Chapter 19

Network Flow V - Min-cost flow

By Sariel Har-Peled, December 30, 2014[①] Version: 1.04

19.1. Minimum Average Cost Cycle

Let $\mathsf{G} = (V, E)$ be a ***digraph*** (i.e., a directed graph) with n vertices and m edges, and $\omega : E \to \mathbb{R}$ be a weight function on the edges. A ***directed cycle*** is closed walk $\mathsf{C} = (v_0, v_1, \ldots, v_t)$, where $v_t = v_0$ and $(v_i \to v_{i+1}) \in E$, for $i = 0, \ldots, t-1$. The ***average cost of a directed cycle*** is $\text{AvgCost}(\mathsf{C}) = \omega(C)/t = \left(\sum_{e \in C} \omega(e)\right)/t$.

For each $k = 0, 1, \ldots$, and $v \in V$, let $d_k(v)$ denote the minimum length of a walk with exactly k edges, ending at v (note, that the walk can start anywhere). So, for each v, we have

$$d_0(v) = 0 \quad \text{and} \quad d_{k+1}(v) = \min_{e=(u \to v) \in E} \Big(d_k(u) + \omega(e) \Big).$$

Thus, we can compute $d_i(v)$, for $i = 0, \ldots, n$ and $v \in V(G)$ in $O(nm)$ time using dynamic programming.

Let

$$\text{MinAvgCostCycle}(\mathsf{G}) = \min_{\mathsf{C} \text{ is a cycle in } \mathsf{G}} \text{AvgCost}(\mathsf{C})$$

denote the average cost of the ***minimum average cost cycle*** in G.

The following theorem is somewhat surprising.

Theorem 19.1.1. *The minimum average cost of a directed cycle in* G *is equal to*

$$\alpha = \min_{v \in V} \max_{k=0}^{n-1} \frac{d_n(v) - d_k(v)}{n-k}.$$

Namely, $\alpha = \text{MinAvgCostCycle}(\mathsf{G})$.

Proof: Note, that adding a quantity r to the weight of every edge of G increases the average cost of a cycle $\text{AvgCost}(\mathsf{C})$ by r. Similarly, α would also increase by r. In particular, we can assume that the price of the minimum average cost cycle is zero. This implies that now all cycles have non-negative (average) cost.

Thus, from this point on we assume that $\text{MinAvgCostCycle}(\mathsf{G}) = 0$, and we prove that $\alpha = 0$ in this case. This in turn would imply the theorem – indeed, given a graph where $\text{MinAvgCostCycle}(\mathsf{G}) \neq 0$, then we will shift the costs the edges so that it is zero, use the proof below, and then shift it back.

MinAvgCostCycle(G) = 0 $\implies$ $\alpha \geq 0$: We can rewrite α as $\alpha = \min_{u \in V} \beta(u)$, where

$$\beta(u) = \max_{k=0}^{n-1} \frac{d_n(u) - d_k(u)}{n-k}.$$

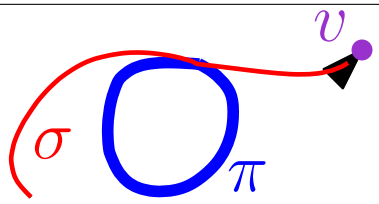


Figure 19.1: Decomposing P_n into a path σ and a cycle π.

Assume, that α is realized by a vertex v; that is $\alpha = \beta(v)$. Let P_n be a walk with n edges ending at v, of length $d_n(v)$. Since there are n vertices in G, it must be that P_n must contain a cycle. So, let us decompose P_n into a cycle π of length $n-k$ and a path σ of length k (k depends on the length of the cycle in P_n). We have that

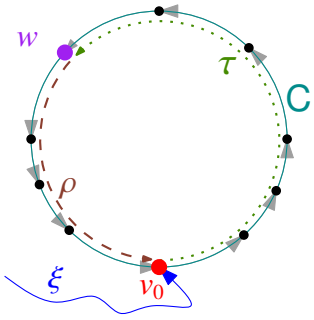
$$d_n(v) = \omega(P_n) = \omega(\pi) + \omega(\sigma) \geq \omega(\sigma) \geq d_k(v),$$

since $\omega(\pi) \geq 0$ as π is a cycle (and we assumed that all cycles have zero or positive cost). As such, we have $d_n(v) - d_k(v) \geq 0$. As such, $\frac{d_n(v)-d_k(v)}{n-k} \geq 0$. Let

$$\beta(v) = \max_{j=0}^{n-1} \frac{d_n(v) - d_j(v)}{n-j} \geq \frac{d_n(v) - d_k(v)}{n-k} \geq 0.$$

Now, $\alpha = \beta(v) \geq 0$, by the choice of v.

