Migrating an existing model to Gurobi Optimizer

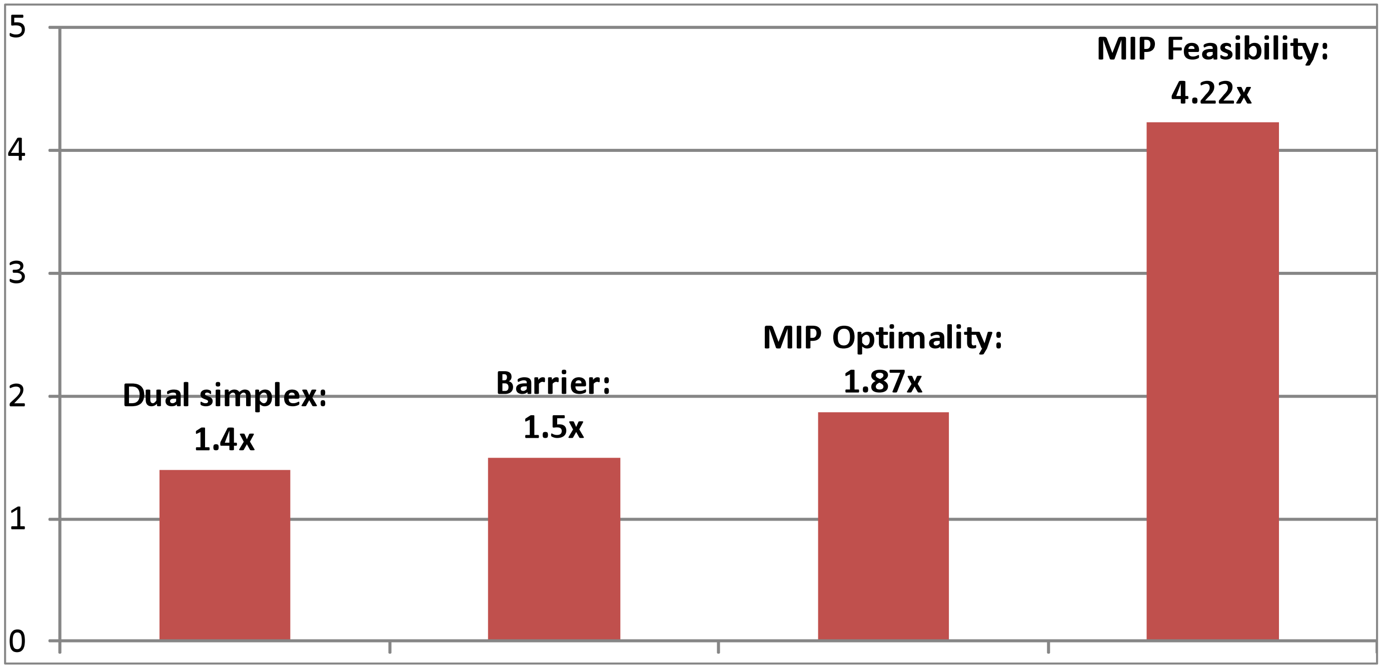
Gurobi
Optimization

Migrating to Gurobi

- Why switch?
  - Performance
  - Cost of Ownership
  - Support
- Migration guide
  - Simple steps for moving your model to Gurobi

Performance

- Gurobi gives better performance



Geometric mean performance ratios (versus CPLEX 12.1) for the Mittelmann LP/MIP benchmarks; data available at http://plato.asu.edu/bench.html

Cost of Ownership

- Using an optimization model in production requires a deployment license
  - Deployment license cost (typical 4-core server):
    - CPLEX: $316,400 (as quoted online by several IBM resellers)
    - Gurobi: $20,400
- Gurobi licenses are flexible
  - All licenses can be used for development or deployment (or both)
  - A single license can be used for more than one application

