Lecture notes on "Analysis of Algorithms":

Minimum cost flow and weighted bipartite matching

Lecturer: *Uri Zwick* *

May 28, 2013

Abstract

We present strongly polynomial time algorithms for the *minimum cost flow* problem and for the *weighted bipartite matching* problem, also known as the *assignment* problem.

1 The minimum cost flow problem

A *flow network* with *costs*, *demands* and *capacities* $N = (G, a, b, c)$ is composed of a directed graph $G = (V, E)$, a *cost* function $a : E \to \mathbb{R}$, a *demand* function $b : V \to \mathbb{R}$, and a *capacity* function $c : E \to \mathbb{R}^+$. A feasible flow f in N is a function $f : E \to \mathbb{R}^+$ that satisfies the following two conditions:

$$0 \leq f(e) \leq c(e) \quad , \quad e \in E \tag{1}$$

$$f(in(v)) - f(out(v)) \;=\; b(v) \quad , \quad v \in V \tag{2}$$

The *cost* of the flow f is:

$$Cost(f) \;=\; \sum_{e \in E} a(e) f(e) \; .$$

In the above definitions, $in(v)$ is the set of edges *entering* v, and $out(v)$ is the set of edges *emanating* from v. Also if $X \subseteq E$ is a subset of edges, then $f(X) = \sum_{e \in X} f(e)$.

The minimum cost flow problem: Given a flow network $N = (G, a, b, c)$ with costs, demands and capacities, find a feasible flow, if any, of minimum cost.

It is not difficult to transform a flow network with a general demand function $b : V \to \mathbb{R}$ into a flow network with a single source and a single sink. Simply define a new source vertex s, and new sink vertex t, and add an edge $e_{s,v} = (s, v)$ with $a(e_{s,v}) = 0$ and $c(e_{s,v}) = -b(v)$, for every $v \in V$ with $b(v) < 0$, and an edge $e_{v,t} = (v, t)$ with $a(e_{v,t}) = 0$ and $c(e_{v,t}) = b(v)$, for every $v \in V$ with

*School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel. E–mail: zwick@cs.tau.ac.il

$b(v) > 0$. Let $N' = (G', a, c, s, t)$ denote the new network obtained. Then, it is easy to see that N has a feasible flow if and only if there is a feasible flow in N' that saturates all the edges emanating from s, or, equivalently, all the edges entering t. (Note that $b(V) = \sum_{v \in V} b(v) = 0$ is a necessary condition for the existence of a feasible flow.) If f is such a flow in N' then the restriction of f to N is a feasible flow in N. We therefore focus in these notes on the following version of the problem:

The minimum cost maximum flow problem: Given a flow network $N = (G, a, c, s, t)$ with costs, capacities, a source and a sink, find a *maximum* flow in N of *minimum* cost.

Note that a flow $f : E \to \mathbb{R}^+$ in a flow network $N = (G, a, c, s, t)$ with a source s and a sink t satisfies

$$f(in(v)) - f(out(v)) \;=\; 0 \quad , \quad v \in V \setminus \{s, t\} \tag{3}$$

The *value* of a flow f is defined to be

$$Val(f) \;=\; f(out(s)) \;=\; f(in(t)) \;,$$

i.e., the amount of flow shipped from s to t.

Both versions of the minimum cost flow problem are easily cast as *linear programs*. (The variables are $f(e)$, for every $e \in E$. The objective function and all the constraints are then linear.) Thus, the problem can be solved in polynomial time using existing polynomial time linear programming algorithms. The resulting algorithms would be polynomial, but not *strongly* polynomial, i.e., their running time would depend (polynomially) on the number of bits needed to represent the (integral) costs and capacities of the network, and not only on the number of vertices and edges in the network.

Our main goal is to obtain strongly polynomial algorithms for the minimum cost flow problem, and for a very interesting special case of it, the *minimum weight bipartite matching*. This later problem is also known as the *assignment* problem. (See Section 7 for a definition.)

For a more thorough treatment of the minimum cost flow problem, including more efficient weakly and strongly polynomial time algorithms for the problem, see Ahuja, Magnanti and Orlin [AMO93]. We cover only a small portion of the material contained there.

2 Flow decomposition

Let $N = (G, a, c, s, t)$ be a flow network. For every simple directed path P from s to t we define a flow $f_P : E \to \mathbb{R}^+$ as follows: If $e \in P$ then $f_P(e) = 1$, otherwise $f_P(e) = 0$. Note that f is not necessarily a feasible flow as it may violate the capacity constraint $f(e) \leq c(e)$ on edges of P. Similarly, for every directed cycle C in G we define a flow f_C.

