GUROBI
OPTIMIZATION

Hidden Gems

Useful Gurobi Features you may not know

In this session, we will review the latest features of version 11.0. In addition to exciting performance improvements, we'll highlight powerful features that make the model building and solving process easier. We will provide an overview of tools that help make it easy to incorporate multiple objectives, scenario analysis, nonlinear functions, and more. Learning how to correctly use these features can streamline your modeling efforts, inspire new application ideas, and help you present the results of your work to upper management.

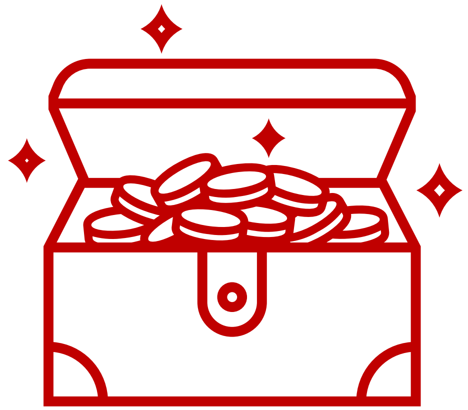
Outline

Hidden Gems

Useful Gurobi Features you may not know

Modeling

- Multiple Objectives
- Multiple Scenarios
- Solution Pool
- General Constraints
- Infeasibility Analysis



Performance

- Variable Start & Hint Values
- NoRel

GitHub

- GRBlogtools
- Gurobi Machine Learning
- Gurobi's Ill Conditioning Explainer
- gurobi-pandas

Hidden Gems: Modeling

Multiple Objectives

- Real-world optimization problems often have multiple, competing objectives

Maximize Profit
&
Minimize Late Orders

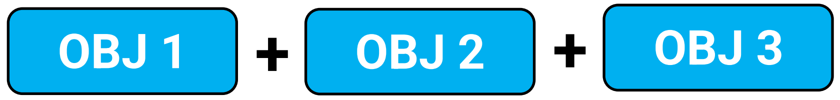
Minimize Shift Count
&
Maximize Worker Satisfaction

Minimize Cost
&
Maximize Product Durability

Maximize Profit
&
Minimize Risk

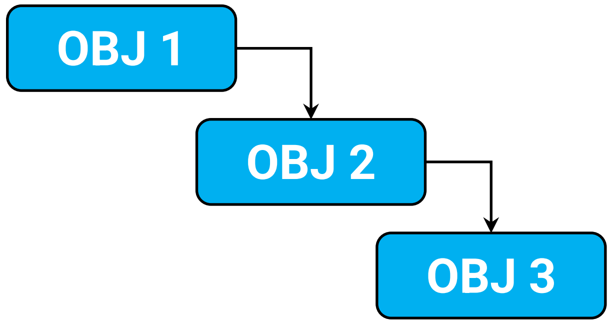
How does Gurobi handle the trade-offs?

- **Weighted:** Optimize a weighted combination of the individual objectives



$$\begin{aligned} \min \quad & w_1 f_1(x) + w_2 f_2(x) + w_3 f_3(x) \\ \text{s.t:} \quad & x \in \mathcal{C} \end{aligned}$$

- **Hierarchical (Lexicographical):** Optimize each objective in a priority order given while limiting the degradation of the higher-priority objectives



$$\begin{aligned} \min \quad & f_1(x) \\ \text{s.t:} \quad & x \in \mathcal{C} \end{aligned}$$

$$\begin{aligned} \min \quad & f_2(x) \\ \text{s.t:} \quad & x \in \mathcal{C} \\ & f_1(x) \leq \epsilon_1 \end{aligned}$$

$$\begin{aligned} \min \quad & f_3(x) \\ \text{s.t:} \quad & x \in \mathcal{C} \\ & f_1(x) \leq \epsilon_1 \\ & f_2(x) \leq \epsilon_2 \end{aligned}$$

- **Weighted + Hierarchical**

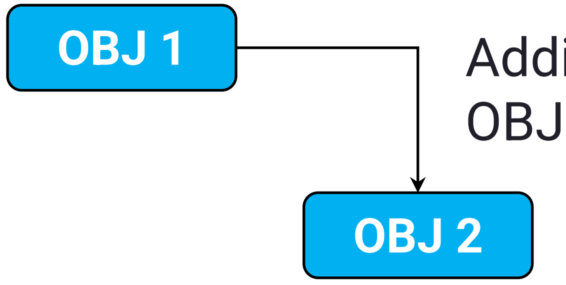
Multiple Objectives API

- Python API

```
Model.setObjectiveN(LinExpr, index, priority=0, weight=1,
                    abstol=1e-6, reltol=0, name="" )
```

- `LinExpr`: The objective expressions should be linear
- `index`: Index for new objective (Used to set different parameters/query the solution per objective)
- `priority`: Objectives' priority (`ObjNPriority` attribute)
- `weight`: Objectives' weight (`ObjNWeight` attribute)
- *`abstol`*: Absolute tolerance used in calculating the allowable degradation (`ObjNAbsTol` attribute)
- `reltol`: Relative tolerance used in calculating the allowable degradation (`ObjNAbsTol` attribute)

How is the degradation value calculated?



Additional Constraint:
OBJ 1 <= rhs

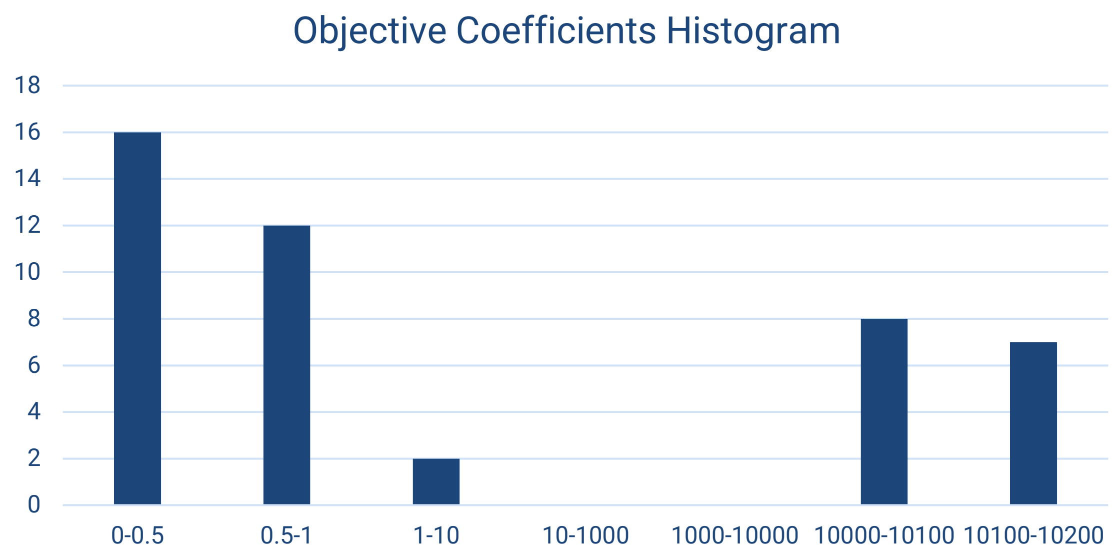
$$\begin{aligned} \min \quad & f_2(x) \\ \text{s.t:} \quad & x \in \mathcal{C} \\ & f_1(x) \le \epsilon_1 \end{aligned}$$

```
base_value = max(objbnd +|objval|*MIPGap, objbnd + MIPGapAbs, objval)
relaxed     = max(ObjNRelTol*|base_value|, ObjNAbsTol)
```
rhs (ϵ_1) = base_value + relaxed

```
objbnd     : best bound of objective OBJ 1
objval     : best solution value for objective OBJ 1
MIPGap     : relative MIP gap
MIPGapAbs  : absolute MIP gap
ObjNRelTol: further allowable relative degradation for OBJ 1
ObjNAbsTol: further allowable absolute degradation for OBJ 1
```

Why use the Multiple Objectives API?