Support

- Gurobi has the most experienced and accomplished team in the industry
- Gurobi is committed to making you successful with optimization
  - Gurobi Optimization is focused solely on developing and supporting math programming solvers
- We don't view support as a cost
  - It's an integral part of our product offering

Migration is easy

- Gurobi has rich yet lightweight interfaces
  - Similar structure to other optimization engines
  - Find migration option suitable for your code
- Gurobi customers say that code migration is surprisingly easy

Gurobi Optimization

Sample migration scenarios

- Model is written in AMPL
- C program uses matrix interface to
  - CPLEX Callable Library
  - Xpress-Optimizer
- Java program uses Concert Technology
- We'll cover these situations and more

Gurobi Optimization

Migration options

- Migrating model files
- Using a modeling system
- Porting existing code
  - Matrix-based
  - Object-based
- Gurobi parameters
- Advanced concepts

Migration options

- Migrating model files
- Using a modeling system
- Porting existing code
  - Matrix-based
  - Object-based
- Gurobi parameters
- Advanced concepts

Working with model files

- Gurobi supports MPS and LP formats
  - Write your model file using your existing code
  - Virtually no changes needed to existing code

- "Quick and dirty" approach
  - Useful for performance testing

Solving via command-line

- gurobi_cl lets you solve a model from the command-line
  - Usage: gurobi_cl [parameters] filename
  - Example:
    - `gurobi_cl heuristics=0.1 glass4.mps`
      solves glass4.mps with heuristics set to 0.1

- Limited ability to interact with the solver
  - Control limited to Gurobi parameters

Solving via interactive shell

- A complete programming environment
  - Use Python to create a full application
    - Based on objects
    - Using model files

- For migration, useful for
  - Advanced testing
  - Porting code that uses model files

Gurobi Optimization

Simple shell example

```
m = read("afiro.mps")
m.optimize()
if m.status == GRB.OPTIMAL:
  m.printAttr('X')
```

Simple shell example – 2

```
m = read("afiro.mps")
m.optimize()
if m.status == GRB.OPTIMAL:
  for i in m.getVars():
    print i.VarName, i.X, i.RC
```

Gurobi Optimization

Migration options

- Migrating model files
- Using a modeling system
- Porting existing code
  - Matrix-based
  - Object-based
- Gurobi parameters
- Advanced concepts

Gurobi Optimization

Using a modeling system

- With an independent modeling system, switching to Gurobi is extremely easy
  - Obtain licenses
  - Set solver to Gurobi
    - Use IDE (AIMMS, GAMS, MPL)
    - Change a line in model file (AMPL, GAMS)
  - Convert parameter settings
    - Use IDE (AIMMS, GAMS, MPL)
    - Command-line (GAMS)
    - Change the lines in files (AMPL, GAMS)

Example: Select Gurobi

- AIMMS
  - Select Gurobi via menu:
    Settings > Solver Configuraration
- AMPL
  - In model file, add:
    ```
    option solver gurobi_ampl;
    ```
- GAMS
  - In program file, add either:
    ```
    Option LP = Gurobi;
    Option MIP = Gurobi;
    ```
- MPL
  - Add Gurobi via menu:
    Options > Solver menu
  - Solve via menu:
    Run > Solve Gurobi

Example: Set Gurobi Parameters

- AIMMS
  - In menu, select Settings > Project Options
  - In Option Tree, select Specific solvers > Gurobi
  - Set parameters via GUI
- AMPL
  - In model file, add: `option gurobi_options 'presolve 2';`
- GAMS
  - Use command-line flags, options file or IDE
- MPL
  - In menu, select: Options > Gurobi Parameters
  - Set parameters via GUI

Gurobi Optimization

Migration options

- Migrating model files
- Using a modeling system
- Porting existing code
  - Matrix–based
  - Object–based
- Gurobi parameters
- Advanced concepts

