Hidden Gems

Useful Gurobi Features you may not know

Dr. Jue Xue

Technical Account Manager

April 9, 2024

Outline

Hidden Gems

Useful Gurobi Features you may not know

Modeling
- Multiple Objectives
- Multiple Scenarios
- Multiple Solutions
- General Constraints
- Infeasibility Analysis

Performance
- Variable Start & Hint Values
- Partition Heuristic

GitHub
- GRBlogtools
- Gurobi Machine Learning
- Gurobi's Ill Conditioning Explainer

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

OBJ 1 + OBJ 2 + OBJ 3

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

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

OBJ 1 → OBJ 2 → OBJ 3

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

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

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

- **Weighted + Hierarchical**

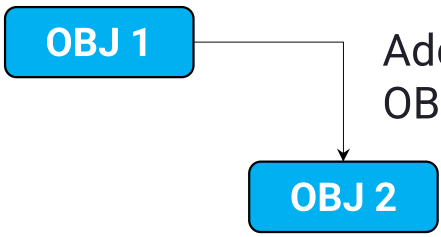
Multiple Objectives API

- Python API

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

- `LinExpr`: Objective expressions
- `index`: Objective index (Used to set different parameters/query the solution per objective)
- `priority`: Objective priority (`ObjNPriority` attribute)
- `weight`: Objective weight (`ObjNWeight` attribute)
- *`abstol`*: Absolute tolerance used in calculating the allowable degradation (`ObjNAbsTol` attribute)
- `reltol`: Relative tolerance used in calculating the allowable degradation (`ObjNrelTol` attribute)

How is the degradation value calculated?



Additional Constraint:
OBJ 1 <= ϵ_1

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

ϵ_1 = base_value + relaxed

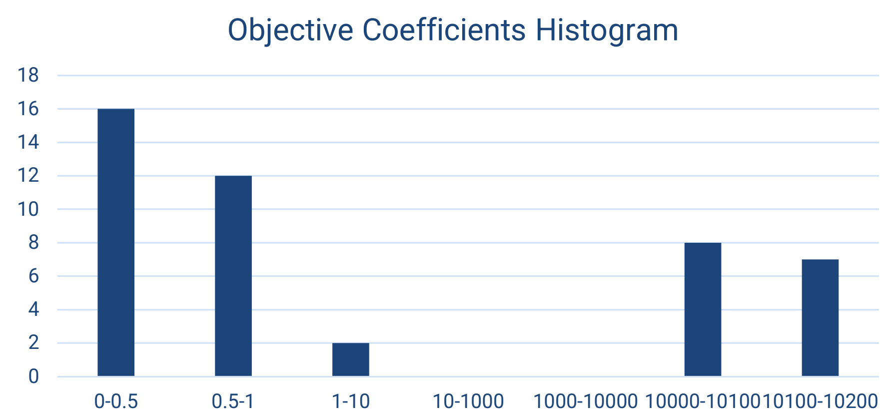
- `base_value = max(objbnd +|objval|*MIPGap, objbnd + MIPGapAbs, objval)`
- `relaxed    = max(ObjNRelTol*|base_value|, ObjNAbsTol)`

```
objbnd     : best bound of objective OBJ1
objval     : best solution value for objective OBJ1
MIPGap     : relative MIP gap
MIPGapAbs  : absolute MIP gap
ObjNRelTol: allowable relative degradation for OBJ1
ObjNAbsTol: allowable absolute degradation for OBJ1
```


Multiple Objectives Details

- A single objective sense for all objectives (`ModelSense` attribute)
- Objective expressions should be linear
- Choosing objective-specific parameters via [multi-objective environments]
- Faster performance from warm starts for hierarchical objectives
- 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 Scenarios

