Hidden Gems

Useful Gurobi Features you may not know

Cara Touretzky & Rodrigo Fuentes

Technical Account Managers

Please Keep in Mind ...

A recording of the webinar and copy of slides will be sent to attendees within a week.

Use the Q&A feature to submit your questions – not the chat box!

Gurobi's Experts will answer questions via Q&A feature.

Questions not answered on Q&A will be addressed live by the presenters at the end.

Questions not ansewered during the event will be answered via email.

Hidden Gems

Useful Gurobi Features you may not know

Cara Touretzky & Rodrigo Fuentes

Technical Account Managers

Outline

Hidden Gems
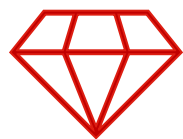
Useful Gurobi Features you may not know

Modeling

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

Performance

- Variable Start & Hint Values
- Partition Heuristic

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
&
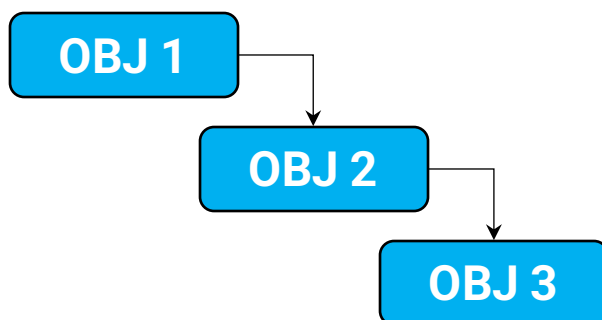Maximize Product
Durability**

**Maximize Profit
&
Minimize Risk**

How does Gurobi handle the trade-offs?

