Modeling: Best Practices & Techniques

Sonja Mars
Outline

- What makes a model difficult
- Numerical issues in models
- Model debugging
WHAT MAKES A MODEL DIFFICULT
What makes a model difficult

- Size
- Frequency – a series of related models
- Integer variables
- Quadratic expressions
- Numerical scaling
Model size

- Models typically become large via copies: regions, products, time, ...
- Gurobi is parallel by default
- Parallel MIP consumes memory

Reduction model size is an art
- What should be modeled
- What should be approximated

Solver considerations
- Have enough physical memory (RAM) to load & solve model in memory
- Use 64-bits
- Try compute server or cloud
Frequency: a series of related models

- Model may not be so easy when there are many to solve

- Improve solve times via warm starts
  - Automatic: modify a model in memory rather than create a new model
  - Manual
    - LP: basis and primal/dual starts
    - MIP: start vectors

- Sometimes warm starts hurt more than they help
- Try solving from scratch via concurrent
Modifying a model

- Change coefficients
  - Objective
  - RHS
  - Matrix
  - Bounds
- Change variable types: continuous, integer, etc.
- Add variables or constraints
- Delete variables or constraints

For small changes, modifying a model is more efficient than creating a new model
  - Reuse existing model data
  - Automatically use prior solution as warm-start for new model
Python example: modifying a model

```python
model = read("usa13509.mps")
model.optimize()

    Solved in 7940 iterations and 0.15 seconds
    Optimal objective  1.959148400e+07

x105 = model.getVarByName("x105")
x105.LB = 0.6
model.optimize()

    Solved in 3 iterations and 0.01 seconds
    Optimal objective  1.959149680e+07

model.reset()
model.optimize()

    Solved in 7931 iterations and 0.14 seconds
    Optimal objective  1.959149680e+07
```
Integer variables

- In most cases, integer variables make a model more difficult
- General integer variables tend to be more difficult than binary (0–1)
- Things to consider
  - Which general integers are necessary
  - Can some variables be approximated
Quadratic expressions

- Quadratic expressions are much more complex than linear
  - Especially for constraints: quadratic constraints require the barrier method

- Quadratic is essential for some applications
  - Financial risk
  - Engineering

- Quadratic constraints should *never* be used for logical expressions
  - Ex: $x = 0$ or $y = 0$ should *not* be modeled by $xy = 0$
  - More about logical expressions later
NUMERICAL ISSUES IN MODELS
Numerical issues

- 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

- Typical causes of numerical errors
  - Scale: too large of a range of numerical coefficients
  - Rounding of numerical coefficients
    - Ex: Don’t write 1/3 as 0.333, if possible multiply constraint with 3
Set parameter Presolve to value 2

Gurobi Optimizer version 6.5.0 build v6.5.0rc1 (linux64)
Copyright (c) 2015, Gurobi Optimization, Inc.

Read MPS format model from file model.mps.bz2
Reading time = 0.02 seconds
model: 1218 rows, 1560 columns, 4997 nonzeros
Optimize a model with 1218 rows, 1560 columns and 4997 nonzeros
Coefficient statistics:
  Matrix range    [9e-06, 6e+01]
  Objective range [1e+00, 1e+00]
  Bounds range    [1e-06, 1e+03]
  RHS range       [0e+00, 0e+00]
Presolve removed 993 rows and 1109 columns
Presolve time: 0.00s
Presolved: 225 rows, 451 columns, 1842 nonzeros

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Objective</th>
<th>Primal Inf.</th>
<th>Dual Inf.</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-3.3756129e+02</td>
<td>1.151348e+05</td>
<td>0.000000e+00</td>
<td>0s</td>
</tr>
</tbody>
</table>

Solved in 635 iterations and 0.04 seconds
Infeasible model
Set parameter `Presolve` to value 2
Set parameter `OptimalityTol` to value `1e-4`
Set parameter `FeasibilityTol` to value `1e-4`

...  
Presolve removed 993 rows and 1109 columns  
Presolve time: 0.00s  
Presolved: 225 rows, 451 columns, 1842 nonzeros

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Objective</th>
<th>Primal Inf.</th>
<th>Dual Inf.</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-3.3756129e+02</td>
<td>1.151348e+05</td>
<td>0.000000e+00</td>
<td>0s</td>
</tr>
<tr>
<td>313</td>
<td>-7.9573909e+01</td>
<td>0.000000e+00</td>
<td>0.000000e+00</td>
<td>0s</td>
</tr>
</tbody>
</table>

Solved in 313 iterations and 0.01 seconds  
Optimal objective: -7.957390880e+01
What happened?

