SolverStudio

An integrated environment for optimisation using modelling languages within Excel

Andrew Mason

28 Feb 2013

http://solverstudio.org/

Documentation written by Oscar Dowson odow003@aucklanduni.ac.nz

Contents

1.0 General

1.0 What is SolverStudio?

SolverStudio is an integrated Excel add-in that makes it easy to develop and deliver optimization models using the familiar Excel environment. SolverStudio adds a text editor to Excel that allows an optimization model to be created using AMPL, GAMS, GMPL, PuLP, COOPR/Pyomo or Gurobi and then embedded and saved within a spreadsheet. SolverStudio provides an integrated data editor that allows model data (such as parameters and sets) to be stored and edited on the spreadsheet. SolverStudio's Solve button runs the model while seamlessly managing data transfers with the spreadsheet. AMPL & GAMS models can be solved locally, or in the cloud using NEOS.

SolverStudio is being developed and supported by Andrew Mason at the Department of Engineering Science, University of Auckland, New Zealand.

This document forms part of the SolverStudio documentation. See also the example spreadsheets which show much of what can be done, and also the web site http://solverstudio.org. This site includes a link to an online SolverStudio/AMPL tutorial.

1.1 Installation

SolverStudio can be downloaded as a .zip file from http://solverstudio.org. This .zip file should be extracted to create a folder (named SolverStudio) containing the SolverStudio add-in, support files needed by the add-in and examples.

To install SolverStudio, first copy the SolverStudio folder (containing the examples and various subdirectories) to the location of your choice.

Next, double click the **setup.exe** file (within a folder called SolverStudio); this will install Microsoft .Net v4 on your computer (if it is not already installed), and then automatically run the SolverStudio.vsto 'click-once' installer which installs the add-in within Excel. (If you already have .Net 4, you can simply run the click-once installer yourself by double-clicking the SolverStudio.vsto file.) If a *Publisher cannot be verified* window appears, click **Install**. This will install the add-in within Excel. Note that this process can take some minutes as Windows verifies the software.

Open Excel workbook to find the SolverStudio Ribbon menu under the **Data** Tab.

To uninstall SolverStudio, use the *Add/Remove Programs* feature in the Windows Control Panel. Depending on your setup, you may also be able to use the Uninstall button in the "About SolverStudio" dialogue.

1.2 FAQ's

Q: I have SolverStudio installed but the SolverStudio Ribbon menu doesn't appear under the Data Tab.
A: This most often happens after you force quit during an optimisation. In Excel, try *File>Options>Add-Ins* then select *Disabled Items* from the drop-down menu at the bottom

and click *Go*. Check SolverStudio and click *Enable*.
A: Uninstall SolverStudio using the *Add/Remove Programs* feature in the Windows Control Panel. Either re-install your existing copy of SolverStudio or download a new copy from here.

Q: How do I save my models?
A: Saving your workbook will also save any models contained in that book. Simply open the book on any computer with SolverStudio installed to load them.

Q: Will SolverStudio run in Excel 2003?
A: SolverStudio will only work in Excel 2007 and later.

1.3 Ribbon Menu

SolverStudio's commands are accessed via the SolverStudio menu in the ribbon, and via the model editing pane opened using "Show Model".

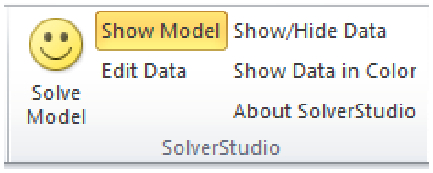


Figure 1 - Ribbon Menu

Solve Model
Solves the current model

Show Model
Shows/hides the model editor. Once you have set up data items on the spread sheet using the SolverStudio Data Items Editor, you use SolverStudio's Show Model button to show and edit your model using PuLP, or any of SolverStudio's other supported modelling languages (as selected using SolverStudio's Language menu).

Edit Data
Shows/hides the Data Items Editor.

Show/Hide Data
Turns on and off the highlighting of data items.

Show Data in Color
Used in conjunction with Show/Hide data. If **Show Data in Color**, the names of the data items will be colored.

About SolverStudio
This brings up a box containing information about SolverStudio.

1.4 Data Items Editor
The Data Items Editor enables the user to give a name to data on a sheet to make that data available within a model.

