Modeling 2
Agenda

• Understanding advanced modeling techniques takes some time and experience
  • No exercises today
  • Ask questions!

• Part 1: Overview of selected modeling techniques
  • Background
  • Range constraints
  • Special functions: absolute value, piecewise linear, min/max
  • Logical conditions on binary variables
  • Logical conditions on constraints
  • Semi-continuous variables
  • Selecting big-M values

• Part 2: We go through the whole model development process
  • From problem description to mathematical model to Python model
Background – It’s automated!

• Gurobi Optimizer 7.0 introduced General Constraints for popular logical expressions
  • Absolute value
  • Min/Max value
  • And/Or over binary variables
  • Indicator (if-then logic)

• The General Constraint syntax is a safe way to implement model logic

• Let’s see
  • How these logical expressions work
  • How to build models with complex logic
Background – Indicator variables and convexity

• Many advanced models are based on binary indicator variables
  • Indicate whether or not some condition holds

• Models with convex regions and convex functions are generally much easier to solve
Background – Special Ordered Sets

• Special Ordered Set of type 1 (SOS-1) – at most one variable in set may be non-zero

• Special Ordered Set of type 2 (SOS-2) – an ordered set where
  • At most two variables may be non-zero
  • Non-zero variables must be adjacent

• Variables need not be integer
Range constraints

• Many models contain constraints like:  \[ L \leq \sum_{i} a_i x_i \leq U \]

• These can be rewritten as:  \[ r + \sum_{i} a_i x_i = U \]

\[ 0 \leq r \leq U - L \]

• The range constraint interface automates this for you (semantic sugar-coating)

• If you need to modify the range
  • Retrieve the additional range variable, named RgYourConstraintName
  • Modify the bounds on that variable

• For full control, it’s easier to model this yourself
Non-linear functions

- General non-linear functions (of decision variables) are not directly supported by Gurobi

- Examples:
  - \( \log(x) \)
  - \( \sqrt{x} \)
  - \( \cos(x), \sin(x), \tan(x), \ldots \)
  - \( 2^x \)
  - \( \ldots \)

- Non-convex quadratic functions are not supported either
  - Directly supported in some special cases (ex: binary decision variables)

- However, we can linearize or approximate some of these through modeling techniques
Absolute value – Convex case

- Simply substitute if absolute value function creates a convex model

\[ \min |x| \quad \Rightarrow \quad \begin{align*}
\min z \\
z &= x_p + x_n \\
x &= x_p - x_n
\end{align*} \]
Absolute value – Non-convex case

- Use indicator variable and arbitrary big-M value to prevent both $x_p$ and $x_n$ positive

\[
\max |x| \quad \rightarrow \\
\max z\\
\begin{align*}
z &= x_p + x_n \\
x &= x_p - x_n \\
x_p &\leq M y \\
x_n &\leq M (1 - y) \\
y &\in \{0, 1\}
\end{align*}
\]

- Q: Any ideas on how to model this if no reasonable, finite big-M exists (ex: $|x|$ can be infinite)?
**Absolute value – SOS-1 constraint**

- Use SOS-1 constraint to prevent both $x_p$ and $x_n$ positive

\[
\max |x| \quad \rightarrow \quad \max z
\]

- $z = x_p + x_n$
- $x = x_p - x_n$
- $x_p, x_n \in \text{SOS-1}$

- No big-M value needed
- Works for both convex and non-convex version

- Q: Which will perform better?
SOS constraints vs big-M representation

• SOS constraints are handled with branching rules (not included in LP relaxation)

• SOS advantages:
  • Always valid (no reasonable M in some cases)
  • Numerically stable

• Big-M advantages:
  • LP relaxation is tighter
  • Typically results in better performance for Gurobi’s algorithms as long as M is relatively small

• In fact, Gurobi will try to reformulate SOS constraints into a big-M representation during presolve
  • User has control over this behavior with `PreSOS1BigM` and `PreSOS2BigM` parameters
  • Establishes limit on the largest big-M necessary
Piecewise linear functions

• Generalization of absolute value functions

• Convex case is easy
  • Function represented by LP

• Non-convex case is more challenging
  • Function represented as MIP or SOS-2 constraints

• Gurobi has an API for piecewise linear objectives
  • Built-in algorithmic support for the convex case
  • Conversion to MIP is transparent to the user

• Q: What are some potential applications?
Piecewise linear functions – Applications

• Piecewise linear functions appear in models all the time

• Examples:
  • Fixed costs in manufacturing due to setup
  • Economies of scale when discounts are applied after buying a certain number of items
  • ...