Theorem 2.1 *Let $N = (G, a, c, s, t)$ be a flow network and let $f : E \to \mathbb{R}^+$ be a flow in G. Then, there is a collection of s-t paths $P_1, P_2, \ldots, P_k$, a collection of cycles $C_1, C_2, \ldots, C_\ell$, and constants $\alpha_1, \alpha_2, \ldots, \alpha_k > 0$ and $\beta_1, \beta_2, \ldots, \beta_\ell > 0$ such that $k + \ell \leq m = |E|$ and $f = \sum_{i=1}^{k} \alpha_i f_{P_i} + \sum_{i=1}^{\ell} \beta_i f_{C_i}$.*

Furthermore, if f is a circulation, i.e., $Val(f) = 0$, then $k = 0$, i.e., the flow can be decomposed into flows on directed cycles.

3 The residual network

Let $N = (G, a, c, s, t)$ be a flow network (with costs, capacities, a source and a sink) and let f be a flow in N. The residual network N_f is defined to be $N_f = (G_f, a_f, c_f, s, t)$, where $G_f = (V, E_f)$ and

$$E_f \;=\; \{\vec{e} \mid e \in E \text{ and } f(e) < c(e)\} \;\cup\; \{\overleftarrow{e} \mid e \in E \text{ and } f(e) > 0\}\ .$$

If $e = (u, v) \in E$, then $\vec{e} = (u, v)$ is an edge with the same endpoints, and $\overleftarrow{e} = (v, u)$ is an edge with reversed endpoints. The *residual capacity* $c_f : E_f \to \mathbb{R}^+$ is defined as follows: $c_f(\vec{e}) = c(e) - f(e)$ and $c_f(\overleftarrow{e}) = f(e)$, for every $e \in E$. The cost function $a_f : E_f \to \mathbb{R}$ is defined as follows: $a_f(\vec{e}) = a(e)$ and $a_f(\overleftarrow{e}) = -a(e)$, for every $e \in E$. (Note that the definitions of $G_f = (V, E_f)$ and $c_f : E_f \to \mathbb{R}^+$ are the same as those used when studying the maximum flow problem. As the dependence of the cost function a_f on f is minimal, we write a instead of a_f when no confusion may arise.)

Recall from our discussion of the maximum flow problem that if f is a feasible flow in a network N and g is a feasible flow in the residual network N_f, then $f + g$ is a feasible flow in the original network N of value $Val(f) + Val(g)$ and cost $Cost(f) + Cost(g)$. The flow $f + g$ is defined as follows: $(f + g)(e) = f(e) + g(\vec{e}) - g(\overleftarrow{e})$, for every $e \in E$. (If $\vec{e} \notin E_f$, we let $g(\vec{e}) = 0$. Similarly, if $\overleftarrow{e} \notin E_f$, we let $g(\overleftarrow{e}) = 0$.) (Check the details again!)

Similarly, if f, f' are two feasible flows in a network N, then $f' - f$ is a feasible flow in the residual network N_f of value $Val(f') - Val(f)$ and cost $Cost(f') - Cost(f)$. The flow $f' - f$ in N_f is defined as follows: If $\vec{e} \in E_f$ then $(f' - f)(\vec{e}) = (f'(e) - f(e))^+$, where $x^+ = \max\{x, 0\}$. Similarly, if $\overleftarrow{e} \in E_f$, then $(f' - f)(\overleftarrow{e}) = (f(e) - f'(e))^+$. (Check the details again!)

4 Augmenting paths and cycles

A directed path P from s to t in N_f is said to be an *augmenting path* for f. A directed cycle C in N_f is said to be an *augmenting cycle* for f. The *residual capacity* of an augmenting path P is defined as $c_f(P) = \min_{e \in P} c_f(e)$. Similarly, $c_f(C) = \min_{e \in C} c_f(e)$.

Lemma 4.1 *Let $N = (G, a, c, s, t)$ be a flow network and let f be a feasible flow in N. Then, f is a minimum cost flow (of value $Val(f)$) if and only if the residual network N_f contains no negative cycles with respect to the cost function a_f.*

Proof: If the residual network N_f contains a cycle C of negative cost, let g be a flow in N_f such that $g(e) = c_f(C)$, for every $e \in C$, and $g(e) = 0$, otherwise. (In other words, $g = c_f(C) f_C$.) Clearly $Val(g) = 0$ and $Cost(g) < 0$. Thus $f + g$ is a cheaper flow in N of value $Val(f)$.

