Minimum Cost Flow

Notations:
- Directed graph \( G = (V, E) \)
- Let \( u \) denote capacities
- Let \( c \) denote edge costs.
- A flow of \( f(v, w) \) units on edge \((v, w)\) contributes cost \( c(v, w)f(v, w) \) to the objective function.

Different (equivalent) formulations
- Find the maximum flow of minimum cost.
- Send \( x \) units of flow from \( s \) to \( t \) as cheaply as possible.
- General version with supplies and demands
  - No source or sink.
  - Each node has a value \( b(v) \).
  - positive \( b(v) \) is a supply
  - negative \( b(v) \) is a demand.
  - Find flow which satisfies supplies and demands and has minimum total cost.
General version of min-cost flow

- Directed graph  $G = (V, E)$
- non-negative edge capacities  $u$
- edge costs  $c$
- Supply/demand  $b$  on each vertex

\[
\begin{align*}
\min & \sum_{(v, w) \in E} c(v, w)f(v, w) \\
\text{subject to} & \\
& f(v, w) \leq u(v, w) \quad \forall (v, w) \in E \\
& \sum_{w \in V} f(v, w) - \sum_{w \in V} f(w, v) = b(v) \quad \forall v \in V \\
& f(v, w) \geq 0 \quad \forall (v, w) \in E
\end{align*}
\]
Assumptions

- if $(v, w) \in E$, then $(w, v) \notin E$
- $\sum_v b(v) = 0$
- Graph is directed
- costs/capacities are integral
- There exists a directed path of infinite capacity between each pair of nodes.
Residual Graph

- Capacity is as for flow (now use $u_f(v,w)$ for residual capacity)
- If $(v, w) \in E$ and $(w, v) \in E_f$ then $c(w, v) = -c(v, w)$. 
Optimality of a flow 1: Negative Cycles

Characterization 1: A feasible flow $f$ is optimal iff $G_f$ has no negative cycles.

Note 1: A feasible flow is one satisfying all supplies/demands. The 0-flow is not feasible (unless all $b(v) = 0$).

Note 2: Flow decomposition for min-cost flow. The difference between any two feasible flows is a collection of cycles.
Node Potentials

• Similar to shortest paths, we use node potentials $\pi(v)$.

• **Reduced cost of edge** $(v, w)$,

  $$c^\pi(v, w) = c(v, w) - \pi(v) + \pi(w)$$

• For any cycle $X$, we have

  $$\sum_{(v,w) \in X} c^\pi(v, w) = \sum_{(v,w) \in X} c(v, w)$$
Optimality 2: Reduced Cost Optimality

**Reduced Cost Optimality:** A feasible flow $f$ is optimal iff there exists potentials $\pi$ such that

$$c^\pi(v, w) \geq 0 \quad \forall (v, w) \in G_f$$
A feasible flow $f$ is optimal iff there exists potentials $\pi$ such that for all edges $(v, w) \in G$

- if $c^\pi(v, w) > 0$ then $f(v, w) = 0$
- if $0 < f(v, w) < u(v, w)$ then $c^\pi(v, w) = 0$
- if $c^\pi(v, w) < 0$ then $f(v, w) = u(v, w)$.
More on $f$ and $\pi$

Two Questions;

- Given an optimal $f$, how do we compute $\pi$?
- Given an optimal $\pi$, how do we compute $f$?
First Answer

Given an optimal $f$, how do we compute $\pi$?

Solution:
- Use Reduced Cost Optimality,
- Compute shortest path distances $d$ in $G_f$,
- Let $\pi = -d$
Given an optimal $\pi$, how do we compute $f$?

Solution

- Use Complimentary Slackness
- Fix $f$ on the edges with $c^\pi(v, w) < 0$ or $c^\pi(v, w) > 0$
- Solve the resulting max flow problem on edges with $c^\pi(v, w) = 0$
There are many algorithms for min cost flow, including:

- Cycle cancelling algorithms (negative cycle optimality)
- Successive Shortest Path algorithms (reduced cost optimality)
- Out-of-Kilter algorithms (complimentary slackness)
- Network Simplex
- Push/Relabel Algorithms
- Dual Cancel and Tighten
- Primal-Dual
- ...
Cycle Cancelling Algorithm

Basic Algorithm (Klein’s Algorithm)

- Find a feasible flow $f$ (solve a maximum flow)
- While there exists a negative cost cycle $X$ in $G_f$
  - Let $\delta = \min_{(v, w) \in X} u_f(v, w)$
  - Send $\delta$ units of flow around $X$

Analysis:

- Let $U = \max_{(v, w) \in E} u(v, w)$
- Let $C = \max_{(v, w) \in E} |c(v, w)|$
- For any feasible flow $-mCU \leq c(f) \leq mCU$
- Each iteration of the Basic Cycle Cancelling Algorithm decreases objective by at least 1.

- **Conclusion:** At most $2mCU$ iterations.
- Running time $= O(nm^2CU)$. Not polynomial.
Ideas for Improvement

– Send flow around most negative cycle. (NP-hard to find)
– How many iterations would that be?
Ideas for Improvement

– Send flow around most negative cycle. (NP-hard to find)
– How many iterations would that be?

Analysis:
– The difference between any two feasible flows is the union of at most $m$ cycles.
– Let $f$ be the current flow, $f^*$ be the optimal flow.
– Consider $f - f^*$. It is the union of at most $m$ cycles.
– The most negative cycle in $f - f^*$ must have cost at least

$$\frac{1}{m}(f^* - f)$$

.
Analysis continued

– Each iteration gets \( \frac{1}{m} \) of the way to the optimal flow.
– Equivalently, each iteration decreases the distance to the optimal flow by a \( 1 - \frac{1}{m} \) factor.
– Initial distance is at most \( 2mCU \).
– Once we get within one of the optimal flow, we are done, since flows, and costs of flows are integers.

