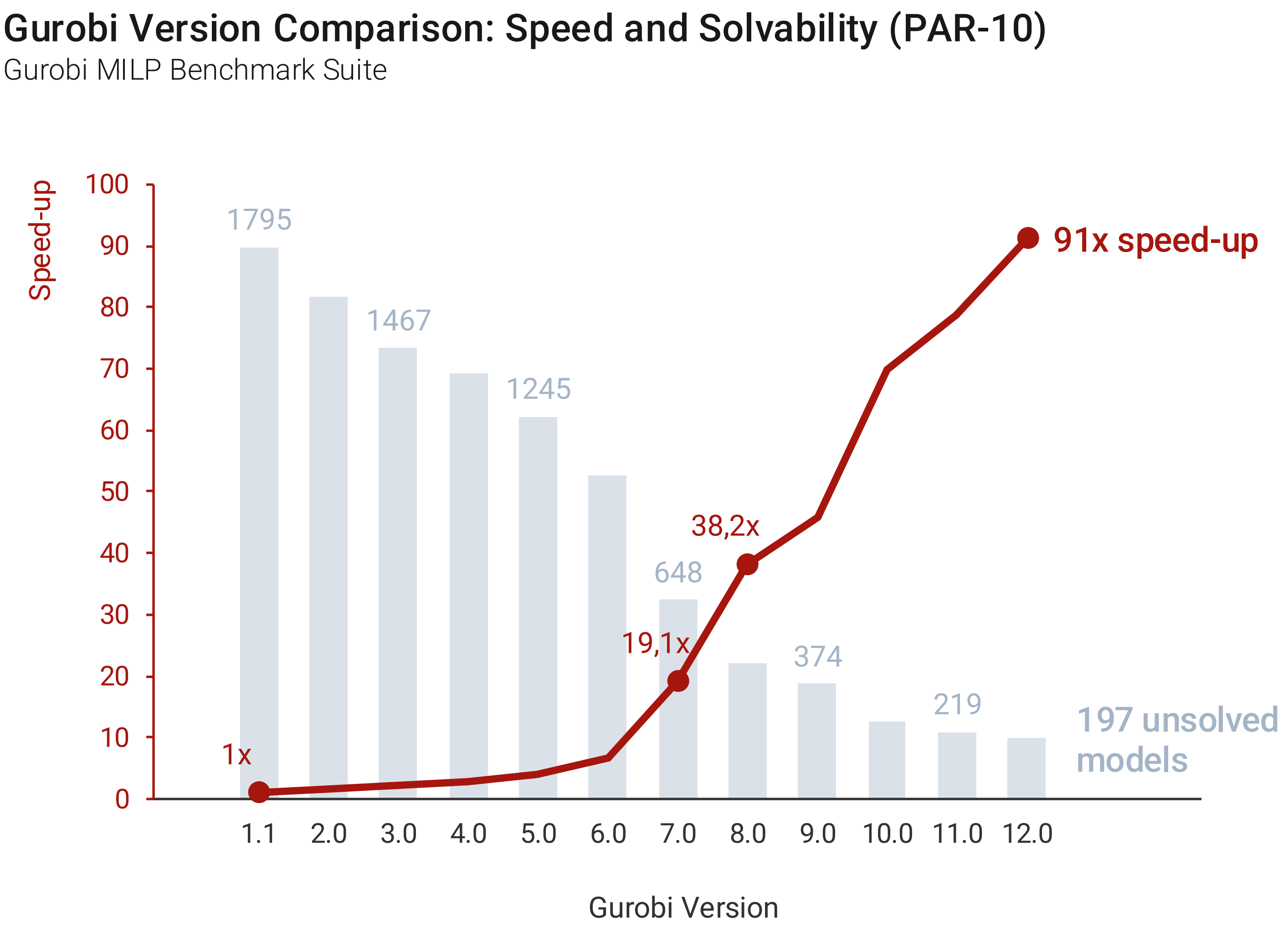
GUROBI OPTIMIZATION

类型	整体提升 (>1 秒)	>100 秒复杂模型
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 提升

- 单纯形 Simplex 7.6%
  - 7个方面的改进 （包括 Harris比率检验和数值计算改进）7 individual improvements (including Harris ratio test and numerical improvements)
- 预优化 Presolve 4.3%
  - 22个方面的改进（包括引申变量缩减）22 individual improvements (including derived variables reduction)
- 切平面 Cutting planes 2.0%
  - 11个方面的改进（包括对偶隐含边界切平面） 11 individual improvements (including dual implied bound cuts)
- 节点预优化 Node presolve 1.6%
  - 5个方面的改进 5 individual improvements
- 并行计算 Parallelism 1.5%
  - 并行同步时截断MIP子问题和节点计算 Preempt sub-MIPs and node LP solves for parallel synchronization
  - 改进超线程的使用 Improved usage of hyper-threads
- 原始启发算法 Primal heuristics 1.2%
  - 4个方面的改进 4 individual improvements
- 分支 Branching 0.9%
  - 更多使用 Driebeek 惩罚而不是强分支 Use Driebeek penalties more often instead of strong branching
- 冲突分析 Conflict analysis 0.3%
  - 不忽略翻转基变量的 Farkas 证明 Do not ignore Farkas proofs with flipped basic variables
- 内存管理 Memory management 0.2%
  - 减少内存分配 Reduced number of memory allocations



MILP

性能演变

在Gurobi 内部 MILP 数据集测试中， 最新版本取得了如下成果:

- 在几何平均(PAR-10)的求解时间上， 比版本1.1提速91倍
- 限定求解时间10,000秒，只有 **197** 个模型未能求解. 测试集包括至少能被一个版本求解出的所有模型。

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 性能提升汇总

GUROBI OPTIMIZATION

类型	整体提升 (>1 秒)	>100 秒复杂模型
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

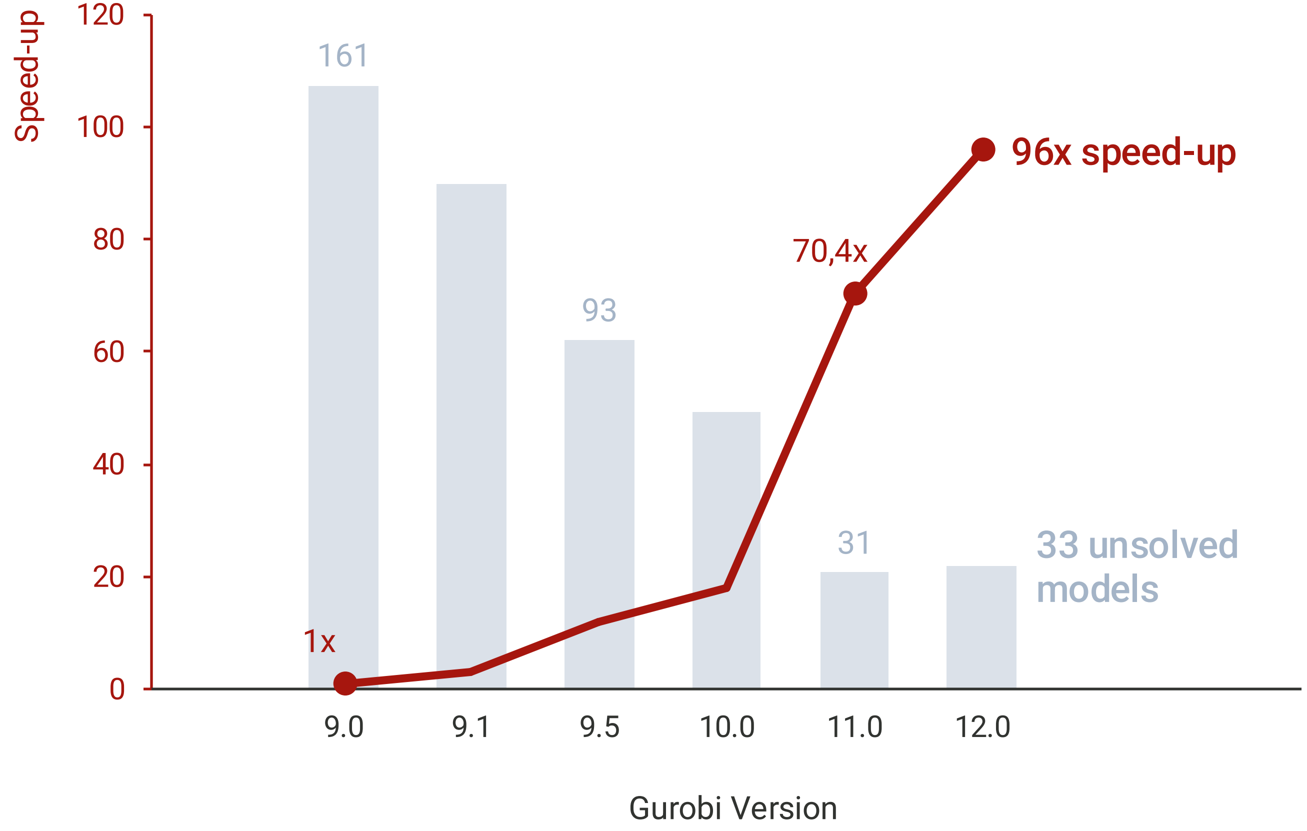
- 节点预优化 Node presolve 10.2%
  - 改进递减成本固定 Improved reduced cost fixing
- 预优化 Presolve 7.1%
  - 改进二次约束到指示约束变换 Improved quadratic constraint to indicator translation
  - 二次项聚合 Quadratic aggregation
- 分支 Branching 4.2%
  - 改进空间分支数值选择 Improved spatial branching value selection
  - 在空间分支中使用更多强分支 Use more strong branching in spatial branching
- 重启 Restarts 2.6%
  - 对于二次和非线性模型引入更积极的重启 More aggressive restarts for quadratic and nonlinear models
- 原始启发算法 Primal heuristics 1.7%
  - 改进在 RINS 之类启发算法中的变量固定策略 Improved fixing strategy in RINS-like heuristic
- 用于非凸 MIQCP 的单纯形提升 Simplex improvements for nonconvex MIQCP 0.7%
  - 停用有数值问题的 McCormick 约束 Deactivate numerically problematic McCormick constraints
- MIP 和单纯形的改进也经常有帮助 (not explicitly measured)

非凸MIQCP

性能演变

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

Gurobi Nonconvex MIQCP Benchmark Suite



在Gurobi 内部非凸MIQCP 数据集测试中， 最新版本取得了如下成果:

- 在几何平均(PAR-10)的求解时间上， 比版本9.0提速96倍
- 限定求解时间10,000秒，只有 **33** 个模型未能求解. 测试集包括至少能被一个版本求解出的所有模型。

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 性能提升汇总

GUROBI OPTIMIZATION

类型	整体提升 (>1 秒)	>100 秒复杂模型
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 提升