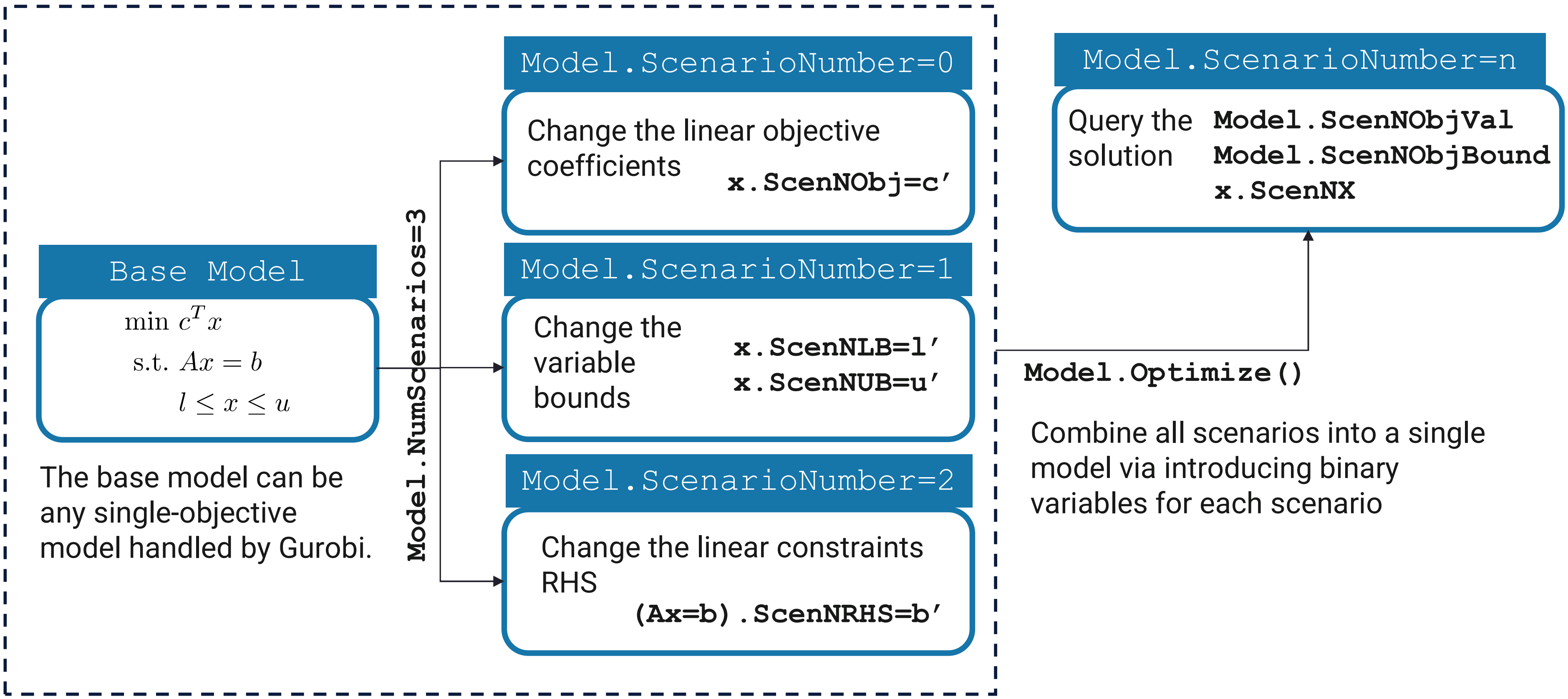
- In many real-world applications, the following may occur:
  - The input data is not accurate
  - The input data is not known in advance and can take multiple values in real time
  - The input data is seasonal or periodic
  - The input data has a range of possible values

- 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

Base Model

$$\begin{aligned} \min\ & c^T x \\ \text{s.t. } & Ax = b \\ & l \leq x \leq u \end{aligned}$$

