Gurobify

A GAP interface to Gurobi.

0.1

23/02/2017

Jesse Lansdown

Jesse Lansdown
Email: jesse.lansdown@research.uwa.edu.au
Homepage: www.jesselansdown.com
Address: Jesse Lansdown
Lehrstuhl B für Mathematik
RWTH Aachen
Pontdriesch 10 - 16
52062 Aachen
Germany

Abstract

Gurobify provides an interface to the Gurobi Optimizer software from GAP. It enables the creation and modification of mixed integer and linear programmming models which can be solved directly by Gurobi from within the GAP environment.

Copyright

Acknowledgements

TODO

Contents

Chapter 1

Introduction

1.1 What is Gurobify?

[GAP16] [gur16a] [GH16] [Hor16] Gurobify is a GAP package which provides an interface to the optimisation software Gurobi.

1.2 Prerequisites

Gurobi requires the following software to be installed.

- GAP 4.8 (or later)
- Gurobi 7.0
- Autotools

Autotools may be installed on MacOSC using homebrew with the commands *brew install autoconf* and *brew install automake*. If you want to regenerate the documentation for any reason, then the following will also be required.

- AutoDoc 2016.03.08 (or later)
- GAPDoc 1.5 (or later)

Although Gurobi is proprietary software, it is available free for academic use. According to the Gurobi website,

"Gurobi makes most of our solvers available to recognized degree-granting academic institutions free of charge" [gur16c],

and

"The free Academic License for Gurobi has all the features and performance of the full Gurobi Optimizer. A free Academic License has no limits on model size. The only restrictions on the use of these licenses are:

- They can only be used by faculty, students, or staff of a degree-granting academic institution
- They can only be used for research or educational purposes
- They must be validated from a recognized academic domain, as described below.

Note, free academic licenses expire twelve (12) months after the date on which your license was generated, but eligible faculty, students, or staff can renew a license by repeating the above process" [gur16b].

For up to date information on Gurobi licences, please refer to the Gurobi website. A link can be found in the Appendix, or through a simple search online.

1.3 Installation

To install Gurobify, first unpack it in the pkg directory of the GAP installation directory. You may place Gurobify in a different location, so long as its parent directory is called "pkg". Installing Gurobify outside of the GAP root pkg directory is not recommended, but is useful for example when administrative privelages are needed to access the GAP root directory. Next run the following commands in the terminal from within the Gurobify directory.

Example
```
./autogen.sh
./congigure --with-gurobi=<gurobi path> [--with-gaproot=<gap path>]
make
```

The --with-gurobi=<gurobi path> is a necessary argument, and normally looks something like this on MacOSX,

--with-gurobi=/Library/gurobi701/mac64

or something like this on Linux,

--with-gurobi=/opt/gurobi701/linux64

The [--with-gaproot=<gap path>] is an optional argument, and is not normally necessary if Gurobify is placed in the pkg directory of the GAP root directory. If, however, Gurobify is located in a non-root pkg directory, then this argument must be included. It normally looks something like this,

--with-gaproot=/opt/gap4r8/

1.4 Documentation

Within the Gurobify directory there is a subdirectory called doc. This directory contains the documentaion for Gurobify in the form of a pdf file called "manual.pdf" as well as in html form. To access the html version of the manual, open the file called "chap0.html". The documentation can be regenerated by running the following command in the terminal from the Gurobify directory, though this should not be necessary.

Example
```
gap.sh makedoc.g
```

1.5 Loading Gurobify

Open GAP and load Gurobify with the command *LoadPackage("Gurobify");*. You should see something like the following.

Example
```
gap>  LoadPackage("Gurobify");
-----------------------------------------------------------------------------
Loading  Gurobify 1.0 (Gurobify provides an interface to Gurobi from GAP.)
by Jesse Lansdown (www.jesselansdown.com).
Homepage: https://github.com/jesselansdown/Gurobify/
-----------------------------------------------------------------------------
true
```

Note that if you have Gurobify located somewhere other than the GAP root directory's pkg directory, then you must run GAP with the following command to enable GAP to find Gurobify.