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

OBJ 1 **+** OBJ 2 **+** OBJ 3

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

- **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

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

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

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

- **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?
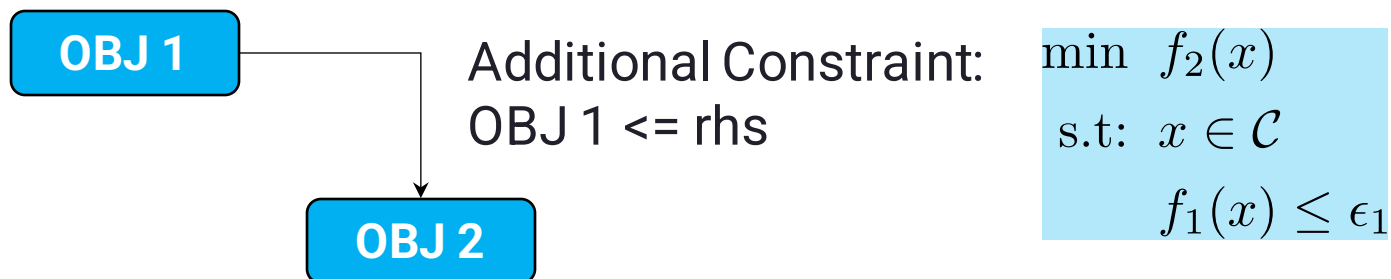

```
base_value = max(objbnd +|objval|*MIPGap, objbnd + MIPGapAbs, objval)
relaxed    = max(ObjNRelTol*|base_value|, ObjNAbsTol)
rhs (ε₁)   = base_value + relaxed


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

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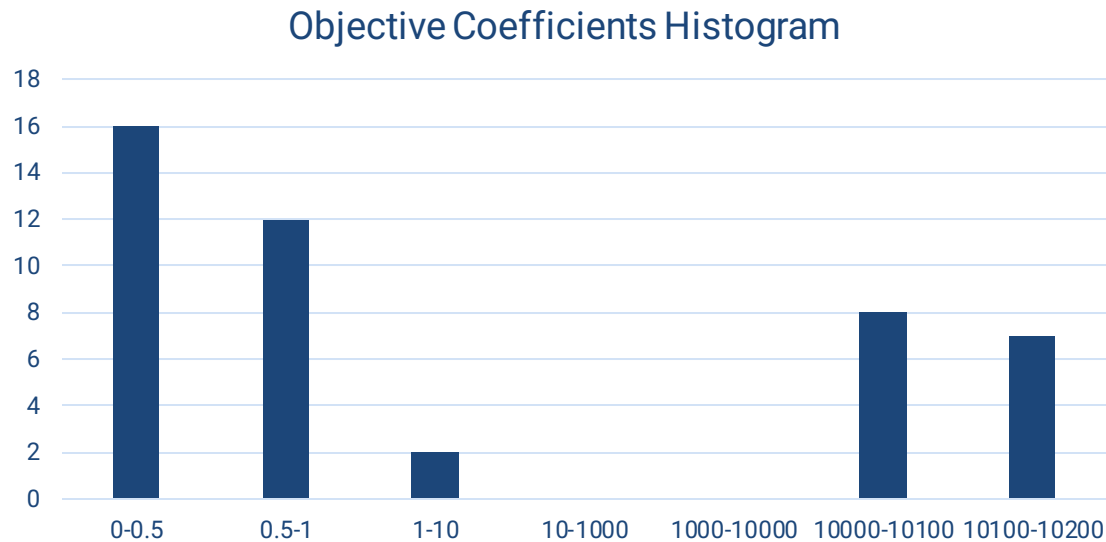
There are 45 coefficients in 2 distinct groups. Is this a multi-objective case in hiding?