Conversely, suppose that f' is a flow in N with $Val(f') = Val(f)$ and $Cost(f') < Cost(f)$. The flow $f' - f$ in the residual network has cost $Cost(f') - Cost(f) < 0$. By Theorem 2.1, the flow $f' - f$ can be decomposed into a collection of flows along directed cycles in N_f. At least one of these cycles must have a negative cost, as required. □

A *cheapest* augmenting path P in N_f is a directed path from s to t in N_f whose cost $a_f(P) = \sum_{e \in P} a_f(e)$ is minimum. A cheapest augmenting path exists if and only if the residual network does not contain negative weight cycles, with respect to the cost function a, that can be reached from s and from which t can be reached. If this condition is satisfied, then a cheapest augmenting path can be found using a shortest paths algorithm. (Note: do not confuse *shortest* augmenting paths used in maximum flow algorithms with *cheapest* augmenting paths considered here.)

Lemma 4.2 *Let $N = (G, a, c, s, t)$ be a flow network and let f be a minimum cost flow (of value $Val(f)$) in N. Let g be a flow of value δ along a cheapest path P from s to t in the residual network N_f, with respect to the cost function a_f. Then, $f + g$ is a minimum cost flow of value of value $Val(f) + \delta$ in N.*

Proof: Let f' be an arbitrary flow of value $Val(f') = Val(f) + \delta$ in N. We have to show that $Cost(f + g) \leq Cost(f')$. Consider the flow $f' - f$ in the residual network N_f. Its value is $Val(f' - f) = \delta$ and it can be represented as the sum of flows along augmenting paths and cycles in N_f. As f is a minimum cost flow, we get, by Lemma 4.1, that N_f contains no negative cost cycles. As each path in N_f is at least as costly as P, we get $Cost(f' - f) \geq a_f(P)\delta$. Thus, $Cost(f + g) = Cost(f) + a_f(P)\delta \leq Cost(f) + Cost(f' - f) = Cost(f')$, as required. □

5 The generic cycle canceling algorithm

Lemma 4.1 immediately suggests the following algorithm for finding a minimum cost maximum flow in a network $N = (G, a, c, s, t)$:

1. Find a maximum flow f in the network (G, c, s, t).
2. As long as the residual network network N_f contains a negative cycle C with respect to the cost function a_f, augment the flow along C, saturating at least one of the edges on C in the residual network.

As an augmentation along a cycle C saturates at least one edge on it, the cycle C is no longer present in the new residual graph. The cycle C was therefore 'canceled', hence the name of the algorithm. The cancelation of a negative cycle C may create, however, new negative cycles in the new residual graph.

If all the capacities $c(e)$ are integral, then the cycle canceling algorithm is guaranteed to halt. (Can you explain why? The proof is not completely obvious when the costs are not assumed to be rational.) As a corollary, we get:

Corollary 5.1 *If $N = (G, a, c, s, t)$ is a flow network with integral capacities, i.e., $c : E \to \mathbb{N}^+$, then there is an integral minimum cost maximum flow f^*.*

As is the case with the Ford-Fulkerson maximum flow algorithm, if some of the capacities are irrational, it is possible to construct pathological sequences of augmenting cycles for which the 'algorithm' would not halt. Furthermore, even if all capacities are integral, the generic cycle canceling algorithm presented above may need to perform an exponential number of augmentations before finding a minimum cost flow.

6 The cheapest augmenting path algorithm

The cycle canceling algorithm of the previous section starts with a maximum flow and gradually reduces its cost until a minimum cost maximum flow is obtained. An alternative approach, pursued in this section, is to start with a minimum cost flow of some value, say 0, and gradually increase its value using cheapest augmenting paths. By Lemma 4.2, each flow obtained by this process is of minimum cost. We end when the flow obtained is a maximum flow. The resulting algorithm is:

1. Find a minimum cost circulation f, i.e., a minimum cost flow of value 0, in the network.
2. As long the residual network N_f contains directed paths from s to t, find a cheapest such path, and augment the flow along this path by its residual capacity.

How do we find a minimum cost circulation? In general, this is as difficult as the general minimum cost flow problem. (Why?) But in certain cases, a minimum cost circulation is easily obtained. If the flow network $N = (G, a, c, s, t)$ contains no negative cycles with respect to a, then the zero flow $f(e) = 0$, for every $e \in E$, is a minimum cost circulation. This is the case, for example, if the graph $G = (V, E)$ is *acyclic*. We will encounter this case in the next section. (Note that a residual network may contain cycles even if the original network is acyclic.)

