1 Assumptions from Last Lecture

First, we have a few things to finish from last lecture. Let $G = (V, E)$, as usual. For our analysis of Δ-Scaling, we assumed that

1. For all $e \in E$, $u_e = \infty$ and $c_e \geq 0$.

2. G is a complete graph.

For the first assumption to be valid, we need a reduction from the general case. In other words, given a general min-cost flow problem, with a graph $G = (V, E)$ having n nodes and m edges, we need a process to obtain an"equivalent" graph $G' = (V', E')$ with $u_e = \infty$, $c_e \geq 0$, for all $e \in E'$. Here, "equivalent" means that we can easily obtain the min-cost flow on G from the min-cost flow on G'. Also, G' cannot be much bigger than G, so we shall have that $|V'| = O(m + n)$ and $|E'| = O(m)$.

The trick is as follows. In the beginning, G' is a copy of G. Let $e = (v, w)$ be a directed edge of G, with b_v, b_w being demands on v and w, and c_e, u_e being the cost and capacity of e. Then, in G', we replace the edge e with the following construction. We make two extra nodes x_e and y_e, and set the demand of x_e to u_e, and the demand of y_e to $-u_e$. The demands of v and w in G' as still b_v and b_w. Then, we make make three directed edges: $(v, x_e), (y_e, x_e)$, and (y_e, w). The capacities of all these edges is ∞. If $c_e \geq 0$, then we set the costs of the three new edges to $c_e, 0, 0$ respectively. If $c_e < 0$, then the costs are $0, -c_e$, and 0.

Claim 1.0.1

1. *G has a feasible flow f with f_e flow on edge $e \in E \iff G'$ has a feasible flow f' with flows $(f_e, u_e - f_e, f_e)$ on the 3 edges that correspond to e.*

2. *If f, f' are corresponding flows as above, then $\sum_{e' \in E'} c_{e'} f'_{e'} = \sum_{e \in E} c_e f_e - \sum_{c_e < 0} c_e u_e$.*

The proof of the claim is clear, since the only possible feasible flow values on the three edges corresponding to e are of the form $(x, u_e - x, x)$ for some x. This x corresponds exactly to f_e. Therefore, G' satisfies all of the properties we required.

We have added two extra nodes for each edge using this process. This does not mean that Dijkstra's has to take longer on the new graph, however. All of the new nodes will have degree at most 2 in the residual graph, so we can have an $O(m)$ process to simplify the graph for Dijkstra, which will still run in the same time as for the original graph G.

Now we must deal with the second assumption, that of G being complete. To make G complete, we can add edges of cost $M = \sum_{e \in E} |c_e u_e| + 1$. Then, any other flow is cheaper than even 1 unit of flow on this edge.

2 The Push/Relabel Algorithm

In this section, we will formulate another algorithm for min-cost flow. This algorithm follows a particularly sound economic analogy.

Recall that a flow f and a price function p on a graph $G = (V, E)$ are at equilibrium if for all $e \in E$, $f_e = u_e$ whenever $c_e^p < 0$ and $f_e = 0$ whenever $c_e^p > 0$. According to this, if a pair (v, w) is in E_f and f, p are at equilibrium, then $c_{(v,w)}^p \geq 0$.

Our algorithm will start at equilibrium. We can do this by making $p(v) = M$, $\forall v$, and some big number M, and setting $f_e = u_e$ if $c_e < 0$ and $f_e = 0$ otherwise. Then, some nodes will have excess flow, i.e. $f^{in}(v) - f^{out}(v) > b_v$, and some nodes will have demand not satisfied by the current flow. Our goal is to have a flow which has no nodes with excess.

Definition 2.0.2 *f, p are at* <u>*ε-equilibirum*</u> *if for all $(v, w) \in E_f$ we have that $c^p_{(v,w)} \geq -\varepsilon$.*

Define an operation $\texttt{push}(v, w)$ as follows. If v has excess, $(v, w) \in E_f$, and $c^p_{(v,w)} < 0$, then $\texttt{push}(v, w)$ pushes $\delta = \min[u^f_{(v,w)}, f^{in}(v) - f^{out}(v) - b_v]$ flow on (v, w). In other words, if there is economic incentive to push flow along (v, w), then $\texttt{push}(v, w)$ pushes as much as possible (up to the excess at v). Every time this operation is executed, it gets the graph closer to a flow with no excess.

However, it is possible that a node v has excess and cannot push to any neighbor, which means that all edges $(v, w) \in E_f$ are such that $c^p_{(v,w)} \geq 0$. Then, we use the operation $\texttt{reprice}(v)$, which simply decreases $p(v)$ by $\min_{(v,w)\in E_f} c^p_{(v,w)} + \varepsilon$.