- Coefficients are numerically difficult
  
  Coefficient statistics:
  
  - Matrix range \([9e-06, 6e+01]\]
  - Objective range \([1e+00, 1e+00]\]
  - Bounds range \([1e-06, 1e+03]\]
  - RHS range \([0e+00, 0e+00]\]

- Ideally, this model should be reformulated
- Setting `NumericFocus` can help

- Setting numerical tolerances is not the way to fix this model
  - (Not the same as termination criteria)
Consequence of numerical issues – 3

Set parameter Presolve to value 2
Set parameter NumericFocus to value 2

...
Presolve removed 993 rows and 1109 columns
Presolve time: 0.00s
Presolved: 225 rows, 451 columns, 1842 nonzeros

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Objective</th>
<th>Primal Inf.</th>
<th>Dual Inf.</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-3.3756129e+02</td>
<td>1.151348e+05</td>
<td>0.0000000e+00</td>
<td>0s</td>
</tr>
<tr>
<td>391</td>
<td>-2.0000000e+01</td>
<td>0.0000000e+00</td>
<td>0.0000000e+00</td>
<td>0s</td>
</tr>
</tbody>
</table>

Solved in 391 iterations and 0.01 seconds
Optimal objective -1.999999999e+01
Understanding Big-M coefficients

- “Big–M” coefficients represent penalty values or logic

- Overly large big–M values can give slow performance or wrong answers
  - Optimal objective from Gurobi Optimizer: \(-1.47e+08\)
  - Optimal objective from other solver: \(-2.72e+07\)
Example: Wrong answer with Big-M

- $y \leq 1000000 \, x$
  - $x$ 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
Check log file for warnings

- Gurobi will tell you, if it encountered numerical issues

- Warning during optimization:
  - Warning: 1 variable dropped from basis
  - Warning: switch to quad precision
  - Warning: Markowitz tolerance tightened to 0.0625

- Warnings after optimization is finished:
  - Warning: max integrality violation (5.0000e-05) exceeds tolerance
  - Warning: max SOS violation (9.9353e-05) exceeds tolerance
  - Warning: max constraint violation (2.5435e-05) exceeds tolerance
  - Warning: max bound violation (2.5435e-05) exceeds tolerance (possibly due to large matrix coefficient range)

Numerical trouble encountered
Looking at the presolved model

- Model.printStats() function in Python API gives a model overview, can also be used on the presolved model.

- Python Interactive Shell:
  ```python
  model=read("misc07.mps")
  presolved=model.presolve()
  presolved.printStats()
  ```

Statistics for model MISC07_pre:
- Linear constraint matrix: 211 Constrs, 232 Vars, 8260 NZs
- Variable types: 0 Continuous, 232 Integer (232 Binary)
- Matrix coefficient range: [1, 7]
- Objective coefficient range: [75, 570]
- Variable bound range: [1, 1]
- RHS coefficient range: [1, 3]
MODEL DEBUGGING
Model debugging

- Basic concepts
  - Naming variables and constraints
  - Model files

- Advanced debugging
  - Covered during troubleshooting session
Naming variables and constraints

- Set the `VarName` and `ConstrName` attributes to meaningful values
  - `flow_Atlanta_Dallas` is more useful than `x3615`

- Don’t reuse names for multiple constraints or variables
  - API doesn’t care about the `VarName` or `ConstrName` attributes
  - Create unique, descriptive names to help with debugging
Model files

**MPS format**
- Machine-readable
- Full precision
- Order is preserved
- Best for testing

**LP format**
- Easy to read and understand
- May truncate some digits
- Order is not preserved
- Best for debugging

**REW format**
- Variable and constraint names anonymized

**RLP format**
- Variable and constraint names anonymized
Write model in different formats

- Python:
  - `model.update()`
  - `model.write("mymodel.mps")`
  - `model.write("mymodel.rew")`
  - `model.write("mymodel.lp")`
  - `model.write("mymodel.rlp")`
Maximize
  \[ x + y + 2z \]
Subject To
  \[ c0: x + 2y + 3z \leq 4 \]
  \[ c1: x + y \geq 1 \]
Bounds
Binaries
  \[ x \ y \ z \]
End