CMPUT 675: Topics in Combinatorics and Optimization **Fall 2016**

Lecture 8 (Sept. 23): Minimum-Cost Flows via Augmenting Paths

Lecturer: Zachary Friggstad *Scribe: Zachary Friggstad*

8.1 Augmenting Cycles

We begin by finishing the following result (see the previous lecture for definitions).

Theorem 1 *A flow f is a minimum-cost flow if and only if $\mathcal{G}_f$ contains no negative-cost cycles.*

Proof. Last lecture, we prove that if $\mathcal{G}_f$ has a negative cost cycle then f is not a minimum-cost flow. So we focus on the other direction. Suppose $\text{cost}(f') < \text{cost}(f)$ for some flow f' with $\text{val}(f') = \text{val}(f)$. We demonstrate that $\mathcal{G}_f$ has a minimum-cost cycle.

Call some $\overline{f} : E \to \mathbb{R}_{\geq 0}$ with $\overline{f}(\delta^{in}(v)) = \overline{f}(\delta^{out}(v))$ a **circulation** (this definition does not discuss capacities). We find a negative-cost cycle in two ways. First, we show that $\overline{G}_f$ has a negative-cost circulation. Then we show a general result that any graph with a negative-cost circulation necessarily has a negative-cost cycle.

First define $g : E \to \mathbb{R}$ by $g(e) = f'(e) - f(e)$. Note $g(\delta^{in}(v)) = g(\delta^{out}(v))$ at each vertex v (including s and t) because it is the difference of two flows with the same value. Also note $\text{cost}(g) = \text{cost}(f') - \text{cost}(f) < 0$.

From g, we get a circulation in $\overline{f}$ in $\mathcal{G}_f$ as follows: for $e \in E_f$ let

$$\overline{f}(e) = \begin{cases} f'(e) - f(e) & \text{if } e \in E \text{ and } g(e) > 0 \\ f(\overleftarrow{e}) - f'(\overleftarrow{e}) & \text{if } \overleftarrow{e} \in E \text{ and } g(\overleftarrow{e}) < 0 \\ 0 & \text{otherwise} \end{cases}$$

Note that each $e \in E$ with $g(e) > 0$ has $e \in E_f$ as $f(e) < f'(e) \leq \mu(e)$ and each $e \in E$ with $g(e) < 0$ has $\overleftarrow{e} \in E_f$ as $0 \leq f'(e) < f(e)$. From this, a careful inspection of the definitions shows $\overline{f}$ is indeed a circulation in $\mathcal{G}_f$ with cost (in $\mathcal{G}_f$) equal to $\text{cost}(g) < 0$.

The proof is completed if we establish the following claim.

Claim
For every circulation $\overline{f}$, there are cycles $C_1, \ldots, C_k$ and scalars $\lambda_1, \ldots, \lambda_k > 0$

This would complete the proof because it is easily seen $\text{cost}(f) = \sum_{i=1}^{k} \lambda_i \cdot c(C_i)$. From this,

$$\min_{1 \leq i \leq k} \lambda_i \cdot c(C_i) \leq \sum_{i=1}^{k} \lambda_i \cdot c(C_i) = \text{cost}(f) < 0.$$

Because $\lambda_i > 0$ for all i, then $c(C_i) < 0$ for some i.

We now prove the claim. If $f(e) = 0$ for all $e \in E$ then we are done (with $k = 0$). Otherwise, let $uv \in E$ be such that $f(uv) > 0$. We inductively construct a walk $u = v_1, v = v_2, v_3, v_4, \ldots, v_n, v_{n+1}$ where, for any $2 \leq i \leq n$ we let v_{i+1} be any vertex such that $v_i v_{i+1} \in E$ and $f(v_i v_{i+1}) > 0$. This is guaranteed to exist because $0 < f(v_{i-1} v_i) \leq f(\delta^{in}(v_i)) = f(\delta^{out}(v_i))$.

Because $|V| = n$, there are indices $1 \leq i < j \leq n$ such that $v_i = v_j$. Choose such a pair with $j - i$ being as small as possible, then $v_i, v_{i+1}, \ldots, v_{j-1}$ are all distinct and $v_i, v_{i+1}, \ldots, v_{j-1}, v_j$ is a cycle using edges $C_1 = \{v_i v_{i+1}, v_{i+1} v_{i+2}, \ldots, v_{j-1} v_j, v_j v_i\}$ with positive f-value.

Let $\lambda_1 = \min_{e \in C} f(e)$ and define $\chi_{C_1} : E \to \{0, 1\}$ (the **indicator vector** for C_1) with

$$\chi_C(e) = \begin{cases} 1 & \text{if } e \in C_1 \\ 0 & \text{if } e \notin C_1 \end{cases}$$

Then by our choices we have $f^* := f - \lambda_1 \cdot \chi_{C_1}$ being a circulation and $|\text{supp}(f^*)| = |\{uv \in E : f^*(uv) > 0\}| < |\text{supp}(f)|$ (because some edge $e \in C_1$ has $f(e) = \lambda_1$ so $f^*(e) = 0$).

We express $f = f^* + \lambda_1 \cdot \chi_{C_1}$. Iterating with f^* and noting each iteration reduces the number of edges with positive flow by at least 1, after at most m iterations we finally see $f = \sum_{i=1}^{k} \lambda_i \cdot \chi_{C_i}$ for some $k \leq m$ and some cycles C_i with weights $\lambda_i \geq 0$. This is the same as saying $f(e) = \sum_{i: e \in C_i} \lambda_i$ for each $e \in E$. ■

8.2 Successive Shortest Paths Algorithm

We now come to our first algorithm for computing a minimum-cost flow. It will not run in polynomial time in general, but it will in the important special case of **minimum-cost perfect matching** in a bipartite graph.