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](#)
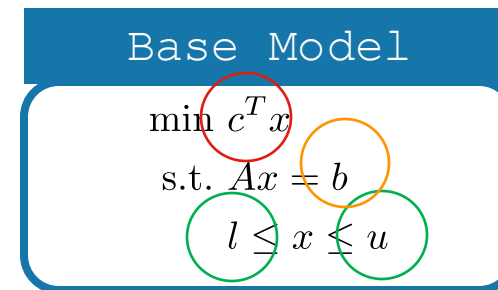
- Callbacks are available

> 💡 **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
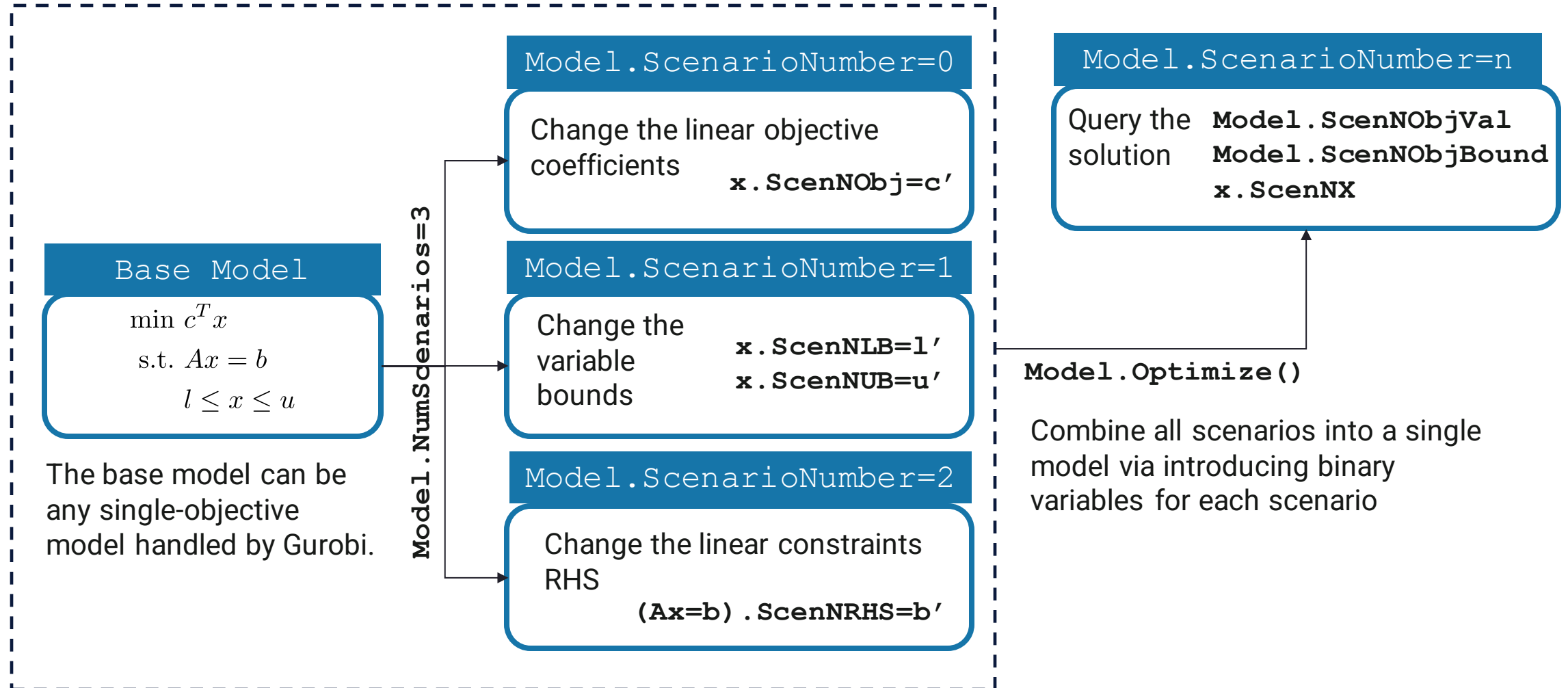
Multiple Scenarios

- 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

Base Model

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

Multiple Scenarios API

Base Model

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

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.ScenarioNumber=n

Query the solution

`Model.ScenNObjVal`
`Model.ScenNObjBound`
`x.ScenNX`

Model.Optimize()

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

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.
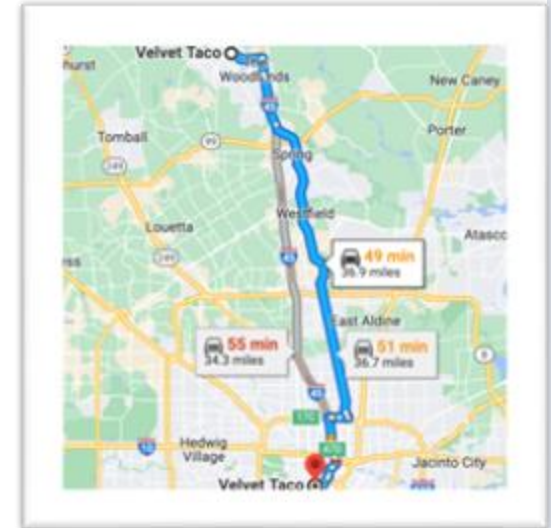
Solution Pool Setup

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.