无关联子集

- MIP IIS 计算的基本操作
  - 移除一部分约束， 求解剩余子MIP
    - 如果不可行, 抛弃移除的约束; 剩余问题仍然不可行
    - 否则, 恢复移除的约束， 移除其他约束
  - 存在无数个选择移除约束的IIS搜索策略
    - 无关联子集可以帮助发现好的可移除约束
- 模型

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

不可行, 当且仅当至少存在一个

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

不可行

IIS 提升

无关联子集

- 探索无关联子集 Exploit disconnected components 8.1%
  - 存在无关联子集的模型，找到1个不可行子集 For models with disconnected components, find one infeasible component
  - 从最小子集开始尝试 Try smallest component first
  - 限制子集求解的节点数量 Apply node limit for component solves
    - 糟糕情况: 困难但可行的小子集, 非常简单的不可行大子集 Bad case: difficult feasible small component, trivially infeasible larger component
  - 从不可行子系统中抛弃其他子集 Discard other components from infeasible subsystem
- 尽早产生无关联子集 Produce disconnected components earlier 9.5%
  - 变更 IIS 搜索策略以便尽早产生无关联子集 Change IIS search strategy to produce disconnected components earlier
  - 在列交叉图中发现隔断 Find separator in row intersection graph
  - 从隔断处尽量一次移除所有行 Try removing all rows in separator at once
    - 如果不可行, 我们会获得无关联的不可行子系统 If this is infeasible: we got a disconnected infeasible subsystem

IIS 提升

邻域探索

- 在"确定"的IIS 行周围扩大邻域 Grow neighborhood of “definite” IIS rows 3.8%
  - IIS 算法发现"确定"的IIS 行 IIS algorithm identifies “definite” IIS rows:
    - 那些必须属于不可行子系统的行 Rows that need to be member of infeasible subsystem
  - 根据行交叉图扩展邻域直到一定规模 Grow neighborhood in row intersection graph until certain size
  - 尽量移除其他 Try removing everything else
    - 如果不可行, 我们获得一个更小的不可行子系统 If this is infeasible: we got a smaller infeasible subsystem
  - 经常会自动聚焦到更小的子集上 Will often automatically zoom in on small component

全局 MINLP

求解 MINLP问题到全局最优

必要技术

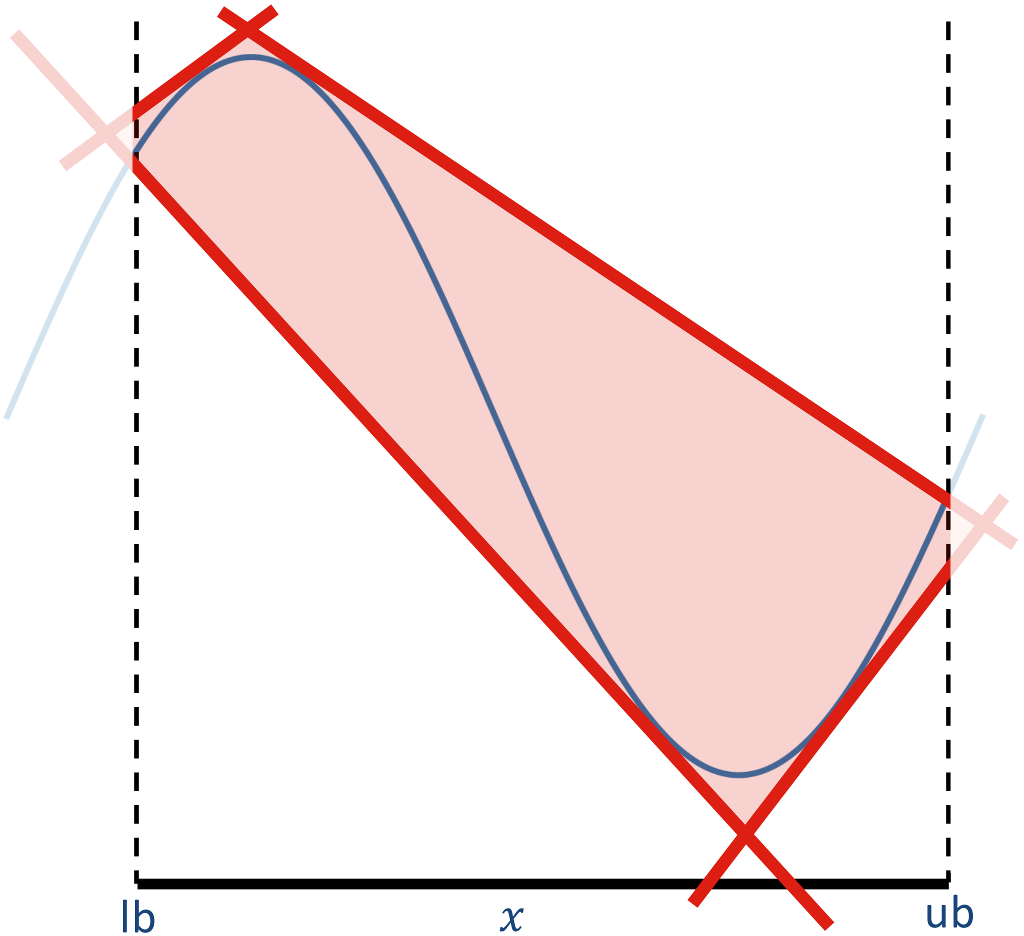
- 外逼近法 Outer approximation
  - 建构 LP 松弛To construct LP relaxation
- 自适应约束 Adaptive constraints
  - 基于搜索树节点局部边界紧缩 LP 松弛To tighten LP relaxation based on local bounds of search tree nodes
- 空间分支定界 Spatial branch-and-bound
  - 解决非线性的约束违反 To resolve nonlinear constraint violations