The cheapest augmenting path algorithm is again guaranteed to halt only if all the capacities are integral. It is easy to see that if all capacities are in $\{1, 2, \ldots, C\}$, then the algorithm will halt after at most nC augmentations.

How do we find cheapest augmenting paths in the residual network $N_f = (G_f, a_f, c_f, s, t)$? As some of the edge costs $a_f(e)$ may be negative, it seems that the costly Bellman-Ford algorithm, whose complexity is $O(mn)$, should be used. We show, however, that a more efficient solution is possible.

Let $\pi : V \to \mathbb{R}$ a potential function defined on the vertices of the network. The *adjusted* cost of an edge $e = (u, v)$ is defined, as before, to be $a_\pi(e) = a_f(e) + \pi(u) - \pi(v)$. (We write a_π instead of $a_{f,\pi}$, when f is clear from the context.)

Let f be a minimum cost flow. By Lemma 4.1, we get that the residual graph G_f contains no negative cycle with respect to the cost function a_f. We showed in the lecture on shortest paths that

there is therefore a potential function π for which $a_\pi(e) \geq 0$, for every $e \in E_f$. We say that such a potential function π is a *feasible* potential function for the residual network N_f.

Suppose that a feasible potential function for the residual network N_f is known to us. As an s-t path is a cheapest path with respect to a_f if and only if it is a cheapest path with respect to a_π, we can use Dijkstra's algorithm to find a cheapest augmenting path in the residual graph G_f. Dijkstra's algorithm would also give us the distances (or costs) $\delta_\pi(s, v)$ from v to all the other vertices in G_f with respect to the cost function a_π. We now claim the following:

Lemma 6.1 *Let $N = (G, a, c, s, t)$ be a flow network with costs, let f be a minimum cost flow in N, let π be a potential function for which $a_\pi(e) \geq 0$, for every $e \in E_f$. For every $v \in V$, let $\delta_\pi(v) = \delta_\pi(s, v)$ be the distance from s to v in G_f with respect to a_π. Let f' be a flow obtained from f using a cheapest augmenting path P in G_f. Then, $\pi' = \pi + \delta_\pi$ is a potential function for which $a_{\pi'}(e) \geq 0$, for every $e \in E_{f'}$.*

Proof: If $e = (u, v)$, then $a_{\pi'}(e) = a_\pi(e) + \delta_\pi(u) - \delta_\pi(v)$. If $e \in E_f$, then by the triangle inequality we have $a_{\pi'}(e) \geq 0$, as required. It remains to consider, therefore, edges $e \in E_{f'} - E_f$, i.e., edges of the residual network $N_{f'}$ that were not present in the previous residual network N_f. An edge $e = (u, v)$ enters the residual network $N_{f'}$ only if its reverse $\overleftarrow{e} = (v, u)$ participates in the augmenting path P. As $\overleftarrow{e}$ appears on a shortest path in G_f, we get that $a_{\pi'}(\overleftarrow{e}) = a_\pi(\overleftarrow{e}) + \delta_\pi(v) - \delta_\pi(u) = 0$. Thus, $a_{\pi'}(e) = -a_{\pi'}(\overleftarrow{e}) = 0$, as required. □

Thus, given a feasible potential function for the initial residual network we can maintain feasible potential functions for all subsequent residual networks obtained by the algorithm. All cheapest augmenting paths can therefore be found using Dijkstra's algorithm, whose running time is $O(m + n \log n)$, instead of the much slower Bellman-Ford algorithm. We thus have:

Theorem 6.2 *Let $N = (G, a, c, s, t)$ be a flow network that contains no negative cycles with respect to a. If $c : E \to \{1, 2, \ldots, C\}$, then an integral minimum cost maximum flow in N can be found in $O((m + n \log n)nC)$ time.*

Proof: As the initial network contains no negative cycles with respect to a, the zero flow $f(e) = 0$, for every $e \in E$, is a minimum cost circulation in N. As all the capacities are integral, each augmentation increases the value of the flow by at least 1. Hence, a maximum flow of minimum cost would be found after at most nC augmentations. □

7 Weighted bipartite matching

Let $G = (U, V, E)$ be a bipartite graph, with $|U| = |V| = n$ and $|E| = m$, and let $w : E \to \mathbb{R}$ be a weight function defined on its edges. In the *minimum cost perfect bipartite matching problem* we

are asked to find a perfect matching $M \subseteq E$ whose cost $w(M) = \sum_{e \in M} w(e)$ is minimized. This problem is also known as the *assignment problem*.

