Moscow Workshops ICPC, Discover Riga 2019, Min Cost Flow Lecture Notes

Pavel Kunyavskiy

September 24, 2019

1 Notation and definitions

A *network* is a graph $G(V, E)$ with capacity function $c_e : E \to \mathbb{R}_+$, and two special vertices s (source) and t (sink). For each edge e we consider there is a *reversed edge* e' going from end vertex of original edge to beginning one, having capacity zero.

A *weighted network* is a network with weight function $w_e : E \to \mathbb{R}$, $w_{e'} = -w_e$.

A *flow* is function $f_e : E \to \mathbb{R}$ satisfying following conditions: $f_e \leqslant c_e$, $f_e = -f_{e'}$, $\sum\limits_u f_{(v,u)} = 0$ for all v, except s and t.

A *value* of flow is sum of flow over edges going out of source.

A *cost* of flow is $\sum\limits_e w_e \cdot f_e$ for edges of initial graph (excluding reversed edges).

A *minimum-cost-k-flow* problem is problem to find a flow of value k with minimum cost.

A *minimum-cost-max-flow* problem is problem to find a flow of maximum value with minimum cost.

A *minimum-cost-flow* problem is problem to find a flow of any value with minimum cost.

A *residual network* with respect to flow G_f is network on same graph with modified capacity function $c'_e = c_e - f_e$. When referring set of edges of residual network, we will consider only edges with non-zero modified capacity.

2 Minimum cost of fixed value criteria

Theorem 1 Given a network, and a flow, this flow if minimum cost among flows of same value if and only if there is no cycles of negative cost in residual network with respect to this flow.

One side is obvious — if there is negative cost cycle, we can modify flow by adding some value to flow on each edge of cycle, and it cost will become less.

For the other side, consider there is no negative cost cycle, but there is flow f' of same value, with smaller cost. Consider flow $f'' = f' - f$. It is correct flow of value 0 with negative cost in G_f. We can decompose it to several cycles. At least one of them is negative, and passing through only edges with non-zero capacity in G_f.

Now we can show correct (but slow) algorithm for minimum-cost-k-flow and minimum-cost-max-flow problems.

Negative weight cycle canceling algorithm. Find any flow of such value. Then, while there is a cycle with negative cost in residual network, find it, and add flow though it.

Choosing good cycles can lead to very fast algorithms, but they are out of scope of today's lecture.

3 Searching for negative cycle

Lets use Ford-Bellman algorithm. We will add fictive vertex, with zero-cost edges to all. For each vertex we will maintain best distance from this vertex, and best previous vertex. After k iterations path would be minimal among all paths of length at most k, and some others. So, if something change after n-th iteration, there is a shortest path with at least n edges, so it must contain cycle, which would be negative. We can find it by restoring changed path.

4 Basic algorithm

Consider the network G and flow f with no negative-cost cycles in G_f. Consider shortest (with minimum sum of weights) path P from s to t in G_f. Let $f' = f + \delta P$ to be a new flow, with added flow δ on this path (it's a valid flow for δ no more than capacity of all edges on P in G_f.

Theorem 2 There is no negative-cost cycles in $G_{f'}$ for flow built as described above.

Consider there is negative cycle C. Consider flow $\frac{\delta}{2}(P + C)$. It's flow of value $\frac{\delta}{2}$ in G_f. We can decompose it in to some paths and cycles, having total cost less than $\frac{\delta}{2} \cdot (cost(P) + cost(C)) < \frac{\delta}{2} \cdot cost(P)$. But total cost of all paths is at least $\frac{\delta}{2} \cdot cost(P)$, because P is shortest one. And all cycles are non-negative, because there was no negative cycles in G_f.

Theorem 3 Shortest path in $G_{f'}$ is not less than in G_f.

Some edges disappeared, this can't help creating small paths. New edges are reversed edges for path P, which also can't help, because they are going from further vertex to closer one.

Solution for min-cost-k-flow

Solve min-cost-0-flow problem in some way. This solution is often easy to find. For example zero flow is a solution, if all weights are positive, or if there is no cycles (or at least no negative cycles) in initial network. Otherwise, we should use Negative weight cycle canceling, to find min-cost-flow of value 0. Then until flow reach needed value, find shortest path, and add flow though it.

Note, that this is solution for both min-cost-k-flow problem and min-cost-max-flow problems, as if we add flow though paths until path exists, it would be max-flow by Ford-Fulkerson theorem.

Also theorem 3 allows to build solution for min-cost-flow problem, as it says that weight of paths increases over iterations of algorithm. So if we stop when it become non-negative, it will be minimal flow overall.

5 Jonson potential

There is one problem with basic algorithm. On each stage it need to search shortest path in graph with negative edges, which can be done in $O(VE)$ time, while with no negative edges, shortest path can be found in $O(ElogE)$ time.

Consider some function $\varphi(v) : V \to R$, called potential. We will modify weight function as $w'_{(v,u)} = w_{(v,u)} + \varphi(v) - \varphi(u)$.

Theorem 4 Modified graph contains negative cycle if and only if original graph contains negative cycle.

Easy to see, that total weight of cycle was not changed.

Theorem 5 Path is shortest in modified graph if and only if it is shortest in original graph cycle.

Easy to see, that total wight of all paths from v to u changed for same value - $\varphi(v) - \varphi(u)$.

This two theorems shows, that we can get some potential, and run all our algorithm on modified graph. So the question is how to find potentials in such way, that all edges become positive.

Let's find shortest distance for all vertices on first iteration using Ford-Bellman algorithm, and set found distances as potentials. Triangle inequality shows, that all edges, present on previous step should be non-negative, as $d(u) \leqslant d(v) + w_{(v,u)}$, so $w_{(v,u)} + d(v) - d(u) \geqslant 0$. New edges, that didn't exist on previous step, are now equal to zero, because they are reversed edges on shortest path and triangle inequality becomes equality for them.

On each of next iterations, we could use Dijkstra algorithm, and add found distances to potentials. Same reasoning shows, all edges will be positive after that.