```python
# 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)
```
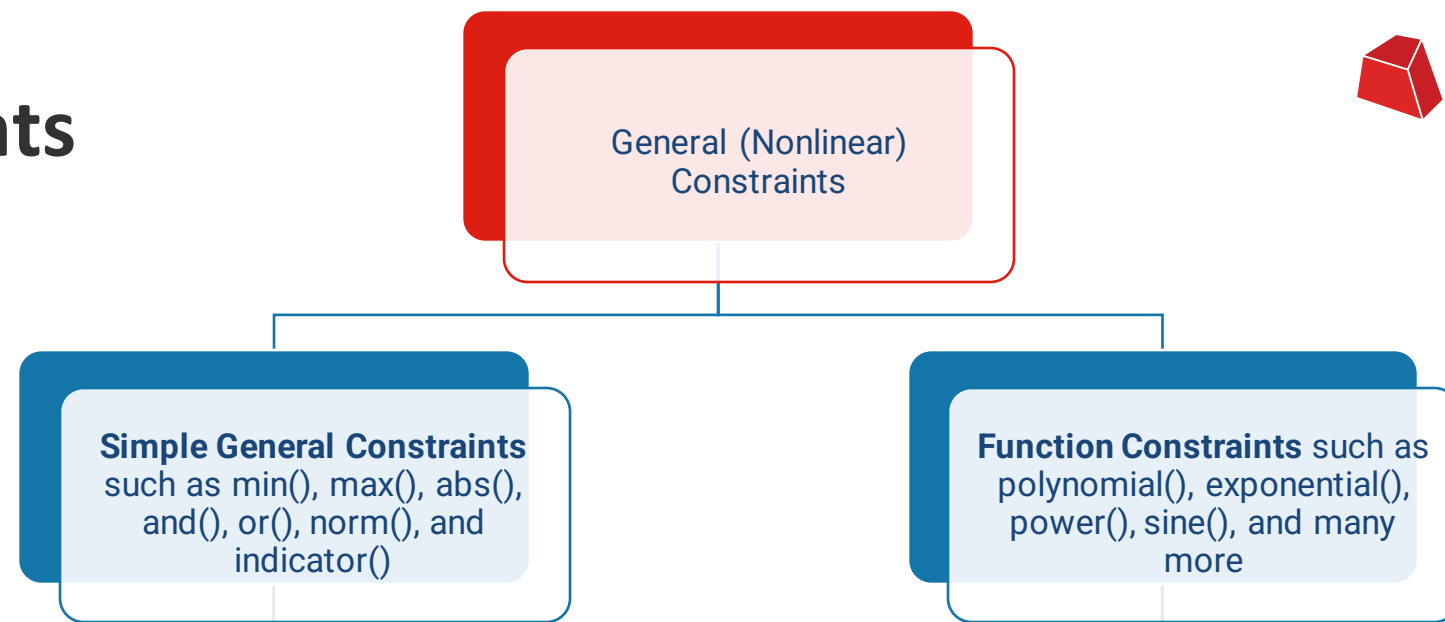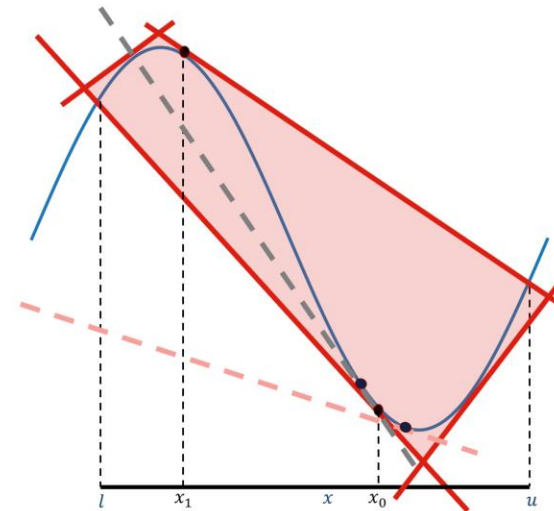
General Constraints

General (Nonlinear) Constraints

- Gurobi supports two types of general 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

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!
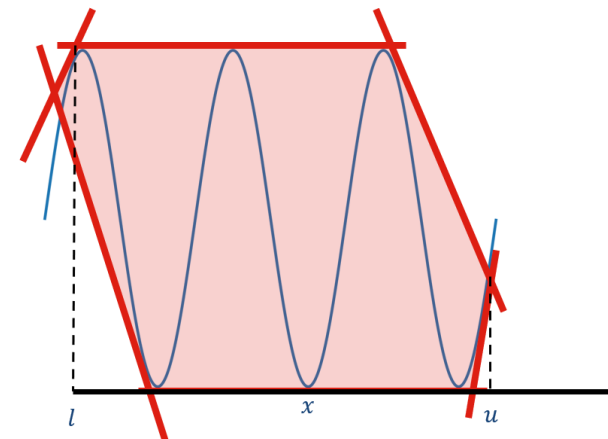
Tighter initial bounds will speed up performance

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
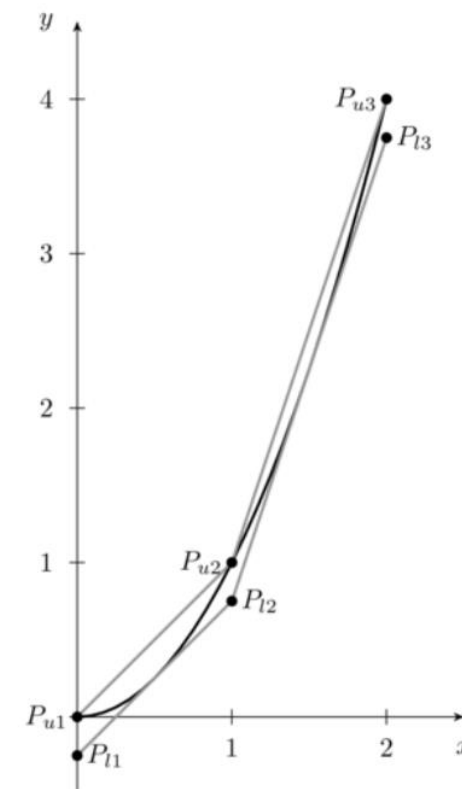
Gurobi API for Function Constraints

GUROBI OPTIMIZATION

Example

$$\min \quad -2x + \boxed{e^x}$$

$$\text{s.t:} \quad 0 \le x \le 1$$

