GNU Linear Programming Kit for C# and the Common Language Interface

Reference Manual

Version 1.10.0

March 16, 2018

Contents

Chapter 1

Introduction

The GNU Linear Programming Kit (GLPK)[2] package supplies a solver for large scale linear programming (LP) and mixed integer programming (MIP). The GLPK project is hosted at http://www.gnu.org/software/glpk.

It has two mailing lists:

- help-glpk@gnu.org and
- bug-glpk@gnu.org.

To subscribe to one of these lists, please, send an empty mail with a Subject: header line of just "subscribe" to the list.

GLPK provides a library written in C and a standalone solver.

The source code provided at ftp://gnu.ftp.org/gnu/glpk/ contains the documentation of the library in file doc/glpk.pdf.

Project GLPK for C#/CLI delivers a Common Language Interface binding for GLPK. It is hosted at http://glpk-cli.sourceforge.net/.

To report problems and suggestions concerning GLPK for C#/CLI, please, send an email to the author at xypron.glpk@gmx.de.

Chapter 2

Getting started

This chapter will run you through the installation of GLPK for C#/CLI and the execution of a trivial example.

2.1 Installation

2.1.1 Windows

The following description assumes:

- You are using a 64-bit version of Windows. Replace folder name w64 by w32 if you are using a 32-bit version.
- The current version of GLPK is 4.65. Please, adjust paths if necessary.
- Your path for program files is "C:\Program Files". Please, adjust paths if necessary.

Download the current version of GLPK for Windows from https://sourceforge.net/projects/winglpk/.

The filename for version 4.65 is winglpk-4.65.zip. Unzip the file. Copy folder glpk-4.65 to "C:\Program Files\GLPK\".

To check the installation run the following command:

```
"C:\Program␣Files\GLPK\w64\glpsol.exe" --version
```

To use GLPK for C#/CLI you need a .NET environment to be installed.

2.1.2 Linux

Download the current version of GLPK source with

```
wget ftp://ftp.gnu.org/gnu/glpk/glpk-4.65.tar.gz
```

Unzip the archive with:

```
tar -xzf glpk-4.65.tar.gz
cd glpk-4.65
```

Configure with

```
./configure
```

Make and install with:

```
make
make check
sudo make install
sudo ldconfig
```

Check the installation with

```
glpsol --version
```

For the next steps you will need the Mono C# compiler to be installed.

You can check the correct installation with the following commands:

```
mcs --version
```

To build GLPK for C#/CLI you will need package SWIG (Simplified Wrapper and Interface Generator, http://www.swig.org/). You can check the installation with the following command:

```
swig -version
```

Most Linux distribution contain a SWIG package. The installation command will depend on the distribution, e.g.

- Debian: sudo apt-get install swig
- Fedora: sudo yum install swig
- Gentoo: sudo emerge swig

Download GLPK for C#/CLI from https://sourceforge.net/projects/glpk-cli/.

Unzip the archive with:

```
tar -xzf glpk-cli-1.10.0.tar.gz
cd glpk-cli-1.10.0
```

Configure with:

```
./configure
```

If libglpk.so is in a special path you may specify this path using parameter LDFLAGS, e.g.

```
./configure LDFLAGS=-L/opt/lib
```

Make and install with:

```
make
make check
sudo make install
sudo ldconfig
```

2.2 Trivial example

In the example we will create a C# class which will write the GLPK version to the console.

With a text editor create a text file test.cs with the following content:

```
using System;
using org.gnu.glpk;

class Program
{
  static void Main (string[] args)
  {
    Console.WriteLine ("GLPK␣" + GLPK.glp_version ());
  }
}
```

2.2.1 Windows

Copy the GLPK for C#/CLI assembly (libglpk-cli.dll) to the directory with file test.cs.

For compiling you need the C# compiler. The location of csc.exe depends on the version of the .NET framework used. Possible paths include:

- C:\Windows\Microsoft.NET\Framework64\v4.0.30319\
- C:\Windows\Microsoft.NET\Framework64\v3.5\
- C:\Windows\Microsoft.NET\Framework64\v2.0.50727\

Compile the application:

```
csc.exe /r:libglpk-cli.dll test.cs
```

To compile a 32 bit application on 64 bit Windows use:

```
csc.exe /platform:x86 /r:libglpk-cli.dll test.cs
```

For running the application both the GLPK library (glpk_4_65.dll) and the GLPK for C#/CLI native library (libglpk_cli_native.dll) have to be in the search path. Set the environment variable PATH accordingly.

```
SET PATH=%PATH%;C:\Program Files\glpk\glpk-4.65\w64
```

Run the application

```
test.exe
```

The output will be the GLPK version number, for example: GLPK 4.65.

