CPS 590.1 - Linear and integer programming

Homework 1: modeling and interpreting problems as linear and integer programs (due Sep. 26 before class)

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

In this assignment, you are asked to study four different computational problems. You will formulate them as linear or (mixed) integer programs, code these formulations up in the modeling language, and study and interpret their duals. Major hint: for the interpretation, it will be very helpful to relate these problems to problems that you have already seen (this will probably allow you to give much shorter answers). Your interpretation of the dual should be written in clear, concise, and logical English. I will not give full credit for unclear writing. To make the duals easy to interpret, you may wish to play around with your formulation a little bit (and try to make it as simple as possible); if your formulation for this part is a little different from the one you coded up, I will not subtract points (as long as they are both correct). Try to keep all your formulations simple, though. Please turn in your writeup as well as printouts of your code (`.mod` files) and output files.

0. Installing the GNU linear programming kit. (0 points.)

You can find the GNU linear programming kit on the Web. You can install it in whatever way you like. If you have trouble, below are instructions that I have used in past undergraduate classes. If you still have trouble, please let me know. After you have successfully installed everything you are highly encouraged to check out the "examples" directory for examples of how to use the modeling language. (While the GNU linear programming kit is nice, once you get to doing your project, you may find that the solver is not powerful enough. More powerful commercial solvers are available if you need them.)

Below are the (slightly modified) instructions from earlier courses.

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.
- `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 cps590.1
cd cps590.1
```

Now you are in the directory cps590.1 (use `pwd` to check this). You can make a subdirectory for this homework:

```
mkdir homework1
cd homework1
```

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/fall12/cps296.1/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:

`/afs/acpub/project/cps/bin/glpsol`

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

`/afs/acpub/project/cps/bin/glpsol --cpxlp`

If you want to solve an LP/MIP expressed using the modeling language, type
`/afs/acpub/project/cps/bin/glpsol --math`
You will also need to specify the file that you want to solve, e.g.
`/afs/acpub/project/cps/bin/glpsol --math problem.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
`/afs/acpub/project/cps/bin/glpsol --math problem.mod -o problem.out`
will instruct the solver to solve the LP/MIP problem.mod, and put the solution in a new file called problem.out.
You will need an editor to read and edit files. One such editor is emacs. For example, typing
`emacs problem.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

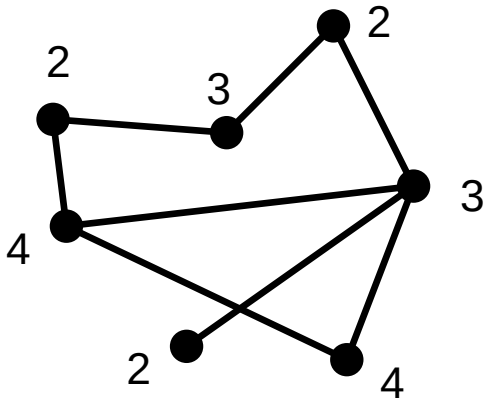

Figure 1: An instance of the maximum weighted independent set problem.

1. Maximum weighted independent set. (25 points.) In the maximum weighted independent set problem, we are given a graph with weights on the vertices. A subset of the vertices is an independent set if no pair of vertices in the subset is connected by an edge. That is, for each edge, one of its endpoints is outside the subset. The weight of a subset of vertices is the sum of the weights of the vertices in the subset. The goal is to find an independent set of maximum weight.

For example, consider the graph in Figure 1. The optimal solution for this graph is to take the upper left, upper right, lower left, and lower right vertices, for a total value of 10.

a. Give an integer program formulation of the maximum independent set problem.

b. Express this integer program in the modeling language and use the GNU solver on this formulation to solve the example instance above.

c. Take the linear program relaxation of the integer program. What problem does this LP relaxation represent? Take the dual.

d. Interpret the dual, and describe the meaning of weak and strong duality and complementary slackness for this problem.

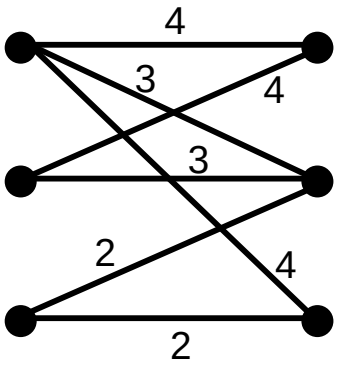

Figure 2: An instance of the maximum weighted bipartite matching problem.

2. Maximum weighted bipartite matching. (25 points.) In this problem, we are given a bipartite graph with weights on the edges. A subset of edges is a feasible matching if no two of the edges in the subset share a vertex. The weight of a matching is the sum of the edges' weights. The goal is to find the feasible matching with maximum weight.

For example, consider the bipartite graph in Figure 2. The optimal solution for this graph is to take the upper-left-to-lower-right, middle-left-to-upper-right, and lower-left-to-middle-right edges, for a total value of 10.

a. Give an integer program formulation of the maximum weighted bipartite matching problem.

b. Express this integer program in the modeling language and use the GNU solver on this formulation to solve the example instance above.

c. Take the LP relaxation of the integer program. Take the dual.

d. Interpret the dual, and describe the meaning of weak and strong duality and complementary slackness for this problem.

3. Dominance. (25 points.) We studied the notion of a minimax strategy in a zero-sum game. A more elementary game-theoretic notion is that of *dominance*. One strategy σ for (say) the column player is said to (strictly) dominate another strategy σ' for the column player if σ gives the column player a strictly higher expected utility than σ', for *every* row. Say that one strategy dominates another strategy by ϵ if the difference in the utilities is always at least ϵ.

For example, consider the following game. Without loss of generality for this purpose, suppose that this is a zero-sum game and the utilities in the matrix are those of the row player (and the column player wishes to minimize these utilities).

9	1	6	**7**
4	9	8	**9**
5	7	4	**8**

We will be interested in dominating the last column of the game (in this case, the fourth column). (This column is in bold font because of its special role in this problem.) In fact, the third column dominates the fourth column—for every row, the row player's utility against the third column is at least 1 lower than the row player's utility against the fourth column: $6 < 7$, $8 < 9$, $4 < 8$. Hence, the third column dominates the fourth column by 1. We can do even better by considering a mixed strategy for the column player: if the column player puts probability $1/2$ on each of the first two columns, then the row player's expected utility is always at least 2 lower against this mixed column strategy than against the fourth column: $(9 + 1)/2 = 5 < 7$, $(4 + 9)/2 = 6.5 < 9$, $(5 + 7)/2 = 6 < 8$. So this mixed column strategy dominates the fourth column by 2.

Our goal is to find a mixed column strategy that dominates the last column by as much as possible.

a. Give a linear program formulation for this dominance problem.

b. Express this linear program in the modeling language and use the GNU solver on this formulation to solve the example dominance problem instance above.

c. Take the dual.

d. Interpret the dual, and describe the meaning of weak and strong duality and complementary slackness for this problem. (Hint: you may wish to normalize the last column.)

4. Combinatorial betting mechanisms. (25 points.) In a *betting mechanism*, people take bets on future events. Many such mechanisms (for example, `intrade.com`) function by letting people buy or sell securities in a particular event. For example, you may be able to buy a security that pays off 100 if basketball team A defeats basketball team B in their next game; let us suppose that this security can currently be bought for 70. If you buy this security, then if A wins, you make a profit of 30, but if B wins, you lose 70.

We will consider a setup in which bettors propose bets to a party that we will call the auctioneer. A bet from a bettor might be "If A wins you have to pay me 30, but if B wins I will pay you 70." The auctioneer is only interested in accepting combinations of bets that result in a guaranteed profit for her. Hence, the auctioneer will not accept the above bet by itself. However, suppose she also has the following bet proposed to her: "If A wins I will pay you 50, but if B wins you have to pay me 60." Now, if the auctioneer accepts both of these bets, then no matter which of A and B wins, the auctioneer will have a profit of at least 10, that is, a guaranteed profit of 10. The auctioneer's goal is to accept a subset of the bets that maximizes her guaranteed profit.

More generally, we suppose that there is some set of states S that can materialize. A bet then consists of $|S|$ numbers, negative or nonnegative, that indicate how much the bettor will pay/receive in each state. For example, in a three-state world, a bet $(3, 3, -8)$ indicates that the bettor will pay 3, unless the third state materializes, in which case he expects to be paid 8.

For example, consider the following three-state instance with four bets:

- $(-2, 6, -3)$
- $(-5, -2, 8)$
- $(9, -2, -2)$
- $(-3, 5, -2)$

The optimal solution for the auctioneer here is to accept the first three bids. The sum of these vectors is $(2, 2, 3)$, indicating that the auctioneer has a guaranteed profit of 2.

a. Give an integer program formulation of the auctioneer's problem of maximizing her guaranteed profit.

b. Express this integer program in the modeling language and use the GNU solver on this formulation to solve the example instance above.

c. Take the linear program relaxation of the integer program. What problem does this LP relaxation represent? Take the dual.

d. Interpret the dual, and describe the meaning of weak and strong duality and complementary slackness for this problem. (Hint: one of the main points of betting mechanisms is to figure out how likely people think various outcomes are...)