We can easily reduce the assignment problem to a minimum cost flow problem as follows. Add a source vertex s and edges $e_{s,u} = (s, u)$, for every $u \in U$, and a sink vertex t and edges $e_{v,t} = (v, t)$, for every $v \in V$. The capacity of all the edges in the new network is 1. The cost $a(e)$ of an edge $e \in E$ is defined to be $a(e) = w(e)$, while the cost of all the new edges is defined to be 0.

It is easy to see that every perfect matching $M \subseteq E$ defines an integral maximum flow in the new network of cost $w(M)$. Similarly, every integral maximum flow f in the new network corresponds to a maximum matching whose weight is $Cost(f)$. As the resulting flow network is acyclic, and as $C = 1$, Theorem 6.2 immediately implies the following result:

Theorem 7.1 *The assignment problem, i.e., the minimum (or maximum) weight perfect bipartite matching problem, can be solved in $O(mn + n^2 \log n)$ time.*

The above mentioned algorithm for the assignment problem actually finds a minimum weight matching of cardinality k, for every $k = 1, 2, \ldots, n$. The algorithm presented is the fastest known strongly polynomial time algorithm for the assignment problem.

8 The minimum mean cost cycle canceling algorithm

We set out to find a strongly polynomial time algorithm for the minimum cost flow problem. We presented a strongly polynomial time algorithm for the assignment problem, but both the algorithms that we presented for the general minimum cost flow problem are not even guaranteed to halt when the capacities are not integral (or rational). Furthermore, even if the capacities are integral, the algorithms that we presented are not guaranteed to run in (weakly) polynomial time. (Note that the $O((m + n \log n)nC)$ running time of the cheapest augmenting path algorithm is not polynomial as it depends linearly on C. Furthermore, the algorithm only works when the input network does not contain negative cycles with respect to the cost function a.)

In this section we obtain a (somewhat slow) *strongly polynomial* time algorithm for the minimum cost flow problem. The algorithm, developed by Goldberg and Tarjan [GT89] is quite simple: Start with some maximum flow in the network $N = (G, a, c, s, t)$. As long as the residual network $N_f = (G, a_f, c_f, s, t)$ contains negative cycles, augment the flow along a cycle of minimum *mean cost*. The algorithm is therefore an instantiation of the generic algorithm of Section 5.

If $C = \langle e_1, e_2, \ldots, e_\ell \rangle$ is a cycle in the residual network $N_f = (G_f, a_f, c_f, s, t)$, its mean cost is defined, as before, to be $a_f(C)/|C| = (\sum_{i=1}^{\ell} a_f(e_i))/\ell$. We saw in one of the previous lectures that a minimum mean cost cycle can be found in $O(mn)$ time using an algorithm of Karp [Kar78].

In the remainder of this section we prove that the minimum mean cost cycle canceling algorithm is a strongly polynomial time algorithm for the minimum cost maximum flow problem:

Theorem 8.1 *The minimum mean cost cycle canceling algorithm finds a minimum cost maximum flow after at most $O(m^2 n \log n)$ augmentations. Its total running time, therefore, is $O(m^3 n^2 \log n)$.*

Theorem 8.1 would follow from a series of claims that we present below.

Let f be a maximum flow in the network $N = (G, a, c, s, t)$. Let $-\mu(f)$ be the minimum mean cost of a cycle in the residual network $N_f = (G, a, c_f, s, t)$. If $\mu(f) \leq 0$, then f is a minimum cost flow (by Lemma 4.1) and we are done. Recall from the lecture on shortest paths that there is a potential function $\pi : V \to \mathbb{R}$ for which $a_\pi(e) \geq -\mu(f)$, for every $e \in E_f$.

Let $f_0, f_1, \ldots, f_\ell$ be the (maximum) flows obtained throughout the operation of the algorithm. Let $\pi_0, \pi_1, \ldots, \pi_\ell$ be the corresponding potential functions for which $a_{\pi_i}(e) \geq -\mu(f_i)$, for every $e \in E_{f_i}$. Let C_i be the cycle of minimum mean cost in E_{f_i} that was used to obtain f_{i+1}.

