INDR 220: Introduction to Computing for Operations Research
Homework 1: The Minimum Cost Flow Problem
Deadline: November 7, 2022, 11:59 PM

In this homework, you will implement a Python script that solves the minimum cost flow problem using CPLEX. Consider a directed and connected network where the N nodes include at least one supply node and at least one demand node. The decision variables are

$$x_{ij} = \text{flow through arc } i \text{ to } j,$$

and the given information includes

$$\begin{aligned} c_{ij} &= \text{cost per unit flow through arc } i \text{ to } j, \\ u_{ij} &= \text{arc capacity for arc } i \text{ to } j, \\ b_i &= \text{net flow generated at node } i. \end{aligned}$$

The value of b_i depends on the nature of node i, where

$$\begin{aligned} b_i > 0 \quad & \text{if node } i \text{ is a supply node,} \\ b_i < 0 \quad & \text{if node } i \text{ is a demand node,} \\ b_i = 0 \quad & \text{if node } i \text{ is a transshipment node.} \end{aligned}$$

The objective is to minimize the total cost of sending the available supply through the network to satisfy the given demand.

By using the convention that *summations are taken only over existing arcs*, the linear programming formulation of this problem is

$$\begin{aligned} \text{minimize} \quad & z = \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} x_{ij} \\ \text{subject to:} \quad & \sum_{j=1}^{N} x_{ij} - \sum_{j=1}^{N} x_{ji} = b_i \quad i = 1, 2, \ldots, N \\ & 0 \le x_{ij} \le u_{ij} \quad i = 1, 2, \ldots, N \quad j = 1, 2, \ldots, N. \end{aligned}$$

The first summation in the *node constraints* represents the total flow *out* of node i, whereas the second summation represents the total flow *into* node i, so the difference is the net flow generated at this node.

An example of a minimum cost flow problem with five nodes is given as

$$\begin{aligned} \text{minimize} \quad & z = 2x_{12} + 4x_{13} + 9x_{14} + 3x_{23} + x_{35} + 3x_{45} + 2x_{54} \\ \text{subject to:} \quad & + x_{12} + x_{13} + x_{14} = 50 \\ & - x_{12} + x_{23} = 40 \\ & - x_{13} - x_{23} + x_{35} = 0 \\ & - x_{14} + x_{45} - x_{54} = -30 \\ & - x_{35} - x_{45} + x_{54} = -60 \\ & 0 \le x_{12} \le 10 \\ & 0 \le x_{13} \le 90 \end{aligned}$$

$$0 \leq x_{14} \leq 90$$
$$0 \leq x_{23} \leq 90$$
$$0 \leq x_{35} \leq 80$$
$$0 \leq x_{45} \leq 90$$
$$0 \leq x_{54} \leq 90.$$

This problem will be represented using three `.txt` files, namely, `costs.txt`, `capacities.txt`, and `flows.txt`.

The first file contains the unit costs (i.e., c_{ij}), and it is composed of the following lines for the example problem:

```
costs.txt
---------
1 2 2
1 3 4
1 4 9
2 3 3
3 5 1
4 5 3
5 4 2
```

The second file contains the arc capacities (i.e., u_{ij}), and it is composed of the following lines for the example problem:

```
capacities.txt
--------------
1 2 10
1 3 90
1 4 90
2 3 90
3 5 80
4 5 90
5 4 90
```

The third file contains the net flows (i.e., b_i), and it is composed of the following lines for the example problem:

```
flows.txt
---------
1 50
2 40
3 0
4 -30
5 -60
```

The optimum solution of the example problem is as follows:

$$x_{12}^{\star} = 0$$
$$x_{13}^{\star} = 40$$
$$x_{14}^{\star} = 10$$
$$x_{23}^{\star} = 40$$

$$x^\star_{35} = 80$$
$$x^\star_{45} = 0$$
$$x^\star_{54} = 20$$
$$z^\star = 490.$$

Implement your algorithm to solve the minimum cost flow problem in a single interactive Python notebook using Azure Lab Services. Your notebook should include at least the following function definition that takes the file paths of three input files as parameters and returns the solution found.

```
def minimum_cost_flow_problem(costs_file, capacities_file, flows_file):
    #implement your algorithm here
    return(x_star, obj_star)
```

What to submit: You need to submit your source code in a single file (`.py` file that you will download from Azure Lab Services by following "File" / "Download as" / "Python (.py)" menu items) named as `STUDENTID.py`, where `STUDENTID` should be replaced with your 7-digit student number.

How to submit: Submit the file you created to Blackboard. Please follow the exact style mentioned and do not send a file named as `STUDENTID.py`. Submissions that do not follow these guidelines will not be graded.

Late submission policy: Late submissions will not be graded.

Cheating policy: Very similar submissions will not be graded.
