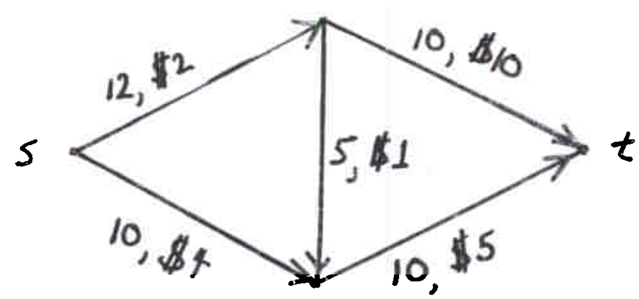
Minimum-Cost Network Flow

In addition to a capacity, each edge has a real-valued cost per unit of flow.

A minimum-cost (maximum) flow is a maximum flow whose total cost (sum of edge flows times edge costs) is minimum.

Problem: find a minimum-cost flow in a given network.



n = # vertices
m = # edges
U = max capacity (if integers)
C = max cost (if integers)

22,-15 40

Two Naive Approaches

(1) Repeat: augment along a cheapest path in the residual network.

Each augmentation takes a shortest path computation.

Shortest paths can be found using Dijkstra's algorithm if costs are kept non-negative using price transformation (primal-dual method of linear programming).

Time: $O(nU(m+n\log n))$ (not polynomial)

(2) repeat { In network of zero-cost residual edges, find a maximum flow. Augment the flow and update the prices. }
(find all paths of a given cost at once)

Time: $O(nC(nm\log(n^2/m)))$ (not polynomial)

$G = (V, E)$ symmetric directed graph

$(v, w) \in E$ iff $(w, v) \in E$

$|V| = n$, $|E| = m$ $\quad m \geq n \geq 2$ $\quad E(v) = \{w \mid (v, w) \in E\}$

arc capacities $u(v, w) : (v, w) \in E$

arc costs $c(v, w) : (v, w) \in E$

cost function is antisymmetric: $c(v, w) = -c(w, v)$

Circulation $f : E \to R$

$f(v, w) \leq u(v, w) \quad \forall (v, w) \in E$ capacity constraints

$f(v, w) = -f(w, v) \quad \forall (v, w) \in E$ flow antisymmetry

$\sum_{w \in E(v)} f(v, w) = 0 \quad \forall v \in V$ flow conservation

Cost of f: $\frac{1}{2} \sum_{(v, w) \in E} f(v, w)\, c(v, w)$

Polynomial - Time Algorithms

Date	Discoverer	Time
1972	Edmonds and Karp	$O(m(\log U)(m + n\log n))$
1980	Rock	$O(m(\log U)(m + n\log n))$
1980	Rock	$O(n(\log C)\ nm\log(n^2/m))$
1984	Tardos	$O(m^4)$
1984	Orlin	$O(m^2(\log n)(m + n\log n))$
1985	Fujishige	$O(m^3\log n)$
1985	Bland and Jensen	$O(n(\log C)\ nm\log(n^2/m))$
1986	Galil and Tardos	$O(n^2(\log n)(m + n\log n))$
1987	Goldberg and Tarjan	$O((\log nC)\min\{n^3, nm\log n\})$

Naive Approach

(1) Repeatedly augment along a cheapest path in the residual graph.

Each augmentation takes a shortest path computation.

Shortest paths can be found using Dijkstra's algorithm if costs are kept non-negative using "prices" to transform costs (primal-dual method).

Time: $O(nU(m + n\log n))$ (not polynomial)

(2) repeat { In network of zero cost edges, find a maximum flow. Augment the flow and update the prices.

Time: $O(nC(nm\log(n^2/m)))$ (not polynomial)

Reformulated problem: Find a circulation of minimum cost: add a return arc from s to t of infinite capacity and large negative cost $(-nC)$.

residual capacity $\quad u_f(v,w) = u(v,w) - f(v,w)$ or $f(w,v)$

residual arc (v,w): $\quad u_f(v,w) > 0$

residual cycle: a (simple) cycle of residual arcs

<u>length</u> of cycle = number of arcs, $\ell(\Gamma)$

<u>cost</u> of cycle = sum of arc costs = $c(\Gamma)$

<u>mean cost</u> of cycle = $c(\Gamma)/\ell(\Gamma)$

<u>negative</u> cycle: $c(\Gamma) < 0$

Theorem (Busacker and Saaty, 1965): A circulation f is minimum-cost iff there is no negative residual cycle.

Algorithm (Klein, 1967)

0. Find any circulation f (by a max flow computation)

1. While $\exists$ negative cycle Γ, cancel Γ by increasing f on all arcs of Γ by $\min\{u_f(v,w) : (v,w) \in \Gamma\}$.

   #iterations can be exponential (or infinite)

How to choose cycles for canceling to minimize #iterations, running time?

minimum cost?

minimum length?

maximum length?

maximum capacity?

maximum cost decrease?

