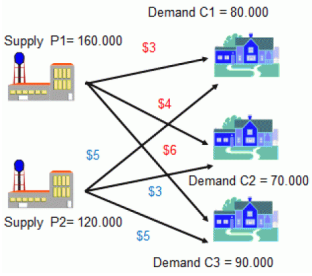
Algorithm Design and Analysis

Victor Adamchik CS 15-451 Spring 2015

Lecture 13 Mar 02, 2015 Carnegie Mellon University

# The Min-Cost Max-Flow Problem

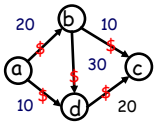



Plan:

Cycle cancelation algorithm

The cheapest augmenting path algorithm

## The problem




Directed graph $G=(V,E)$

Each edge has a capacity $w(e)\geq 0$

Each edge has a cost $c(e)\geq 0$ per unit flow.

The cost of flow is given by $|f| = \sum_{e \in E} f(e)\, c(e)$

The goal is to find the minimum cost flow, subject to

1) $0 \leq f(u, v) \leq w(u, v)$
2) $f(u, v) = -f(v, u)$
3) at a vertex $V-\{s, t\}$: flow-in = flow-out

The problem combines a max flow problem with the shortest path problem.

## Residual Network

Network: $G = (V, E)$ and flow $f$.

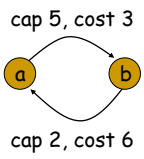
Residual capacity: $w_f(u,v) = w(u,v) - f(u,v)$

Residual graph: $G_f(V, E_f)$, where

$$E_f = \{(u,v) \in V^2 \mid w_f(u,v) > 0\}$$

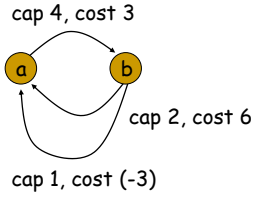
If $(v, u) \in E$ and $(u, v) \in E_f$ then $c(u, v) = -c(v, u)$ .

## Example




Suppose we send 1 flow from a to b

In $G_f$:




Theorem. Given any flow f in G. Let $G_f$ denote the residual graph. f has min-cost iff $G_f$ has no negative cycles (wrt to the cost).

Proof. ⇒) Suppose $G_f$ has a negative cycle.
At least 1 unit of flow can be pushed over the cycle.
The resulting flow will have a lower cost.
⇐) Suppose ∃f', s.t. |f'|=|f| and cost(f') < cost(f).
Consider a flow d=f'-f, where cost(d)<0.
d is a circulation (a flow without s and t).
d can be decomposed into cycles in $G_f$.
At least one of them must be negative.

Theorem (Flow Decomposition). Any feasible flow f can be decomposed into no more than E cycles and s-t paths.

Proof. Run DFS on $G_f$
We get either s-t path P or a cycle C.
Update the flow along P or C by f(u,v) – min_edge.
Update $G_f$. Repeat.
Note, on each step we set a flow to 0 on ≥ one edge.
If there are no edges in $G_f$ leaving s, then
$G_f$ must consists of cycles.
Run DFS on $G_f$ starting at vertex with leaving edges.
We have to come back by the flow conservation law.
Update flow and $G_f$. Repeat.

## Algorithm 1 (cycle canceling)

Given (G, s, t, w, c)
1) Find a max-flow ignoring cost (Ford-Fulkerson)
2) Construct $G_f$
3) Search $G_f$ for a negative cost cycle (Bellman-Ford)
4) If ∄ neg. cost cycle ⇒ done!
5) If ∃ neg. cost cycle ⇒ push a flow around.
6) Go to 2).

Finding the most negative cost cycle is NP-hard.

## Algorithm 1 (cycle canceling)

Termination.

Augmentation along the cycle will saturate at least one edge, so the cycle is no longer in $G_f$.

The cycle was "canceled".

But cancelation may create another neg. cycle.

Each augmentation decreases cost by at least 1

Runtime: $O(V E^2 cap_{max} cost_{max})$

Number of iterations $E\ cap_{max}\ cost_{max}$

## Demand/Supply Problem

We associate to each node v a number b(v) representing its supply/demand.

If b(v) > 0, node v is a surplus node;

if b(v) < 0, node v is a demand node with a demand of −b(i);

and if b(v) = 0, node v is a transshipment node.

Goal: satisfy demand at min cost, subject to

## Demand/Supply Problem

Finding feasible flow.

Consider the network G, but without any cost.

Add two special nodes s and t and add edges as follows.

For each v ∈ V ,

1) if b(v) > 0 then add (s, v) with capacity b(v),

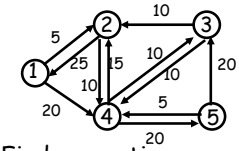
2) if b(v) < 0 then add (v, t) with capacity −b(v).

Now solve a maximum flow problem from s to t.

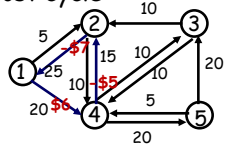
If the max flow saturates all the source edges, then the original network has a feasible flow; otherwise it is infeasible

## Example

1) Find a feasible flow (ignoring cost) of 25

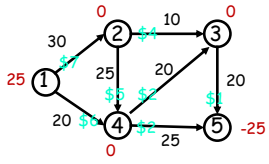



2) Find a negative cost cycle

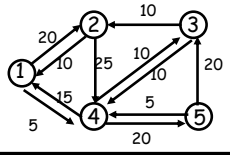



$4 – cost

10 - capacity




3) Send flow around the cycle with the capacity 15.



## Algorithm 2

Given (G, s, t, w, c, F)

1) Stat with |f|=0
2) Construct $G_f$
3) Find the shortest cost path s-t in $G_f$
4) Augment the flow along that path
5) If the flow reaches F, terminate
6) Go to 2)

In step 3), Bellman-Ford or Dijkstra?

## Shortest paths with potentials

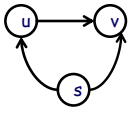
The idea is similar to Johnson's algorithm (lect. 9)
We define a potential $\pi$ for each vertex.

Next we define an adjusted cost

$$c_\pi(u,v) = c(u,v) + \pi(u) - \pi(v) \geq 0$$

Here $\pi(v)$ is the shortest distance from s to v. Thus

$$\pi(v) \leq \pi(u) + c(u, v)$$




## Shortest paths with potentials

Like in Johnson's algorithm, the cost of the path is not changed.
Run Dijkstra's algorithm from s to find $\delta_\pi(s,v)$
Push a flow along that path.

Update potential for each vertex

$$\pi^*(v) = \pi(v) + \delta_\pi(s,v)$$

All the edges remain non-negative in length

Runtime: $O(E \text{ Log } V |F|)$