2.2.2 Linux

Compile the class

```
mcs -r:libglpk-cli -lib:/usr/local/lib/glpk-cli/ test.cs
```

Run the application:

```
export MONO_PATH=/usr/local/lib/glpk-cli
export LD_LIBRARY_PATH=/usr/local/lib/glpk-cli
./test.exe
```

The output will be the GLPK version number, for example: GLPK 4.65.

Chapter 3

Architecture

A GLPK for C#/CLI application will consist of

- the GLPK library
- the GLPK for C#/CLI library
- the GLPK for C#/CLI assembly
- the application code.

Chapter 4

Classes

GLPK for C#/CLI uses the Simplified Wrapper and Interface Generator (SWIG)[3] to create the Common Language Inteface binding for GLPK. Classes are created in namespace org.gnu.glpk.

Class GlpkCallback is called by the MIP solver callback routine.

Interface IGlpkCallbackListener can be implemented to register a listener for class GlpkCallback.

Class GlpkTerminal is called by the MIP solver terminal output routine.

Interface IGlpkTerminalListener can be implemented to register a listener for class GlpkTerminal.

Class GlpkException is thrown if an error occurs.

Class GLPK maps the functions and constants from glpk.h.

The following classes map structures from glpk.h:

- glp_arc
- glp_attr
- glp_bfcp
- glp_cpxcp
- glp_graph
- glp_iocp
- glp_iptcp
- glp_long
- glp_mpscp
- glp_prob
- glp_smcp
- glp_tran
- glp_tree
- glp_vertex

The following classes are used to map pointers:

- SWIGTYPE_p_double
- SWIGTYPE_p_f_p_q_const__char_v_____void
- SWIGTYPE_p_f_p_struct_glp_tree_p_void__void
- SWIGTYPE_p_f_p_void__void
- SWIGTYPE_p_f_p_void_p_q_const__char__int
- SWIGTYPE_p_glp_arc
- SWIGTYPE_p_glp_graph
- SWIGTYPE_p_glp_vertex
- SWIGTYPE_p_int
- SWIGTYPE_p_p_char
- SWIGTYPE_p_p_glp_vertex
- SWIGTYPE_p_size_t
- SWIGTYPE_p_va_list
- SWIGTYPE_p_void

The following clases are used for network problems:

- glp_cli_arc_data
- glp_cli_vertex_data

4.1 Exceptions

When illegal parameters are passed to a function of the GLPK native library an exception GlpkException is thrown. Due to the architecture of GLPK all GLPK objects are invalid when such an exception has occured.

4.1.1 Implementation details

GLPK for C#/CLI registers a function glp_cli_error_hook() to glp_error_hook() before calling an GLPK API function. If an error occurs function glp_free_env is called and a long jump is used to return to the calling environment. Then function glp_cli_throw() is called which throws GlpkException.

4.2 Network problems

For network problems additional data like capacity and cost of arcs or the inflow of vertics has to be specified. The GLPK library does not provide data structures. In GLPK for C#/CLI classes _glp_cli_arc_data and _glp_cli_vertex_data are provided.

When creating a graph the size of the structures for these classes has to be specified. In some routines the offsets to individual fields in the structures are needed. The following constants have been defined:

- GLP_CLI_A_CAP - offset of field cap in arc data
- GLP_CLI_A_COST - offset of field cost in arc data
- GLP_CLI_A_LOW - offset of field low in arc data
- GLP_CLI_A_RC - offset of field rc in arc data
- GLP_CLI_A_X - offset of field x in arc data
- GLP_CLI_A_SIZE - size of arc data
- GLP_CLI_V_CUT - offset of field cut in vertex data
- GLP_CLI_V_PI - offset of field pi in vertex data
- GLP_CLI_V_RHS - offset of field rhs in vertex data
- GLP_CLI_V_SET - offset of field set in vertex data
- GLP_CLI_V_SIZE - size of vertex data

For accessing vertices method GLPK.glp_cli_vertex_get can be used.