• Also useful when approximating non-linear functions
  • More pieces provide for a better approximation

• Examples:
  • Unit commitment models in energy sector
  • ...

Copyright 2016, Gurobi Optimization, Inc.
Piecewise linear functions – API

- Only need to specify function breakpoints
  - No auxiliary variables or constraints necessary

- Python example:
  ```python
  model.setPWLObj(x, [1, 3, 5], [1, 2, 4])
  ```

- $x$ must be non-decreasing
  - Repeat $x$ value for a jump (or discontinuity)
Piecewise linear functions – SOS-2 constraint

- Let \((x_i, y_i)\) represent \(i\)th point in piecewise linear function

- To represent \(y = f(x)\), use:
  \[
  x = \sum_{i} \lambda_i x_i \\
y = \sum_{i} \lambda_i y_i \\
\sum_{i} \lambda_i = 1 \\
\lambda_i \geq 0, \text{ SOS-2}
  \]

- SOS-2 constraint is redundant if \(f\) is convex

- Binary representation also exists
Min/max functions – Convex case

- Easy to minimize the largest value (minimax) or maximize the smallest value (maximin)

\[
\min \left\{ \max_{i} x_{i} \right\} \quad \Rightarrow \quad \min z
\]

\[
z \geq x_{i} \quad \forall i
\]

- Ex: minimize completion time of last job in machine scheduling application
Min/max functions – Non-convex case

- Harder to minimize the smallest value (minimin) or maximize the largest value (maximax)
  - Use multiple indicator variables and a big-M value

\[
\begin{align*}
\min \left\{ \min_i x_i \right\} & \quad \longrightarrow \quad \min z \\
\sum_i y_i & = 1 \\
y_i & \in \{0, 1\}
\end{align*}
\]

\[
z \geq x_i - M(1 - y_i)
\]
General Constraints for Logical Expressions

<table>
<thead>
<tr>
<th>Function</th>
<th>Python syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y = \min{x_1, x_2, x_3}$</td>
<td>addGenConstrMin($y$, [$x_1$, $x_2$, $x_3$])</td>
</tr>
<tr>
<td>$y = \max{x_1, x_2, x_3}$</td>
<td>addGenConstrMax($y$, [$x_1$, $x_2$, $x_3$])</td>
</tr>
<tr>
<td>$y = \abs{} x$</td>
<td>addGenConstrAbs($y$, $x$)</td>
</tr>
</tbody>
</table>

General constraints are also available for C, C++, Java, .NET; we use Python syntax simply for illustration.
Logical conditions on binary variables

• And
  \[ x_1 = 1 \text{ and } x_2 = 1 \]
  \[ x_1 + x_2 = 2 \]

• Or
  \[ x_1 = 1 \text{ or } x_2 = 1 \]
  \[ x_1 + x_2 \geq 1 \]

• Exclusive or (not both)
  \[ x_1 = 1 \text{ xor } x_2 = 1 \]
  \[ x_1 + x_2 = 1 \]

• At least / at most / counting
  \[ x_i = 1 \text{ for at least } 3 \text{ } i \text{'s} \]
  \[ \sum_{i} x_i \geq 3 \]

• If-then
  if \[ x_1 = 1 \text{, then } x_2 = 1 \]
  \[ x_1 \leq x_2 \]
Logical conditions – Variable result

• And
  \[ y = (x_1 = 1 \text{ and } x_2 = 1) \]
  \[ y \leq x_1 \]
  \[ y \leq x_2 \]
  \[ y \geq x_1 + x_2 - 1 \]

• Or
  \[ y = (x_1 = 1 \text{ or } x_2 = 1) \]
  \[ y \geq x_1 \]
  \[ y \geq x_2 \]
  \[ y \leq x_1 + x_2 \]

• Exclusive or (not both)
  \[ y = (x_1 = 1 \text{ xor } x_2 = 1) \]
  \[ y \geq x_1 - x_2 \]
  \[ y \geq x_2 - x_1 \]
  \[ y \leq x_1 + x_2 \]
  \[ y \leq 2 - x_1 - x_2 \]
### General Constraints for Logical Conditions

<table>
<thead>
<tr>
<th>Condition</th>
<th>Python syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y = (x_1 = 1 \text{ and } x_2 = 1)$</td>
<td><code>addGenConstrAnd(y, [x1, x2])</code></td>
</tr>
<tr>
<td>$y = (x_1 = 1 \text{ or } x_2 = 1)$</td>
<td><code>addGenConstrOr(y, [x1, x2])</code></td>
</tr>
</tbody>
</table>

General constraints are also available for C, C++, Java, .NET; we use Python syntax simply for illustration.
Logical conditions on constraints – Overview

• Add indicator variables for each constraint

• Enforce logical conditions via constraints on indicator variables
Logical conditions on constraints – And

• Trivial – constraints are always combined with "and" operator!