Multiple Scenarios API

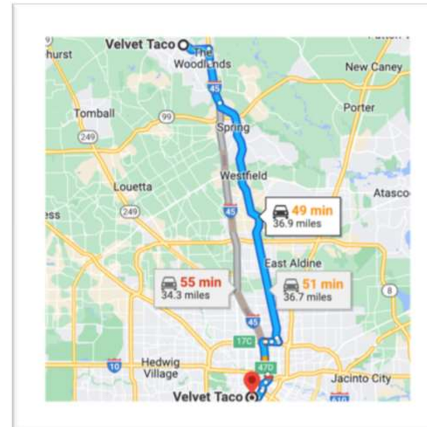


Multiple Scenarios (Tips & Tricks)

- 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 find several feasible solutions?
  - Define a **Solution Pool** and report multiple solutions automatically, and 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 the number and quality of solutions 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 additional 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)
```

Gurobi API for Function Constraints

- Smart translation for periodic functions
- Using actual functions during presolve
- Bound strengthening in presolve for more efficient handling

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()
```

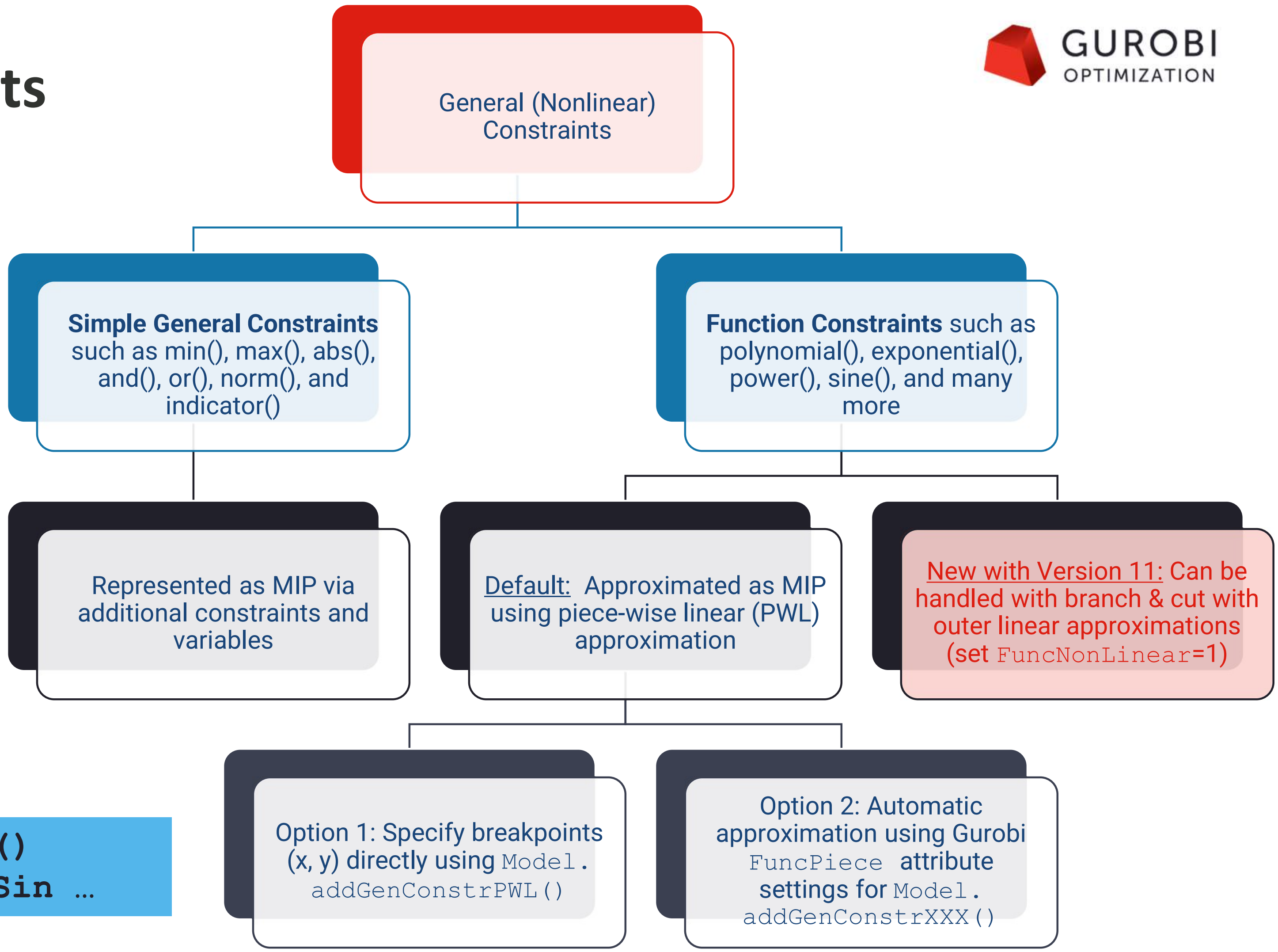
Supported Function Constraints