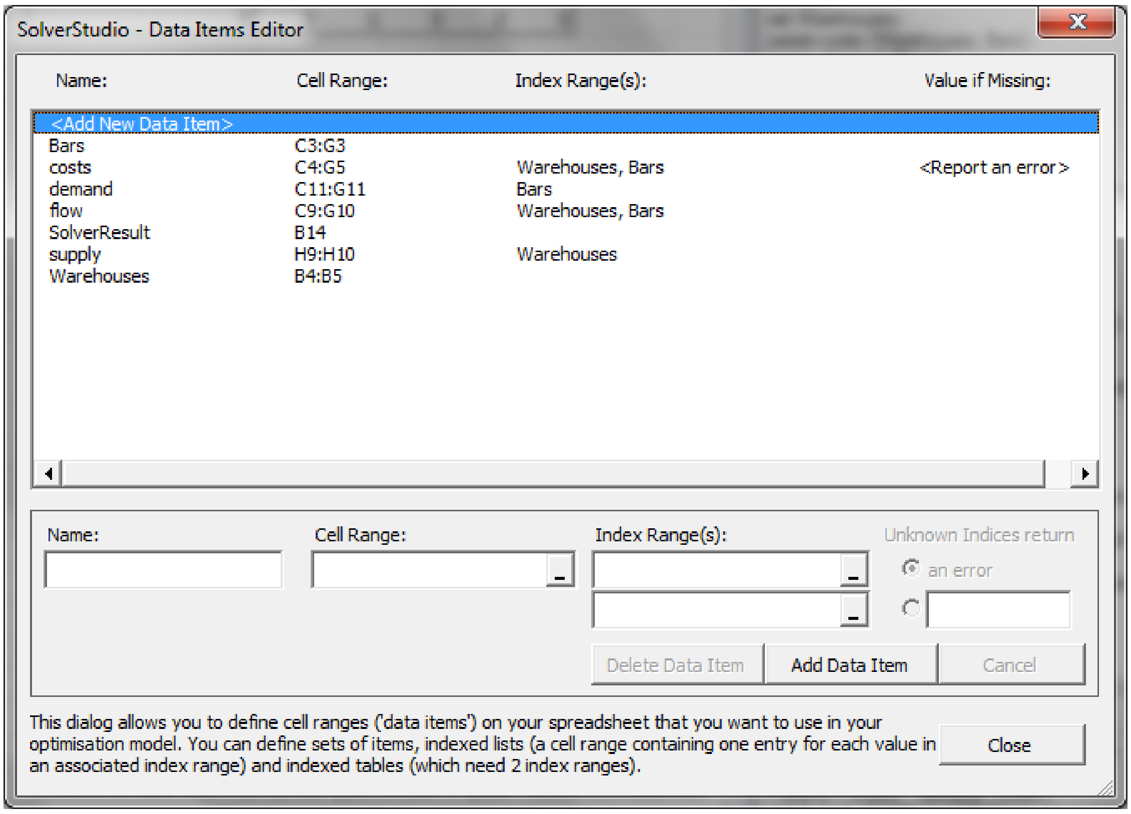
1.4.1 Data Types
There are 3 types of data supported by SolverStudio

- A list of items, such as a list of warehouses, or a list of bars, or a list of arcs each identified by a 'from' and a 'to' node. In PuLP, this will appear as a list of values, or a list of (to, from) tuples respectively. In other modelling languages, this is typically a 'set' of values or tuples.
- A list of indexed items, such as the supply at each warehouse. This will be a dictionary in PuLP of values. Dictionaries can contain values or tuples, and can be indexed by values or tuples. In other modelling languages, this will be an indexed parameter.
- A table, such as the distance from each warehouse to each bar. This table has a row index, e.g. a warehouse, and a column index, e.g. a bar. This appears in PuLP as a dictionary with each entry indexed by a single tuple formed by the row and column indices (in the order specified in the data items dialog). In other modelling languages, this is typically a doubly-indexed parameter. (Higher order indexing, such as using triples, is also supported, if the indices are tuples.)