• All other logical conditions require indicator variables
Logical conditions on inequalities – Or

• Use indicator for the satisfied constraint, plus big-M value

\[
\begin{align*}
\sum_i a_i^1 x_i &\leq b^1 \\
or \quad \sum_i a_i^2 x_i &\leq b^2 \\
or \quad \sum_i a_i^3 x_i &\leq b^3
\end{align*}
\]

\[
\begin{align*}
\sum_i a_i^1 x_i &\leq b^1 + M (1 - y^1) \\
\sum_i a_i^2 x_i &\leq b^2 + M (1 - y^2) \\
\sum_i a_i^3 x_i &\leq b^3 + M (1 - y^3)
\end{align*}
\]

\[
y^1 + y^2 + y^3 \geq 1
\]

\[
y^1, y^2, y^3 \in \{0, 1\}
\]
Logical conditions on equalities – Or

• Add a free slack variable to each equality constraint
• Use indicator variable to designate whether slack is zero

\[ \sum_i a_i^k x_i = b^k \]

\[ \sum_i a_i^k x_i + w^k = b^k \]

\[ w^k \leq M (1 - y^k) \]

\[ w^k \geq -M (1 - y^k) \]

\[ y^k \in \{0, 1\} \]
Logical conditions on constraints – At least

- Generalizes the "or" constraint
- Use indicator for the satisfied constraints
- Count the binding constraints via a constraint on indicator variables

- Ex: at least 4 constraints must be satisfied with \( y_1 + y_2 + \ldots + y_m \geq 4 \)
Logical conditions on constraints – If-then

• Indicator General Constraint represents if-then logic
  • If \( z = 1 \) then \( x_1 + 2x_2 - x_3 \geq 2 \)
  • Syntax: `addGenConstrIndicator(z, 1, x1+2*x2-x3 >= 2)`

• The condition \( (z = 1) \) must be a binary variable (z) and a value (0 or 1)
  • Q: How do you transform this to other types of logic?
Semi-continuous variables

• Many models have special kind of "or" constraint
  \[ x = 0 \text{ or } 40 \leq x \leq 100 \]

• This is a semi-continuous variable

• Semi-continuous variables are common in manufacturing, inventory, power generation, etc.

• A semi-integer variable has a similar form, plus the restriction that the variable must be integer
Two techniques for semi-continuous variables

1. Add the indicator yourself

\[ 40y \leq x \leq 100y, \ y \in \{0,1\} \]

- Good performance but requires explicit upper bound on the semi-continuous variable

2. Let Gurobi handle variables you designate as semi-continuous

- Only practical option when upper bound is large or non-existent
Example – Combined logical constraints

- Limit on number of non-zero semi-continuous variables

- Easy if you use indicator variables

\[ 40y_i \leq x_i \leq 100y_i \]

\[ \sum y_i \leq 30 \]

- By modeling the logic yourself, fewer variables are needed
Selecting big-M values

- Want big-M as tight (small) as possible
  - Ex: for \( x_1 + x_2 \leq 10 + My \), if \( x_1, x_2 \leq 100 \) then \( M = 190 \)

- Presolve will do its best to tighten big-M values

- Tight, constraint-specific big-M values are better than one giant big-M that is large enough for all constraints
  - Too large leads to poor performance and numerical problems
  - Pick big-M values specifically for each constraint
Numerical issues
Numerical issues can be problematic

- Models are solved via a series of continuous (LP/QP) relaxations

- Computer is limited by numerical precision, typically doubles
  - In solving an LP or MIP, billions of numerical calculations can lead to an accumulation of numerical errors

- Can lead to slow performance or wrong answers
  - Optimal objective from Gurobi Optimizer: $-1.47e+08$
  - Optimal objective from other solver: $-2.72e+07$

- Typical causes of numerical errors
  - Rounding of numerical coefficients
    - Ex: Don’t write 1/3 as 0.333
  - Scaling – too large of a range for numerical coefficients
    - Ex: big-M values
Example – Trickle flow with big-M

• $y \leq 1000000 \cdot x$
  $x$ binary
  $y \geq 0$

• With default value of IntFeasTol (1e-5):
  • $x = 0.0000099999, y = 9.9999$ is integer feasible!
  • $y$ can be positive without forcing $x$ to 1
  • $y$ is positive without incurring the expensive fixed charge on $x$
Consequence of numerical issues