可选技术

- 预优化和基于优化的边界紧缩算法(OBBT) Presolve and OBBT
  - 缩小变量取值域和改进问题的表达 To tighten domains of variables and improve problem formulation
- 切平面 Cutting planes
  - 紧缩 LP 松弛 To tighten the LP relaxation
- 节点预优化 Node presolve
  - 紧缩搜索树节点变量的局部边界 To tighten local bounds of variables at search tree nodes
- 原始启发算法 Primal heuristics
  - 帮助发现可行解 To help finding feasible solutions
  - 特别: 对于连续NLP用内点法发现局部最优解 In particular: interior point algorithm to get locally optimal solutions for continuous NLPs

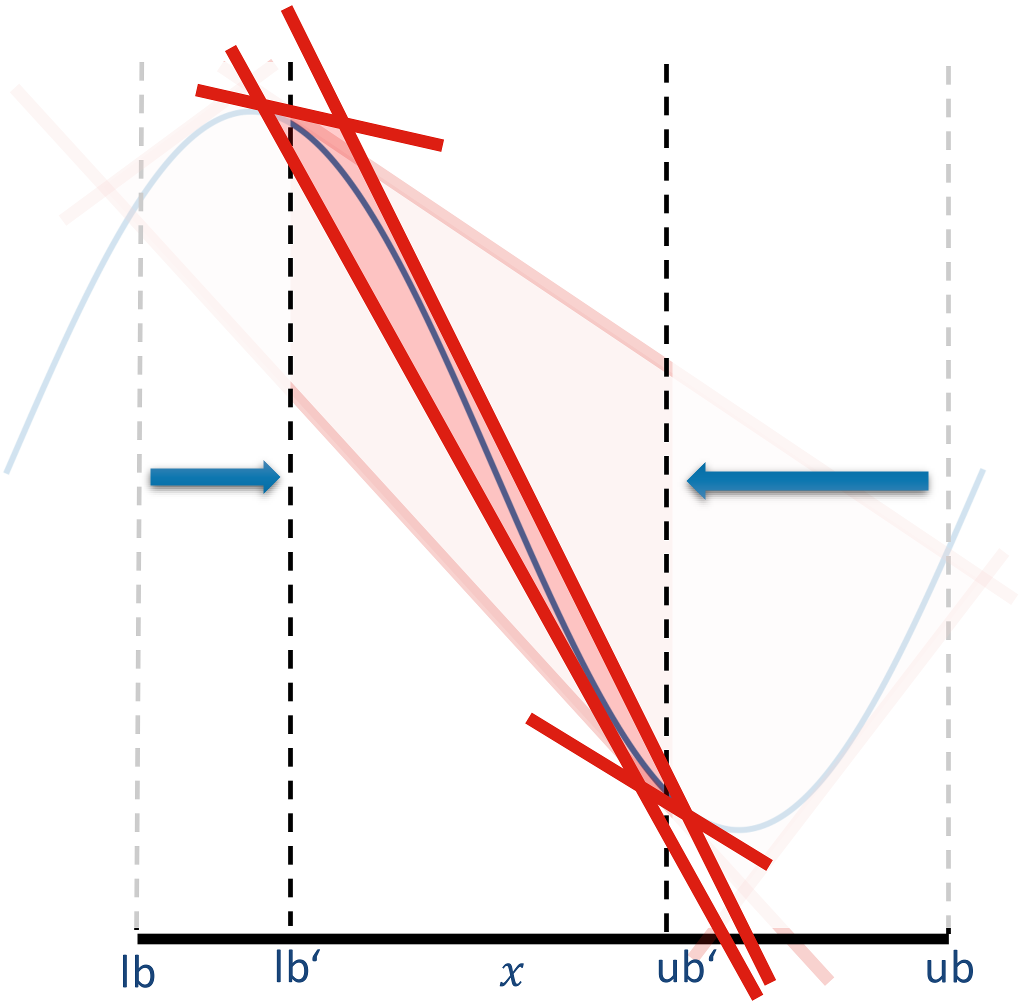
动态外逼近法 Dynamic Outer Approximation

- 非线性约束的凸包的线性松弛 Linear relaxation of convex hull of nonlinear constraint
- 基于变量的全局边界 Based on global bounds of variables
- 采用小数量的切线和割线 Use small number of tangents and secants



动态外逼近法 Dynamic Outer Approximation

- 更紧的边界（局部搜索树的节点） Tighter bounds (at local search tree nodes)
- 获得更紧的外部近似 Yield tighter outer approximation
- 采用自适应约束的自动计算 Automatically calculated using “adaptive constraints”
    - 运行中调整 LP 松弛的系数和右边项 Change coefficients and rhs values in LP relaxation on the fly



广义非线性约束

- Gurobi 9.0 及之后版本提供了单变量广义函数约束的表达方式
  - $y = e^x, y = a^x$
  - $y = \ln(x), y = \log_a(x)$
  - $y = \sin(x), y = \cos(x), y = \tan(x)$
  - $y = x^a$
  - $y = ax^3 + bx^2 + cx + d$

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

- Gurobi 9.0 – 10.0:
  - 函数约束在预优化阶段被分段线性近似所替换
- Gurobi 11.0:
  - 用户可以选择将函数约束用动态外逼近方法准确表达
- Gurobi 12.0:
  - 支持用一个动态外逼近来处理广义非线性约束
    - 多变量复合非线性约束

