Using GMPL/GLPK for IE 1079/2079

May 15, 2009

1 Introduction

Many facility location models can be described and solved using linear programming. However, for any real problems, these models can quickly become too large for the free versions of LINDO and the Excel Solver (Frontline Systems) that are used in the introductory linear programming course. Therefore, this course will require use of the Gnu Linear Programming Toolkit/Gnu Math Programming Language (GLPK/GMPL). The package is freely downloadable for use under the GNU Public License (GPL). You will have to use both the GLPSOL solver as well as the C API.

The Gnu Linear Programming Toolkit (GLPK) is an open source linear programming solver. Therefore, it can be freely downloaded and used. In addition, the source code is available for inspection and use in accordance with its license. This later quality has led to a number of enhancements and it use in other programs. It also means that versions of GLPK have been made available on a number of computer platforms, including Windows, Mac OSX and Linux/Unix.

A tutorial for using the Gnu Math Programming Language (GMPL) can be found here:

http://www.engr.pitt.edu/hunsaker/2082/glpsol_tutorial.pdf

In addition, there is a tutorial at the IBM website on the GMPL.

- http://www.ibm.com/developerworks/linux/library/l-glpk1/
- http://www.ibm.com/developerworks/linux/library/l-glpk2/
- http://www.ibm.com/developerworks/linux/library/l-glpk3/

There are three ways of using GLPK/GMPL. The most basic method is through the glpsol program, which takes a GMPL model and data file and finds the solution. For more complex problems, you will need to use the Application Programming Interface (API). The second way is through the C API. For this you will need a C compiler (e.g. Visual C Express for Windows, Xcode for Mac, GCC for other platforms.) The third way is throught the glpk libarary in the R Project for data analysis. (Note that there are projects that incorporate GLPK in Java and other programming languages and other API, but they will require some programming skill.)

2 GLPK API

2.1 Windows

GLPK can be downloaded for Windows at the following:

- http://downloads.sourceforge.net/gnuwin32/glpk-4.34-doc.zip which includes

5 Gnu Linear Programming Toolkit Reference Manual

6 Gnu Linear Programming Toolkit Modeling Language MathProg

In addition, to build applications with Visual C for Windows, you need to download a linked library built with Visual C

- http://sourceforge.net/projects/winglpk/

The GLPK API can be used with the Visual Studio Express for Windows. Visual Studio C++ Express for windows can be freely obtained from the Microsoft website at http://www.microsoft.com/express/vc/

Some instruction on using GLPK with Visual C++ are here:
http://www.engr.pitt.edu/hunsaker/3051/visual_studio.pdf

In summary, to use GLPK, you need to specify the following (assuming you are using version 4.38):

1. Where include files are located "glpk.h"
2. Libraries that are needed - Go to the project and right-click to choose "Properties." Then under "Configuration Properties" choose Linker -¿ Input. In the "Additional Dependencies" line, enter glpk_4_38.lib"
3. Indicating where to find libraries - In the Winglpk directory, choose glpk-4.38/win32.
4. Add .dll files to the project directory - Copy the glpk-4.38/w32/glpk-4.38.dll to both the project directory as well as the directory that contains the executable.

A sample C file is at the end of this document

2.2 Mac OS X and Unix/Linux

Mac OS X and Unix/Linux development is the same and requires the `gcc` compiler and associated tools. To do programming development work on the Mac, you will need to install the Xcode programming tools. These should have been included on the DVD that came with the Mac, or you can go to the Xcode website at Apple.

http://developer.apple.com/tools/xcode/

You can see if you already have the GCC compiler from Apple by opening up a terminal window and entering `gcc`. If available, you will get a message letting you know that it expects parameters. `gcc:  no input files`

A sample makefile and C file are at the end of this document. Let me know if you are going this route to determine if you need other assistance from me.