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

General Constraints

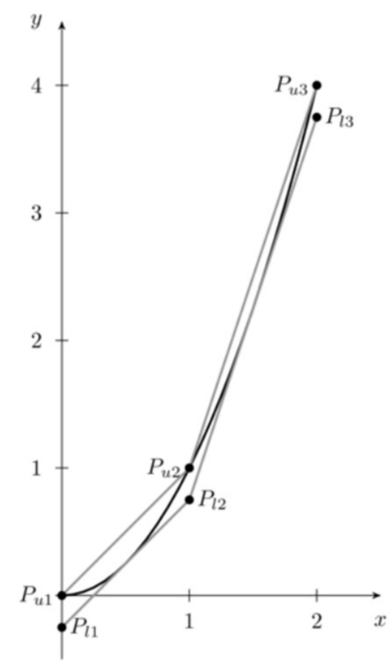
- Gurobi supports two types of general constraints
- There are different strategies for the algorithmic implementation…
- …but the API is consistent

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



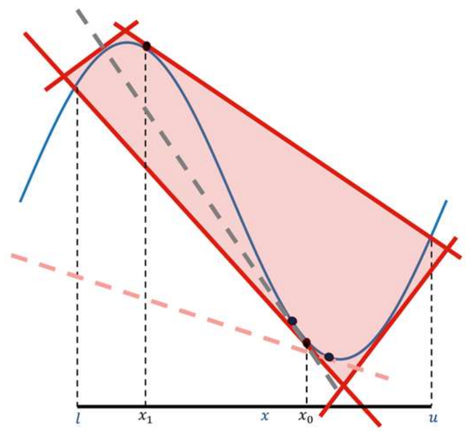
Options for Automatic PWL Translation

- 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
- Note
  - Constraint Attributes: Applied to a specific function constraint
  - Parameters: Applied to all function constraints

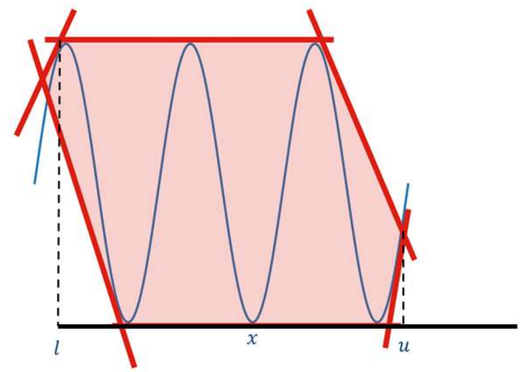


Function Constraints with Outer Approximations

- 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)`



Note:
Branching on x tightens the relaxation quickly!