广义非线性约束

- Gurobi 支持特定单变量函数约束 $y = f(x)$
  - 三角函数, 指数, 对数, 幂指数等
  - 可以用于构造多变量、复合广义非线性约束
- 举例：我们希望建模

$$z = \ln w$$

$$v = \sqrt{u} \qquad w = x + v$$

$$f(x) = \sqrt{1 + x^2} + \ln\left(x + \sqrt{1 + x^2}\right) \leq 2,\ x \geq 0$$

$$u = 1 + x^2$$

- 我们引入辅助变量 $u, v, w, z \geq 0$ 以及如下约束:
  - $u = 1 + x^2,\ u = v^2,\ w = x + v,\ z = \ln w$
  - 那么 $f(x) \leq 2$ 可以表示为 $v + z \leq 2$

分解和可行性容差

$$u - x^2 = 1$$
$$v - \sqrt{u} = 0$$
$$w - x - v = 0 \, ($$
$$z - \ln w = 0$$
$$v + z \le 2$$

$$\Leftrightarrow$$

$$\sqrt{1 + x^2} + \ln\left(x + \sqrt{1 + x^2}\right) \le 2$$

分解和可行性容差

- 每个分解后的约束都受制于可行性容差限制
- 结果可能导致广义非线性约束的总违反量更多

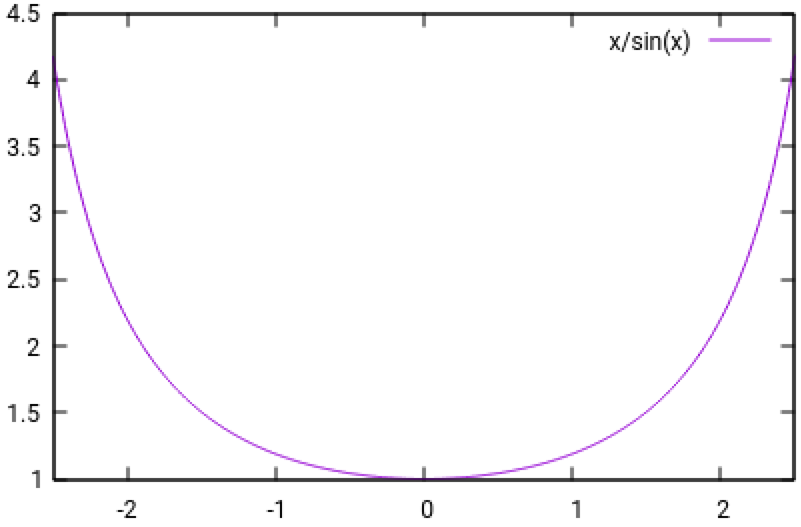
$$u - x^2 = 1\ (\pm\varepsilon)$$
$$v - \sqrt{u} = 0\ (\pm\varepsilon)$$
$$w - x - v = 0\ (\pm\varepsilon)$$
$$z - \ln w = 0\ (\pm\varepsilon)$$
$$v + z \le 2\ (+\varepsilon)$$

$$\not\Leftrightarrow$$

$$\sqrt{1 + x^2} + \ln\left(x + \sqrt{1 + x^2}\right) \le 2\ (+\varepsilon)$$

分解和可行性容差

- 举例: $y = \frac{x}{\sin x}$
- 一个解:
  - $x' = 0.0001$
  - $y' = 1.0000000016666666$



- 分解: $u = \sin x\,, v = u^{-1}, y = x \cdot v$
- 一个解:
  - $x' = 0.0001$
  - $u' = 0.0000999999998333333343$
  - $v' = 10000.000016666666$
  - $y' = 1.0000000016666666$

- 一个满足容差在 10^{-6} 内 的解:
  - $x'' = 0.0001$
  - $u'' = 0.0000989999998333333343$　　$u'' = \sin x'' - 10^{-6}$
  - $v'' = 10101.010118015167$
  - $y'' = 1.010101010118015167$

辅助变量约束中 10^{-6} 的违反量导致了复合约束中 10^{-2} 的违反量

复合约束表现良好在区间 $x \in [-2.5,2.5]$: 没有发生数值问题

分解和可行性容差

- 每个分解后的约束都受制于可行性容差限制
- 结果可能导致广义非线性约束的总违反量更多

GUROBI 11

$$u - x^2 = 1\ (\pm\varepsilon)$$
$$v - \sqrt{u} = 0\ (\pm\varepsilon)$$
$$w - x - v = 0\ (\pm\varepsilon)$$
$$z - \ln w = 0\ (\pm\varepsilon)$$
$$v + z \le 2\ (+\varepsilon)$$

$\not\Leftrightarrow$

GUROBI 12

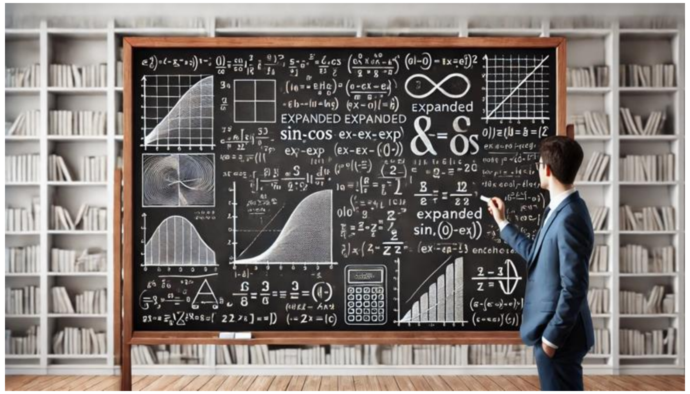
$$\sqrt{1 + x^2} + \ln\left(x + \sqrt{1 + x^2}\right) \le 2\ (+\varepsilon)$$