MinAvgCostCycle(G) = 0 $\implies$ $\alpha \leq 0$: Let $\mathsf{C} = (v_0, v_1, \ldots, v_t)$ be the directed cycle of weight 0 in the graph. Observe, that $\min_{j=0}^{\infty} d_j(v_0)$ must be realized (for the first time) by an index $r < n$, since if it is longer, we can always shorten it by removing cycles and improve its price (since cycles have non-negative price). Let ξ denote this walk of length r ending at v_0. Let w be a vertex on C reached by walking $n-r$ edges on C starting from v_0, and let τ denote this walk (i.e., $|\tau| = n-r$). We have that



$$d_n(w) \leq \omega\Big(\xi \parallel \tau\Big) = d_r(v_0) + \omega(\tau)\,, \tag{19.1}$$

where $\xi \parallel \tau$ denotes the path formed by concatenating the path τ to ξ.

Similarly, let ρ be the walk formed by walking on C from w all the way back to v_0. Note that $\tau \parallel \rho$ goes around C several times, and as such, $\omega(\tau \parallel \rho) = 0$, as $\omega(\mathsf{C}) = 0$. Next, for any k, since the shortest path with k edges arriving to w can be extended to a path that arrives to v_0, by concatenating ρ to it, we have that

$$d_k(w) + \omega(\rho) \geq d_{k+|\rho|}(v_0) \geq d_r(v_0) \geq d_n(w) - \omega(\tau)\,,$$

by Eq. (19.1). Rearranging, we have that $\omega(\rho) \geq d_n(w) - \omega(\tau) - d_k(w)$. Namely, we have

$$\forall k \qquad 0 = \omega(\tau \parallel \rho) = \omega(\rho) + \omega(\tau) \geq \Big(d_n(w) - \omega(\tau) - d_k(w)\Big) + \omega(\tau) = d_n(w) - d_k(w).$$

$$\implies \qquad \forall k \qquad \frac{d_n(w) - d_k(w)}{n-k} \leq 0$$

$$\implies \qquad \beta(w) = \max_{k=0}^{n-1} \frac{d_n(w) - d_k(w)}{n-k} \leq 0.$$

As such, $\alpha = \min_{v \in V(\mathsf{G})} \beta(v) \leq \beta(w) \leq 0$, and we conclude that $\alpha = 0$. ∎

Finding the minimum average cost cycle is now not too hard. We compute the vertex v that realizes α in Theorem 19.1.1. Next, we add $-\alpha$ to all the edges in the graph. We now know that we are looking for a cycle with price 0. We update the values $d_i(v)$ to agree with the new weights of the edges.

Now, v is the vertex realizing the quantity $0 = \alpha = \min_{u \in V} \max_{k=0}^{n-1} \frac{d_n(u) - d_k(u)}{n-k}$. Namely, we have that for the vertex v it holds

$$\max_{k=0}^{n-1} \frac{d_n(v) - d_k(v)}{n-k} = 0 \implies \forall k \in \{0, \ldots, n-1\} \quad \frac{d_n(v) - d_k(v)}{n-k} \leq 0$$
$$\implies \forall k \in \{0, \ldots, n-1\} \quad d_n(v) - d_k(v) \leq 0.$$

This implies that $d_n(v) \leq d_i(v)$, for all i. Now, we repeat the proof of Theorem 19.1.1. Let P_n be the path with n edges realizing $d_n(v)$. We decompose it into a path π of length k and a cycle τ. We know that $\omega(\tau) \geq 0$ (since all cycles have non-negative weights now). Now, $\omega(\pi) \geq d_k(v)$. As such, $\omega(\tau) = d_n(v) - \omega(\pi) \leq d_n(v) - d_k(v) \leq 0$, as π is a path of length k ending at v, and its cost is $\geq d_k(v)$. Namely, the cycle τ has $\omega(\tau) \leq 0$, and it the required cycle and computing it required $O(nm)$ time.

Note, that the above reweighting in fact was not necessary. All we have to do is to compute the node realizing α, extract P_n, and compute the cycle in P_n, and we are guaranteed by the above argumentation, that this is the cheapest average cycle.