To use the Make and GCC, save the GMPL model `transp.mod`, C program `glpktest.c` and Makefile at the end of this document to a working directory. Then, use the commands `make glpktest` and `./glpktest` (note the leading './') as follows:

```
prompt$ Make glpktest

gcc  glpktest.c -o glpktest -I/usr/include -lglpk

prompt$ ./glpktest

Reading model section from transp.mod...
Reading data section from transp.mod...
63 lines were read
Generating cost...
Generating supply...
Generating demand...
Model has been successfully generated
This problem has 6 rows, 6 columns, and 18 nonzeros.
The upper bound on the first column is 0.000000.
      0: obj =   0.000000000e+00  infeas =  9.000e+02 (0)
*     4: obj =   1.561500000e+02  infeas =  0.000e+00 (0)
*     5: obj =   1.536750000e+02  infeas =  0.000e+00 (0)
OPTIMAL SOLUTION FOUND
Integer optimization begins...
+     5: mip =     not found yet >=              -inf        (1; 0)
+     5: >>>>>   1.536750000e+02 >=   1.536750000e+02   0.0% (1; 0)
+     5: mip =   1.536750000e+02 >=     tree is empty   0.0% (0; 1)
INTEGER OPTIMAL SOLUTION FOUND
Found optimal solution!
Objective value is 153.675000.
Solution:
```

```
x[Seattle,New-York] 50.000000
x[Seattle,Chicago] 300.000000
x[Seattle,Topeka] 0.000000
x[San-Diego,New-York] 275.000000
x[San-Diego,Chicago] 0.000000
x[San-Diego,Topeka] 275.000000

It took 5 iterations to solve.
```

3 R - GLPK

The other way of using the GLPK API is to use the GLPK library of the R-Project. The R-Project (http://cran.r-project.org) is an environment for data analysis and computing. It is best known as used by statisticians, although it is used heavily by financial modelers and bioinformatics as well. It is easier to install and use then C and GLPK, but it requires more learning to get started since most of you have programmed in C before. Fortunately you don't need to to know very much of R for this class. If you go this route you need a few things.

- The R Program itself. http://cran.r-project.org and there is a link for a Mac OSX version
- The GLPK package - From R, enter "install.packages()" Then choose a server (PA). Then choose 'glpk'
- From inside R, Enter "help.start()" (I think there is also a menu item for this). Look for a link called 'Packages'. Click on this, and find 'glpk'. There is something called a 'Vignette' that will help you.
- Or you can go straight here for the tutorial: http://cran.r-project.org/web/packages/glpk/vignettes/glpk-intro.pdf. Note that this document assumes you know linear programming and some idea of using GLPK/GMPL already.

If you use R, you should work through a guide introducing you to R. I recommend this: http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf R was created for statistics and data analysis, so going through this will give you a fast review of statistics as well.

4 Examples

4.1 GMPL model

This is the Transportation model, similar to the model found in the GMPL documentation.

transp.mod

```
# A TRANSPORTATION PROBLEM
#
# This problem finds a least cost shipping schedule that meets
# requirements at markets and supplies at factories.
#
#  References:
#              Dantzig G B, "Linear Programming and Extensions."
#              Princeton University Press, Princeton, New Jersey, 1963,
#              Chapter 3-3.

set I;
/* canning plants */

set J;
/* markets */

param a{i in I};
/* capacity of plant i in cases */

param b{j in J};
/* demand at market j in cases */

param d{i in I, j in J};
/* distance in thousands of miles */

param f;
/* freight in dollars per case per thousand miles */

param c{i in I, j in J} := f * d[i,j] / 1000;
/* transport cost in thousands of dollars per case */

var x{i in I, j in J} >= 0;
/* shipment quantities in cases */

minimize cost: sum{i in I, j in J} c[i,j] * x[i,j];
/* total transportation costs in thousands of dollars */

s.t. supply{i in I}: sum{j in J} x[i,j] <= a[i];
/* observe supply limit at plant i */

s.t. demand{j in J}: sum{i in I} x[i,j] >= b[j];
/* satisfy demand at market j */

data;

set I := Seattle San-Diego;
set J := New-York Chicago Topeka;

param a := Seattle     350
           San-Diego   600;

param b := New-York    325
           Chicago     300
           Topeka      275;

param d :              New-York   Chicago   Topeka :=
           Seattle     2.5        1.7       1.8
           San-Diego   2.5        1.8       1.4  ;

param f := 90;

end;
```