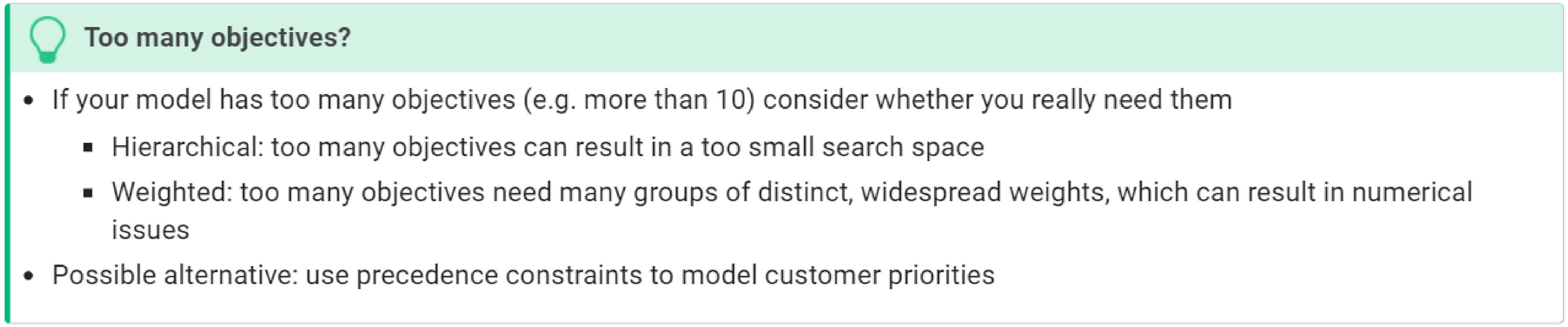
- Make the objective functions easy to understand and maintain
- Get faster performance with warm starts for hierarchical objectives
- Multiple objectives can help avoiding numerical issues with large objective coefficients
  - Soft constraints with large penalty variables



There are 45 coefficients in 2 distinct groups.
Is this a multi-objective case in hiding?

Multiple Objectives API - Details

- There is a single objective sense controlled via the `ModelSense` attribute
- Objective function expressions must be linear
- Different parameters can be set for each objective pass via multi-objective environments
- Callbacks are available



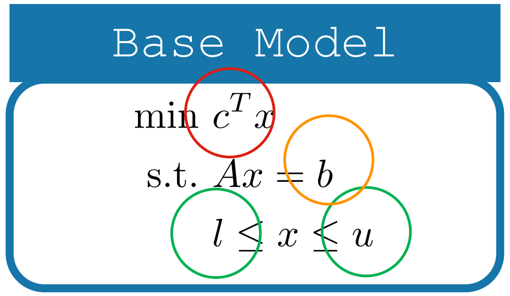
Too many objectives?

- If your model has too many objectives (e.g. more than 10) consider whether you really need them
  - Hierarchical: too many objectives can result in a too small search space
  - Weighted: too many objectives need many groups of distinct, widespread weights, which can result in numerical issues
- Possible alternative: use precedence constraints to model customer priorities

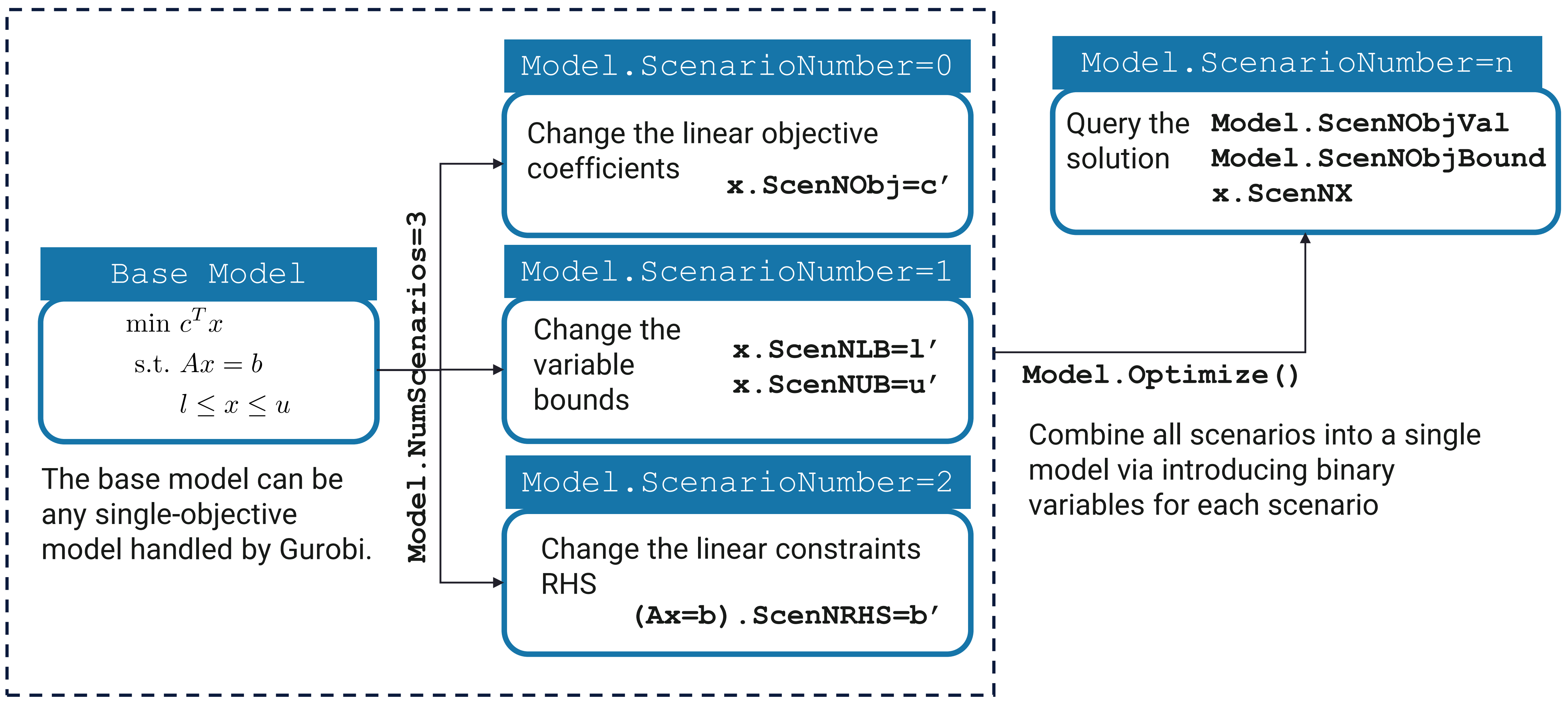
Multiple Scenarios

- In many real-world applications, the following may occur:
  - The input data is not accurate
  - The input data is not known in advance and can take multiple values in real time
  - The input data is seasonal or periodic
  - The input data has a range of possible values