```
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)

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

Infeasible model

$$\min \quad c^T x$$
$$\text{s.t.} \quad Ax \le b$$
$$x \ge 0$$

Feasibility relaxation

$$\min \quad \|(s, u)\|_p$$
$$\text{s.t.} \quad Ax - s \le b$$
$$x + u \ge 0$$
$$s, u \ge 0$$

GUROBI OPTIMIZATION

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 |

    - Run 2:
      Redo plan starting in 2nd month

      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**

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

x_1	x_2	...	x_k	x_{k+1}	...	x_n

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

x_1	x_2	...	x_k	x_{k+1}	...	x_n

Solve a sub-MIP to optimize unfixed variables

x_1	x_2	...	x_k	x_{k+1}	...	x_n

- 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

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.

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 kth sub-MIP and fix 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

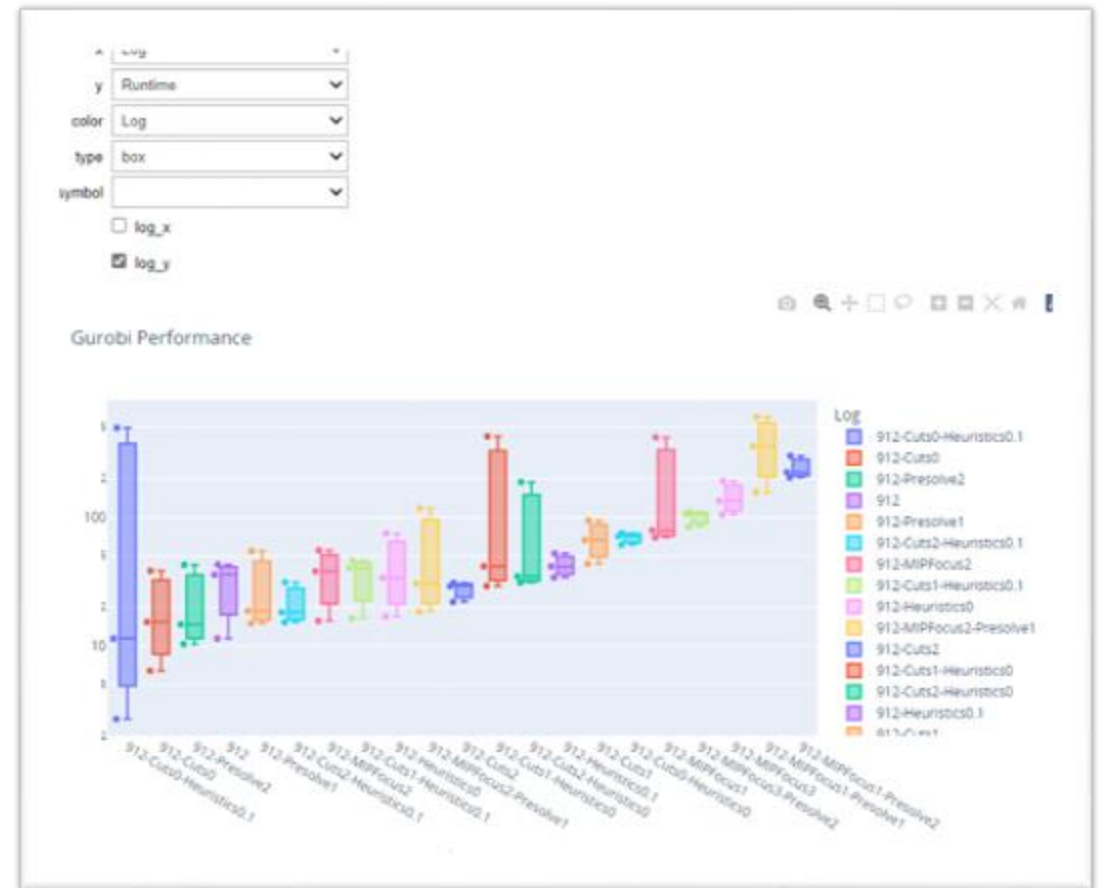| 16	8	4	2	1
Before the root relaxation	Start of root cut loop	End of root cut loop	Nodes in the branch-and-cut search	End of branch-and-cut search