Gurobi-specific modeling features

- Gurobi environment
- Lazy updates
- Attributes

- These modeling features need to be considered when porting existing code

Gurobi Optimization

Gurobi environment

- Models are built from an environment
- Parameters are set on an environment

- A model gets its own copy of the environment
  - Once a model is created, subsequent parameter changes in the parent environment are not reflected in the model environment
  - Use getEnv() functions to get the environment from a model

Lazy updates

- Gurobi updates models in batch mode
- Must call update() to use model elements
  - Ex: Call update() after creating a variable before using it in a constraint

- Model creation and updates are efficient
- May require changes to code for other solvers

Accessing attributes

- Object interface
  - get/set methods on the objects
  - C++ example

```
nz = model.get(GRB_IntAttr_NumNZs);
var.set(GRB_DoubleAttr_UB, 1.0);
```

- Matrix interface
  - get/set functions by type (int, double, char, string)
  - C example

```
status = GRBgetintattr(model, "NumNZs", &nz);
status = GRBsetdblattrelement(model, "UB", varidx, 1.0);
```

Role of attributes

- Unified system to access model elements
  - Attributes work the same across all Gurobi interfaces – C, C++, Java, .NET, Python
- Attributes refer to model elements
  - Access via a basic set of get and set functions
    - Attribute name is specified as a parameter
  - Replaces many functions used by other solvers
- Full list in Attributes section of Reference Manual

Selected attributes

- The model itself
  - Number of variables, constraints, nonzeros
  - Solve time
  - Solution status (optimal, infeasible, etc.)
- Individual variables
  - Solution value, upper bound, lower bound
  - Objective coefficients
  - Type – continuous, binary, general integer, etc.
- Individual constraints
  - Values for right-hand side, slack, dual

Gurobi Optimization

Gurobi interfaces

- Matrix-based
  - C
- Object-based
  - C++, Java, .NET, Python

Sparse matrix format

- Compressed sparse row format
  - GRBaddconstrs()
- Compressed sparse column format
  - GRBaddvars()

- Standard formats used by many solvers
  - Use simple arrays to represent
    - Matrix coefficients
    - Index positions for these coefficients
  - Virtually no changes required to existing code

Gurobi Optimization

Object modeling interfaces

- Represent models using objects
  - Objects for variables
  - Objects for constraints
- Function methods to create constraints, columns

- Migrating existing code may require updates to all lines of model building code

Objects in a simple constraint: $x + y \geq 1$

C++

```
model.addConstr(x+y>=1,
  "c1");
```

Java

```
expr = new GRBLinExpr();
expr.addTerm(1.0, x);
expr.addTerm(1.0, y);
model.addConstr(expr,
  GRB.GREATER_EQUAL, 1.0,
  "c1");
```

Gurobi Optimization

Objects in aggregate constraint: $x_1 + \ldots + x_n \leq 2$

C++

```
GRBLinExpr lhs = 0;
for (int i=0; i<n; ++i) {
  lhs += x[i];
}
model.addConstr( lhs <= 2,
  "ub" );
```

Java

```
GRBLinExpr lhs = new
  GRBLinExpr();
for (int i=0; i<n; ++i) {
  lhs.addTerm(1.0, x[i]);
}
model.addConstr(lhs,
  GRB.LESS_EQUAL, 2, "ub");
```

Gurobi Optimization

Objects in objective

- In Gurobi, objective coefficients are specified via attributes on variables
- In other solvers, objective may be specified using an expression

- Pragmatic migration
  - Assign objective expression to an object
    - z = x[1] + x[2] + x[3]
  - Set objective coefficient on the object z

Column modeling via objects

- Similar principle as adding constraints
  - Create column object
  - Add terms
    - Individually
    - Iteratively
  - Add new variable using column object
    - addVar() method

Gurobi Optimization

Error handling

- C matrix interface
  - Virtually every function returns status
  - Nonzero status represents an error code
- Object interface
  - Enclose Gurobi functions in a try block
  - Catch Gurobi exceptions