Example
```
gap.sh -l ";/alternative/path/to/Gurobify"
```

where */alternative/path/to/Gurobify* is the path to the directory which contains */pkg/Gurobify* as subdirectories.

Chapter 2

Getting Started

2.1 Getting Started

TODO

2.2 Minimal working example?

TODO

Example

```
gap> model := GurobiNewModel(["BINARY", "BINARY", "BINARY"], [1.,2.,1.]);
<object>
gap> GurobiSetIntegerAttribute(model, "ModelSense", -1);
gap> GurobiAddConstraint(model, [2, 2, 2], "<", 6, "Initial Constraint");
gap> GurobiAddConstraint(model, [1, 2, 3], ">", 5, "Initial Constraint");
gap> GurobiOptimizeModel(model);
2
gap> GetSolution(model);
[ 1., 1., 1. ]
gap> GurobiReset(model);
gap> GurobiSetIntegerAttribute(model, "ModelSense", 1);
gap> GurobiOptimizeModel(model);
2
gap> GetSolution(model);
[ 0., 1., 1. ]
gap> GurobiWriteToFile(model, "test.lp");
gap> re_model := GurobiReadModel("test.lp");
<object>
gap> GurobiAddConstraint(re_model, [1, 1, 1], ">", 3, "Other Constraint");
gap> GurobiOptimizeModel(re_model);
2
gap> GetSolution(re_model);
[ 1., 1., 1. ]
gap> GurobiAddConstraint(re_model, [0, 1, 1], "<", 1, "Other Constraint");
gap> GurobiOptimizeModel(re_model);
3
gap> GurobiDeleteAllConstraintsWithName(re_model, "Other Constraint");
gap> GurobiOptimizeModel(re_model);
2
```

```
gap> GetSolution(re_model);
[ 0., 1., 1. ]
gap> SetTimeLimit(re_model, 0.0001);
true
gap> GurobiOptimizeModel(re_model);
9
gap> SetTimeLimit(re_model, 0.01);
true
gap> GurobiOptimizeModel(re_model);
2
```

Chapter 3

Using Gurobify

3.1 Creating or reading a model

TODO intro to section

3.1.1 GurobiReadModel

▷ `GurobiReadModel(`*`ModelFile`*`)` (function)

Returns: a Gurobi model.

Takes a model file, reads it and creates a Gurobi model from it. ModelFile is the name of the file as a string, with the appropriate extension, and including the path if the file is not located in the current GAP working directory. Gurobi accepts files of type .mps, .rew, .lp, .rlp, .ilp, or .opb. Refer to the gurobi documentation for more infomation on which file types can be read.

3.1.2 GurobiNewModel

▷ `GurobiNewModel(`*`VariableTypes, ObjectiveFunction`*`)` (function)

Returns: a Gurobi model.

Creates a gurobi model with variables defined by VariableTypes and an objective function given by ObjectiveFunction. VariableTypes must be a list, with entries indexed by the set of variables, and entries corresponding to the type of variable, as a string. Accepted variable types are "CONTINUOUS", "BINARY", "INTEGER", "SEMICONT", or "SEMIINT". Refer to the Gurobi documentation for more information on the variable types. ObjectiveFunction is a list, with entries indexed by the set of variables, where each entry corresponds to the coefficient of the variable in the objective function. ObjectiveFunction takes only double values.

3.2 Adding and deleting constraints

TODO

3.2.1 GurobiAddConstraint

▷ `GurobiAddConstraint(`*`Model, ConstraintEquation, ConstraintSense, ConstraintRHSValue, ConstraintName`*`)` (function)

Returns:

Adds a constraint to a gurobi model. ConstraintEquation must be a list, with entries indexed by the variable set, such that each entry is the coefficient of the corresponding variable in the constraint equation. The ConstraintSense must be one of "<", ">" or "=", where Gurobi interprets < as <= and > as >=. The ConstraintRHSValue is the value on the right hand side of the constraint. A constraint may also be given a name, which helps to identify the constraint if it is to be deleted at some point. May also take an empty string "" if no name is needed. Note that a model must be updated or optimised before any additional constraints become effective.