- 如果用户手动分解 (需要 Gurobi 11)
  - Gurobi 无法知道是否存在底层的复合约束
  - 获得的解只对分解后的模型满足可行性容差
- 如果用户直接使用广义非线性约束 (需要 Gurobi 12)
  - Gurobi 可以检查复合约束的可行性容差，并且抛弃违反的解

Gurobi 12 的非线性约束 API 接口

GUROBI OPTIMIZATION

- 推荐的 Gurobi API 接口: gurobipy
  - 对矩阵友好的数学建模
- 第三方建模工具
  - AMPL
  - GAMS
  - JuMP
  - 其他厂商可能会很快支持 Gurobi 12 非线性约束
- 基于数组的通用语言 API
  - C
  - C++
  - Java
  - .NET
  - LP and MPS files



gurobipy 中的非线性表示

- gurobipy 中有一个新的非线性对象 NLExpr

```
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  # NLExpr
expr4 = x / y      # NLExpr

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 ≥ 0 instead

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

gurobipy 示范: AC 最优电流 AC Optimal Power Flow

输入数据和向量变量

```
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 示范: AC 最优电流 AC Optimal Power Flow

非线性矩阵约束和Numpy 广播方式

```
# 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()
```


gurobipy 示范: AC 最优电流 AC Optimal Power Flow

非线性矩阵约束和Numpy 广播方式

```
# 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()
```

