CPS 173 - Computational Microeconomics

Programming Assignment 1: linear and mixed integer programming (due Feb. 8 before class)

Please read the rules for assignments on the course web page. Contact Nate (xiaoming@cs.duke.edu) and/or Vince (conitzer@cs.duke.edu) with any questions.

Getting set up. (Please ask questions if you have trouble doing this; don't worry about asking a question that appears dumb. Also, you are welcome to use a different setup if you know how; you can find the GNU linear programming kit on the Web. The point of this exercise is to get everyone comfortable solving linear and (mixed) integer programs.)

You will need to be able to connect to login.oit.duke.edu using SSH. One way of doing this is to download the program PuTTY (search the Web for "download putty"). Once you have installed this, run it. Under "Host Name" type login.oit.duke.edu, select SSH, (if you want to, type a name under saved sessions and press "Save",) and press "Open". Use your NetID and password to log in.

Useful commands to try out:

- `pwd`: tells you the directory that you are in
- `ls`: lists the contents of the directory that you are in
- `cd name`: changes to directory "name", if such a directory exists (cd .. brings you up one level)
- `mkdir name`: creates a directory called "name"
- `rmdir name`: removes directory "name"
- `cp name1 name2`: makes a copy of file "name1" and calls it "name2"
- `rm name`: deletes (removes) file "name"
- `wget url`: gets the file at the given url into the current directory. E.g., `wget http://www.cs.duke.edu/courses/spring12/cps173/painting.lp`
- `logout`: logs you out

Also, pressing the up arrow will take you back to commands you entered earlier. Pressing Tab will try to complete the command you are typing for you.

You probably want to make a directory for the course, e.g.

```
mkdir cps173
cd cps173
```

Now you are in the directory cps173 (use `pwd` to check this). You can make a subdirectory for this homework:

```
mkdir homework1
cd homework1
```

To get all the files relevant to this homework into this directory:

```
wget http://www.cs.duke.edu/courses/spring12/cps173/homework1.tar
tar -xvf homework1.tar
```

(A .tar file is an archive file that contains multiple files, and `tar -xvf` extracts the files.) Use `ls` to check that you have all the files.

To be able to run the GLPK solver, you need to enter the following commands (only the first time):

```
cd ~
wget http://www.cs.duke.edu/courses/spring12/cps173/glpk.conf
cp .cshrc cshrcbackup
cat glpk.conf >> .cshrc
```

After completing the above commands, logout and login again. After this, you will be able to run the GLPK solver.

The command to run the GLPK solver is:

`glpsol`

If you want to solve an LP/MIP expressed in the standard (CPLEX) LP format, type

`glpsol --cpxlp`

If you want to solve an LP/MIP expressed using the modeling language, type

`glpsol --math`

You will also need to specify the file that you want to solve, e.g.

`glpsol --math cell.mod`

and you will also need to specify a name for a file in which the output will be stored, preceded by -o. So, typing

`glpsol --math cell.mod -o cell.mod.out`

will instruct the solver to solve the LP/MIP cell.mod, and put the solution in a new file called cell.mod.out.

You will need an editor to read and edit files. One such editor is emacs. Going to the right directory and typing

`emacs cell.mod.out`
will allow you to read the output file.

Inside emacs there are all sorts of commands. You can find emacs commands on the Web, but a few useful ones are:

- Ctrl-x Ctrl-c: exit emacs
- Ctrl-x Ctrl-s: save the file you are editing
- Ctrl-s: search the file for a string (string=sequence of characters)
- Ctrl-r: search the file backwards
- Ctrl-g: if you accidentally typed part of some emacs command and you want to get back to editing, type this
- ESC-%: allows you to replace one string with another throughout the file; for each occurrence it will check with you, press spacebar to confirm the change, n to cancel it
- Ctrl-k: delete a whole line of text
- Ctrl-Shift-_: undo

Try playing around with all of this. In particular, check that you can solve all the example files, and read the solutions. Then, solve the following questions (which refer to the examples from class).

1. (10 points.) Knapsack. Modify[1] the knapsack.lp file (*not* knapsack.mod) so that object B has weight 4.25kg, volume 3.5 liters, sells for $3.75, and has 4.25 units available. (Do not add integrality constraints, that is, allow the solution to be fractional.) Solve it using `glpsol --cpxlp` and put the output in knapsack.out. Check that the solution makes sense.

2. (30 points.) Hot dogs. Modify the hotdogs.mod file to solve the following instance of the hot-dog problem: you now have 3 hot-dog stands (call the third one s3). The customers are now as follows:

- location: 2, #customers: 5, willing to walk: 2
- location: 5, #customers: 4, willing to walk: 1
- location: 6, #customers: 3, willing to walk: 1
- location: 8, #customers: 5, willing to walk: 1
- location: 9, #customers: 2, willing to walk: 1

[1] If you want to keep the original knapsack.lp file, you can do `cp knapsack.lp knapsack.original.lp` first to create a backup.

- location: 11, #customers: 3, willing to walk: 6
- location: 13, #customers: 4, willing to walk: 1
- location: 15, #customers: 6, willing to walk: 7

Solve using `glpsol --math` and put the output in hotdog.out. Check that the solution makes sense.

3. (60 points.) Choosing courses.

Bob is a student at the University of Interdisciplinarity. At this university, students must obtain 10 points in the natural sciences, 10 points in the social sciences, and 10 points in the humanities, to satisfy their general education requirements. Rather cynically, Bob is only interested in minimizing the amount of effort that he has to put in to satisfy these requirements. (Let's at least say it's because Bob is passionate about a particular specialization and really wants to focus his efforts there instead...)

There are four courses that can give Bob these points.

1. "Introduction to neuroscience and its implications for social behavior" gives 8 natural science points, 6 social science points, and 2 humanities points. It requires 5 units of effort.
2. "The history of the popular perception of statistical facts" gives 2 natural science points, 6 social science points, and 8 humanities points. It requires 5 units of effort.
3. "The use of biophysics in sports" gives 4 natural science points, 2 social science points, and 3 humanities points. It requires 2 units of effort.
4. "A brief introduction to global warming" gives 5 natural science points, 2 social science points, and 1 humanities point. It requires 2 units of effort.

Which courses should Bob take? What if courses can be taken partially, meaning that you take, say, half of the course, spend half of the effort, and get half of the points in each category? (Of courses, fractions other than 1/2 are also allowed, but the fraction has to be between 0 and 1.)

Write an entirely new file courses.mod (just type `emacs courses.mod`), in which you use the modeling language to solve Bob's problem instances (both without and with the option of taking courses partially). As always with the modeling language, you should write the file so that it is possible to modify only the `data` part of the file to solve similar problem instances. Your file should start with the lines:

```
set REQUIREMENTS;
set COURSES;

param points_required{i in REQUIREMENTS};
param points_contributed{i in REQUIREMENTS, j in COURSES};
param effort{j in COURSES};
```

The rest is up to you to fill in. Solve using `glpsol --math` and put the output in courses_integral.out and courses_fractional.out. Check that your solutions make sense.

Instructions for turning in your homework: Send the files that you have created to xiaoming@cs.duke.edu (this time, you are expected to send seven files: knapsack.lp, knapsack.out, hotdogs.mod, hotdogs.out, courses.mod (leave it in the state where it computes fractional solutions), courses_integral.out, and courses_fractional.out). You can type `pine` for a simple e-mail program. You are welcome to send Nate and/or Vince e-mail with questions.