Lemma 8.2 *$\mu(f_{i+1}) \leq \mu(f_i)$, for $i = 0, 1, \ldots, \ell - 1$.*

Proof: Recall that $a_{\pi_i}(e) \geq -\mu(f_i)$, for every $e \in E_{f_i}$. As the mean cost of the cycle C_i is $-\mu(f_i)$, we have $a_{\pi_i}(e) = -\mu(f_i)$ for every $e \in C_i$. We show that for every $e \in E_{f_{i+1}}$ we still have $a_{\pi_i}(e) \geq -\mu(f_i)$. This would imply that $\mu(f_{i+1}) \leq \mu(f_i)$. (Note that we are using here the potential function π_i for the edges of $E_{f_{i+1}}$.) If $e \in E_{f_i}$, then this clearly holds, as $a_{\pi_i}(e)$ did not change. If $e \in E_{f_{i+1}} - E_{f_i}$, then $\overleftarrow{e}$ is an edge on C_i and therefore $a_{\pi_i}(\overleftarrow{e}) = -\mu(f_i)$. Thus, $a_{\pi_i}(e) = -a_{\pi_i}(\overleftarrow{e}) = \mu(f_i) \geq 0 \geq -\mu(f_i)$, as required. □

Lemma 8.3 *$\mu(f_{i+m}) \leq (1 - \frac{1}{n})\,\mu(f_i)$, for every $i = 1, 2, \ldots, \ell - m$.*

Proof: Let π_i be a potential function for which $a_{\pi_i}(e) \geq -\mu(f_i)$, for every $e \in E_{f_i}$. Let $C_i, C_{i+1}, \ldots, C_{i+m-1}$ be the cycles along which the flow was augmented while moving from f_i to f_{i+m}. We first show that if all the edges on all the cycles $C_i, C_{i+1}, \ldots, C_{i+m-1}$ have negative cost, with respect to a_{π_i}, then $\mu(f_{i+m}) = 0$ and the claim of the lemma holds. (It would also follow that f_{i+m} is a minimum cost flow.)

Suppose that all the edges on the cycles $C_i, C_{i+1}, \ldots, C_{i+m-1}$ have strictly negative costs with respect to a_{π_i}. Let $i \leq j < i + m$. The flow augmentation along C_j saturates at least one edge $e \in C_j$ and removes it from the residual network. In addition, a reverse edge $\overleftarrow{e}$ is added to $E_{f_{j+1}}$, if it is not already present in E_{f_j}, for every $e \in C_j$. If $a_{\pi_i}(e) < 0$, for every $e \in C_j$, then for every edge $\overleftarrow{e}$ added to $E_{f_{j+1}}$ we have $a_{\pi_i}(\overleftarrow{e}) = -a_{\pi_i}(e) > 0$. Thus, the number of edges $e \in E_{f_{j+1}}$ with negative adjusted cost $a_{\pi_i}(e)$ decreases by at least 1. As there are at most m edges $e \in E_{f_i}$ with $a_{\pi_i}(e) < 0$, it follows that for every $e \in E_{f_{i+m}}$ we have $a_{\pi_i}(e) \geq 0$. Thus $\mu(f_{i+m}) \geq 0$, as claimed.

It remains to consider the case in which at least one of the cycles $C_i, C_{i+1}, \ldots, C_{i+m-1}$ contains an edge e with a non-negative adjusted cost $a_{\pi_i}(e) \geq 0$. Let $i \leq j < i+m$ be the index of the *first* such cycle and let $e' \in C_j$ have $a_{\pi_i}(e') \geq 0$. As $a_{\pi_i}(e') \geq 0$ and $a_{\pi_i}(e) \geq -\mu(f_i)$, for every $e \in C_j - \{e'\}$, we get that

$$a(C_j) \;=\; \sum_{e \in C_j} a(e) \;=\; \sum_{e \in C_j} a_{\pi_i}(e) \;\geq\; -(|C_j| - 1)\,\mu(f_i)\,.$$

The mean cost of C_j, which equals $-\mu(f_j)$, therefore satisfies

$$-\mu(f_j) \;=\; \frac{a(C_j)}{|C_j|} \;\geq\; -\frac{(|C_j|-1)\,\mu(f_i)}{|C_j|} \;\geq\; -(1-\frac{1}{n})\,\mu(f_i)\;.$$

By Lemma 8.2 we thus have $\mu(f_{i+m}) \leq \mu(f_j) \leq (1-\frac{1}{n})\,\mu(f_i)$, as required. □

Using Lemmas 8.2 and 8.3 we obtain:

Corollary 8.4 *If $a : E \to \mathbb{Z}$ and $-A \leq a(e) \leq A$, for every $e \in E$, then the minimum mean cost cycle canceling algorithm performs only $O(mn \log(nA))$ cycle cancelations.*