- The Gurobi Optimizer includes scenario analysis features which are useful to understand the sensitivity of the computed solution with respect to changes in the inputs:
  - Linear objective function coefficients
  - Variable lower and upper bounds
  - Linear constraint right-hand side values



Multiple Scenarios API



`Base Model`

$$\min\ c^T x$$
$$\text{s.t. } Ax = b$$
$$l \le x \le u$$

The base model can be any single-objective model handled by Gurobi.

`Model.NumScenarios=3`

`Model.ScenarioNumber=0`

Change the linear objective coefficients `x.ScenNObj=c′`

`Model.ScenarioNumber=1`

Change the variable bounds `x.ScenNLB=l′` `x.ScenNUB=u′`

`Model.ScenarioNumber=2`

Change the linear constraints RHS `(Ax=b).ScenNRHS=b′`

`Model.Optimize()`

Combine all scenarios into a single model via introducing binary variables for each scenario

`Model.ScenarioNumber=n`

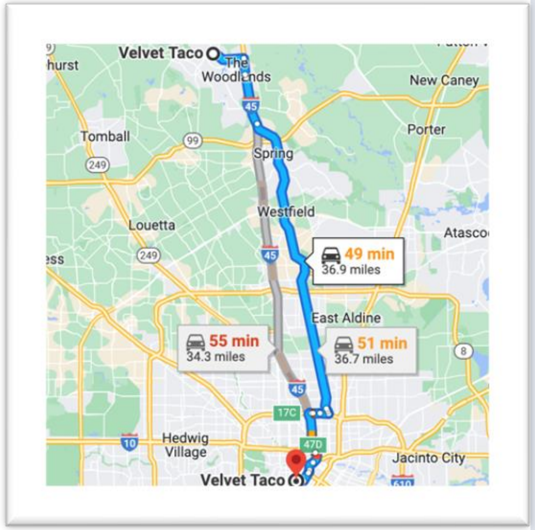
Query the solution `Model.ScenNObjVal` `Model.ScenNObjBound` `x.ScenNX`

Multiple Scenarios (Tips & Tricks)

- The multiple scenarios API is restricted. For example, it is not possible to explicitly
  - Add/remove variables or constraints
  - Change the variable types
  - Change the sense of constraints
  - …
- However, we can circumvent some of the restrictions using useful tricks
  - To remove a variable, set its bounds to zero
  - To add a variable to a scenario, add it to the base model with bounds set to zero and then change the bounds accordingly
  - To remove a constraint, change its RHS values to GRB.INFINITY/-GRB.INFINITY
  - To add a constraint to a scenario or change its sense, add it as a pair of inequalities to the base model and change its RHS values accordingly

Multiple Solutions

- You may want to report several solutions, not just the optimal solution
  - The model may lack implicit elements like preferences, or some aspects of the objective may be difficult to quantify
  - Demonstrate value by comparing alternatives to the optimal solution
  - Gives a greater feeling of control
  - Get feedback, may learn about missing model elements if an alternate solution should have been the optimal one based on real-world knowledge



- How can you quickly report several feasible solutions?
  - ~~Re-run the model with a constraint manually added to cut off the previously reported solution and get a new one~~
  - Define a **Solution Pool** and report multiple solutions automatically, efficiently after a single run

- Note there are some subtleties and limitations. e.g., continuous variables - multiple equivalent solutions will not be reported per our definitions.

Solution Pool Setup

Controlled via model parameters (documentation)

Parameter Settings	Behavior
`PoolSearchMode = 0`	Stores all solutions found in the regular optimization. No additional tree search performed.

`PoolSearchMode = 1`
`PoolSolutions  = n`
	Stores n-1 **additional solutions** to the optimal solution.
	PoolSolutions Controls how many solutions to save.

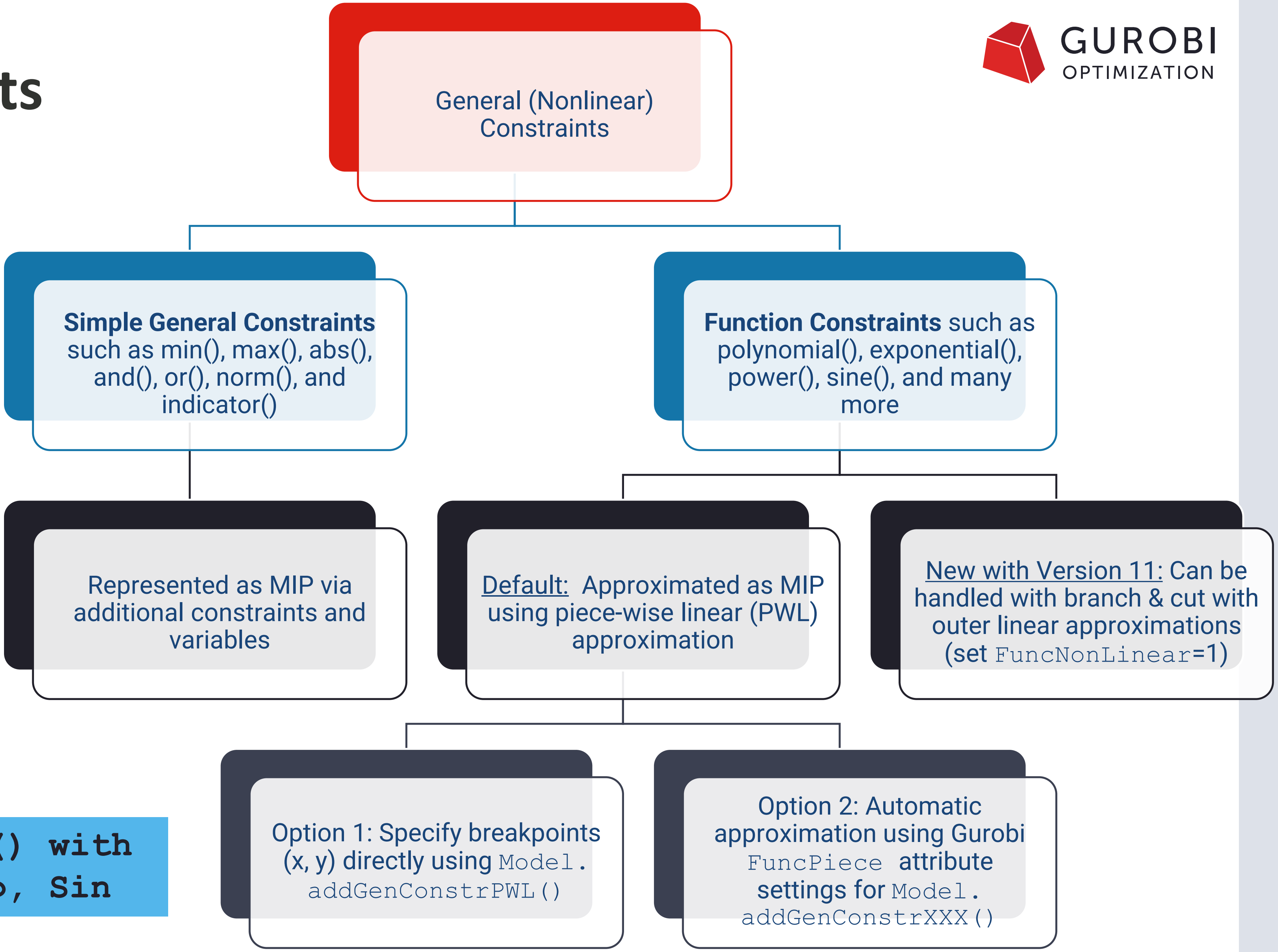
`PoolSearchMode = 2`
`PoolSolutions  = n`
`PoolGap        = x`
	Stores n-1 **best solutions** with a MIPGap less than x% in addition to the optimal solution.
	Requires exploring the tree search more than setting PoolSearchMode=1.

```
# Limit how many solutions to collect
model.setParam(GRB.Param.PoolSolutions, 100)

# Limit the search space by setting a gap for the worst possible solution that will be accepted
model.setParam(GRB.Param.PoolGap, 0.10)

# do a systematic search for the k-best solutions
model.setParam(GRB.Param.PoolSearchMode, 2)
```