Corollary 19.1.2. *Given a direct graph G with n vertices and m edges, and a weight function $\omega(\cdot)$ on the edges, one can compute the cycle with minimum average cost in $O(nm)$ time.*

19.2. Potentials

In general computing the shortest path in a graph that have negative weights is harder than just using the Dijkstra algorithm (that works only for graphs with non-negative weights on its edges). One can use Bellman-Ford algorithm[2] in this case, but it considerably slower (i.e., it takes $O(mn)$ time). We next present a case where one can still use Dijkstra algorithm, with slight modifications.

The following is only required in the analysis of the minimum-cost flow algorithm we present later in this chapter. We describe it here in full detail since its simple and interesting.

For a directed graph $G = (V, E)$ with weight $w(\cdot)$ on the edges, let $\mathbf{d}_\omega(s, t)$ denote the length of the shortest path between s and t in G under the weight function w. Note, that w might assign negative weights to edges in G.

A ***potential*** $p(\cdot)$ is a function that assigns a real value to each vertex of G, such that if $e = (u \to v) \in G$ then $w(e) \geq p(v) - p(u)$.

Lemma 19.2.1. *(i) There exists a potential $p(\cdot)$ for G if and only if G has no negative cycles (with respect to $w(\cdot)$).*

(ii) Given a potential function $p(\cdot)$, for an edge $e = (u \to v) \in E(G)$, let $\ell(e) = w(e) - p(v) + p(u)$. Then $\ell(\cdot)$ is non-negative for the edges in the graph and for any pair of vertices $s, t \in V(G)$, we have that the shortest path π realizing $\mathbf{d}_\ell(s, t)$ also realizes $\mathbf{d}_\omega(s, t)$.

(iii) Given G and a potential function $p(\cdot)$, one can compute the shortest path from s to all the vertices of G in $O(n \log n + m)$ time, where G has n vertices and m edges

Proof: (i) Consider a cycle C, and assume there is a potential $p(\cdot)$ for G, and observe that

$$w(\mathsf{C}) = \sum_{(u \to v) \in E(\mathsf{C})} w(e) \geq \sum_{(u \to v) \in E(\mathsf{C})} (p(v) - p(u)) = 0,$$

[2] http://en.wikipedia.org/wiki/Bellman-Ford_algorithm

as required.

For a vertex $v \in V(G)$, let $p(v)$ denote the shortest walk that ends at v in G. We claim that $p(v)$ is a potential. Since G does not have negative cycles, the quantity $p(v)$ is well defined. Observe that $p(v) \leq p(u) + w(u \to v)$ since we can always continue a walk to u into v by traversing $(u \to v)$. Thus, $p(v) - p(u) \leq w(u \to v)$, as required.

(ii) Since $\ell(e) = w(e) - p(v) + p(u)$ we have that $w(e) \geq p(v) - p(u)$ since $p(\cdot)$ is a potential function. As such $w(e) - p(v) + p(u) \geq 0$, as required.

As for the other claim, observe that for any path π in G starting at s and ending at t we have that

$$\ell(\pi) = \sum_{e=(u\to v)\in\pi} (w(e) - p(v) + p(u)) = w(\pi) + p(s) - p(t),$$

which implies that $\mathbf{d}_\ell(s,t) = \mathbf{d}_\omega(s,t) + p(s) - p(t)$. Implying the claim.

(iii) Just use the Dijkstra algorithm on the distances defined by $\ell(\cdot)$. The shortest paths are preserved under this distance by (ii), and this distance function is always positive. ∎

19.3. Minimum cost flow

Given a network flow $\mathsf{G} = (V, E)$ with source s and sink t, capacities $\mathsf{c}(\cdot)$ on the edges, a real number ϕ, and a cost function $\kappa(\cdot)$ on the edges. The ***cost*** of a flow f is defined to be

$$\text{cost}(\mathsf{f}) = \sum_{e\in E} \kappa(e) * \mathsf{f}(e).$$

The ***minimum-cost s-t flow problem*** ask to find the flow f that minimizes the cost and has value ϕ.

It would be easier to look on the problem of ***minimum-cost circulation problem***. Here, we are given instead of ϕ a lower-bound $\ell(\cdot)$ on the flow on every edge (and the regular upper bound $\mathsf{c}(\cdot)$ on the capacities of the edges). All the flow coming into a node must leave this node. It is easy to verify that if we can solve the minimum-cost circulation problem, then we can solve the min-cost flow problem. Thus, we will concentrate on the min-cost circulation problem.

An important technicality is that all the circulations we discuss here have zero demands on the vertices. As such, a circulation can be conceptually considered to be a flow going around in cycles in the graph without ever stopping. In particular, for these circulations, the conservation of flow property should hold for all the vertices in the graph.