Hidden Gems: GitHub

GRBlogtools

Open-source Python package to analyze multiple Gurobi log files

Easily compare results and logs from:

- Multiple model instances

- Different parameter sets
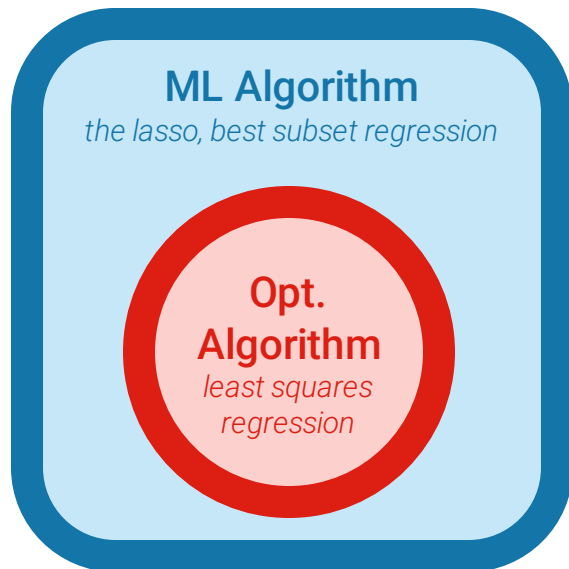
- Different computers

How it works:

- Read log data into pandas

- Plot values using Plotly
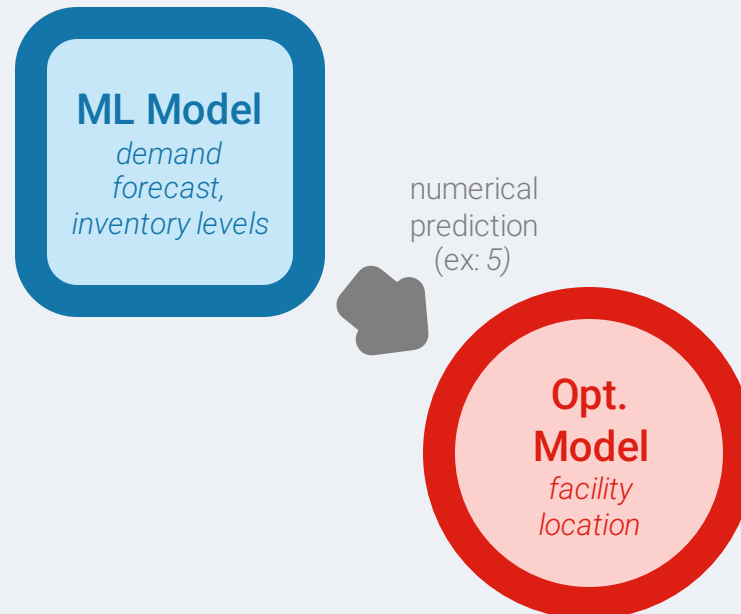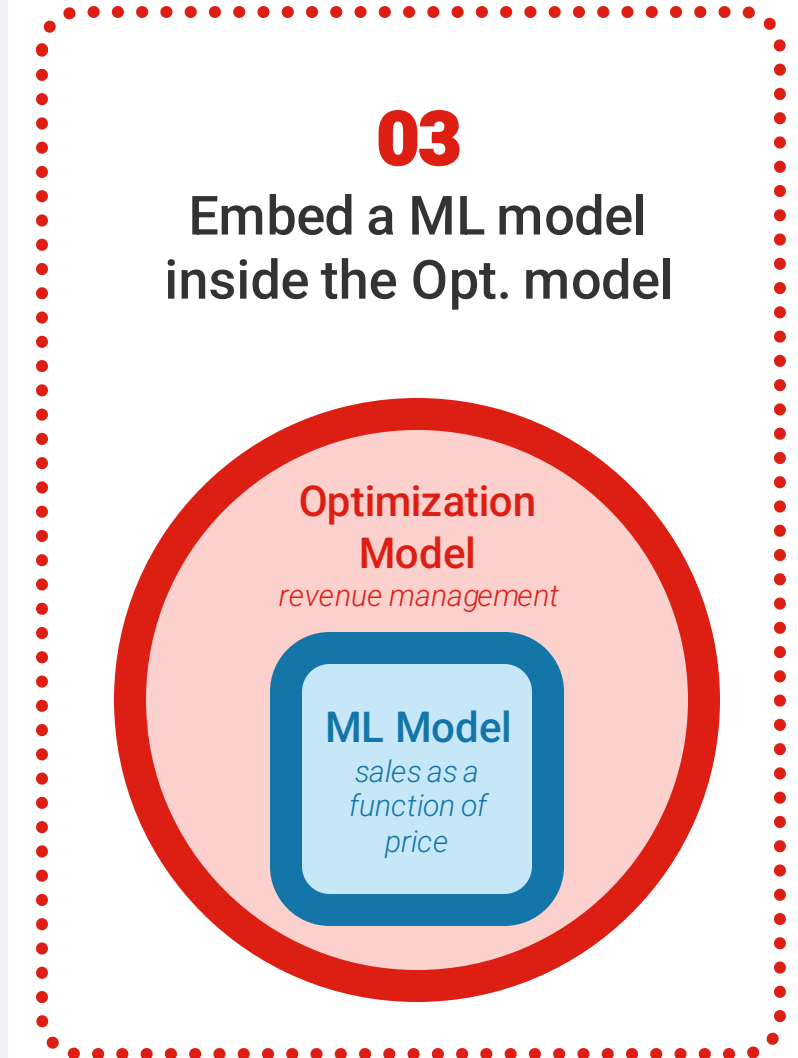
- Convert log data to Excel spreadsheets