General Constraints

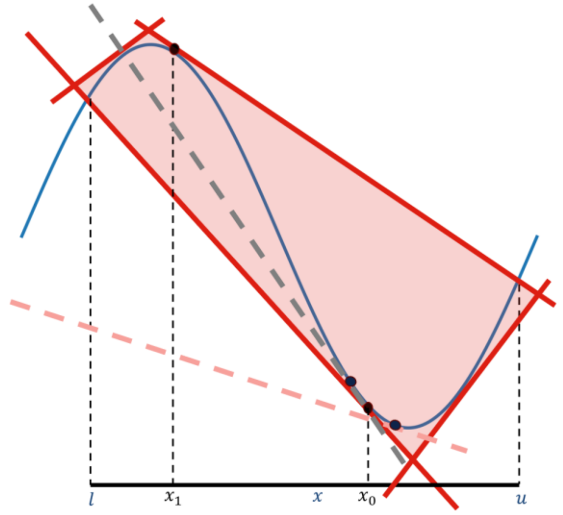
- Gurobi supports two types of general constraints
- There are different strategies for the algorithmic implementation…
- …but the API is consistent

```
Model.addGenConstrXXX() with
XXX = Max, Min, …, Exp, Sin
```

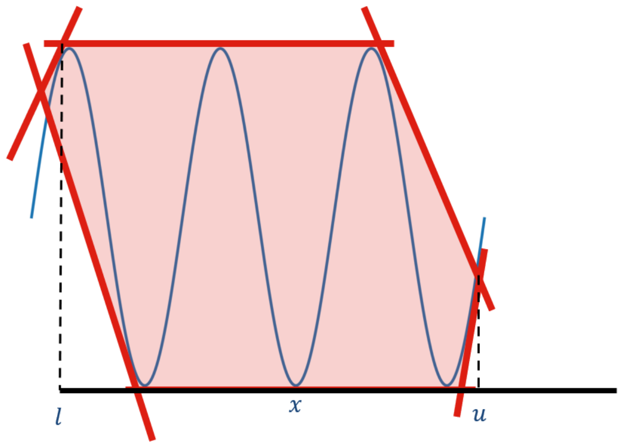

General (Nonlinear) Constraints
Simple General Constraints such as min(), max(), abs(), and(), or(), norm(), and indicator()
Function Constraints such as polynomial(), exponential(), power(), sine(), and many more
Represented as MIP via additional constraints and variables
Default: Approximated as MIP using piece-wise linear (PWL) approximation
New with Version 11: Can be handled with branch & cut with outer linear approximations (set FuncNonLinear=1)
Option 1: Specify breakpoints (x, y) directly using Model.addGenConstrPWL()
Option 2: Automatic approximation using Gurobi FuncPiece attribute settings for Model.addGenConstrXXX()

Function Constraints with Outer Approximations

- Available with Gurobi Version 11
- Derives hyperplane cuts to add to LP relaxation. Adding more tangents at various points improves the relaxation.
- Options
  - **`FuncNonlinear =  1`**
    `(enable Non-Linear Constraint)`
  - **`FuncNonlinear = -1`**
    `(default, PWL approximation)`
  - Constraint Attribute: Applied to a specific function constraint
  - Parameter: Applied to all function constraints



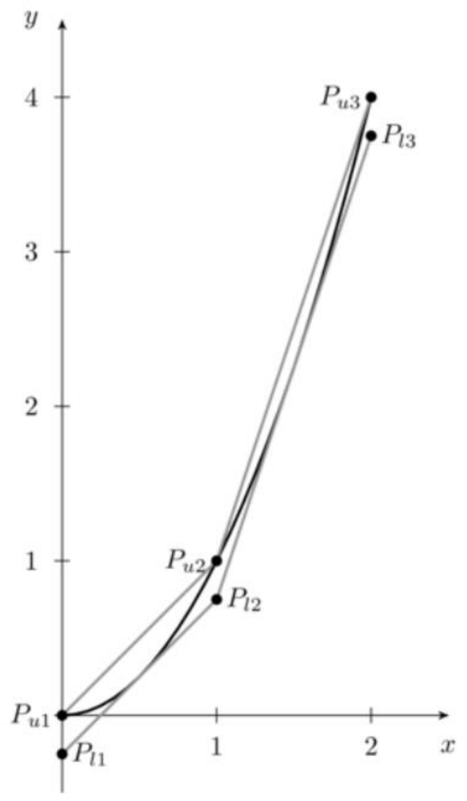
Note: Branching on x tightens the relaxation quickly!