The example below shows how the data items editor is used to named various data itewms on the sheet. This contains examples of each type of data:

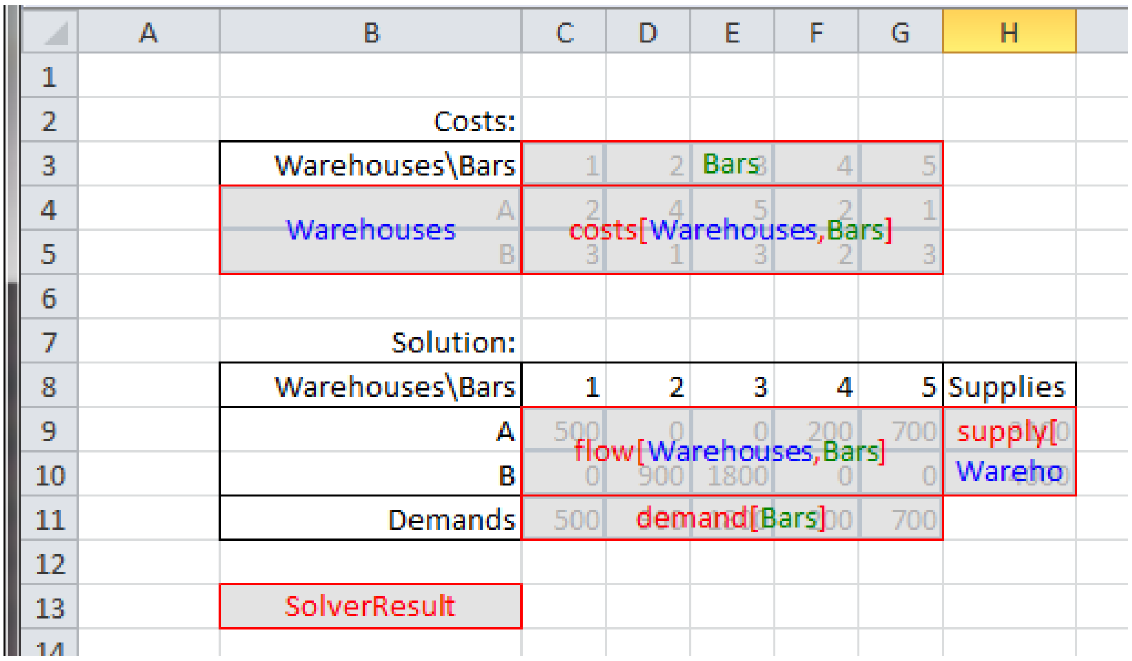
-*Bars*, *Warehouses* and *SolverResult* are lists; they are not indexed.
-*supply* and *demand* are both single indexed parameters.
-*costs* and *flow* are both doubly indexed parameters.

Advanced Users: It is possible to give the range an index or value as either an excel range "A1:B2", a name of an existing item "Bars" or as a formula "=OFFSET("A1",0,0,COUNTA("A1:A9"),1)".


SolverStudio - Data Items Editor

Name:	Cell Range:	Index Range(s):	Value if Missing:
<Add New Data Item>			
Bars	C3:G3		
costs	C4:G5	Warehouses, Bars	<Report an error>
demand	C11:G11	Bars	
flow	C9:G10	Warehouses, Bars	
SolverResult	B14		
supply	H9:H10	Warehouses	
Warehouses	B4:B5		

This dialog allows you to define cell ranges ('data items') on your spreadsheet that you want to use in your optimisation model. You can define sets of items, indexed lists (a cell range containing one entry for each value in an associated index range) and indexed tables (which need 2 index ranges).



	A	B	C	D	E	F	G	H
1								
2		Costs:						
3		Warehouses\Bars	1	2	3	4	5	
4		A	2	4	5	2	1	
5		B	3	1	3	2	3	
6								
7		Solution:						
8		Warehouses\Bars	1	2	3	4	5	Supplies
9		A	500	0	0	200	700	
10		B	0	900	1800	0	0	
11		Demands	500				700	
12								
13		SolverResult						

Figure 2 - Data Items Editor (top) and highlighted example (bottom)

