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**

$$\min \sum_{(v,w)\in E} c(v,w)f(v,w)$$

subject to

$$\begin{aligned}
f(v,w) &\leq u(v,w) & \forall (v,w) \in E \\
\sum_{w\in V} f(v,w) - \sum_{w\in V} f(w,v) &= b(v) & \forall v \in V \\
f(v,w) &\geq 0 & \forall (v,w) \in E
\end{aligned}$$

Assumptions

- **if** $(v, w) \in E$ **, then** $(w, v) \notin E$
- $\Sigma_v\, b(v) = 0$
- **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 exsits potentials π such that

$$c^{\pi}(v, w) \geq 0 \quad \forall (v, w) \in G_f$$

Optimality 3: Complimentary Slackness

A feasible flow f is optimal iff there exsits potentials π 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 π

Two Questions;

- **Given an optimal f, how do we compute π ?**
- **Given an optimal π, how do we compute f ?**

First Answer

- **Given an optimal f , how do we compute π ?**

Solution:

- **Use Reduced Cost Optimality,**
- **Compute shortest path distances d in G_f ,**
- **Let $\pi = -d$**

Seond Answer

- **Given an optimal** π **, 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** $c^\pi(v,w) = 0$

Algorithms for Minimum Cost Flow

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 δ 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 $-mCU \le c(f) \le mCU$
- 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

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

.

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

$$\lg_{1-1/m}(mCU)$$

.

Analysis:

$$\begin{aligned} \lg_{1-1/m}(mCU) &= \frac{\lg(mCU)}{\lg(1-\frac{1}{m})} \\ &\approx \frac{\lg(mCU)}{\frac{1}{m}} \\ &= m\lg(mCU) \end{aligned}$$

There are $O(m\lg(mCU))$ 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 "aproximates" the most negative cycle well.**

Mnimum 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 δ units of flow around X (Maintain potentials π at nodes).**

Note: Flows are always feasible in this algorithm

Def: A flow f is ϵ-optimal if there exists potentials π such that

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

ϵ-optimality

Lemma:

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

Main Theorem

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

Choosing π, given f

- **Note that f is not optimal, so we cannot just run shortest paths to find an optimal π**
- **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**

$$\epsilon(f) = -\mu(f)$$

More analysis

Lemma: **Let f be a feasible non-optimal flow. Let X be the minimum mean cycle in G_f. Then there exist π s.t.**

$$c^{\pi}(v,w) = \mu(f) = -\epsilon(f) \;\; \forall (v,w) \in X$$

Progress

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** $\epsilon(f') \leq \epsilon(f) = \epsilon$

Measured Progress

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, ϵ decreases by a $1 - 1/n$ factor.
- In nm iterations, ϵ 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 $O(n^2 m^2 \log(nC))$

Nice feature of algorithm: No explicit scaling. Eplicit 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 $O(n^2 m^3 \log n)$ time.**

Analysis

Ideas for strongly polynomail algorithm

- If, at some point $|c^{\pi}(v, w)| >> \epsilon(f)$, 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 ϵ -fixed if the flow on that edge is the same for all $\epsilon^{'}$ -optimal flows, for all $\epsilon^{'} \leq \epsilon$.
- Once an edge is ϵ -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** $|c^\pi(v,w)| \geq 2n\epsilon(f)|$ **, then** (v,w) **is** ϵ **-fixed.**

Analysis Continued

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

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