Proof: Let f_0 be the initial flow and let $f_{\ell-1}$ be the next to last flow obtained by the algorithm. Then, $\mu(f_0) \leq A$ and $\mu(f_{\ell-1}) \geq \frac{1}{n}$. As every m consecutive iterations reduce $\mu(f_i)$ by a factor of at least $(1-\frac{1}{n})$, the claim follows. □

The next lemma shows that when the absolute value of adjusted cost $a_{\pi_i}(e)$ of an edge e becomes sufficiently large with respect to $\mu(f_i)$, the flow on the edge e remains *fixed* throughout the remaining iterations of the algorithm. (The flow on e would in fact be fixed to either 0 or $c(e)$.)

Lemma 8.5 *If $|a_{\pi_i}(e)| \geq 2n\,\mu(f_i)$, for some $e \in E$ and $0 \leq i \leq \ell$, then the flow on e does not change after the i-th augmentation. (In other words $f_i(e) = f_{i+1}(e) = \cdots = f_\ell(e)$.)*

Proof: Assume at first that $a_{\pi_i}(e) \geq 2n\,\mu(f_i)$. If $f_i(e) > 0$, then $\overleftarrow{e} \in E_{f_i}$ and $a_{\pi_i}(\overleftarrow{e}) \leq -2n\,\mu(f_i)$, contradicting the assumption that $a_{\pi_i}(e) \geq -\mu(f_i)$, for every $e \in E_{f_i}$. Thus $f_i(e) = 0$.

Suppose now, for the sake of contradiction, that $f_j(e) > 0$, for some $i < j \leq \ell$. The flow $f_j - f_i$ in the residual network N_{f_i} can be decomposed into flows along directed cycles of N_{f_i}. At least one of these cycles C must contain e. As $a_{\pi_i}(e) \geq 2n\,\mu(f_i)$ and $a_{\pi_i}(e') \geq -\mu(f_i)$ for any other edge $e' \in C$, we get that

$$\frac{a(C)}{|C|} \;\geq\; \frac{2n\,\mu(f_i) - (|C|-1)\mu(f_i)}{|C|} \;>\; \mu(f_i)\;.$$

It is not difficult to check that $\overleftarrow{C}$, the cycle obtained by reversing the direction of all the edges in C, is a directed cycle in the residual network N_{f_j}. Now

$$-\mu(f_j) \;\leq\; \frac{a(\overleftarrow{C})}{|\,\overleftarrow{C}\,|} \;=\; -\frac{a(C)}{|C|} \;<\; -\mu(f_i)\;.$$

This, however, contradicts Lemma 8.2. It follows that $f_j(e) = 0$, for every $i \leq j \leq \ell$.

The proof for the case $a_{\pi_i}(e) \leq -2n\,\mu(f_i)$ is similar and is left as an exercise. □

We are now in a position to prove Theorem 8.1.

Proof of Theorem 8.1: We show that every $\lceil mn \ln(2n) \rceil$ consecutive augmentations performed by the algorithm fix the flow on at least one new edge. As the number of edges fixed during the operation of the algorithm is at most m, the claim of the theorem follows.

Let $k = \lceil mn \ln(2n) \rceil$ and let $0 \leq i \leq \ell - k$. By Lemma 8.3 we get that $\mu(f_{i+k}) \leq \frac{\mu(f_i)}{2n}$. Let C_i be the cycle canceled in N_{f_i}. Let π_{i+k} be a potential function for which $a_{\pi_{i+k}}(e) \geq -\mu(f_{i+k})$, for every $e \in E_{f_{i+k}}$. By the definitions of $\mu(f_i)$ and C_i we get that

$$\frac{a(C_i)}{|C_i|} \;=\; \frac{a_{\pi_{i+k}}(C_i)}{|C_i|} \;=\; -\mu(f_i) \;\leq\; -2n\mu(f_{i+k}) \;.$$

Thus, there is at least one edge $e \in C_i$ for which $a_{\pi_{i+k}}(e) \leq -2n\,\mu(f_{i+k})$. Let $\bar{e} \in E$ be the edge in the original network that corresponds to e. As $\bar{e}$ is either e itself or $\overleftarrow{e}$, we get that $|a_{\pi_{i+k}}(\bar{e})| \geq 2n\,\mu(f_{i+k})$. By Lemma 8.5, the flow on $\bar{e}$ would not change after the $(i+k)$-th augmentation. As the flow on $\bar{e}$ does change in the i-th iteration, at least one *new* edge is fixed while moving from f_i to f_{i+k}, as required. □