Memory management

- C
  - Gurobi copies your arrays; you can free them
  - At end, you should free the model & environment
- C++
  - Some get functions create new objects on the heap; your code should free these when finished
  - At end, you should free the model & environment
- Others: use automatic garbage collector

- See examples subdirectory for best practices

Migration options

- Migrating model files
- Using a modeling system
- Porting existing code
  - Matrix-based
  - Object-based
- Gurobi parameters
- Advanced concepts

Gurobi parameters

- Parameters control Gurobi algorithms
  - Termination criteria
  - Tolerances
  - Behavior of LP, MIP, Presolve, IIS
  - Output logs
  - Threads used

- Full list in Parameters section of Reference Manual

Setting Gurobi parameters

- Parameters are set on an environment
- A model gets its own copy of the environment
  - Once a model is created, subsequent parameter changes in the parent environment are not reflected in the model environment
  - Use getEnv() functions to get the environment from a model

Setting parameters from C

- Set time limit of 3600 seconds on master environment

```
status = GRBsetdblparam(env, "TimeLimit", 3600);
```

- Set presolve level to 2 on model

```
status = GRBsetintparam(GRBgetenv(model),
  "Presolve", 2);
```

Setting parameters from Java

- Set time limit of 3600 seconds on master environment

```
env.set(GRB.DoubleParam.TimeLimit, 3600);
```

- Set presolve level to 2 on model

```
model.getEnv().set(GRB.IntParam.Presolve, 2);
```

Common parameters: termination

- TimeLimit: stop after specified seconds
- SolutionLimit: stop after specified number of integer feasible solutions
- NodeLimit: stop after specified number of MIP nodes

Common parameters: tolerances

- MIPGap: stop when the specified relative MIP gap is reached
- MIPGapAbs: stop when the specified absolute MIP gap is reached

Gurobi Optimization

Common parameters: control

- LPMethod: LP algorithm used for nodes & continuous models
- RootMethod: LP algorithm used for root
- Heuristics: Frequency to apply MIP heuristics
- MIPFocus: Whether to focus on optimality, feasibility or a blend
- Cuts: Level of MIP cuts to generate
  - Parameters available for individual cut types

Migration options

- Migrating model files
- Using a modeling system
- Porting existing code
  - Matrix-based
  - Object-based
- Gurobi parameters
- Advanced concepts

Callbacks

- Get information during optimization
  - Ex: LP relaxation values, MIP progress
  - Use for heuristics, solution progress, etc.
- Modify the solver behavior
  - Add MIP cuts
  - Provide a MIP feasible solution
  - Terminate

Informational callbacks

- Implement by writing a function or class
  - Specify where (when) to run callback
    - presolve, simplex, barrier, MIP solution, MIP node, etc.
- Use the cbget function
  - Specify what to query
    - Objective value, best bound, number of integer solutions, etc.
- Illustrated in callback example

Piecewise linear functions

- Gurobi has no modeling feature for piecewise linear functions
- Gurobi does support special ordered sets
  - SOS2 is efficient for piecewise linear functions
  - http://yetanothermathprogrammingconsultant.blogspot.com/2009/06/gams-piecewise-linear-functions-with.html
- Absolute value function is a special case

Semi-continuous variables

- Gurobi supports semi-continuous variables
  - Ex: $x = 0$ or $200 \leq x \leq 400$
- Two steps to model this in Gurobi
  - Specify bounds on the variable
    - 200 and 400 in example above
  - Set variable VType attribute to 'S'

Gurobi Optimization

Logical expressions

- Gurobi does not have modeling features for logical expressions
  - Ex: and, or, not, implies, if and only if

- Model this yourself using standard LP/MIP techniques
  - Examples in many textbooks such as Model Building in Mathematical Programming by H. P. Williams

Try it yourself!

Download a trial copy of Gurobi Optimizer:

http://www.gurobi.com/html/freetrial.html