Recitation 1

In this recitation, we will review how to download AMPL and to use AMPL to solve simple linear and integer programs.

You can work in pairs or individually. Answer the questions in the blank space provided, and turn this handout in to your section TA. If you work with a partner, you and your partner need only turn in one copy.

Name and NetID: Section:

1 Downloading and setting up AMPL

1. Go to `http://www.ampl.com/DOWNLOADS/details.html#Tables`
2. Download the `amplcml.zip` file to your Desktop.
3. Unzip the file: Double-click `amplcml.zip` and click `Extract all files`. Following the instructions in the Extraction Wizard, extract the files to a folder called `amplcml` on your Desktop. Then, delete the zip file. [Note: it is not necessary to do this on your Desktop; they can be saved elsewhere on your computer.]
4. To begin using AMPL: double click the application `sw` in the `amplcml` folder..
5. You will see a command window with a blank line preceded with the prompt `sw:`, where you will enter your commands. Type `ampl` and press enter. You are now ready to run AMPL

2 Solving an LP using AMPL

1. AMPL describes linear programs using *model* files: files that have the suffix `.mod`, which you can open and edit using Notepad .

   The model file is a general model, which might not include actual numbers or "data". The data for the AMPL model can be included in a separate file, a *data* file, which are files with the suffix `.dat`, which you can open and edit using Notepad as well.

   The `amplcml` package you downloaded above comes with some example model files, stored under the subfolder `MODELS`. Open this subfolder and find the file `net1.mod`. By right-clicking, open this file using Notepad. Try to understand what this model is doing.

   In the space provided below, write the analogous linear program in standard mathematical notation. Clearly specify what the decision variables, the objective function, and the constraints are.

2. Now, open the corresponding *data* file `net1.dat` using Notepad. Notice that the model file does not contain the actual numbers, but the data file supply this concrete information. Convert this input to a graph that you should draw below.

3. Try to figure out the optimal solution by hand. Do not use the simplex method, but try to guess a solution and try to reason that it is optimal. Draw the solution (flow values on each edge) on your graph above. (This exercise is meant to build up your intuition, which is important. Do not worry if your solution here does not turn out to be the "correct" one.)

4. Now, let us use AMPL to solve this input. Make sure that you have opened the `sw` application in the `amplcml` folder and that you have invoked AMPL by typing `ampl` and pressing enter at the `sw:` prompt.

   Read your model and your data files into AMPL, then solve the model. To tell AMPL what your model file is, type `model <filename>;` and press enter. To tell AMPL what your data file is, type `data <filename>;` and press enter. To solve, type `solve;` and enter. You should now see the following:

```
ampl: model MODELS/net1.mod;
ampl: data MODELS/net1.dat;
ampl: solve;
MINOS 5.5: optimal solution found.
3 iterations, objective ____
```

`ampl:`

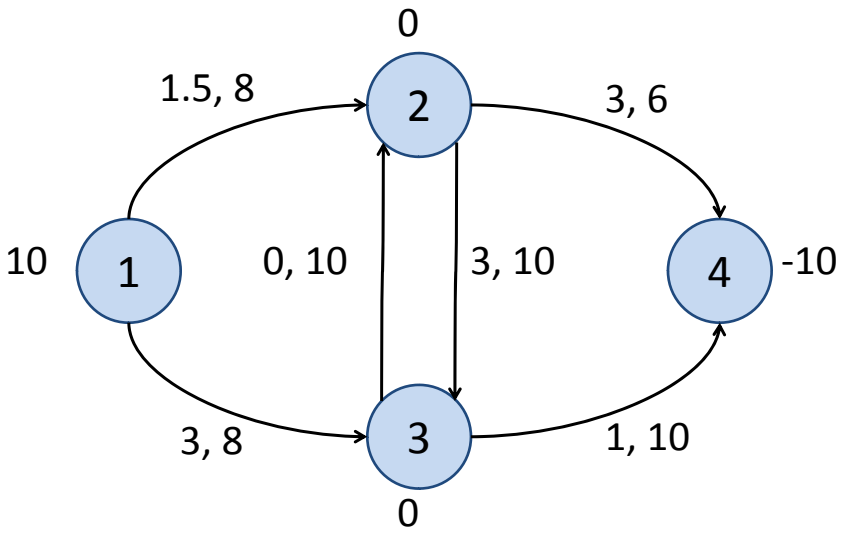
Note:

- You can only have one model file
- You must end each AMPL command with a semicolon ";"
- If your file is located in a subfolder or in a different folder, make sure you specify this location accordingly.