Tighter initial bounds will speed up performance

Options for Automatic PWL Translation

- Gurobi API for function constraints automatically approximates them as PWL constraints
  - Using actual functions during presolve
  - Bound strengthening in presolve for potentially more efficient PWL translation
- Options
  - `FuncPieces`, `FuncPieceLength`, `FuncPieceError` – there is a speed vs. accuracy tradeoff when choosing piece length, number of pieces, or maximum allowed error
  - `FuncPieceRatio` – Choices for having the approximation as an underestimate, overestimate, or somewhere in between of the actual function
  - Constraint Attributes: Applied to a specific function constraint
  - Parameters: Applied to all function constraints



Gurobi API for Function Constraints

Example

$$\min \quad -2x + \boxed{e^x}$$
$$\text{s.t:} \quad 0 \leq x \leq 1$$

```
model = gp.Model(”gen")
x = model.addVar(lb=0, ub=1, name="x")
y = model.addVar(name="y")
gc = model.addGenConstrExp(x, y)
model.setObjective(-2 * x + y)
model.optimize()
```

Supported Function Constraints

- Polynomial
- Natural exponential
- Exponential
- Logarithm
- Logistic
- Power
- Sine
- Cosine
- Tangent

- How to decide which option should you use?
  - **`FuncNonlinear =  1`**
    `(enable Non-Linear Constraint)`
  - **`FuncNonlinear = -1`**
    `(default, PWL approximation)`

Infeasibility Analysis

- Why the model is infeasible?
  - Compute an Irreducible Inconsistent (Infeasible) System (IIS)
- What changes do I need to make to recover feasibility?
  - Compute the smallest perturbation needed to recover feasibility

```
Gurobi Optimizer version 10.0.1 build v10.0.1rc0
…
Optimize a model with 14 rows, 72 columns and 72 nonzeros
...
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    4.6400000e+02   4.400000e+01   0.000000e+00      0s

Solved in 1 iterations and 0.00 seconds (0.00 work units)
Infeasible model
```

workforce1.py example in Gurobi Python examples

Irreducible Inconsistent System (IIS)

- Given an infeasible system of constraints
  - Find a subset of constraints/variable bounds that
    - It is infeasible
    - Removing a single constraint/bound makes it feasible
  - IIS is minimal and not minimum
- Meant to be read and analyzed by a human
  - The smaller, the better
- Computational complexity
  - Cheap for LP and expensive for MIP

API

```
if  model.Status == GRB.INFEASIBLE:
      model.computeIIS()
      model.write("iis.ilp")
```

ILP File Format

```
\ Model assignment_copy
\ LP format - for model browsing. Use MPS
format to capture full model detail.
Minimize

Subject To
 Thu4: x[Cathy,Thu4] + x[Ed,Thu4] = 4
Bounds
 -infinity <= x[Cathy,Thu4] <= 1
 -infinity <= x[Ed,Thu4] <= 1
End
```

Options for IIS

- Method used to compute IIS
  - `IISMethod` as a solver parameter

- User control to guide IIS computation
  - Attributes to either include or exclude constraints/bounds from the IIS
    - `IISConstrForce`, `IISLBForce`, `IISUBForce`, `IISSOSForce`, `IISQConstrForce`, `IISGenConstrForce`
  - Useful in identifying which changes made an already feasible model infeasible

Feasibility Relaxation

- The feasibility relaxation model minimizes the amount by which the violation of bounds and the linear constraints of the original model is minimized
- The violation is measured with respect to
  - Number of violations (0-norm)
  - Sum of the violations (1-norm)
  - Sum of the squares of violations (2-norm)

Infeasible model

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

Feasibility relaxation

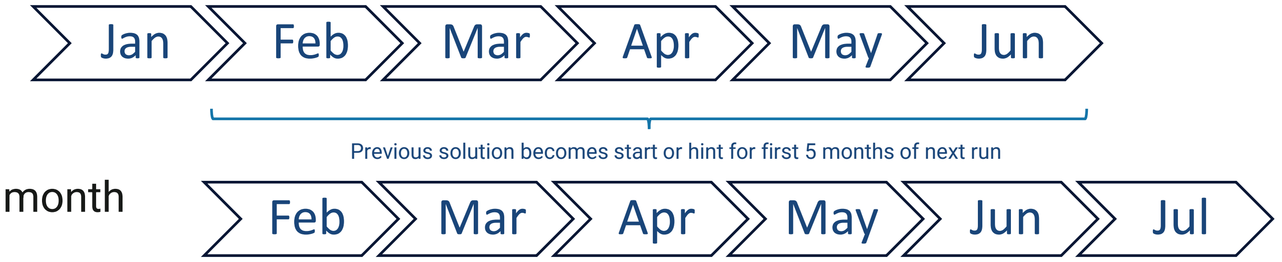
$$\begin{aligned} \min \quad & \|(s, u)\|_p \\ \text{s.t.} \quad & Ax - s \leq b \\ & x + u \geq 0 \\ & s, u \geq 0 \end{aligned}$$

- There are two different APIs:
  - `feasRelaxS(relaxobjtype, minrelax, vrelax, crelax)` (**Python only**)
  - `feasRelax(relaxobjtype, minrelax, vars, lbpen, ubpen, constrs, rhspen)`

Hidden Gems: Performance

Variable Start & Hint Values

- Take advantage of previous solutions & model insight to improve performance
  - Knowledge of some variable values may be available from previous solves
  - Example: Rolling horizon planning application
    - Run 1: 6mo plan
    - Run 2:
      Redo plan starting in 2nd month


Jan
Feb
Mar
Apr
May
Jun
Previous solution becomes start or hint for first 5 months of next run
Feb
Mar
Apr
May
Jun
Jul

- Idea: Reduce solve times by specifying these values in the solver
  - There are 2 options for how to provide this information
    - **Start values**: to generate an initial solution. (Full or partial MIP starts can be used)
    - **Variable hints**: to influence the MIP search

Variable Start & Hint Values – Candidates

- Values from prior solves are most common
- Other candidates
  - Preferences: Use the most efficient resource
  - Heuristics: Apply use case insight
  - Penalties: Avoid an expensive penalty resource
  - Symmetry: Pick one value as a start
  - Only the objective changes
  - Only new variables are added
- Values are specific to the model

```
# Guess at the starting point: close the plant with
the highest fixed costs;
# open all others

# First open all plants
for p in plants:
    open[p].Start = 1.0

# Now close the plant with the highest fixed cost
print('Initial guess:')
maxFixed = max(fixedCosts)
for p in plants:
    if fixedCosts[p] == maxFixed:
        open[p].Start = 0.0
        print('Closing plant %s' % p)
        break
```

Variable Start & Hint Values – Comparison

Start Values

- **Generate initial integer solution**, which is improved via MIP search
- **Can specify partial solution**, to be completed by solver (typically don't specify 0 values)

- **Controlled** via `Start` variable attribute (or load a .mst MIP start file)
- **Supports multiple start values** via `NumStart` model attribute and `StartNumber` parameter

Variable Hints

- **Guide MIP search** toward anticipated values
- **Can specify hints for subset of integer variables**, to be used by solver (albeit with less guidance)

- **Controlled** via `VarHintVal` variable attribute
- **Express your confidence for each hint** via `VarHintPri` variable attribute
- **Supports only one hint per variable**

NoRel Motivation

Sometimes default MIP solve is not perfect

Slow root relaxation

```
 9442783    9.8014026e+10   8.192461e+03   0.000000e+00  39779s
 9451658    9.8014026e+10   7.339784e+03   0.000000e+00  39786s
 9470156    9.8014026e+10   7.289652e+03   0.000000e+00  39793s
 9476934    9.8014026e+10   6.589656e+03   0.000000e+00  39800s
 9488697    9.8014026e+10   6.595012e+03   0.000000e+00  39819s
 9495174    9.8014027e+10   0.000000e+00   0.000000e+00  39903s
 9495280    9.8014027e+10   1.788495e+01   0.000000e+00  39920s
 9495293    9.8014027e+10   0.000000e+00   0.000000e+00  39936s

Root relaxation: objective 9.801403e+10, 9495293 iterations, 37539.78 seconds (35149.49 work units)
```