Linear constraint matrix : 25050 Constrs, 15820 Vars, 94874 NZs
Variable types : 14836 Continuous, 984 Integer
Matrix coefficient range : [ 0.00099, 6e+06 ]
Objective coefficient range : [ 0.2, 65 ]
Variable bound range : [ 0, 5e+07 ]
RHS coefficient range : [ 1, 5e+07 ]

• Big-M values create too large of a range of coefficients
• By reformulating the model, user got fast, reliable results
Numeric issues – Objective function

• Avoid large spread for objective coefficients
  • Often arises from penalties

• Example: minimize $100000 \, x + 5000 \, y + 0.001 \, z$
  • Coefficient on $x$ is large relative to others

• If $x$ takes small values, rescale $x$
  • Change scale from units to thousandths of units
  • Generally limited to continuous variables

• If $x$ takes large values, use hierarchical objectives
  • Optimize terms sequentially
  • Value of previous term introduced as a constraint
From the business problem to the mathematical problem to the Python implementation
Factory Planning

• Example from our website: http://www.gurobi.com/resources/examples/factory-planning-l

• Download Jupyter Notebook and Python source http://files.gurobi.com/training/factory.zip

• In production planning problems, choices must be made about how many of what products to produce using what resources (variables) in order to maximize profits or minimize costs (objective function), while meeting a range of constraints. These problems are common across a broad range of manufacturing situations.

• We will develop the mathematical model, the Python Implementation and a nice tabular output of the result all within a single Jupyter Notebook.
Factory Planning: Interactive Model Development

Chapter 3: Factory Planning

Problem Description
A factory makes several products. Each type of product is manufactured on one of several machines, including:

- Grinder
- Moulder
- Moulder 2
- Milling machine

Each product has a different production rate on each machine. The factory wants to optimize its production by maximizing profits. The objective is to minimize the cost of production while meeting the demand for each product.

In the example, we will assume that the factory has a limited number of machines and that the demand for each product is fixed. The goal is to determine the optimal production plan for each machine.

Model Formulation

Sets
- Set of products: \( P \)
- Set of machines: \( M \)
- Time horizon: \( T \)

Parameters
- Production rate of product \( p \) on machine \( m \): \( r_{p,m} \)
- Demand for product \( p \) in time period \( t \): \( d_t \)
- Cost of operating machine \( m \): \( c_m \)

Decision Variables
- Production of product \( p \) on machine \( m \) in time period \( t \): \( x_{p,m,t} \)

The objective is to minimize the total cost of production:

\[
\min \sum_{p \in P} \sum_{m \in M} \sum_{t \in T} c_m x_{p,m,t}
\]

Subject to:
- Production capacity:
  \[
  \sum_{p \in P} \sum_{m \in M} x_{p,m,t} \leq r_{m} \quad \forall m \in M, t \in T
  \]
- Demand:
  \[
  \sum_{m \in M} \sum_{t \in T} x_{p,m,t} = d_t \quad \forall p \in P, t \in T
  \]

Python Implementation

```python
# Import necessary modules
import gurobipy

# Define sets P, M, T

# Define data
machines = ["Grinder", "Moulder1", "Moulder2", "Milling", "Planer"]
months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun"]

# Define profit contribution per product
profit_contribution = {"Product1": 10, "Product2": 6, "Product3": 8, "Product4": 4, "Product5": 11, "Product6": 9, "Product7": 3}

# Define constraints
for m in machines:
    for t in months:
        model.addConstr(gurobipy.quicksum(x[p, m, t] for p in products) <= r[m], name=f"Cap_{m}_{t}")

# Define objective
model.setObjective(gurobipy.quicksum(profit_contribution[p] * x[p, m, t] for p in products for m in machines for t in months), gurobipy.GRB.MAXIMIZE)
```

Copyright 2016, Gurobi Optimization, Inc.
Additional resources

- Visit [http://www.gurobi.com/documentation/](http://www.gurobi.com/documentation/) for more information on Gurobi interfaces
  - Quick Start Guide
  - Reference Manual

- Explore our examples at [http://www.gurobi.com/resources/examples/example-models-overview](http://www.gurobi.com/resources/examples/example-models-overview)
  - Functional Examples
  - Modeling Examples
  - Interactive Examples

- Read *Model Building in Mathematical Programming* by H. Paul Williams
  - Great introduction to modeling business problems with math programming

- For more guidance on numeric issues, refer to [http://files.gurobi.com/Numerics.pdf](http://files.gurobi.com/Numerics.pdf)
Thank you – Questions?