Use the "display" command to display your optimal solution. Write down the optimal solution and the corresponding objective value in the space below. Did you get the right solution before?

3 Writing your own AMPL model

In this section, we will review the structure of the AMPL model and data files. We will do this by writing an AMPL model for the following simple minimum-cost network flow problem. The number next to each node is the supply value of the node. The first number next to each edge is the edge cost while the second number is the edge capacity.



1. First write linear program for this problem in standard mathematical notation. Clearly specify what the decision variables, the objective function, and the constraints are.

   In particular, for each of the four nodes, write the flow-conservation constraint explicitly.

2. The remainder of this section will guide you step-by-step how to model this problem in AMPL. If you feel comfortable with AMPL, feel free to work on this on your own and skip ahead to page 8 to write down your solution. Then, proceed to page 9.

   Note that although this model will be similar to `net1.mod`, we advise that you to start from an empty file, typing everything by hand.

   Next, based on the above linear program, we will write the model file, which describe a general minimum-cost network flow problem. (So, this model file should not specifically mention the four nodes and six edges in the above graph.)

   After opening Notepad, save your document as `Rec1.mod`. Make sure to specify "Save as type" as "All files".

   The following are the basic components of an AMPL model file. (It is best, for now, that you write the components your AMPL model file in this order to avoid potential errors.)

   (a) Set declarations
   (b) Parameter declarations
   (c) Input checks
   (d) Variable declarations
   (e) Objective function
   (f) Constraints

   We will elaborate each of the components below:

   Set declarations Here, you tell AMPL what sets you will be using. In our problem, we deal with the set of nodes and the set of edges in the graph. To declare a set, type `set <setname>;`. So, we will type:

   ```
   set Nodes;
   set Edges;
   ```

In our mathematical notation, each edge is given by a pair of nodes. To specify to AMPL that we also want the set `Edges` to take the form of `(node1, node2)`, revise our set declaration above to:

```
set Nodes;
set Edges within (Nodes cross Nodes);
```

Parameter declarations Parameters are the constants that appear in your linear program. In this case, our parameters are the supply values for each node, and the edge costs and edge capacities for each edge.

To declare a parameter `supply` for each element of the set `Nodes`, type:

```
param supply {Nodes};
```

or (another, equivalent option):

```
param supply {i in Nodes};
```

Now, in the space below (as well as in your `.mod` file), declare parameters `cost` and `capacity` for each element of the set `Edges`.

Input checks Sometimes, we want our input (which we'll specify in the data file) to satisfy certain model assumptions. For example, in our problem, we want the edge capacities to be nonnegative numbers:

```
check {(i, j) in Edges}: capacity[i, j] >= 0;
```

We also want to make sure that total supply is equal to total demand. I.e. the sum of the supply values is zero.

In a standard mathematical notation, if s_i= supply at node i, then we want $\sum_{i \in Nodes} s_i = 0$.

The AMPL notation is similar:

```
check: sum { i in Nodes } supply[i] = 0;
```

Variable declarations We need a decision variable for the flow on each edge. Suppose that we call our decision variables `flow`, then try to guess how you should declare them. Make sure that you specify a flow variable for each element of the set edge. Write it down below.

Objective function The general form of the objective function is as follows.

```
Minimize <objectivename>: <objective function expression>;
```

or

```
Maximize <objectivename>: <objective function expression>;
```

where `<objective function expression>` will involve some (or all of) your decision variables. The name `<objectivename>` can be anything, but cannot be separated by spaces.

We will review the AMPL summation notation one more time. In your mathematical formulation of the LP, you should have the following objective:

$$\min c_{12}x_{12} + c_{13}x_{13} + c_{23}x_{23} + c_{32}x_{32} + c_{24}x_{24} + c_{34}x_{34},$$

or more succinctly

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij},$$

where c_{ij} denote the cost on the edge (i, j).

The AMPL notation for $\sum_{(i,j)\in E} c_{ij}x_{ij}$ in terms of the parameters and variables we declared above is very similar:

```
sum {(i, j) in Edges} cost[i, j] * flow[i, j]
```

Based on this, in the space below, write the complete line to specify your objective function:

Constraints While you can only have one objective function, you can have a lot of constraints. The general form of a constraint is as follows.

```
Subject to <constraintname>: <constraint>;
```

Sometimes, though, you might also want to write a few very similar constraints. For example, you would like to write the "same" constraint for each element in a set `Set1`. In this case, your constraint will take the form

```
Subject to <constraintname> {i in Set1}: <constraint, in terms of i>;
```

In the space provided below, write the flow conservation constraints; recall that there is one constraint for each element of the set `Nodes`.