1.3.2 Data Representation (advanced users)

SolverStudio uses Excel's Name Manager to store all its data. The following VBA could be used to add the Data Item *costs* in the example above:

```
ActiveSheet.Names.Add Name:="costs", _
   RefersTo:="='Sheet1'!$C$4:$G$5"
ActiveSheet.Names.Add Name:="costs.rowindex", _
   RefersTo:="='Sheet1'!$B$4:$B$5", Visible:=False
ActiveSheet.Names.Add Name:="costs.rowindex.dirn", _
   RefersTo:="row", Visible:= False
ActiveSheet.Names.Add Name:="costs.columnindex", _
   RefersTo:="='Sheet1'!$C$3:$G$3", Visible:= False
ActiveSheet.Names.Add Name:="costs.columnindex.dirn", _
   RefersTo:="column", Visible:= False
ActiveSheet.Names.Add Name:="costs.badindex", _
   RefersTo:=1, Visible:= False
'ActiveSheet.Names.Add Name:="costs.dirn", _
   RefersTo:="", Visible:=False
```

2.0 Languages

2.1 AMPL

AMPL is a modelling language for both linear and non-linear optimization problems that was developed at Bell Laboratories. In SolverStudio, AMPL can be accessed by selecting *'AMPL'* from the language menu.

2.1.1 Installation

AMPL is not included with SolverStudio. To install it you can either purchase it here or use the **Install AMPL Student Version** menu item (detailed below).

You can test your installation by opening the *'AMPL(NEOS) + GMPL Examples.xlsx'* workbook included with SolverStudio and running an example that ends in '-AMPL'.

2.1.2 Data Command

When the user clicks **Solve Model** SolverStudio will write all declared items in the model file with matching data items to a text file called '*SheetData.dat*'. To allow your model to access this, your model should include *data SheetData.dat;* somewhere before the solve command.

2.1.3Menu Items

View Last Data File

This will open the last *'SheetData.dat'* file. This is useful when debugging, and checking that SolverStudio has written the data in the way you expect.

Import Data File

This allows you to import an AMPL data file.

Fix Minor Errors

When ticked, upon execution SolverStudio will attempt to correct any common errors that users tend to (or choose to) make. These include:

- A missing *data SheetData.dat;*
- A missing *solve;*
- A missing *display item1 > Sheet;*
- The wrong version of CPLEX (student/full)

Install AMPL Student Version

This will download and install the AMPL Student version. The student version is limited to problems of 300 variables and 300 constraints and objectives.

2.1.4 Advanced Options

So long as **Fix Minor Errors** is checked, these commands are optional.

data SheetData.dat – this must come before the *solve* statement. To use another data file change *SheetData.dat* to *path/to/APML/data/file.dat*

option solver cplex – this must come before the *solve* statement. To use another solve replace *cplex* with the solver of your choice. The solver path must be in the system environment variable *PATH*.

(optional) option display_1col9999999 – this must come after the *solve* statement. Use this option if SolverStudio fails to read in your AMPL results. It should not be needed in SolverStudio 0.5 or later.

display ***name*** *> Sheet* - this must come after the *solve* statement. Use this to return items to the sheet. You can return the duals on constraints be creating a data item with the same name as a constraint, and adding *display constrName > Sheet;* to the bottom of your model file.

References:

http://solverstudio.org/languages/ampl/
http://ampl.com/
http://ampl.com/BOOK/index.html

2.2 GAMS

GAMS is a widely used commercial modelling environment. You can access GAMS in SolverStudio by selecting *'GAMS'* from the language menu.

In this latest release of SolverStudio, we have moved away from storing the data as text and introduced integrated GDX support (using the C# API after GAMS generously allowed us to ship the GDX DLL's). This means SolverStudio will be much faster, especially for larger data sets.

This results in two necessary changes to existing SolverStudio GAMS models.

1. *$INCLUDE "SheetData.dat"* should be replaced with *$GDXIN "SheetData.gdx"*
2. SolverStudio will return to the sheet any items with matching data items specified by *display var1, [var2]*. To return the marginal of a variable use *display var1.m*. [For those more familiar with GAMS, the display command gets changed to *execute_unload "results.gdx" var1, [var2].* This *results.gdx* is saved to the SolverStudio working directory (*file>View Working* Directory) You could import it using the **Import a GDX file** menu command (detailed below)]

2.2.1 Installation

GAMS is not included with SolverStudio. To install it, go to the GAMS Download page and download and install the latest version (this will give you access to a demo version, with limited numbers of variables and constraints). You can upgrade to a full version by purchasing a license.

Alternatively, to run GAMS models without a version of GAMS installed, try GAMS on NEOS.

You can test your installation by opening the *'GAMS(NEOS) Examples.xlsx'* workbook included with SolverStudio and running an example that ends in '-GAMS'.