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

Combining Machine Learning and Optimization

01
Training a ML model

ML Algorithm
the lasso, best subset regression

Opt. Algorithm
least squares regression

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

Optimization Model
revenue management

ML Model
sales as a function of price

Gurobi Machine Learning

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

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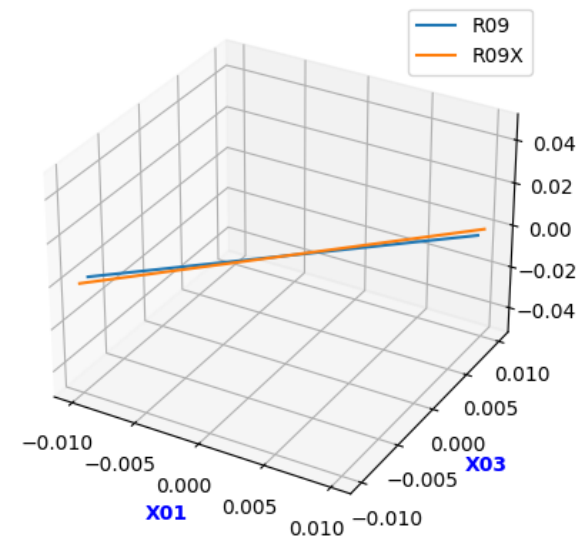
`kappa_explain()` will generate a new LP or MPS file, containing the ill-conditioning certificate:

```
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: - 0.999999900000001 X01 + X03 = 0
 (mult=-2696322.6896988587)R09: - X01 + X03 = 0
 (mult=0.2787787486643817)X46: - X03 + 0.109 X22 <= 0
 (mult=0.03038688360441706)R19: X23 - X22 + X24 + X25 = 0
 (mult=0.03038688360441706)X45: - X25 <= 0
 (mult=0.03038688360441706)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

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

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

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

```python
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
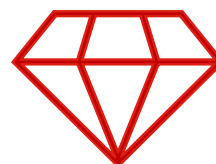- Variable Start & Hint Values
- Partition Heuristic

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

Q&A

Webinar recording and copies of slides
will be available to all attendees

Hidden Gems

Useful Gurobi Features you may not know

Cara Touretzky & Rodrigo Fuentes

Technical Account Managers

Contact Information:

Fuentes@gurobi.com

Touretzky@gurobi.com