What other constraints do you need? Add them to your model file.

3. Now, write the data file corresponding to the above graph. In this file, you will specify the nodes, edges, supply values, edge costs, and edge capacities.

   Open a new file on Notepad and save it as `Rec1.dat`. Remember to set the "Save as type" as "All files".

   Your data file must specify the elements of each of the sets mentioned in `.mod` and must specify the values of each of the parameters.

Sets We specify the elements of a set by as follows; separating each element by a space or a tab, ending with a semicolon.

```
Set1 := element1 element2 element3;
```

In the space below, write the element of the sets `Nodes` and `Edges`.

Parameters Suppose that our model file specify a parameter called `g`, which is just one number. Also suppose that in our data file, we want to set the value of `g` to be 9.81. Then, we only need to write in the data file:

```
param g := 9.81;
```

Parameters with one index On the other hand, if the parameter is specified over an index set, as we have for the supply parameter, `param supply {i in Nodes}`, then we write:

```
param supply:=
 1 10
 2 0
 3 0
 4 -10;
```

We note that the first column consists of the members of `Nodes` and the secon column consists of the corresponding values of `supply`. Note that the column format is not required. You could have equivalently write

```
param supply:= 1 10 2 0 3 0 4 -10;
```

or

```
param supply:= 1 10, 2 0, 3 0, 4 -10;
```

Finally, note that many nodes in our problem has supply value zero. Rather than having to specify these zero values by hand, you could let zero be a default value and only specify the other values. For example:

```
param supply default 0 :=
 1 10
 4 -10;
```

Parameters with two or more indices If the parameter is specified over a pair (or more) of indices, as we have for the parameter cost which is specified over edges (i, j), then we write:

```
param cost :=
1 2   1.5
1 3   3
2 3   3
2 4   3
3 2   0
3 4   1;
```

Several parameters with the same indices Suppose that we have several parameters that are specified over the same index sets. For example, the parameters `<parameter1>` and `<parameter2>` are both specified over the set with elements `index1, index2, index3, index4`. Then, we can declare them together by using

```
param: <parametername1> <parametername2> :=
 index1 value11 value21
 index2 value12 value22
 index3 value13 value23
 index4 value14 value24;
```

That is, the above is equivalent to declaring them separately:

```
param <parametername1> :=
 index1 value11
 index2 value12
 index3 value13
 index4 value14;

param <parametername2> :=
 index1 value21
 index2 value22
 index3 value23
 index4 value24;
```

In your data file, declare cost and capacity together.

4. Having finished your AMPL model (model and data files), call them to AMPL and solve it. In the space below, write down your optimal objective value and the corresponding optimal solution.

4 Your turn!

1. Next, you will create your own model for a simple "2-commodity flow problem":

   The input graph consists of a square, that is, 4 nodes connected by 4 edges in a cycle. Call the nodes, as you go clockwise, nodes 1, 2, 3, and 4. There are two commodities, A and B. Commodity A originates from node 1 with the final destination at node 3; commodity B originates from node 2 with the final destination at node 4. We can send the commodity on the edges in either direction, but the total amount of flow in both directions, for each edge, is at most 3 units. We have an additional constraint that requries that the total amount of the first commodity (sent from node 1 to node 3) must be equal to the total amount of the second commodity (sent from node 2 to node 4).

   You wish to maximize the total amount of the total of the two commodities that are shipped to their destinations.

   (a) Sketch the graph and other relevant information below. Try to determine the optimal solution by hand and write it down.

   (b) Now, write a linear programming formulation of this model, where the decision variables are $x^k_{(i,j)}$, the amount of commodity k (which is either A or B), that is shipped along the edge from i to j (in that direction).

(c) Next, try to write an AMPL model (using Notepad) for this and solve it. You can write the model and the data in separate `.mod` and `.dat` files, but you don't have to. You can incorporate the data in the `.mod` file. When saving your file, make sure that you choose "Save as type" as "All Files". [It is advised, but not necessary, that you save your file either under the `amplcml` folder, or under a new subfolder in `amplcml`.]

Write down the solution that you obtain using AMPL below, including the corresponding objective value. You do not need to write down your `.mod` and `.dat` files when you hand in your solutions.