2.2.2 Menu Items

View last GAMS result file

This will open the most recent list file (.lst) created by GAMS. This is useful when debugging your model.

View last GAMS input data file in GAMSIDE

This menu item will open the last GDX file created by SolverStudio in GAMSIDE (provided with the free demo version of GAMS). This is useful for checking SolverStudio has built your data in the way you expect.

Fix Minor Errors
Due to the change to integrated GDX support, SolverStudio models no longer require *'$INCLUDE "SheetData.dat"*. Instead they require *'$GDXIN "SheetData.gdx"* followed by *'$LOAD item1,[item2],[item3]'*. If **Fix Minor Errors** is selected, SolverStudio will attempt to fix any errors in these commands (such as $INCLUDE instead of $GDXIN) before compilation. If the model does not specify a *'$LOAD item1, item2, … '* SolverStudio will add all declared sets and parameters with matching data items.

Import a GDX file
Use this menu item to import a GDX file into Excel and add the items to the Data Items Editor. You have the option to import variables only, and to group data items for easier viewing. **You do not need a version of GAMS installed to use this.**

2.2.3 Advanced Options
So long as **Fix Minor Errors** is checked, these commands are optional.

$GDXIN "SheetData.gdx"	- this will load the GDX file created from the data on
$LOAD item1, item2, item3, …	- the sheet. To use a different one, change
$GDXIN	- *SheetData.gdx* to the name of your GDX file.
	- without a load command, SolverStudio will load all

- sets and parameters defined in your model with matching data items.

display var1,var2, … ; - use this command to selectively return variables to the Sheet. You can return the values of constraints be creating a data item with the same name as a constraint and going *display constraintName, ….* You can return the dual of a constraint of variable by adding the suffix *.m* to the name of the item.

References:
http://solverstudio.org/languages/gams/
http://www.gams.com/default.htm
http://www.gams.com/docs/document.htm

2.3 NEOS
The NEOS Server is a free server that allows models to be solved in the cloud using various languages and solvers. SolverStudio can use NEOS to solve models written in either GAMS or AMPL.

Models should be identical to those used with the local copy of AMPL/GAMS (indeed you should be able to solve the same model locally or in the cloud by selecting either 'AMPL/GAMS' or 'AMPL/GAMS on NEOS' from the language menu.

NOTE: All jobs submitted to NEOS are logged and available for downloading, and so using SolverStudio with NEOS results in your model and data becoming public. You also agree to the NEOS terms and conditions.

2.3.1 GAMS on NEOS
This can be accessed in SolverStudio by selecting *'GAMS on NEOS'* from the language menu. **You do not need a version of GAMS installed to use GAMS on NEOS.**

GAMS on NEOS returns its data in the form of a GDX file. If you have a large number of variables (or you wish to see the whole thing rather than only the variables returned via display) you can always import this using the **Import a GDX file** menu option.

All examples in the *'GAMS(NEOS) Examples.xlsx'* workbook included with SolverStudio can be solved via NEOS.

2.3.2 AMPL on NEOS

This can be accessed in SolverStudio by selecting *'AMPL on NEOS'* from the language menu. You do not need a version of AMPL installed to use AMPL on NEOS.

All the AMPL examples included in the *'AMPL(NEOS) + GMPL Examples.xlsx'* workbook can be solved via NEOS.

2.3.3 Menu Items

Run In Short Queue

This runs you model in the NEOS short queue (time limit of 5 minutes). Only select this if you are running a small problem.

Choose Solver

Use this menu to select which solver you send you model to (if using AMPL on NEOS, this will update the last *option solver xxx;* in your model file).

Presently only some solvers are supported by SolverStudio.

Update NEOS Solvers

The list of NEOS solvers is not generated dynamically. Therefore, this list may get out of date as NEOS adds/removes various solvers. Use this menu item to update the list.

http://solverstudio.org/languages/neos/
http://www.neos-server.org/neos/

2.4 COOPR/Pyomo

Coopr is a collection of open-source optimization-related Python packages that supports a diverse set of optimization capabilities for formulating and analysing optimization models. You can access COOPR in SolverStudio by selecting *'COOPR'* from the language menu.

2.4.1 Installation

Coopr is not included with SolverStudio. The user will first need a version of CPython (2.6 or 2.7) which they can get here, if not already installed.