4.2 C example

glpktest.cpp

```
/* glpktest.c : Defines the entry point for the console application.
Based on Brian Hunsaker http://www.engr.pitt.edu/hunsaker/3051/simple_glpk.c
Note that the 'lpx' prefix for many GLPK functions is being replaced by a
'glp' prefix in more recent versions of GLPK
*/

/* "stdafx.h" here for use by Visual C++ Express */
/* #include "stdafx.h" */

#include <stdio.h>
#include <stdlib.h>
#include "glpk.h"

int main(int argc, const char* argv[])
{
  /* Pointer to the lp object */
  LPX *lp;
  int ret;  /* return code */
  char *filename = "transp.mod";
  int nrows;  /* number of rows */
  int ncols;  /* number of columns */
  int nelem; /* number of nonzeros */
  double upper_bound; /* upper bound of first col */
```

```
 /* Read in an instance from an MPS file */
 /* lp = lpx_read_mps( filename );*/
 /* lp = lpx_read_freemps( filename ); */

 /* Read in an instance from a GMPL file */
 lp = lpx_read_model(filename, NULL, NULL);

 /* Check that we read it correctly */
 if (lp == NULL)
   {
     printf("Error reading file %s\n", filename);
     exit(1);
   }

 /* Display some information about the instance */
 nrows = lpx_get_num_rows(lp);
 ncols = lpx_get_num_cols(lp);
 nelem = lpx_get_num_nz(lp);
 printf("This problem has %d rows, %d columns, and %d nonzeros.\n",
nrows, ncols, nelem);

 upper_bound = lpx_get_col_ub(lp, 1);
 /* NOTE: GLPK starts indices at 1 instead of 0!! */
 printf("The upper bound on the first column is %lf.\n", upper_bound);
 /* All the information about the instance is available with similar
    routines */


 /* Before solving, indicate some parameters */
 lpx_set_int_parm(lp, LPX_K_ITLIM, 50);
 lpx_set_real_parm(lp, LPX_K_TOLBND, 0.001 );


 /* Solve the (relaxation of the) problem */
 ret = lpx_simplex(lp);
 ret = lpx_integer(lp);

 /* Check the solution */
 if ( ret == LPX_E_OK && lpx_get_status(lp) == LPX_OPT )
   {
     int n; /* number of columns */
     int i; /* iterator for loop */

     printf("Found optimal solution!\n");
     printf("Objective value is %lf.\n", lpx_get_obj_val(lp) );
```

```
      /* Examine solution */
      n = lpx_get_num_cols(lp);

      printf("Solution: \n");
      for (i = 0; i < n; i++){
        printf("%s ", lpx_get_col_name(lp, i+1) );
        printf("%lf ", lpx_get_col_prim(lp, i+1) );
        printf("\n");
      }
      printf("\n");

      printf("It took %d iterations to solve.\n",
            lpx_get_int_parm(lp, LPX_K_ITCNT) );
    }
  else
    {
      printf("Didn't find optimal solution.\n");

      /* Check other status functions.  What happened? */
      if (lpx_get_status(lp) == LPX_NOFEAS)
       printf("Problem is proven to be infeasible.\n");
      if (lpx_get_status(lp) == LPX_UNBND)
       printf("Problem is proven dual infeasible.\n");
      if (ret == LPX_E_ITLIM)
       printf("Reached iteration limit.\n");
    }

  return 0;
}
```

4.3 Makefile

Makefile

```
# Based on Oualline, (1997), Practical C Programming, 3rd Ed., Ch. 7
# Makefile for GLPK programming with gcc compiler
#

CC=g++
CFLAGES=-g -Wall

#
# Compiler flags:
# -g -- Enable debugging
# -Wall -- Turn on all warnings
# -D__USE_FIXED_PROTOTYPES__
# -- force the compiler to use the correct headers
# -ansi -- Don't use GNU extensions, Stick to ANSI C
#

# Note:  The -I/usr/include and -L/usr/lib are not strictly necessary,
# if the include files (.h) and the library files (libglpk) are in their
# standard locations.  If they are elsewhere, then replace the '/usr/include'
# and '/usr/lib' with the appropriate directories

glpktest: glpktest.cpp
$(CC) $(CFLAGS) glpktest.cpp -o glpktest -I/usr/include -L/usr/lib -lglpk
```