The algorithm initially starts with the all-0 flow f and augments it along minimum-cost $s - t$ paths in $\mathcal{G}_f$. It also maintains a potential ϕ for $(\mathcal{G}_f, c_f)$ in each step. This will both allow us to quickly find the augmenting path (using Dijkstra's) and will also certify at each step that we have a minimum-cost flow.

Assumption
We will assume the maximum flow value in $\mathcal{G}$ is at most γ. This can be enforced by creating a new vertex s' that has a single outgoing edge to s, setting $\mu(s's) = \gamma$ and $c(s's) = 0$.

Algorithm 1 Successive Shortest Paths Algorithm for the Minimum-Cost Flow Problem

---

Input: $\mathcal{G} = (V; E)$ (directed), capacities $\mu : E \to \mathbb{R}_{\geq 0}$, costs $c : E \to \mathbb{R}_{\geq 0}$, distinct $s, t \in V$, target flow $\gamma \geq 0$.
Output: A minimum-cost flow of value γ plus a potential for $\mathcal{G}_f$, or None.

```
add a new vertex s' to V and a new edge s's to E with μ(s's) = γ, c(s's) = 0.
f(e) ← 0 for each e ∈ E
φ(v) ← 0 for each v ∈ V        {We will prove φ is always a potential for the flow f.}
while there is an s' − t path in G_f do
  P ← a minimum-cost s' − t path in G_f       {Computed via Dijkstras using costs c_φ.}
  for each v ∈ V, ℓ(v) ← cost of a minimum-cost s' − v path in G_f under edge-costs c_φ
  φ(v) ← φ(v) + ℓ(v)
  Augment f along P.
end while
if val(f) < γ then
  return None
else
  return f, φ (excluding the entries f(s's) and φ(s'))
end if
```

Alternatively, instead of adding the new vertex s' and edge $s's$, we could just ensure the augmenting step never augments the flow past value γ.

This correctly decides if there is even a flow with value γ because it operates just like the Ford-Fulkerson algorithm. We will verify that each iteration leaves ϕ being a potential for f. Given this, we see each iteration runs in $O(m + n \log n)$ time because every step takes linear time apart from the Single Source Shortest Path calculation, which can be executed in $O(m + n \log n)$ time using Dijkstra's algorithm on $\mathcal{G}_f$ and the potential ϕ.

Unfortunately this suffers from the same running time issue as the Ford-Fulkerson algorithm. The number of iterations may not be polynomial in the input size. We will briefly discuss (next lecture) how it works well for minimum-cost bipartite matching. We will, however, establish that in each iteration f is a minimum-cost flow.

Theorem 2 *Initiallly and after each iteration, f is a minimum-cost flow and ϕ is a potential for $\mathcal{G}_f$.*

Proof. Really we just have to establish that ϕ is a potential for $\mathcal{G}_f$, because this certifies $\mathcal{G}_f$ has no negative-cost cycles (the proof was given in the previous lecture) and then Theorem 1 shows f is a negative-cost cycle.

To that end, note that ϕ is initially a potential because $c_\phi(uv) = c(uv) + \phi(u) - \phi(v) = c(uv) \geq 0$ because only original edges of E lie in the initial residual graph $\mathcal{G}_f$ (as f is the all-0 flow) and we assume $c(e) \geq 0$ for each $e \in E$.

Let f_i, ϕ_i denote the flow and potential after the i'th step (and f_0 the initial flow, ϕ_0 the initial potential). Inductively, assume ϕ_i is a potential for $\mathcal{G}_{f_i}$. Also let $\ell_i(v)$ denote the lengths calculated in the i'th iteration. Consider some $uv \in E_{f_{i+1}}$. If $uv \in E_{f_i}$ we have

$$\ell_i(v) \leq \ell_i(u) + c_{\phi_i}(uv) = \ell_i(u) + c(uv) + \phi_i(u) - \phi_i(v).$$

So

$$c_{\phi_{i+1}}(uv) = c(uv) + \phi_{i+1}(u) - \phi_{i+1}(v) = c(uv) + \phi_i(u) + \ell_i(u) - \phi_i(v) - \ell_i(v) \geq 0$$

where the bound is exactly what was just shown above.

If $uv \notin E_{f_i}$ then because $uv \in E_{f_i}$ it must be that vu was on the minimum-cost path P. In this case, we know

$$\ell_i(u) = \ell_i(v) + c_{\phi_i}(vu).$$

Then

$$\begin{aligned} c_{\phi_{i+1}}(vu) &= c(vu) + \phi_{i+1}(v) - \phi(i+1)(u) \\ &= c(vu) + \phi_i(v) + \ell_i(v) - \phi(u) - \ell_i(u) \\ &= c_{\phi_i}(vu) + \ell_i(v) - \ell_i(u) \\ &= 0. \end{aligned}$$

Then by the simple observation that $c_{f_{i+1}}(uv) = -c_{f_{i+1}}(vu)$ for any $vu \in E$, we also have $c_{\phi_{i+1}}(uv) = 0$. ■

The algorithm can easily be adapted to work if some input costs can be negative even if there are negative-cost cycles.

- If there is a negative-cost cycle using only edges with infinite capacity (which, technically, isn't allowed by our definition but we could easily extend it to include this), then there is no notion of a *cheapest* flow because we could push arbitrarily-high amounts of flow across this cycle.
- Otherwise, we start with the all-o flow and then iteratively augmenting the flow across negative-cost cycles in $\mathcal{G}_f$. One can prove this eventually stops if the capacities are rational values. Once this is done, we get our initial potential for the current residual network and then begin augmenting along minimum-cost augmenting paths until there are no more augmenting paths.