Now we can actually state the full Push/Relabel Algorithm (due to Goldberg and Tarjan).

```
Start with f and p at equilibrium
While ∃v with excess flow
   Select such a node v
      If ∃w such that push(v,w) applies, run push(v,w)
      else reprice(v)
end while
```

To be convinced that this algorithm works, we must first prove some properties.

Claim 2.0.3 *f, p are at ε-equilibrium throughout this algorithm.*

Proof: The $\texttt{push}(v, w)$ operation maintains the ε-equilibrium since before it is executed, we have that $0 > c^p_{(v,w)} \geq -\varepsilon$. The $\texttt{push}$ operation pushes at most ε, so the new edge (w, v) in the residual graph has cost $0 < c^p_{(w,v)} \leq \varepsilon$, since $c^p_{(w,v)} = -c^p_{(v,w)}$.

$\texttt{reprice}(v)$ maintains the ε-equilibrium as well. There are two types of edges to consider.

- For edges (v, w), we know that $c^p_{(v,w)} \geq 0$, since that is a necessary condition for $\texttt{reprice}$. We decrease the price at v by $\min_{(v,w)\in E_f} c^p_{(v,w)} + \varepsilon$, so after the $\texttt{reprice}$ operation, $c^p_{(v,w)} \geq -\varepsilon$.
- For edges (w, v), we know that $c^p_{(w,v)} \geq -\varepsilon$ since we are at an ε-equilibrium. The $\texttt{reprice}$ operation only increases the cost of these edges, so afterwards, we still have that $c^p_{(v,w)} \geq -\varepsilon$.

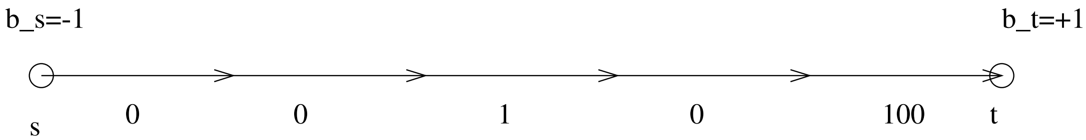
Since we start the algorithm in an ε-equilibrium, this proves the claim. □

When this algorithm terminates, f is a feasible flow because there are no nodes with excess. However, it may not be a min-cost flow, since the only thing we know is that f, p are at ε-equilibrium.

Theorem 2.0.4 *If $0 < \varepsilon < \frac{1}{n}$, where n is the number of vertices in the graph, and f, p are at ε-equilibrium, then f is a min-cost flow. (We assume costs are integral).*

Proof: We already know that a flow f is minimum-cost if there are no negative-cost cycles in the residual graph, i.e. for all cycles C in E_f, we have that $\sum_{(v,w)\in C} c_{(v,w)} \geq 0$. Consider an arbitrary cycle C in E_f. Then,

$$\sum_{(v,w)\in C} c^p_{(v,w)} = \sum_{(v,w)\in C} [c_{(v,w)} + p(v) - p(w)] = \sum_{(v,w)\in C} c_{(v,w)}$$



because the prices cancel since C is a cycle. We know that $\sum_{(v,w)\in C} c^p_{(v,w)} \geq -\varepsilon|C|$ since $c^p_{(v,w)} \geq -\varepsilon$ for each edge because we are at ε-equilibrium. By integrality of the costs, we know that if $\varepsilon < \frac{1}{|C|}$, then $\sum_{(v,w)\in C} c^p_{(v,w)} \geq 0$ because $-\varepsilon|C| > -1$. Therefore, if we take $\varepsilon < \frac{1}{n}$, then

$$\sum_{(v,w)\in C} c_{(v,w)} = \sum_{(v,w)\in C} c^p_{(v,w)} \geq 0$$

for every cycle C. Therefore, f is min-cost. $\square$

We have now proven that this algorithm works and actually produces a minimum-cost flow. However, its running time is very bad. Consider an example shown above of a directed path of 6 nodes with costs as shown, and infinite capacities, where the demand of all the nodes is 0 except for s and t. This algorithm pushes 1 unit of flow back and forth on the first two edges n times, until the prices become low enough that it can push over the third edge. It takes n times because the prices only drop by $\varepsilon < \frac{1}{n}$ every step. Then, it pushes 1 unit of flow back and forth $100n$ times before terminating. The way to make this algorithm run faster is by starting with a large ε, and slowly decreasing it. This process is similar to the Δ-scaling we discussed in the previous lecture.