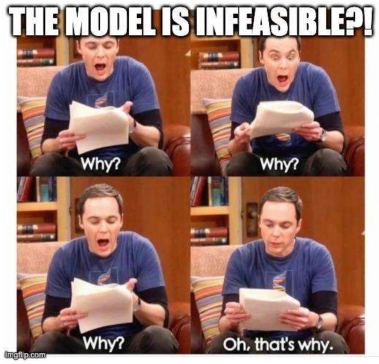


Tighter initial bounds will speed up performance

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)

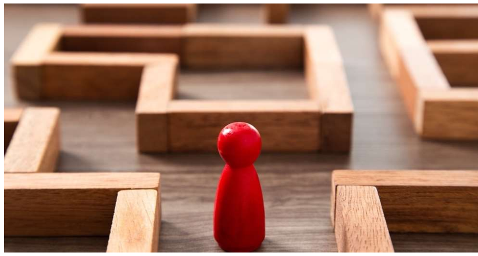
- Given an infeasible system of constraints
  - Find a subset of constraints/variable bounds that
    - It is infeasible
    - Removing a single constraint/bound makes it feasible
  - IIS is minimal and not minimum
- Meant to be read and analyzed by a human
  - The smaller, the better
- Computational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

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

```
\ 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)`
  - `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

    Jan Feb Mar Apr May Jun

    Previous solution becomes start or hint for first 5 months of next run

  - Run 2:
    Redo plan starting in 2nd month

    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 – 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**

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
```


Partition Heuristic

- Partition heuristic is typically useful if there is a natural grouping in the model
  - Improve the scheduling of jobs assigned to the same machine
  - Improve the allocation of warehouses to an open facility
  - Improve the production plan over time periods for a specific product

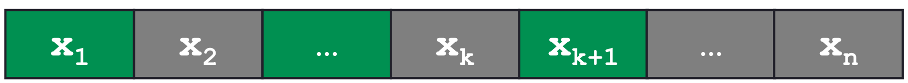
- If variables are partitioned into different groups, a separate sub-MIP is solved for each partition.

Partition Heuristic

- Improvement heuristics based on the idea of neighborhood search are used in Gurobi
  - Start from the current incumbent
  - Make a perturbation to the current incumbent
  - Solve a new MIP

Current incumbent

Select a subset of variables to be fixed at the current incumbent



Solve a sub-MIP to optimize unfixed variables

- How to decide which variables to fix?
  - **Relaxed Induced Neighborhood Search (RINS)**: Fix variables whose values agree in both the current incumbent and the current node relaxation
  - **Partitioning**: User provides guidance via variable grouping

Options for Partition Heuristic

- `Partition`, a variable attribute, to indicate which group the variable belongs to
  - -1: Fix the variable in all sub-MIPs (if set for all variables, no partition heuristic)
  - 0 : Unfix the variable in all sub-MIPs
  - k : Unfix the variable in the k^{th} sub-MIP and fix it in the rest

- `PartitionPlace`, a solver parameter controlling where the heuristic runs
  - The parameter value is a bit vector, with each bit turning on/off the heuristic
  - Example: `PartitionPlace = 10` runs the heuristic at the start of the root node and at all nodes

16	8	4	2	1
Before the root relaxation	Start of root cut loop	End of root cut loop	Nodes in the branch-and-cut search	End of branch-and-cut search