The ***residual graph*** of f is the graph $\mathsf{G}_\mathsf{f} = (V, E_\mathsf{f})$ where

$$E_\mathsf{f} = \left\{ e = (u \to v) \in V \times V \;\middle|\; \mathsf{f}(e) < \mathsf{c}(e) \text{ or } \mathsf{f}\left(e^{-1}\right) > \ell\left(e^{-1}\right) \right\}.$$

where $e^{-1} = (v \to u)$ if $e = (u \to v)$. Note, that the definition of the residual network takes into account the lower-bound on the capacity of the edges.

Assumption 19.3.1. To simplify the exposition, we will assume that if $(u \to v) \in E(\mathsf{G})$ then $(v \to u) \notin E(G)$, for all $u, v \in V(G)$. This can be easily enforced by introducing a vertex in the middle of every edge of G. This is acceptable, since we are more concerned with solving the problem at hand in polynomial time, than the exact complexity. Note, that our discussion can be extended to handle the slightly more general case, with a bit of care.

We extend the cost function to be anti-symmetric; namely,

$$\forall\, (u \to v) \in E_{\mathsf{f}} \qquad \kappa\Big((u \to v)\Big) = -\kappa\Big((v \to u)\Big).$$

Consider a directed cycle C in G_{f}. For an edge $e = (u \to v) \in E$, we define

$$\chi_C(e) = \begin{cases} 1 & e \in C \\ -1 & e^{-1} = (v \to u) \in C \\ 0 & \text{otherwise;} \end{cases}$$

that is, we pay 1 if e is in C and -1 if we travel e in the "wrong" direction.

The ***cost*** of a directed cycle C in G_{f} is defined as

$$\kappa(\mathsf{C}) = \sum_{e \in \mathsf{C}} \kappa(e).$$

We will refer to a circulation that comply with the capacity and lower-bounds constraints as being ***valid***. A function that just comply with the conservation property (i.e., all incoming flow into a vertex leaves it), is a ***weak circulation***. In particular, a weak circulation might not comply with the capacity and lower bounds constraints of the given instance, and as such is not a valid circulation.

We need the following easy technical lemmas.

Lemma 19.3.2. *Let* f *and* g *be two valid circulations in* $\mathsf{G} = (V, E)$. *Consider the function* $\mathsf{h} = \mathsf{g} - \mathsf{f}$. *Then,* h *is a weak circulation, and if* $\mathsf{h}(u \to v) > 0$ *then the edge* $(u \to v) \in \mathsf{G}_{\mathsf{f}}$.

Proof: The fact that h is a circulation is trivial, as it is the difference between two circulations, and as such the same amount of flow that comes into a vertex leaves it, and thus it is a circulation. (Note, that h might not be a valid circulation, since it might not comply with the lower-bounds on the edges.)

Observe, that if $\mathsf{h}(u \to v)$ is negative, then $\mathsf{h}(v \to u) = -\mathsf{h}(u \to v)$ by the anti-symmetry of f and g, which implies the same property holds for h.

Consider an arbitrary edge $e = (u \to v)$ such that $\mathsf{h}(u \to v) > 0$.

There are two possibilities. First, if $e = (u \to v) \in E$, and $\mathsf{f}(e) < \mathsf{c}(e)$, then the claim trivially holds, since then $e \in \mathsf{G}_{\mathsf{f}}$. Thus, consider the case when $\mathsf{f}(e) = \mathsf{c}(e)$, but then $\mathsf{h}(e) = \mathsf{g}(e) - \mathsf{f}(e) \le 0$. Which contradicts our assumption that $\mathsf{h}(u \to v) > 0$.

The second possibility, is that $e = (u \to v) \notin E$. But then $e^{-1} = (v \to u)$ must be in E, and it holds $0 > \mathsf{h}\left(e^{-1}\right) = \mathsf{g}\left(e^{-1}\right) - \mathsf{f}\left(e^{-1}\right)$. Implying that $\mathsf{f}\left(e^{-1}\right) > \mathsf{g}\left(e^{-1}\right) \ge \ell\left(e^{-1}\right)$. Namely, there is a flow by f in G going in the direction of e^{-1} which larger than the lower bound. Since we can return this flow in the other direction, it must be that $e \in \mathsf{G}_{\mathsf{f}}$. ∎

Lemma 19.3.3. *Let* f *be a circulation in a graph* G. *Then,* f *can be decomposed into at most* m *cycles,* $\mathsf{C}_1, \ldots, \mathsf{C}_m$, *such that, for any* $e \in E(G)$, *we have*

$$\mathsf{f}(e) = \sum_{i=1}^{t} \lambda_i \cdot \chi_{\mathsf{C}_i}(e),$$

where $\lambda_1, \ldots, \lambda_t > 0$ *and* $t \le m$, *where* m *is the number of edges in* G.