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

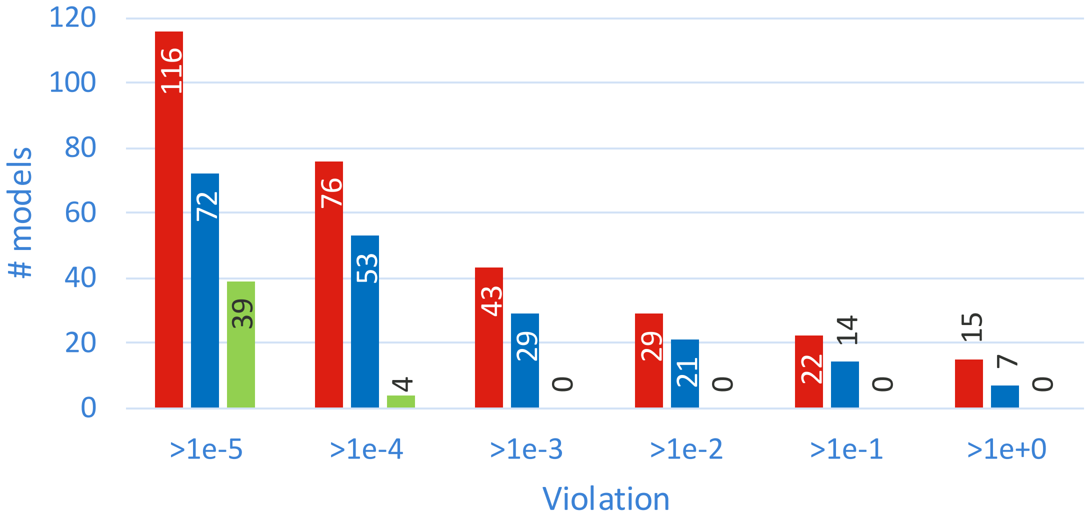
GUROBI
OPTIMIZATION

全局 MINLP

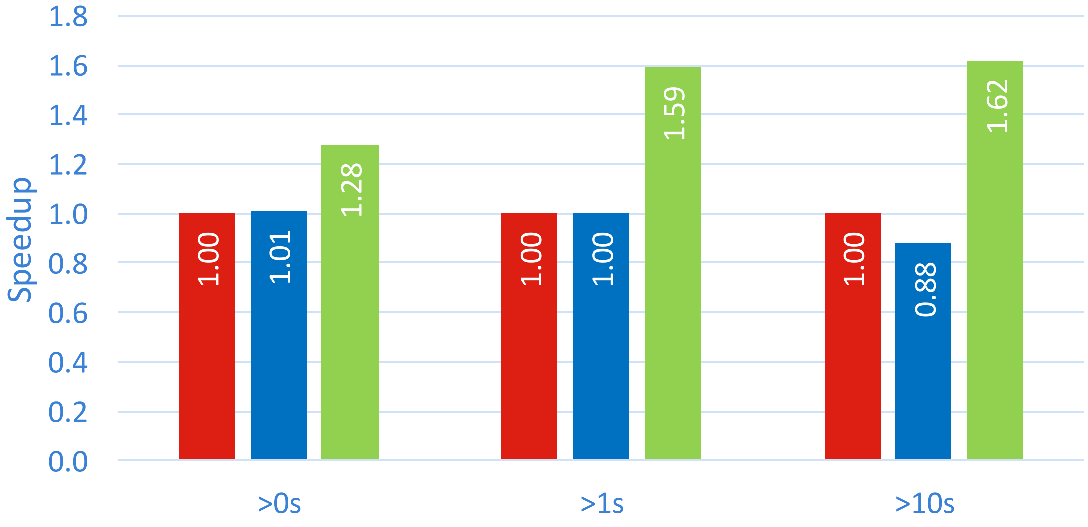
求解质量和速度

- Gurobi 11:
  - 函数约束（单变量）模型
- Gurobi 12:
  - 函数约束（单变量）模型
  - 非线性约束（多变量）模型

- 更好的求解质量: v11 < v12 单变量 < v12 多变量



- 单变量速度相当, 多变量速度更快



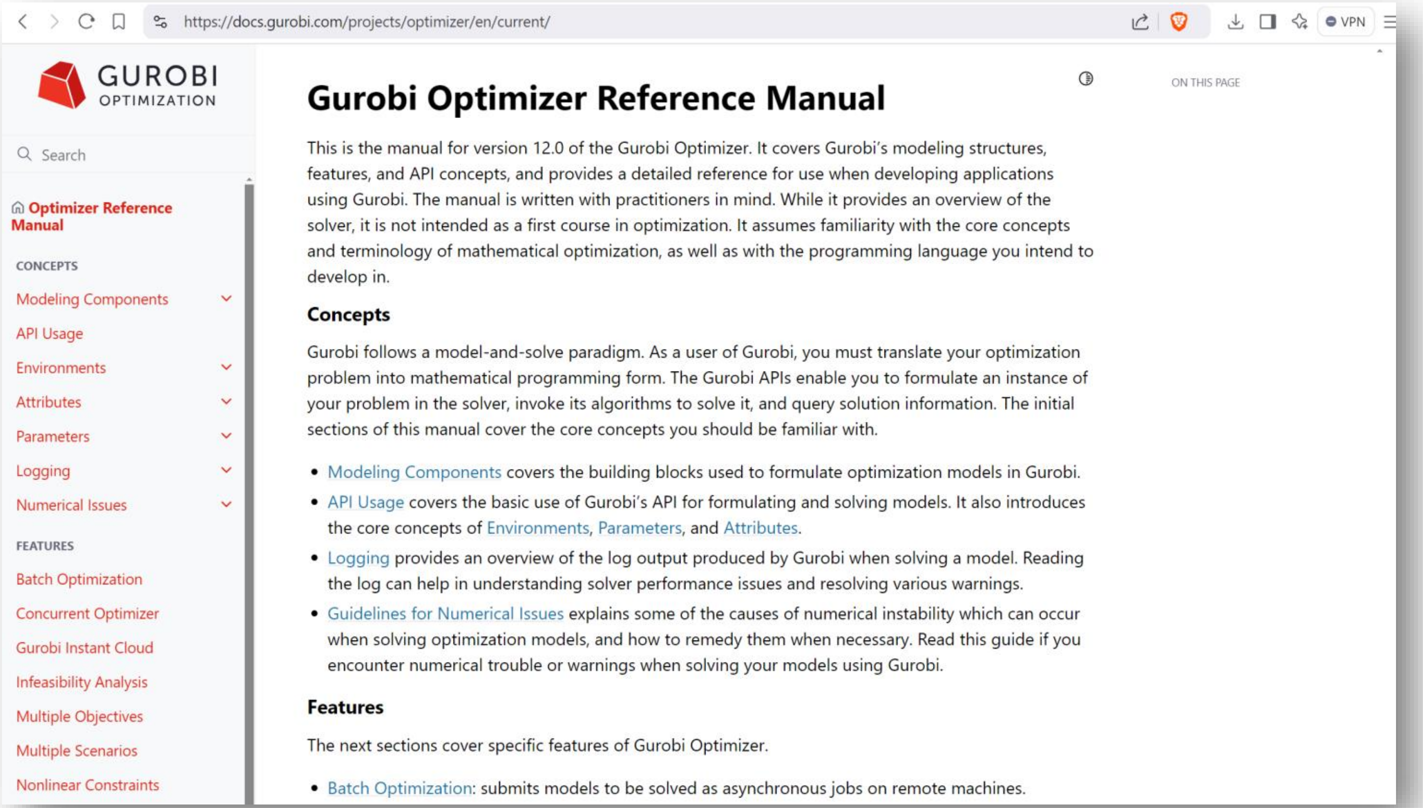
使用文档

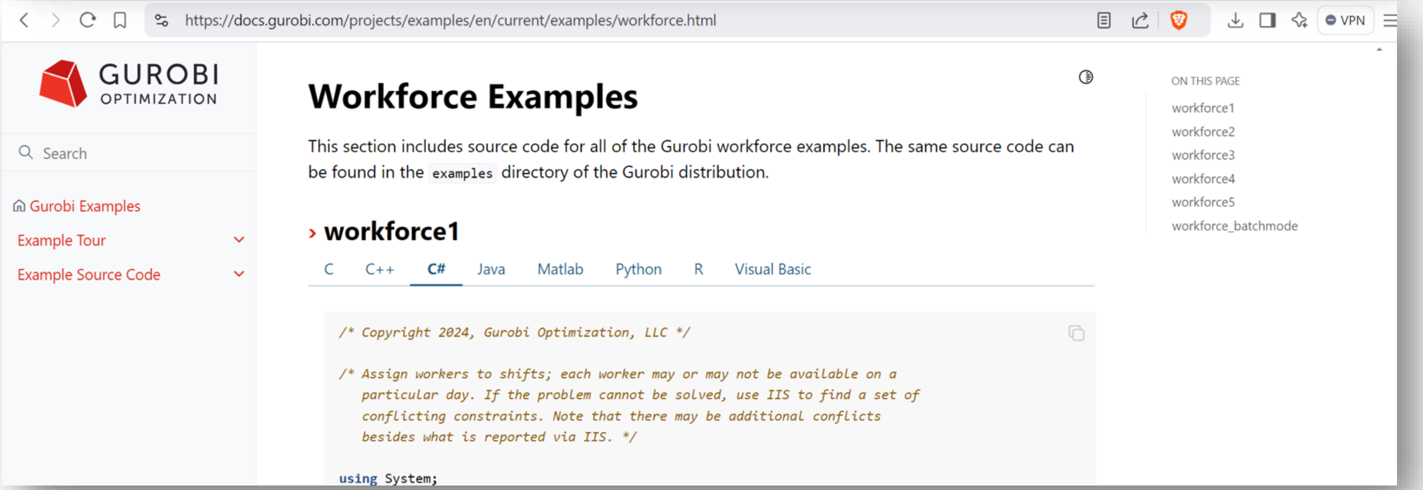
GURUBI OPTIMIZATION

新文档系统

- 全部更新的文档系统
- 基于 Sphinx
- 托管在 readthedocs

- 更符合现代风格的设计
- 更好的结构，更多交叉引用
- 各种语言的示范代码以分页形式排列
- 更容易维护
  - 持续不断地补充和修订





其他功能

回调功能 Callbacks

- 对于运算服务器许可支持在回调函数中设置可行解
  - 传递用户启发式结果给优化过程
  - 例如修补违反惰性约束的解
- 关于多目标回调函数的更多信息
  - 优化状态 Solving status
  - 运行时间 runtime
  - 工作 work
  - 迭代数量 iteration count
  - 节点数量 node count
  - 剩余节点数量 number of nodes left
  - 当前可行解 incumbent value