To install Coopr, go to the COOPR Downloads page and download the latest *Coopr_xxx_setup.exe* (at the time of writing this was 3.2.6148). Run the executable.

Once completed by sure to add the path of *'pyomo.exe'* to the system path environment variable. Unless you have made changes, this will be '*\Python2X\Scripts*'.

You can test your installation by opening the *'COOPR Examples.xlsx'* workbook included with SolverStudio and running any example.

2.4.2 Menu Items

View last data Item

This will open the last COOPR data file created by SolverStudio. This is useful for checking that SolverStudio has created the data in the format you expected.

Choose Solver

SolverStudio now includes a directory (SolverStudio/Solvers) containing some ready-to-run solvers. You can use these to solve your Coopr models, or any other solver (such as Gurobi) installed on your machine. Simply select which solver you would like to use using Solver item in the COOPR menu, if it appears there, and a tick will appear beside that Solver. (This menu only lists solvers in the SolverStudio/Solvers directory; see PyomoOptions next for using other solvers installed on your machine.) If you do not select a solver, Coopr will default to the COIN-OR solver CBC. To add solvers to this list, copy any ready-to-run executable into the Solver directory. They will be added to the list and available to use next time Excel starts up.

2.4.3 PyomoOptions

An optional way to pass additional commands to *'pyomo.exe'* is to create a named range called **PyomoOptions** (be careful with spelling). This should be a 2-column table, whose values are the command line options (the - and -- are optional. Case is ignored) and their values (if any), such as:

solver	Gurobi
summary	

For options such as '--summary', the left hand 'option' cell could contain 'summary' or 's' (or indeed '--summary' or '–s'), while the right hand 'value' cell is left blank.

This is (currently) the only way to select solvers such as Gurobi and CPLEX that are installed on your machine, but not present in the SolverStudio/Solvers directory.

Note that if you have defined a PyomoOption 'solver' on the spreadsheet, then this will specify the solver used when running Pyomo. Changes to the solver made by the user using the COOPR... Solver... menu option will be reflected in the appropriate PyomoOption cell. If no PyomoOption data item has been defined on the sheet with a 'solver' entry, then the choice of solver is stored internally within the spreadsheet by SolverStudio.

References:

http://solverstudio.org/languages/cooprpyomo/
https://software.sandia.gov/trac/coopr
https://software.sandia.gov/trac/coopr/wiki/Documentation

2.5 GUROBI

Gurobi is a commercial product for solving linear and mixed integer models using python. You can access Gurobi in SolverStudio by selecting *'Gurobi (Python)'* in the language menu.

2.5.1 Installation

Gurobi is not included with SolverStudio. To install it, go to the Gurobi Downloads page and download the latest version. Run the executable.

Gurobi requires a license for each user on each machine. To obtain one, go to the Gurobi License page. Once you have obtained a license, you can install it by clicking *start* and then copy and pasting the license key (*grbgetkey xxxx)* into the search box. Alternatively, you can let SolverStudio manage you Gurobi licenses using the SolverStudio Gurobi License Handler (detailed below).

You can test your installation by opening the *'Gurobi Examples.xlsx'* workbook included with SolverStudio and running any of the examples.

2.5.2 Menu Items

View last data (Gurobi Input) file

This will open the last Gurobi data file created by SolverStudio. This is useful for checking that SolverStudio has created the data in the format you expected.

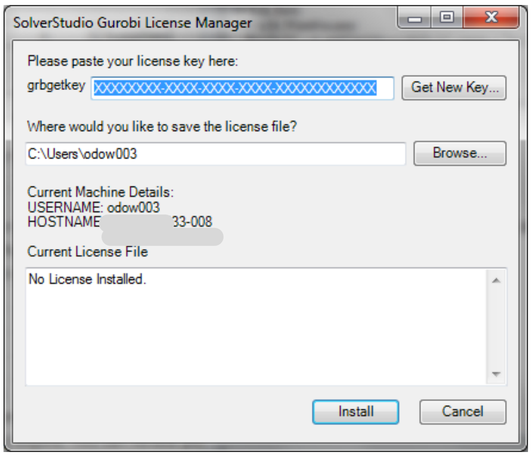
View last results (Gurobi Output) file

This will open the last Gurobi results file returned from Gurobi. This is useful for checking that Gurobi has solved your model and SolverStudio has read it back correctly.

