

CS5310 Network Flow Assignment Report

Brandon Rodriguez

December 8, 2019

Note that all implementations use the custom built "Graph Library" Python project to help manage data.

1 Solution By Hand

See handwritten solution in documents folder.

2 Ford Fulkerson Implementation

The "Ford Fulkerson" algorithm implementation revolves around "augmenting paths". These "augmenting paths" revolve around traveling from sink node to source node (or vice versa), ideally using a breadth-first search, such that every edge travelled in this path has at least 1 open capacity.

The algorithm ends when no more augmenting paths can be found. How these paths are found is not specified by the book.

My implementation essentially finds all possible unique, non-repeating augment paths at once, before proceeding. Then, in order from shortest to longest, it travels through the edges in the graph, updating edges when possible.

It repeats these steps until no augmenting paths can be found.

3 Simplex Implementation

The "Simplex" algorithm implementation converts the graph into a set of linear equations, which are then solved with the "Linear Programming Simplex Method". My implementation uses the custom built "Simplex" Python project to solve the problem, after initial values are passed in.

However, my implementation of the algorithm doesn't seem to work quite right. Whenever it runs, it unconditionally sets **all** edges in the graph to the maximum

possible flow capacity, regardless of it actually makes sense in the relation of the entire graph.

I spent quite a few hours working on and attempting to debug this. At this point, I'm pretty confident I just don't understand some detail in regards to converting the graph node/edge/capacity values into the linear equation format. Even after consulting online references, I'm unsure of what I'm doing wrong.

Having said that, I'll note what I currently understand the logic is supposed to be.

I note that all the following is discussing the simplex from a "Simplex Tableau" point of view.

First, each column in the tableau table corresponds to a single edge. All default values in the tableau (aka, any column/row values I do not explicitly state) are assumed to default to 0.

For the objective function, we take all edges leading out from the source, and set those columns to "1". For each edge leading into the sink, we set column values to "-1".

For the constraints, we look at two things:

1) We examine all individual edge capacity values, creating the constraint " $\text{edge}_x \leq \text{edge}_x \text{ capacity}$ ". As far as the tableau, this results in a "1" in the corresponding left-hand side column value, and then the edge capacity in the corresponding right-hand side constant value. As is normal with simplex, each one of these constraints is a new row in the tableau.

2) We examine all nodes individually. For each node, we look at all edges coming inward (edge_i) and edges going outward (edge_o). For each edge_i , we set the corresponding column value in the row to 1. For each edge_o , we set the corresponding column value in the row to -1. Each node (and corresponding set of directly connected edges) gets its own row in the tableau. The right-hand side constant value for these rows should always be "= 0", as we expect the amount of flow entering the node to be equal to the amount exiting.

From what I understand, that should be all that's needed to get the Simplex algorithm to work. But as stated above, it does not seem able to solve properly

and I'm baffled as to why. Given how close it is to the end of the semester, I unfortunately don't have further time to attempt to debug this.