Slow node progress / Very weak model

```
    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0    0.00000    0 1762  168.00000    0.00000  100%     - 4505s
     0     0   -0.00000    0 1623  168.00000    0.00000  100%     - 6380s
     0     0   -0.00000    0 1776  168.00000    0.00000  100%     - 7788s
H    0     0                     162.0000000    0.00000  100%     - 13016s
     0     0    0.00000    0 2751  162.00000    0.00000  100%     - 19249s
     0     0    0.00000    0 2644  162.00000    0.00000  100%     - 23132s
     0     0    0.00000    0 2624  162.00000    0.00000  100%     - 26440s
     0     0    0.00000    0 2480  162.00000    0.00000  100%     - 28800s
```

Slow/no feasible solutions found

```
    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 -5385.7647    0 13339          - -5385.7647      -     -   28s
     0     0 -5277.9010    0 15005          - -5277.9010      -     -   81s
   ...
     0     2 -4610.2942    0 14077          - -4610.2942      -     - 2142s
     1     4 -4354.7802    1 14100          - -4608.2243      -  8993 2149s
     3     8 -4116.7369    2 13907          - -4593.4772      -  5149 2191s
     7    12 -4078.5283    3 14300          - -4492.2159      -  4986 2200s
    11    14 -4082.4056    3 14166          - -4491.9484      -  4514 2215s
    15    18 -4064.6967    4 14256          - -4491.9484      -  3672 2243s
    19    22 -3829.7080    4 14160          - -4491.9484      -  5519 2248s
   ...
   993   531 -3157.0875   25 13633          - -4185.2719      -  1190 2638s
```

Metaheuristics offer a different approach

Examples of metaheuristics

- Ant Colony Optimization
- Genetic Algorithms
- Evolutionary Algorithms
- Particle Swarm Optimization
- Very Large-Neighborhood Search
- Simulated Annealing

Metaheuristics:

1. Try to find high-quality feasible solutions
2. Can keep running forever

In practice, implementing a metaheuristic yourself can be:

1. Extremely time consuming.
2. Difficult to maintain.
3. Difficult to extend.

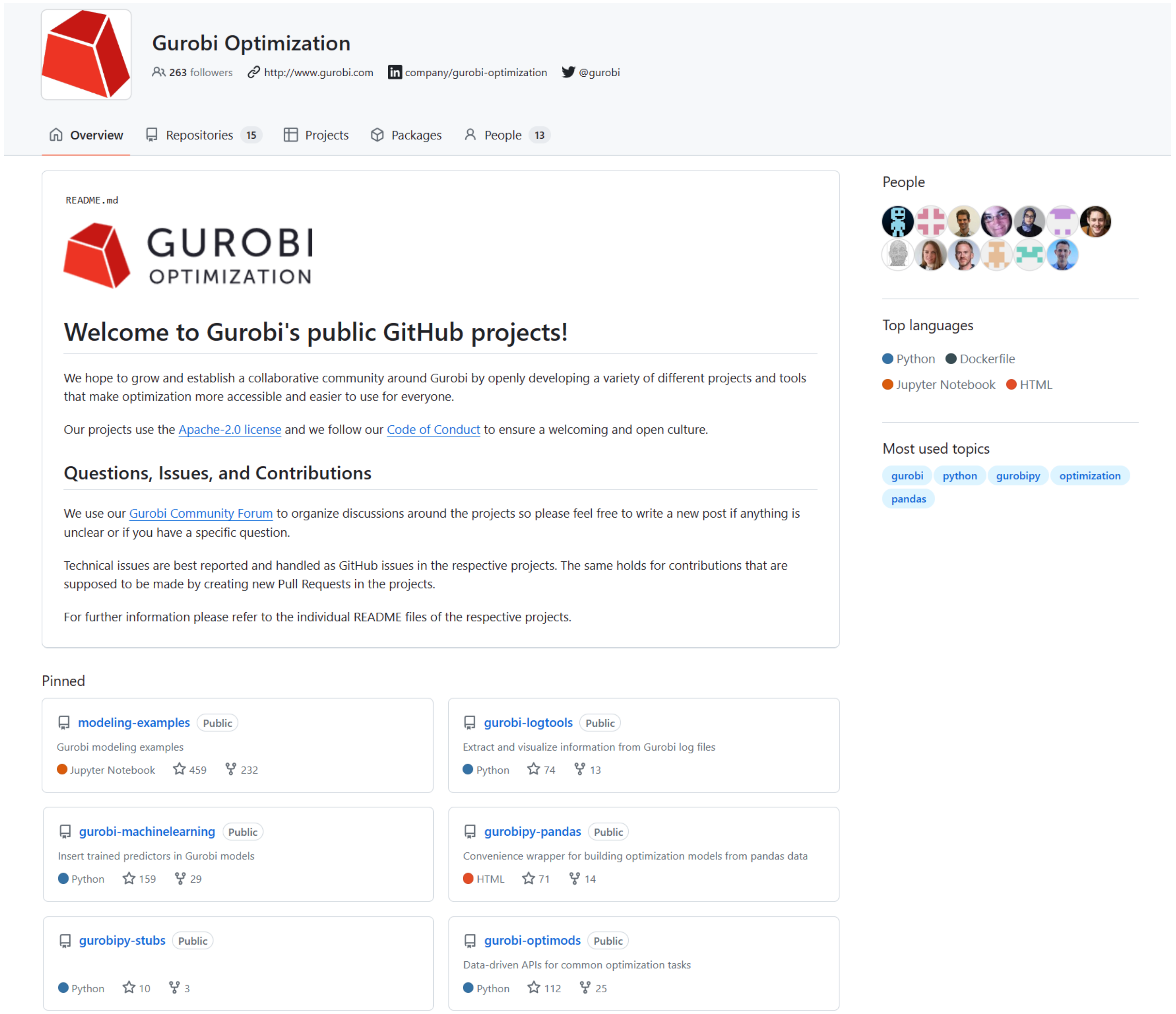
We developed our own!
NoRel

This heuristic searches for high-quality feasible solutions before solving the root relaxation. It can be quite useful on models where the root relaxation is particularly expensive.

NoRel (extremely powerful in practice!)

- The approach can be applied to almost all problem types
- To use:
  - Upgrade to Gurobi 9.5 or newer.
  - Set the NoRelHeurTime parameter
- Models that are good candidates to benefit:
  - Very large models
  - Models where finding a feasible solution is difficult
  - Weak models
  - Models with a slow linear relaxation
- Parallelizes extremely well for large machines
- The NoRel heuristic has 2 phases. In the first phase, the heuristic focuses on finding a feasible solution. The relaxation value tells how far the current infeasible solution is from a feasible one. The lower the value, the closer the heuristic is to shifting to phase 2. In phase 2, a feasible point is available and the heuristics tries to improve the objective value.