Primal Network Simplex Algorithm: Definitions

~~[illegible]~~

$\{v,w\}$ is residual if both (v,w) and (w,v) are residual

f is basic if the set of residual edges forms a forest

The algorithm maintains a basic circulation f and a basis tree T such that T contains every residual edge.

Any nontree arc (v,w) defines a basic cycle $\Gamma_T(v,w)$ consisting of (v,w) and the path of tree arcs from w to v.

(We regard each tree edge as consisting of a pair of tree arcs.)

An arc (v,w) is pseudoresidual if it is residual or a tree arc.

A simple cycle is pseudoresidual if it consists only of pseudoresidual arcs.

Our Results

Minimum-mean cycle canceling: Always cancel a cycle of minimum mean cost.

<u>Theorem</u>: # cancellations $= O(nm^2 \log n)$. If costs are integers of maximum magnitude C, # cancellations $= O(nm \log(nC))$.

Time to find a minimum mean cycle $= O(nm)$ (Karp, 1978)

A variant of this approach gives a "practical" algorithm with a running time of $O(nm \log n \min\{\log(nC), m \log n\})$.

Price Function (Dual Variables)

$p: V \to \mathbb{R}$ reduced arc cost $c_p(v,w) = c(v,w) + p(v) - p(w)$

<u>Theorem</u> (Ford and Fulkerson, 1962): A circulation f is minimum-cost iff $\exists p$ such that, $\forall (v,w) \in E$,

$u_f(v,w) > 0$ implies $c_p(v,w) \geq 0$.

ϵ-optimality

For $\epsilon > 0$, a circulation f is ϵ-optimal with respect to a price function p iff, $\forall (v,w) \in E$,

$u_f(v,w) > 0$ implies $c_p(v,w) \geq -\epsilon$.

$\epsilon(f)$ = minimum $\epsilon \geq 0$ for which f is ϵ-optimal with respect to some p.

<u>Theorem</u> (Bertsekas, 1986): If costs are integral and $\epsilon < 1/n$, any ϵ-optimal circulation is minimum-cost.

Analysis of Minimum Mean Cycle Canceling

<u>Lemma</u>: Canceling a minimum mean cycle cannot increase $\epsilon(f)$.

<u>Lemma</u>: After m cancellations, $\epsilon(f)$ has decreased by a factor of $(n-1)/n$ or better.

<u>Theorem</u>: # cancellations = $O(nm \log(nC))$.

<u>Lemma</u>: If f and f' are both ϵ-optimal and $|c_p(v,w)| > 2n\epsilon$, where f is ϵ-optimal with respect to p, then $f(v,w) = f'(v,w)$.

<u>Theorem</u>: # cancellations = $O(nm^2 \log n)$.

Idea: minimum mean cycle canceling reduces $\epsilon(f)$ by a measurable amount, after enough cancellations.

Note: The cost of a cycle is the same as its reduced cost.

Key question: What is $\epsilon(f)$?

Let $\mu(f)$ be the mean cost of a minimum mean residual cycle with respect to circulation f.

<u>Theorem</u>: $\epsilon(f) = \max\{0, -\mu(f)\}$.

<u>Proof</u>: Use properties of shortest paths, e.g. shortest paths exist iff there are no negative cycles.

A Practical Variant

Maintain a price function p along with a circulation f.

Call an arc (v,w) <u>eligible</u> if $u_f(v,w) > 0$ and $c_p(v,w) < 0$.

Let $\epsilon(f,p) = -\min\left(\{c_p(v,w) \mid u_f(v,w) > 0\} \cup \{0\}\right)$.

Algorithm

0. Let f be any circulation and let $p = 0$.

1. Repeat until $\epsilon(f,p) < 1/n$:

   a. Find an eligible cycle and cancel it.

   b. If the subgraph of eligible arcs is acyclic, modify p to reduce $\epsilon(f,p)$ by a factor of at least $(n-1)/n$.

Analysis

There are at most m iterations of 1a between iterations of 1b.

All iterations of 1a between two iterations of 1b take a total of $O(m \log n)$ time using dynamic trees.

One iteration of 1b takes $O(m)$ time.

$O(n)$ iterations of 1b reduce $\epsilon(f,p)$ by a constant factor.

$\therefore\ O(nm \log n \log(nC))$ total time.

If every n^{th} iteration of 1b reduces $\epsilon(f,p)$ as much as possible, then the amortized time per iteration of 1b is still $O(m)$ (every n^{th} takes $O(nm)$).

$\therefore\ O(nm^2 (\log n)^2)$ total time.

Scaling Approach

Add a bit of precision (to capacities or costs) at a time. Start with exact solution to approximate problem. Use it as an approximate solution to a more exact problem. Improve the solution to an exact solution.

Scaling capacities (Edmonds & Karp; Rock)

For each bit of capacity precision, must find $O(m)$ cheapest paths.
Find a cheapest path using Dijkstra's shortest path algorithm; use price transform.

Time: $O(m \log U (m + n \log n))$

Scaling costs (Rock; Bland & Jensen)

For each bit of cost precision, must find $O(n)$ maximum flows.

Time: $O(n \log C (nm \log (n^2/m)))$