3.2.2 GurobiDeleteAllConstraintsWithName

▷ `GurobiDeleteAllConstraintsWithName(`*`Model, ConstraintName`*`)` (function)

Returns:

Deletes all constraints from a model with the name ConstraintName. Returns the updated model.

3.3 Optimizing a model

TODO

3.3.1 GurobiOptimizeModel

▷ `GurobiOptimizeModel(`*`Model`*`)` (function)

Returns: Optimisation status.

Takes a Gurobi model and optimises it. Returns the optimisation status code which indicates the outcome of the optimisation. A status code of 2 indicates that a feasible solution was found, a status code of 3 indicates the model is infeasible. There a number of other status codes. Refer to the Gurobi documentation for more information about status codes. The model itself is altered to reflect the optimisation, and more information about can be obatained using other functions, in particular the GurobiGetAttribute and GurobiGetAttributeArray functions.

3.3.2 GurobiReset

▷ `GurobiReset(`*`Model`*`)` (function)

Returns:

Reset all information associated with a solution for the model.

3.3.3 GetSolution (for IsGurobiModel)

▷ `GetSolution(`*`Model`*`)` (operation)

Returns: Solution

Display the solution found for a successfuly optimised model.

3.4 Querying attributes and parameters

TODO

3.5 Querying other attributes and parameters

In addition to these specific queries given in the previous section, all other gurobi parameters and attributes which take integer or double values can be queried using GurobiGetIntegerParameter("ParameterName"), GurobiGetDoubleParameter("ParameterName"), GurobiGetIntegerAttribute("AttributeName") or GurobiGetDoubleAttribute("AttributeName") respectively, where "ParameterName" and "AttributeName" are strings given exactly as in the Gurobi documentation. See the Appendix for links to the relevant documentation.

3.5.1 GurobiGetIntegerParameter

▷ `GurobiGetIntegerParameter(`*`Model, ParameterName`*`)` (function)

Returns: parameter value

Takes a Gurobi model and retrieve the value of a integer-valued parameter. Refer to the Gurobi documentation for a list of parameters and their types.

3.5.2 GurobiGetDoubleParameter

▷ `GurobiGetDoubleParameter(`*`Model, ParameterName`*`)` (function)

Returns: parameter value

Takes a Gurobi model and retrieve the value of a double-valued parameter. Refer to the Gurobi documentation for a list of parameters and their types.

3.5.3 GurobiGetIntegerAttribute

▷ `GurobiGetIntegerAttribute(`*`Model, AttributeName`*`)` (function)

Returns: attibute value

Takes a Gurobi model and retrieve the value of an integer-valued attribute. Refer to the Gurobi documentation for a list of attributes and their types.

3.5.4 GurobiGetDoubleAttribute

▷ `GurobiGetDoubleAttribute(`*`Model, AttributeName`*`)` (function)

Returns: attibute value

Takes a Gurobi model and retrieve the value of a double-valued attribute. Refer to the Gurobi documentation for a list of attributes and their types.

3.5.5 GurobiGetAttributeArray

▷ `GurobiGetAttributeArray(`*`Model, AttributeName`*`)` (function)

Returns: attibute array

Takes a Gurobi model and retrieve an attribute array. Can only get value of attributes arrays which take integer or double values, Refer to the Gurobi documentation for a list of attributes and their types.

3.6 Modifying attributes and parameters

TODO

3.6.1 SetTimeLimit (for IsGurobiModel, IsFloat)

▷ `SetTimeLimit(`*`Model, TimeLimit`*`)` (operation)

Returns: true

Set a time limit for a Gurobi model. Note that TimeLimit should be a float, however an integer value can be given which will be automatically converted to a float.

