Network Optimization (Fall 2024): Suggested Projects

- Choose **one** of the two suggested projects, **or** discuss an alternative option with me.
- Each student should work on their project alone.
- Your submission must include a **concise** project report detailing your approach, and summarizing the results, along with files for any code written for the project. In particular, do **not** include your code files in the report itself; submit them separately.
- Submit all files, including the report in a **zipped** folder. Email you submission to kbala@wsu.edu.
- **You will not be allowed to include .py files in your email attachment. Notebooks (.ipynb files) are allowed. Another option is to rename your .py files as .txt instead.**
- Your folder name should identify you in the usual fashion. If you are Maxflo Sewie Sewerton, you could name your submission MaxfloSewerton_Proj.zip, for instance.
- **Begin the SUBJECT of your email submission with the same `FirstnameLastname`, expression, e.g., "MaxfloSewerton Project submission".**
- **This project is due by 11:59 PM on Thursday, December 12.**

1 Average Run Times of Max flow Algorithms

We discuss(ed) the following algorithms in class for solving the maximum flow problem (also listed are their worst-case running times).

- Shortest augmenting path algorithm; $O(n^2m)$.
- Capacity scaling algorithm; $O(nm \log U)$.
- FIFO preflow push algorithm; $O(n^3)$.

This project aims to compare the running times of these algorithms in practice to their worst-case running times.

1. (20) Implement the capacity scaling algorithm (in Matlab or another package/language). You should have implemented the other two algorithms as part of homework assignments. If you did not implement them as part of homework, do so now for this project!

2. (80) Describe a procedure to generate **random** networks having n nodes, m arcs, and each arc having an integer capacity randomly chosen from the interval $[1, U]$. You want to compute the average running times taken by the three algorithms for solving the max flow problem on these instances.

   For each set of (n, m, U) values, generate at least fifty (50) different random networks using your procedure, and record the running time taken by each algorithm to solve the max flow problem on each instance. To ensure uniformity, you should run the three algorithms on the *same* instance or network in each case (i.e., do not generate new networks for each algorithm). Vary each parameter—$n, m,$ and U—across appropriate ranges, and plot the observed average running times for each algorithm along with the worst-case running time. If the running time is $O(f(n, m, U))$, use just $f(n, m, U)$ as an estimate of the worst-case running time. Briefly summarize your findings on how each parameter affects the running times of each of the algorithms. Based on your computational experiments, would you prefer using one of the algorithms over the other two in most cases?

2 Max-Min Flow for Flushing

This project studies the water pipe network in the city of Sewerboro. The network consists of one central pumping station, and 34 demand areas. The central pumping station can pump water at any required rate in order to satisfy all the demands. Each demand area has a demand of $10,000$ GPH (gallons per hour). Our network models the central pumping station and the demand areas as nodes, and has a set of *main pipes* connecting several of these nodes. The network has 35 nodes and 64 arcs (corresponding to main pipes), and the forward star representation of the network including the capacity of each arc is given in the file `SewerboroFStar.txt` (available on the course web page).

Node 35 is the central pumping station. Each demand area has subnetworks of *local pipes*, which are not captured in the network. Some of the main pipes have a bigger capacity of 100,000 GPH, while the remaining pipes each have a capacity of 30,000 GPH. The bigger pipes are the ones going out of the central pumping station, and a few other ones going out of demand areas which are close to the pumping station. The network is directed.

1. (40) **Max-Min flow:** We would like to maintain a minimum rate of flow in each of the main pipes so that the sediments accumulating in the main pipes get cleared out during the normal supply of water to the demand areas. Specifically, we want a set of flows in the main pipes that satisfies all demands, and the *minimum* amount of water flowing in any pipe is *as large as possible*. We are permitted to *oversupply* some of the demand areas (nodes) in order to achieve this goal. Clearly explain a network method/algorithm to solve this *max-min flow* problem, and find a flow that gives the largest minimum flow. You're expected to devise a **network algorithm**, and not a direct optimization/LP formulation.

2. (60) **Flushing:** We are interested in setting up a *flushing schedule* for the local pipes in the demand areas (which are not included in the forward-star representation of the network). The idea is to run water at high speed through the local pipes so as to flush out the sediments which collect in these pipes during normal use. The main pipes are not included in the flushing process, but they continue to supply water to the remainder of the network when we are actually flushing the local pipes. Thus, some amount of flushed sediment will be carried to previously flushed demand areas, if they are "downstream" from the area that is currently being flushed. For modeling purposes, we will assume that such contamination occurs only to areas that are immediately downstream from the area that is currently getting flushed (i.e., nodes that are just one out-arc away from the current node, which is being flushed). Further, we will assume that the amount of contamination is proportional to the rate at which water is flowing along the main pipe from the area that is being flushed to the area immediately downstream. It is often convenient to perform flushing on small groups of demand areas simultaneously, until all local pipes are flushed.

   Use the flow obtained as a solution to Part (1) to set the contamination amounts in the demand areas. Determine an optimal, or a well-justified heuristic flushing schedule for the network in question. Your method should clearly identify the *groups* of demand areas that are flushed simultaneously, as well as the order in which to flush these groups. The goal of such a flushing schedule should be to minimize the total amount of re-contamination of already flushed demand areas.