Hidden Gems: GitHub



Gurobi Optimization

263 followers · http://www.gurobi.com · company/gurobi-optimization · @gurobi

Overview · Repositories 15 · Projects · Packages · People 13

README.md

Welcome to Gurobi's public GitHub projects!

We hope to grow and establish a collaborative community around Gurobi by openly developing a variety of different projects and tools that make optimization more accessible and easier to use for everyone.

Our projects use the Apache-2.0 license and we follow our Code of Conduct to ensure a welcoming and open culture.

Questions, Issues, and Contributions

We use our Gurobi Community Forum to organize discussions around the projects so please feel free to write a new post if anything is unclear or if you have a specific question.

Technical issues are best reported and handled as GitHub issues in the respective projects. The same holds for contributions that are supposed to be made by creating new Pull Requests in the projects.

For further information please refer to the individual README files of the respective projects.

Pinned

Repository	Description	Language	Stars	Forks
modeling-examples (Public)	Gurobi modeling examples	Jupyter Notebook	459	232
gurobi-logtools (Public)	Extract and visualize information from Gurobi log files	Python	74	13
gurobi-machinelearning (Public)	Insert trained predictors in Gurobi models	Python	159	29
gurobipy-pandas (Public)	Convenience wrapper for building optimization models from pandas data	HTML	71	14
gurobipy-stubs (Public)		Python	10	3
gurobi-optimods (Public)	Data-driven APIs for common optimization tasks	Python	112	25

People

Top languages

● Python ● Dockerfile ● Jupyter Notebook ● HTML

Most used topics

gurobi · python · gurobipy · optimization · pandas

GRBlogtools

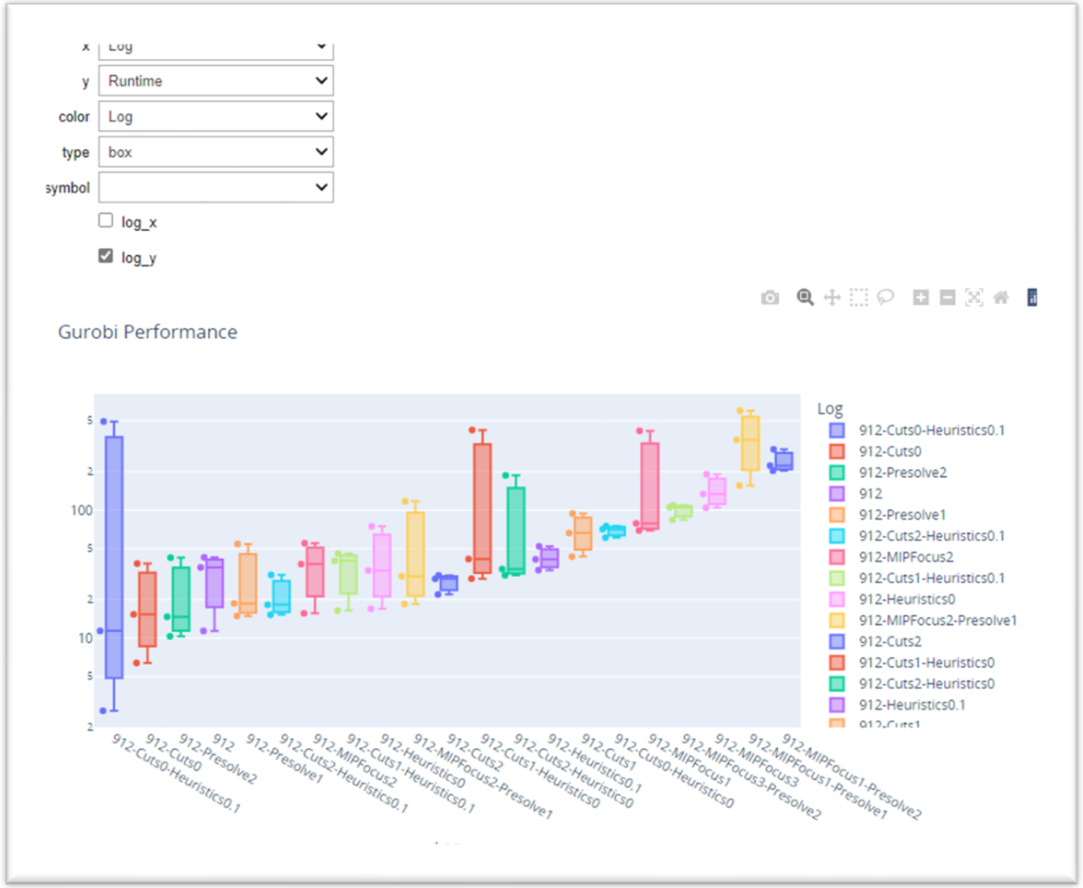
Open-source Python package to analyze multiple Gurobi log files

Easily compare results and logs from:

- Multiple model instances
- Different parameter sets
- Different computers

How it works:

- Read log data into pandas
- Plot values using Plotly
- Convert log data to Excel spreadsheets

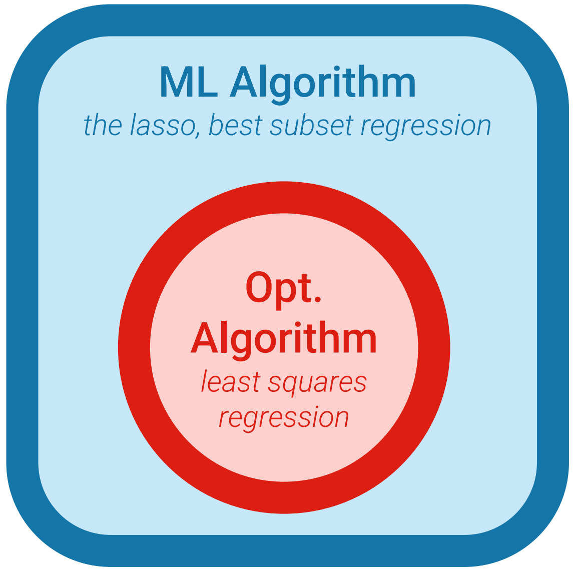


x: Log
y: Runtime
color: Log
type: box
symbol:
☐ log_x
☑ log_y

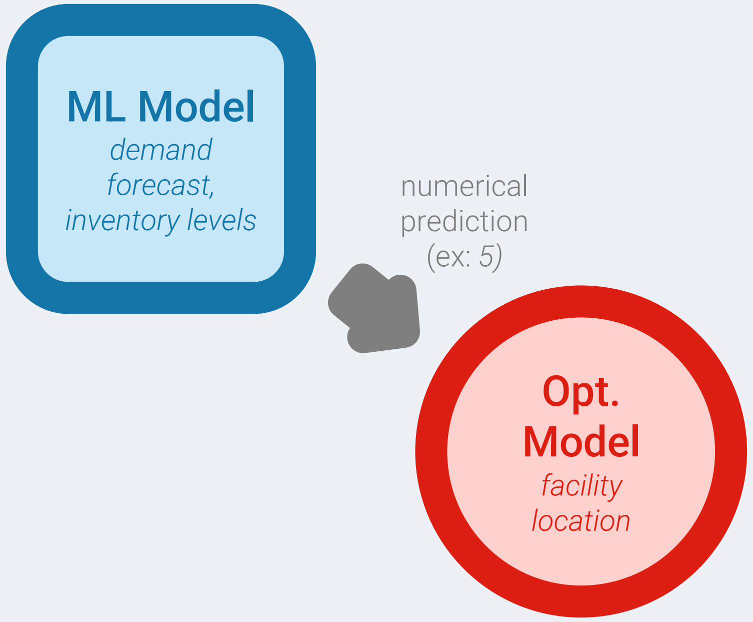
Details: https://github.com/Gurobi/grblogtools