3.6.2 SetBestObjectiveBoundStop (for IsGurobiModel, IsFloat)

▷ `SetBestObjectiveBoundStop(`*`Model, BestObjectiveBoundStop`*`)` (operation)

Returns: true

Optimisation will terminate if a feasible solution is found with objective value at least as good as BestObjectiveBoundStop. Note that BestObjectiveBoundStop should be a float, however an integer value can be given which will be automatically converted to a float.

3.6.3 SetCutOff (for IsGurobiModel, IsFloat)

▷ `SetCutOff(`*`Model, CutOff`*`)` (operation)

Returns: true

Optimisation will terminate if the objective value is worse than CutOff. Note that CutOff should be a float, an integer value can be given which will be automatically converted to a float.

3.7 Modifying other attributes and parameters

3.7.1 GurobiSetIntegerParameter

▷ `GurobiSetIntegerParameter(`*`Model, ParameterName, ParameterValue`*`)` (function)

Returns:

Takes a Gurobi model and assigns a value to a given integer-valued parameter. ParameterValue must be a integer value. Refer to the Gurobi documentation for a list of parameters and their types.

3.7.2 GurobiSetDoubleParameter

▷ `GurobiSetDoubleParameter(`*`Model, ParameterName, ParameterValue`*`)` (function)

Returns:

Takes a Gurobi model and assigns a value to a given double-valued parameter. ParameterValue must be a double value. Refer to the Gurobi documentation for a list of parameters and their types.

3.7.3 GurobiSetIntegerAttribute

▷ `GurobiSetIntegerAttribute(`*`Model, AttributeName, AttributeValue`*`)` (function)

Returns:

Takes a Gurobi model and assigns a value to a given integer-valued attribute. AttributeValue must be a double value Refer to the Gurobi documentation for a list of attributes and their types.

3.7.4 GurobiSetDoubleAttribute

▷ `GurobiSetDoubleAttribute(`*`Model, AttributeName, AttributeValue`*`)` (function)

Returns:

Takes a Gurobi model and assigns a value to a given double-valued attribute. AttributeValue must be a double value Refer to the Gurobi documentation for a list of attributes and their types.

3.8 Other

3.8.1 GurobiWriteToFile

▷ `GurobiWriteToFile(`*`Model, FileName`*`)` (function)

Returns:

Takes a model and writes it to a file. File type written is determined by the FileName suffix. File types include .mps, .rew, .lp, .rlp, .ilp, .sol, or .prm Refer to the gurobi documentation for more infomation on which file types can be read.

3.8.2 GurobiUpdateModel

▷ `GurobiUpdateModel(`*`Model`*`)` (function)

Returns:

Takes a model and updates it. Changes to parameters or constraints are not processed until the model is either updated or optimised.

Chapter 4

Examples

4.1 Examples

TODO

Chapter 5

Appendix

5.1 Links to some Gurobi documentation

For more information on Gurobi parameters, attributes, and status codes, see the following links:

- Attributes: http://www.gurobi.com/documentation/7.0/refman/attributes.html
- Parameters: http://www.gurobi.com/documentation/7.0/refman/parameters.html
- Status codes: https://www.gurobi.com/documentation/7.0/refman/optimization_status_codes.html
- Licencing: http://www.gurobi.com/products/licensing-pricing/licensing-overview

References

[GAP16] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.8.6*, 2016. 4

[GH16] Sebastian Gutsche and Max Horn. *AutoDoc – a GAP package, Version 2016.03.08*, 2016. 4

[gur16a] Gurobi Optimization, Inc. *Gurobi Optimizer Reference Manual*, 2016. 4

[gur16b] Gurobi Optimizer Academic Program. http://www.gurobi.com/pdfs/Pricing-Academic.pdf, 2016. [Online; accessed 17-March-2017]. 5

[gur16c] Licensing Overview. http://www.gurobi.com/products/licensing-pricing/licensing-overview, 2016. [Online; accessed 17-March-2017]. 4

[Hor16] Max Horn. *PackageMaker – a GAP package, Version 0.8*, 2016. 4

Index