Migrating an existing model to Gurobi Optimizer
Migrating to Gurobi

Why switch?
- Higher Performance
- Lower Cost of Ownership
- Outstanding Support

This migration guide
- Simple steps for moving your model to Gurobi

Other migration resources
- Switching from CPLEX
- Switching from XPRESS
Higher Performance

- Gurobi gives better performance

Geometric mean performance ratios (versus CPLEX 12.2.0.2) for the Mittelmann LP/MIP benchmarks; data available at http://plato.asu.edu/bench.html
Lower Cost of Ownership

- Using an optimization model in production requires a deployment license
  - Deployment license cost (typical 1 CPU; 4-core server):
    - CPLEX: $72,240 (as quoted online by several IBM resellers)
    - Gurobi: $24,000

- Support for both CPLEX and Gurobi is 20% of license list cost
  - Annual support for the Server license described above
    - CPLEX: $14,448
    - Gurobi: $4,800

- Gurobi licenses are more flexible
  - All licenses can be used for development or deployment (or both)
  - A single license can be used for more than one application
Outstanding Support

- Gurobi gives you direct access to optimization experts
  - Each has a PhD and years of experience working with commercial models

- 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 easier than you may think

- Gurobi has rich yet lightweight interfaces
  - Similar structure to other optimization engines
  - Find migration option suitable for your code

- Includes support for both MPS and LP file formats

- Gurobi customers say that code migration is surprisingly easy
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
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
Simple shell example

```python
m = read("afiro.mps")
m.optimize()
if m.status == GRB.OPTIMAL:
    m.printAttr('X')
```
Simple shell example – 2

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

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

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
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 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 interfaces

- Matrix-based
  - C, MATLAB, R

- 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
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++**

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

**Java**

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

- Similar principle as adding constraints
  - Create column object
  - Add terms
    - Individually
    - Iteratively
  - Add new variable using column object
    - addVar() method
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
  
  ```c
  status = GRBsetdblparam(env, "TimeLimit", 3600);
  ```

- Set presolve level to 2 on model
  
  ```c
  status = GRBsetintparam(GRBgetenv(model), "Presolve", 2);
  ```
Setting parameters from Java

- Set time limit of 3600 seconds on master environment
  
  ```java
  env.set(GRB.DoubleParam.TimeLimit, 3600);
  ```

- Set presolve level to 2 on model
  
  ```java
  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
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
Piecwise linear functions

- Gurobi has no modeling feature for piecewise linear functions
- Gurobi does support special ordered sets
  - SOS2 is efficient for piecewise linear functions
- 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'
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/products/gurobi-optimizer/try-for-yourself

Explore more on switching from CPLEX or XPRESS:
http://www.gurobi.com/resources/switching-to-gurobi/switching-from-cplex
http://www.gurobi.com/resources/switching-to-gurobi/switching-from-xpress

Contact us with any questions:
support@gurobi.com, sales@gurobi.com, or +1 713 871 9341