For accessing the data areas of arcs and vertices methods

- GLPK.glp_cli_arc_get_data,
- GLPK.glp_cli_vertex_data_get, and
- GLPK.glp_cli_vertex_get_data

can be used.

```
glp_arc arc;
glp_cli_arc_data adata;
glp_cli_vertex_data vdata;

glp_graph graph =
        GLPK.glp_create_graph(
        GLPKConstants.GLP_CLI_V_SIZE,
        GLPK.GLP_CLI_A_SIZE);
GLPK.glp_set_graph_name(graph,
        MinimumCostFlow.class.getName());

int ret = GLPK.glp_add_vertices(graph, 9);

GLPK.glp_set_vertex_name(graph, 1, "v1");
GLPK.glp_set_vertex_name(graph, 2, "v2");
GLPK.glp_set_vertex_name(graph, 3, "v3");
GLPK.glp_set_vertex_name(graph, 4, "v4");
GLPK.glp_set_vertex_name(graph, 5, "v5");
GLPK.glp_set_vertex_name(graph, 6, "v6");
GLPK.glp_set_vertex_name(graph, 7, "v7");
GLPK.glp_set_vertex_name(graph, 8, "v8");
GLPK.glp_set_vertex_name(graph, 9, "v9");
```

```
vdata = GLPK.glp_cli_vertex_data_get(graph, 1);
vdata.setRhs(20);
vdata = GLPK.glp_cli_vertex_data_get(graph, 9);
vdata.setRhs(-20);

arc = GLPK.glp_add_arc(graph, 1, 2);
adata = GLPK.glp_cli_arc_get_data(arc);
adata.setLow(0); adata.setCap(14); adata.setCost(0);

...

GLPK.glp_write_mincost(graph,
        GLPKConstants.GLP_CLI_V_RHS,
        GLPKConstants.GLP_CLI_A_LOW,
        GLPKConstants.GLP_CLI_A_CAP,
        GLPKConstants.GLP_CLI_A_COST,
        "mincost.dimacs");
GLPK.glp_delete_graph(graph);
```

4.3 Callbacks

The MIP solver provides a callback functionality. This is used to call method callback of class GlpkCallback. A CLI program can listen to the callbacks by instantiating a class implementing interface IGlpkCallbackListener and registering the object with method addListener() of class GlpkCallback. The listener can be deregistered with method removeListener(). The listener can use method GLPK.glp_ios_reason() to find out why it is called. For details see the GLPK library documentation.

4.4 Output listener

GLPK provides a hook for terminal output. A CLI program can listen to the callbacks by instantiating a class implementing interface IGlpkTerminalListener and registering the object with method addListener of class GlpkTerminal. The listener can be dregistered with method removeListener(). After a call to glp_free_env() the GlpkTerminal has to registered again by calling GLPK.glp_term_hook(null, null). glp_free_env() is called if an exception GlpkException occurs.

4.5 Aborting a GLPK library call

Method void GLPK.glp_cli_error(String message) can be used to abort any call to the GLPK library. An exception GlpkException will occur. As GLPK is not threadsafe the call must be placed in the same thread as the initial call that is to be aborted. The output method of a GlpkTerminalListener can be used for this purpose.

4.6 Debugging support

Method void GLPK.glp_cli_set_msg_lvl(int msg_lvl) can be used to enable extra output signaling when a GLPK library function is entered or left using value with GLPKConstants.GLP_CLI_MSG_LVL_ALL. The output is disabled by a call with value GLPKConstants.GLP_CLI_MSG_LVL_OFF.

4.7 Locales

Method void GLPK.glp_cli_set_numeric_locale(String locale) can be used to set the locale for numeric formatting. When importing model files the GLPK library expects to be using locale "C".

4.8 Threads

The GLPK library is not thread safe. Never two threads should be running that access the GLPK library at the same time. When a new thread accesses the library it should call GLPK.glp_free_env(). When using an GlpkTerminalListener it is necessary to register GlpkTerminal again by calling GLPK.glp_term_hook(null, null).

When writing a GUI application it is advisable to use a separate thread for the calls to GLPK. Otherwise the GUI cannot react to events during the call to the GLPK libary.

Chapter 5

License

GLPK for C#/CLI is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License[1] as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

GLPK for C#/CLI is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with GLPK for C#/CLI. If not, see http://www.gnu.org/licenses/.

Bibliography

[1] Free Software Foundation, Inc. *GNU General Public License*, 2007.

[2] Andrew Makhorin. *GNU Linear Programming Kit*. GNU Software Foundation, 2010.

[3] SWIG.org. *Simplified Wrapper and Interface Generator*, 2010.

Index