Conclusion: The number of iterations is

\[
\log_{1/(1-1/m)}(mCU)
\]

Analysis:

\[
\log_{1/(1-1/m)}(mCU) = \frac{\log(mCU)}{\log(1/(1 - \frac{1}{m}))} \\
\approx \frac{\log(mCU)}{\frac{1}{m+1}} \\
= (m + 1) \log(mCU)
\]

There are \( O(m \log(mCU)) \) iterations.
Cycle Cancelling

• If we could find most negative cycle, there would be a polynomial number of iterations.

• Finding the most negative cycle is NP-hard.

• Solution: Find minimum mean cycle and cancel it.

• We will show that the minimum mean cycle “approximates” the most negative cycle well.
Minimum Mean Cycle Algorithm

• Find a feasible flow $f$ (solve a maximum flow)
• While there exists a negative cost cycle $X$ in $G_f$
  – Let $X$ be the minimum mean cycle
  – Let $\delta = \min_{(v, w) \in X} u_f(v, w)$
  – Send $\delta$ units of flow around $X$ (Maintain potentials $\pi$ at nodes).

Note: Flows are always feasible in this algorithm

Def: A flow $f$ is $\epsilon$-optimal if there exists potentials $\pi$ such that

$$c^\pi(v, w) \geq -\epsilon \ \forall (v, w) \in G_f$$
Lemma:

- Any feasible flow is $C$-optimal.
- If $\epsilon < 1/n$, then an $\epsilon$-optimal flow is optimal.
Main Theorem

Defining $\epsilon$ given $f$ and $\pi$: Given $\pi$ and $f$, let $\epsilon^\pi(f) = -\min_{(v,w)\in G_f}\{c^\pi(v,w)\}$. This value is the smallest $\epsilon$ for which the flow $f$ is $\epsilon$-optimal.

Choosing $\pi$, given $f$

- Note that $f$ is not optimal, so we cannot just run shortest paths to find an optimal $\pi$.
- Let $\epsilon(f) = \min_\pi \epsilon^\pi(f)$.
- Let $\mu(f)$ be the minimum mean cycle value in $G_f$.

Theorem  Given any feasible flow $f$

$$\epsilon(f) = -\mu(f)$$
Lemma: Let $f$ be a feasible non-optimal flow. Let $X$ be the minimum mean cycle in $G_f$. Then there exist $\pi$ s.t.

$$c^\pi(v,w) = \mu(f) = -\epsilon(f) \quad \forall (v,w) \in X$$
Lemma: Let $f$ be a feasible non-optimal flow. Let $X$ be the minimum mean cycle in $G_f$. Suppose we push flow around $X$ to obtain $f'$. Then $\epsilon(f') \leq \epsilon(f) = \epsilon$. 
Lemma: Let $f$ be a feasible non-optimal flow. Suppose that we execute $m$ iterations of the minimum-mean cycle algorithm to obtain $f$. Then, if the algorithm has not terminated, we have that

$$\epsilon(f') \leq \left(1 - \frac{1}{n}\right) \epsilon(f)$$

.
Summary

• In $m$ iterations, $\epsilon$ decreases by a $1 - 1/n$ factor.
• In $nm$ iterations, $\epsilon$ decreases by a $(1 - 1/n)^n \approx 1/e$ factor.
• Initially $\epsilon \leq C$
• We stop when $\epsilon \leq 1/n$
• Decrease by a factor of $e \ln(nC)$ times.
• Therefore, number of iterations is $O(nm \log(nC))$
• Running time is $O(n^2m^2 \log(nC))$

Nice feature of algorithm: No explicit scaling. Explicit scaling enforces a lower bound.
Strongly Polynomial Algorithm

- Recall that strongly polynomial means polynomials in $n$ and $m$ and “independent” of $C$ and $U$.
- We have seen strongly polynomial algorithms for maximum flow.
- No strongly polynomial algorithm is known for linear programming.
- No strongly polynomial algorithm is known for multicommodity flow.
- We will see a strongly polynomial algorithm for minimum cost flow, one of the “hardest” problems for which such an algorithm exists.
- Strongly polynomial is mainly a theoretical issue.

**Theorem:** The minimum mean cycle algorithm runs in $O(n^2m^3 \log n)$ time.
Analysis

Ideas for strongly polynomial algorithm

• If, at some point $|c^\pi(v, w)| \gg \epsilon(f)$, then (v, w) if fixed, the flow will never change.
  
  – If $c^\pi(v, w)$ large positive, you never want to put most flow on it.
  – If $c^\pi(v, w)$ large negative, you never want to remove flow from it.

More precisely

• An edge if $\epsilon$-fixed if the flow on that edge is the same for all $\epsilon'$-optimal flows, for all $\epsilon' \leq \epsilon$.

• Once an edge is $\epsilon$-fixed, we can freeze the flow on that edge, and ignore the edge for the remainder of the algorithm.

• We therefore have a notion of progress that depends on the number of edges of the graph.
Analysis

**Theorem** If $|c^π(v, w)| ≥ 2nε(f)|$, then $(v, w)$ is $ε$-fixed.
Analysis Continued

**Theorem:** Every $nm(\ln n + 1)$ iterations, at least one edge becomes $\epsilon$-fixed.

**Corollary:** Total of $O(nm^2 \lg n)$ iterations and $O(n^2m^3 \lg n)$ running time.