Install or Renew Gurobi License

This will open the SolverStudio Gurobi License Manager. If you have not already obtained a key for the current machine, click *'Get New Key…'* which will take you to the Gurobi License page. Once you have a key, copy and paste it (minus the grbgetkey part) into the highlighted section. You can choose where to save the license file by clicking *'Browse…'* When ready, click *'Install'* to install the license.

Some license types (such as Academic) need to be renewed. If you have used the SolverStudio Gurobi License Manager to install the license originally, clicking install will renew it.


SolverStudio Gurobi License Manager

Please paste your license key here:

grbgetkey XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

Get New Key...

Where would you like to save the license file?

C:\Users\odow003

Browse...

Current Machine Details:
USERNAME: odow003
HOSTNAME [redacted] 33-008

Current License File

No License Installed.

Install

Cancel

Figure 2 - SolverStudio Gurobi License Manager

http://solverstudio.org/languages/gurobi/
http://www.gurobi.com/
http://www.gurobi.com/resources/documentation

2.6 SimPy

SimPy (*'SIMulation in Python'*) is a simulation language for Python that is included in SolverStudio by default and runs under the IronPython compiler. You can access SimPy in SolverStudio by selecting *'PuLP (IronPython)'* from the language menu.

As SolverStudio includes SimPy by default, no installation is necessary. Try running the examples contained in the *'SimPyExamples.xlsx'* workbook included with SolverStudio.

NOTE: Not all modules of SimPy are included with SolverStudio (notably SimPlot). You could try installing them yourself, or simply use Excels plotting capabilities. The first example (*'SimPy-Queuing'*) shows how you can access a range of Excel functions from your Python model.

References:

http://simpy.sourceforge.net/
http://simpy.sourceforge.net/SimPy_Manual/index.html

2.7 PuLP

PuLP is an LP modeller written in Python by Stuart Mitchell. It is included in SolverStudio by default and runs under the IronPython compiler. You can access PuLP in SolverStudio by selecting *'PuLP (IronPython)'* from the language menu.

As SolverStudio includes PuLP by default, no installation is necessary. Try running the examples contained in the *'PuLP Examples.xlsx'* workbook included with SolverStudio.

References:

http://solverstudio.org/languages/pulp/
http://code.google.com/p/pulp-or/
https://pulp-or.googlecode.com/files/pulp1.4.7.pdf

2.8 GMPL

GMPL (GNU Math Programming Language) is an open source AMPL look-alike developed as part of GLPK (GNU Linear Programming Kit). You can access GMPL in SolverStudio by selecting *'GMPL'* from the language menu.

As SolverStudio includes GMPL by default, no installation is necessary. Try running the GMPL example contained in the *'AMPL(NEOS) + GMPL Examples.xlsx'* workbook included with SolverStudio.

References:

http://en.wikibooks.org/wiki/GLPK/GMPL_%28MathProg%29

2.9 CPython

CPython is no longer included with the SolverStudio download, but instead the user can download it here or here. See also:

http://python.org/
http://docs.python.org/2/
http://www.activestate.com/activepython

If you already have Python installed, SolverStudio will use it. This allows the user to access any Python modules they have installed themselves.

You can access CPython in SolverStudio by selecting *'Python (external)'* from the language menu.

Note: SolverStudio still has the ability to run a copy of CPython in the SolverStudioPath/SolverStudio/CPython directory (where SolverStudioPath is the path to your SolverStudio installation). If you are unable to install Python, then this may be useful. Note that SolverStudio will look for SolverStudioPath/SolverStudio/CPython/python.exe.

3.0 Advanced Usage

3.1 Language Paths

By default, SolverStudio will attempt to find the relevant support files for each language (eg AMPL) by searching for the appropriate EXE file (eg AMPL.exe) in all the directories in your system PATH variable. However, you can manually change the directory SolverStudio uses for each supported languae by setting the corresponding Environment Variables. (These can be set by right-clicking on My Computer and choosing Properties.. Advanced System Settings... Environment Variables, or using a command line command such as 'set SolverStudio_AMPL_PATH "*C:\Program Files\AMPL\ampl.exe"*.)