Combining Machine Learning and Optimization

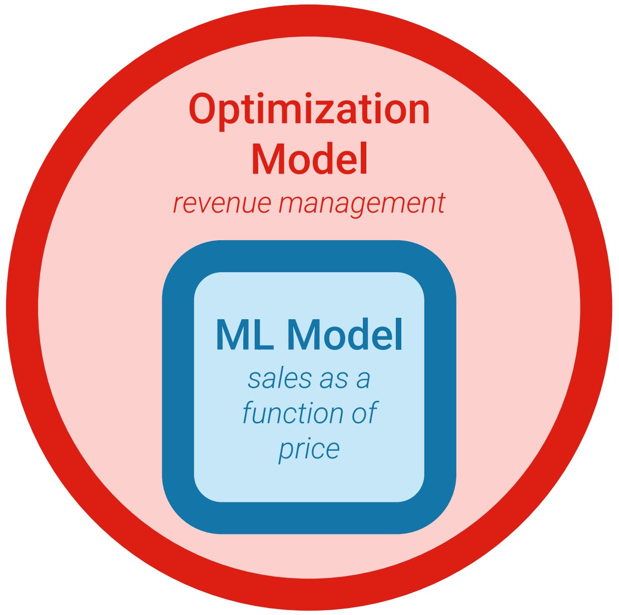
01
Training a ML model



02
Use the ML predictions to define the Opt. Model


ML Model
demand forecast, inventory levels
numerical prediction (ex: 5)
Opt. Model
facility location

03
Embed a ML model inside the Opt. model



Gurobi Machine Learning

GUROBI OPTIMIZATION

An open-source Python package to embed trained regression models in a gurobipy model to be solved with Gurobi

scikit learn

PyTorch

dmlc XGBoost

K Keras

Gurobi’s Ill Conditioning Explainer

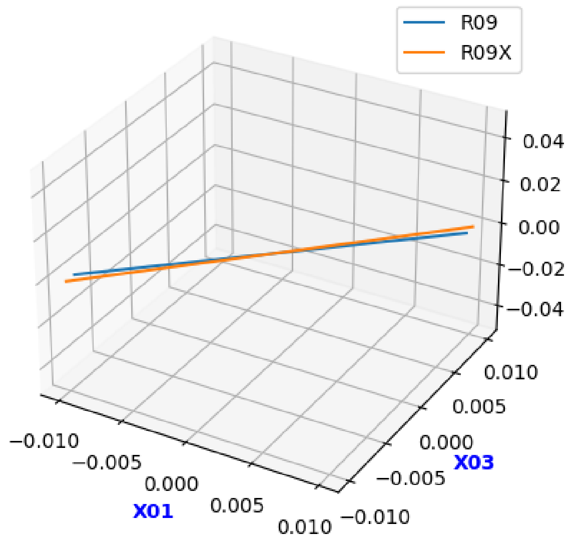
Open-source Python package to calculate explanations of ill-conditioned basis matrices

Motivation:

- Find sources of numerical instability (not infeasibility). I know Kappa is large, but then what?

How it works:

- Root lp inspection for MIPs
- *kappa_explain()* (row or column based explanation)
- *angle_explain()* (pairs of rows or columns)
- And more!



`kappa_explain()` will generate a new LP or MPS file, containing the ill-conditioning certificate:

```
Minimize
  0 X36 + 0 X04 + 0 X15 + 0 X16 + 0 X26 + 0 X38 + 0 X37
Subject To
 GRB_Combined_Row: 0.0303868836044176 X23 + 4.80518e-10 X01
   - 4.65661e-10 X03 = 0
 (mult=2696322.968477607)R09x: - 0.9999999000000001 X01 + X03 = 0
 (mult=-2696322.6896988587)R09: - X01 + X03 = 0
 (mult=0.2787787486643817)X46: - X03 + 0.109 X22 <= 0
 (mult=0.030386883604417606)R19: X23 - X22 + X24 + X25 = 0
 (mult=0.030386883604417606)X45: - X25 <= 0
 (mult=0.030386883604417606)X48: 0.301 X01 - X24 <= 0
Bounds
End
```

Details: https://github.com/Gurobi/gurobi-modelanalyzer

gurobi-pandas

Open-source Python package to connect pandas with gurobipy

Motivation:

- Make it easier to build optimization models from DataFrames, and return solutions as Panda objects.

How it works:

- Add variables and constraints

  using DataFrame.gppd accessors or

  gppd.add_vars(), gppd.add_constrs() functions

  .

- Use gppd series accessor to extract solutions

Details: https://github.com/Gurobi/gurobipy_pandas

$$\begin{array}{ll} \max & \sum_{i\in I}\sum_{j\in J} p_{ij}x_{ij} \\ \text{s.t.} & x_{ij} \in \{0,1\} \quad \forall (i,j) \end{array}$$

$$\sum_{i\in I} w_i x_{ij} \leq c_j \quad \forall j \in J$$

```
import pandas as pd
import gurobipy as gp
from gurobipy import GRB
import gurobipy_pandas as gppd

projects = pd.read_csv(projects_csv, index_col="project")
teams = pd.read_csv(teams_csv, index_col="team")
project_vals = pd.read_csv(project_values_csv,index_col=["project", "team"])

model = gp.Model()
model.ModelSense = GRB.MAXIMIZE
x = gppd.add_vars(model, project_values, vtype=GRB.BINARY, obj="profit", name="x")

capacity_constraints = gppd.add_constrs(
    model,
    (
        (projects["resource"] * x)
        .groupby("team").sum()
    ),
    GRB.LESS_EQUAL,
    teams["capacity"],
    name='capacity',
)
```

Summary

~~Hidden~~ Revealed Gems

Modeling

- Multiple Objectives
- Multiple Scenarios
- Solution Pool
- General Constraints
- Infeasibility Analysis

Performance

- Variable Start & Hint Values
- NoRel

GitHub

- GRBlogtools
- Gurobi Machine Learning
- Gurobi's Ill Conditioning Explainer
- gurobi-pandas

There are still Gems to discover!

Partition Heuristic, Callbacks, VarBranch Priorities, Distributed Optimization, Auto-tuner, Optimods, … and More