  - 对偶边界 dual bound
  - MIP间隙值MIP gap
  - 对于最近目标函数提供子模型的信息

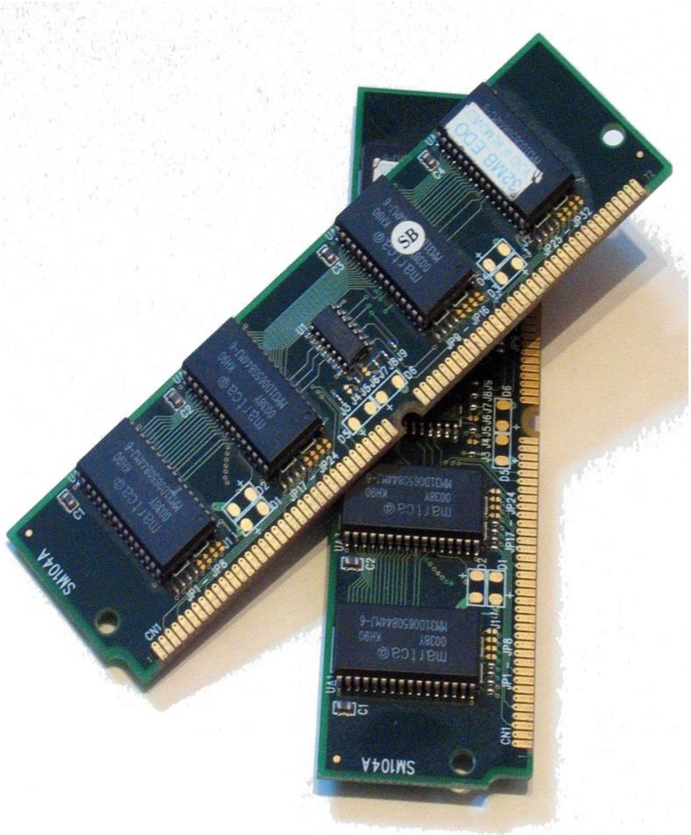
- 允许在回调中设定的优化参数
  - TimeLimit
  - WorkLimit
  - NodeLimit
  - BarIterLimit
  - PumpPasses
  - 简化用户终止条件

调参工具Tuning Tool

- 限定调参在有限参数之内
  - 例如仅调参切平面参数和启发算法参数
- 对多目标模型调参

GUROBI OPTIMIZATION

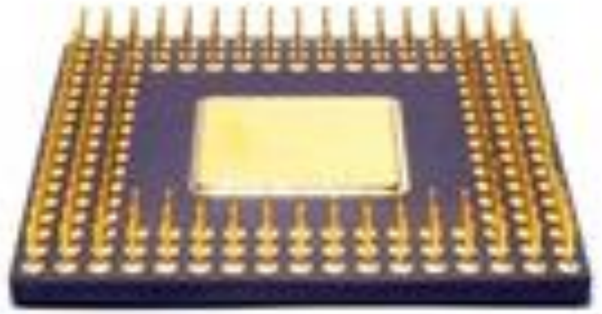
硬件资源管理



内存管理

- 分配更少内存
  - 避免由于内存碎片导致的内存不足问题
  - 对于复杂MIP 问题产生了0.5% 速度提升
- 对于规模较大的解池消耗更少的内存
  - 对解池向量的稀疏存储
  - 复杂MIP问题有1.4% 速度提升
    - 设定 PoolSearchMode=2 和 PoolSolutions=1000
    - 46 内存限度 → 35 内存限度
- 查询内存消耗的属性
  - 当前和最大内存消耗
  - 优化中 (通过回调功能) 和优化后

GUROBI OPTIMIZATION



CPU利用率

- 更多根节点的并行处理
  - 在根节点用多线程运行 MIP 子模型策略
  - 叠加在并行根节点切平面循环之上
  - 0.4% 整体性能提升 (4 核)
- 更好的MIP 搜索树并发
  - 截断过长时间的启发算法和节点计算
  - 在MIP同步后继续启发算法和节点LP计算
  - 1.4% 整体性能提升 (4 核)

GUROBI OPTIMIZATION

欢迎提问