Setting up GLPK

Sam McCauley

November 7, 2021

We will be using the open-source GLPK solver to solve optimization problems in this class.

This solver does not achieve quite the performance of proprietary solvers like CPLEX, Gurobi, Google OR tools, etc. But, it's free and it's easier to set up and use.

GLPK is already installed on the lab machine and the below is just a reference if you want to set it up yourself. Note that I haven't thoroughly tested the below. Also, I'm unsure if Section 2 is going to be at all useful; honestly I'm including it mostly because I wrote it already.

Please let me know if you find anything wrong—or if there's anything useful you found that I should add!

1 Setting Up on Your Personal Machine

I'll assume you have administrator/root access for these instructions.

The easiest way to set GLPK up on your personal machine is to install it using available binaries. Note that I haven't tried all of the options below, but hopefully they should work.

Once you have set up the binary, proceed to Section 3 ("Using GLPK".)

1.1 Windows

Download and install "GLPK for Windows" from `http://winglpk.sourceforge.net/#quickguide`.

Note that for windows, you have to modify how you run the program a bit—you'll be using `glpsol.exe` rather than just `glpsol`. Everything else should hopefully stay the same. This is detailed in Section 3—you should be able to follow the instructions there; just something you keep in mind.

1.2 Mac OSX

If you have `brew` on your computer, the likely easiest way is to use

```
% brew install glpk
```

After that, you're done! This installs the program on your computer, allowing you to run GLPK from anywhere.

Alternatively, you can follow the instructions in Section 2; they will also work. If you follow these instructions you will need to run `glpsol` from your `lp/` directory.

1.3 Linux (with root access)

In short: use your package manager to install GLPK.

For example, on Ubuntu you'd use:

```
% sudo apt install glpk-utils
```

2 Setting Up on a Linux Machine Without Root Access

Create a directory to hold the files you're creating. I created a `lp/` directory in my home directory (the place you land when you first ssh into a lab machine). To make the directory, run:

```
% mkdir lp
```

and then cd into it:

```
% cd lp/
```

First, you need to download the necessary files to the machine. One easy way is to use wget:

```
% wget http://mirror.team-cymru.com/gnu/glpk/glpk-4.35.tar.gz
```

This will give you an archive (a tar file). You need to extract that file to get a folder:

```
% tar -xf glpk-4.35.tar.gz
```

Go into this new folder:

```
% cd glpk-4.35/
```

Unclear if this one step is necessary: Before we start installing the program we need to tell `gcc` where to find these new libraries.

```
% export CPATH=$HOME/lp/glpk-4.35/include:$CPATH
% export LD_LIBRARY_PATH=$HOME/lp/glpk-4.35/src/.libs:$LD_LIBRARY_PATH
% export LIBRARY_PATH=$LD_LIBRARY_PATH:$LIBRARY_PATH
```

The `configure` script looks around the machine and chooses some basic settings to help with compilation. Run this script:

```
% ./configure
```

This prints a lot of "checking..." messages; ignore these unless there is some kind of error.
Now you're ready to make:

```
% make
```

Again, this will output quite a lot of stuff; don't worry about it unless there's an error.
The program is all set to run, but it's in an annoying place. Cd back into the `lp/` directory:

```
% cd ../
```

Now, we want to create a symlink to make the program easier to run. A symlink is a file that just points to a file in another location.

```
% ln -s glpk-4.35/examples/glpsol glpsol
```

Now we're ready to use the program! Whenever you want to use it, be sure to navigate to your `lp/` directory.

3 Using GLPK

First, download `dietSol.lp` from the course website.

- If you installed GLPK using a symlink (i.e. if you followed the directions in Section 2), navigate to your `lp/` directory. Copy the test file into your `lp/` directory.
- Otherwise, navigate to the directory where you have the test file.

Now, you're ready to run GLPK! Use the following command. We'll be using the `--cpxlp` option to let GLPK know that we're using the CPLEX file format, and we'll be outputting the result to `test.out`.

GLPK is also going to output some information to your terminal (including some potentially-alarming information—for example, when following the Section 2 instructions it always tells me it is "Crashing!" for some reason). Don't worry about this so long as your `test.out` results are correct. Let me know if things seem off.

3.1 Unix (Linux and OSX)

The following should hopefully just work.

```
% glpsol --cpxlp dietSol.lp -o dietSol.out
```

3.2 Unix if Installed Without Root Access

The first attempt is the following:

```
% glpsol.sh --cpxlp dietSol.lp -o dietSol.out
```

If you created the symlink, you may need to specify that `glpsol.sh` can be found in the current directory:

```
% ./glpsol.sh --cpxlp dietSol.lp -o dietSol.out
```

Finally, if you're having problems or experiencing some weird errors, you may want to invoke bash directly:

```
% bash glpsol.sh --cpxlp dietSol.lp -o dietSol.out
```

3.3 Windows

The following should hopefully just work using the command line or powershell. If you're using WSL or git-bash, you should probably follow the unix instructions instead.

```
% glpsol.exe --cpxlp dietSol.lp -o dietSol.out
```

4 Expected Output

For me, the output to the command line looks like this:

```
GLPSOL--GLPK LP/MIP Solver 5.0
Parameter(s) specified in the command line:
 --cpxlp dietSol.lp -o dietSol.out
```

```
Reading problem data from 'dietSol.lp'...
2 rows, 3 columns, 5 non-zeros
12 lines were read
GLPK Simplex Optimizer 5.0
2 rows, 3 columns, 5 non-zeros
Preprocessing...
2 rows, 3 columns, 5 non-zeros
Scaling...
 A: min|aij| =  2.500e+00  max|aij| =  2.870e+01  ratio =  1.148e+01
GM: min|aij| =  5.677e-01  max|aij| =  1.761e+00  ratio =  3.103e+00
EQ: min|aij| =  3.223e-01  max|aij| =  1.000e+00  ratio =  3.103e+00
Constructing initial basis...
Size of triangular part is 2
      0: obj =   0.000000000e+00 inf =   6.743e+00 (2)
      2: obj =   5.742284526e+00 inf =   0.000e+00 (0)
*     3: obj =   4.314562212e+00 inf =   0.000e+00 (0)
OPTIMAL LP SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.0 Mb (32525 bytes)
Writing basic solution to 'dietSol.out'...
```

And the contents of `dietSol.out` are:

```
Problem:
Rows:       2
Columns:    3
Non-zeros:  5
Status:     OPTIMAL
Objective:  obj = 4.314562212 (MINimum)

   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 protein      NL             46            46                 0.0212298
     2 carbs        NL            130           130                 0.0256768

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 p            NL              0             0                   1.06227
     2 r            B         2.92166             0
     3 c            B         2.86636             0

Karush-Kuhn-Tucker optimality conditions:

KKT.PE: max.abs.err = 0.00e+00 on row 0
        max.rel.err = 0.00e+00 on row 0
        High quality

KKT.PB: max.abs.err = 0.00e+00 on row 0
        max.rel.err = 0.00e+00 on row 0
        High quality

KKT.DE: max.abs.err = 0.00e+00 on column 0
        max.rel.err = 0.00e+00 on column 0
```

```
        High quality

KKT.DB: max.abs.err = 0.00e+00 on row 0
        max.rel.err = 0.00e+00 on row 0
        High quality

End of output
```