SolverStudio_AMPL_PATH - The full path to the AMPL executable '*ampl.exe'*
e.g.. 'C:\Program Files\AMPL\ampl.exe'

SolverStudio_GAMS_PATH - The full path to the GAMS executable *'gams.exe'*
e.g. 'C:\Program Files\GAMS\gams.exe'

SolverStudio_GUROBI_PATH - The full path to the Gurobi batch file *'gurobi.bat'*
e.g. 'C:\Program Files\Gurobi\bin\gurobi.bat'

SolverStudio_COOPR_PATH - The full path to the COOPR executable *'pyomo.exe'*
e.g. 'C:\Python27\Scripts\pyomo.exe'

SolverStudio_GLPSOL_PATH - The full path to the GMPL executable *'glpsol.exe'*
e.g. 'C:\Program Files\GLPK\glpsol.exe'

SolverStudio_CPYTHON_PATH - The full path to a CPython executable *'python.exe'*
e.g. 'C:\Python27\python.exe'

Note: The paths above should be set as shown, but will also work if the executable name is dropped, and or if then the trailing \ is also dropped. So, for example, '*C:\Python27*' will tell SolverStudio to run Python by finding 'python.exe' (the default executable name) in the Python27 directory.

Note: Environment variable names are case insensitive

When SolverStudio runs a model, it does so in a new environment with a modified PATH environment variable. (Your main PATH is not modified, just the PATH used when running the model.) Typically, for each language, the directory containing the EXE file (eg AMPL.exe) is added to the PATH, and then the directory containing the Solvers (eg path\to\solverstudio\Solvers\32bit (or \64bit on a 64bit sytem). This ensures, for example, that AMPL runs any copy of lpsolve provided by AMPL, and not another copy provided by SolverStudio. This is important as AMPL will often require a 'stub' program, such as lpsolve.exe or Gurobi.exe which they provide; this stub then calls the underlying solver which may be in a DLL for example.

3.2 Advanced GAMS GDX support

The following commands are able to be used in SolverStudio via the "Pulp (IronPython)" language (the default way of accessing IronPython directly in SolverStudio). They use the GAMS 23.7 C# API.

```
//This opens a GDX for reading or writing
int PGX = SolverStudio.OpenGDX(string filepath, bool read = true, bool append = false)
       filepath: the full path to the GDX being opened
       read: true = reading, false = writing
       append:  true = open an existing GDX to append new data items

//This closes an open GDX
void SolverStudio.CloseGDX(int PGX)
       PGX: integer corresponding to a GDX file

//This imports the contents of a GDX into a new sheet in the workbook
void SolverStudio.ImportGDX(int PGX, string filepath, bool VarOnly,bool Group)
       PGX: integer corresponding to a GDX file
       filepath: the full path to the GDX being opened
       VarOnly: import the variables only
       Group: group the rows on the imported sheet

//Reads the contents of a symbol into an existing data item
void   SolverStudio.ReadSymbolFromGDX(int   PGX,   string   name,   int   level,   ref
ManagedDataItems DataItems)
       PGX: integer corresponding to a GDX file
       name: name of the symbol
       level: index level of the value (some levels will not be available)
              0 = value
              1 = marginal
              2 = Lower bound
              3 = Upper bound
       DataItems: Special type used by SolverStudio to store data.
              use SolverStudio.DataItems
```

Here is an example that could be run under the Pulp (IronPython) Language.

```
#-------------------------------------------------------------------------------------#
# This example demonstrates the various GDX methods exposed by SolverStudio          #
# To run, first solve the Transportation Example in 'GAMS(NEOS) Examples.xlsx'       #
# Change the language to PuLP (IronPython) and copy and paste this model into        #
# the model editor. Click 'Solve Model.'                                              #
#-------------------------------------------------------------------------------------#
#
# Path to the GDX file
path = SolverStudio.WorkingDirectory() + "\\results.gdx"
# Name of existing data item
syName = "flow"
#
```

```
# Open the GDX for reading
PGX = SolverStudio.OpenGDX(path)
# Read in the marginal of 'flow' to its existing data item on the current sheet
SolverStudio.ReadSymbolFromGDX(PGX, syName, 1, SolverStudio.DataItems)
# Import the entire contents to the GDX to a new sheet
SolverStudio.ImportGDX(PGX, path, False, False)
# Close the GDX
SolverStudio.CloseGDX(PGX)
#-------------------------------------------------------------------------------------#
```

You may also find the GAMS page on APIs and Tools useful.