Exercises

Exercise 1 Show that a flow $f : E \to \mathbb{R}^+$ is a minimum cost flow in a flow network $G = (V, E)$ with cost function $a : E \to \mathbb{R}$ and capacity function $c : E \to \mathbb{R}^+$ if and only if there exists a potential function $\pi : V \to \mathbb{R}$ such that

1. if $a_\pi(e) > 0$, then $f(e) = 0$,
2. if $0 < f(e) < c(e)$, then $a_\pi(e) = 0$,
3. if $a_\pi(e) < 0$, then $f(e) = c(e)$,

where $a_\pi(e) = a(e) + \pi(u) - \pi(v)$, for every $e = (u, v) \in E$. (Hint: We have shown in class that f is a minimum cost flow if and only if there are no negative cycles in the *residual* network.)

Exercise 2 A *circulation* is a flow of value 0, i.e., a flow in which the conservation constraints are satisfied at all vertices. Describe a simple reduction from the problem of finding maximum flow to the problem of finding minimum cost circulation.

Exercise 3 An instance of the *uncapacitated* minimum cost flow problem with demands is specified by giving a graph $G = (V, E)$, a cost function $a : E \to \mathbb{R}$ and demand function $b : V \to \mathbb{R}$. The capacity of each edge is *infinite*. Describe a reduction from the standard capacitated minimum cost flow problem, with a source and a sink, to the uncapacitated version of the minimum cost flow problem with demands. (Hint: For every edge $e = (u, v) \in E$, introduce a new *vertex* $\bar{e}$. Replace the edge e by the two edges $(u, \bar{e})$ and $(v, \bar{e})$. Define the costs of these edges appropriately, and adjust the demands at u, v and $\bar{e}$.)

Exercise 4 (The *hopping airplane problem*, taken from Ahuja *et al.* [AMO93].) An airplane is scheduled to sequentially visit n cities. For every $1 \leq i < j \leq n$, there are b_{ij} passengers that would like to fly from city i to city j. Each one of these passengers is willing to pay f_{ij} for the trip. The airplane can only carry p passengers at any given time. Describe a polynomial time algorithm for deciding which passengers to admit so as to maximize the total fare collected. (Hint: Use a reduction to minimum cost flow.)

Exercise 5 (a) Show that the generic cycle canceling algorithm is *not* a polynomial time algorithm.
(b) Show that the cheapest augmenting path algorithm is *not* a polynomial time algorithm.
Recall that an algorithm is polynomial if its running time is polynomial in the size of the network and in the size of the binary encoding of the capacities and costs, which are all assumed here to be integral. (Hint: In both cases, use an appropriate modification of the examples used to show that the Ford-Fulkerson max flow algorithm is not a polynomial time algorithm.)

Exercise 7 We have shown that a flow f is a minimum cost flow if and only if the residual network G_f contains no negative cycles. If the residual network G_f contains a negative cycle C, then 'canceling' this cycle reduces the cost of the flow by $-a_f(C)c_f(C)$, where $a_f(C) = \sum_{e \in C} a_f(e)$ and $c_f(C) = \min_{e \in C} c_f(e)$. (Note that $a_f(C) < 0$.) We refer to $a_f(C)c_f(C)$ as the *improvement* obtained by using C. Show that if we had a way of finding a cycle of maximum improvement, then we could have found a minimum cost flow using at most $O(m \log(mAC))$ negative cycle cancelations, where $A = \max_{e \in E} |a(e)|$ and $C = \max_{e \in E} c(e)$. (Hint: Use the flow decomposition theorem applied to $f - f^*$, where f^* is a minimum cost flow.)

Exercise 8 Let f be an optimal solution to a minimum cost flow problem $N = (G, a, b, c)$, where $a, b, c : E \to \mathbb{Z}^+$. (I.e., all costs, demands and capacities are integral.) Let $e \in E$.
(a) Suppose that we increase or decrease $c(e)$, the capacity of e, by one unit. How efficiently can a new optimal solution be found?
(b) Suppose that we increase or decrease $a(e)$, the cost of e, by one unit. How efficiently can a new optimal solution be found?

References

[AMO93] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network flows – Theory, algorithms and applications*. Prentice Hall, 1993.

[GT89] A.V. Goldberg and R.E. Tarjan. Finding minimum-cost circulations by canceling negative cycles. *Journal of the ACM*, 36(4):873–886, 1989.

[Kar78] R.M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23(3):309–311, 1978.