Hidden Gems: GitHub

gurobi-logtools

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

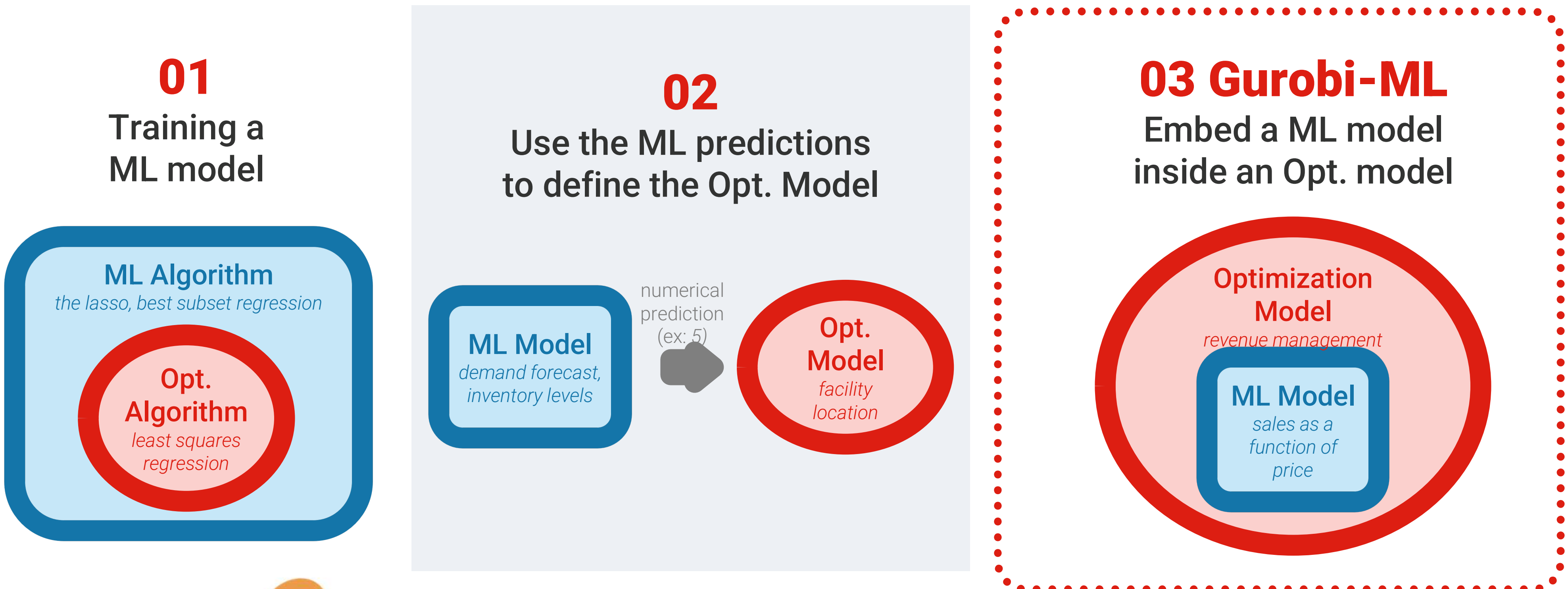
Details: https://github.com/Gurobi/gurobi-logtools

Machine Learning and Optimization

GUROBI OPTIMIZATION

Gurobi-ML: an open-source Python package

Embed trained regression models* in an optimization model, solved by Gurobi



scikit learn · dmlc XGBoost · Keras · PyTorch

Gurobi's Ill Conditioning Explainer

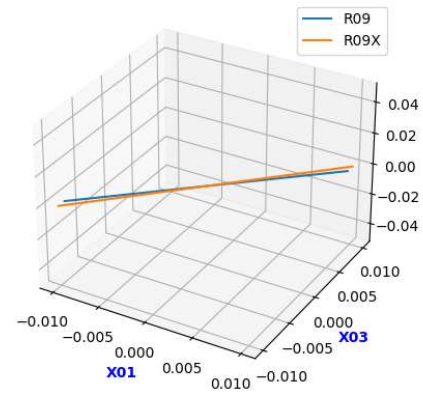
Open-source Python package to calculate explanations of ill-conditioned basis matrices

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

GUROBI OPTIMIZATION

$$\max \sum_{i \in I} \sum_{j \in J} p_{ij} x_{ij}$$

$$\text{s.t.} \quad x_{ij} \in \{0,1\} \quad \forall (i,j)$$

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

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

```
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',
)
```

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

Summary

~~Hidden~~ Revealed Gems

Modeling

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

Performance

- Variable Start & Hint Values
- Partition Heuristic

GitHub

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

There are still Gems to discover!

NoRel Heuristic, VarBranch Priorities, Callbacks, Distributed Optimization, Optimods, … and More

GUROBI
OPTIMIZATION

Thank You

For more information: gurobi.com