Proof: Since f is a circulation, and the amount of flow into a node is equal to the amount of flow leaving the node, it follows that as long as f not zero, one can find a cycle in f. Indeed, start with a vertex which has non-zero amount of flow into it, and walk on an adjacent edge that has positive flow on it. Repeat this process, till you visit a vertex that was already visited. Now, extract the cycle contained in this walk.

Let C_1 be such a cycle, and observe that every edge of C_1 has positive flow on it, let λ_1 be the smallest amount of flow on any edge of C_1, and let e_1 denote this edge. Consider the new flow $\mathsf{g} = \mathsf{f} - \lambda_1 \cdot \chi_{\mathsf{C}_1}$. Clearly, g has zero flow on e_1, and it is a circulation. Thus, we can remove e_1 from G, and let H denote the new graph. By induction, applied to g on H, the flow g can be decomposed into $m - 1$ cycles with positive coefficients. Putting these cycles together with λ_1 and C_1 implies the claim. ∎

Theorem 19.3.4. *A flow f is a minimum cost feasible circulation if and only if each directed cycle of G_f has nonnegative cost.*

Proof: Let C be a negative cost cycle in G_f. Then, we can circulate more flow on C and get a flow with smaller price. In particular, let $\varepsilon > 0$ be a sufficiently small constant, such that $\mathsf{g} = \mathsf{f} + \varepsilon * \chi_{\mathsf{C}}$ is still a feasible circulation (observe, that since the edges of C are G_f, all of them have residual capacity that can be used to this end). Now, we have that

$$\mathrm{cost}(\mathsf{g}) = \mathrm{cost}(\mathsf{f}) + \sum_{e \in \mathsf{C}} \kappa(e) * \varepsilon = \mathrm{cost}(\mathsf{f}) + \varepsilon * \sum_{e \in \mathsf{C}} \kappa(e) = \mathrm{cost}(\mathsf{f}) + \varepsilon * \kappa(\mathsf{C}) < \mathrm{cost}(\mathsf{f}),$$

since $\kappa(\mathsf{C}) < 0$, which is a contradiction to the minimality of f.

As for the other direction, assume that all the cycles in G_f have non-negative cost. Then, let g be any feasible circulation. Consider the circulation $\mathsf{h} = \mathsf{g} - \mathsf{f}$. By Lemma 19.3.2, all the edges used by h are in G_f, and by Lemma 19.3.3 we can find $t \leq |E(\mathsf{G}_\mathsf{f})|$ cycles $\mathsf{C}_1, \ldots, \mathsf{C}_t$ in G_f, and coefficients $\lambda_1, \ldots, \lambda_t$, such that

$$\mathsf{h}(e) = \sum_{i=1}^{t} \lambda_i \chi_{\mathsf{C}_i}(e).$$

We have that

$$\mathrm{cost}(\mathsf{g}) - \mathrm{cost}(\mathsf{f}) = \mathrm{cost}(\mathsf{h}) = \mathrm{cost}\left(\sum_{i=1}^{t} \lambda_i \chi_{\mathsf{C}_i}\right) = \sum_{i=1}^{t} \lambda_i \mathrm{cost}(\chi_{\mathsf{C}_i}) = \sum_{i=1}^{t} \lambda_i \kappa(\mathsf{C}_i) \geq 0,$$

as $\kappa(\mathsf{C}_i) \geq 0$, since there are no negative cycles in G_f. This implies that $\mathrm{cost}(\mathsf{g}) \geq \mathrm{cost}(\mathsf{f})$. Namely, f is a minimum-cost circulation. ∎

19.4. A Strongly Polynomial Time Algorithm for Min-Cost Flow

The algorithm would start from a feasible circulation f. We know how to compute such a flow f using the standard max-flow algorithm. At each iteration, it would find the cycle C of minimum average cost cycle in G_f (using the algorithm of Section 19.1). If the cost of C is non-negative, we are done since we had arrived to the minimum cost circulation, by Theorem 19.3.4.

Otherwise, we circulate as much flow as possible along C (without violating the lower-bound constraints and capacity constraints), and reduce the price of the flow f. By Corollary 19.1.2, we can compute such a cycle in $O(mn)$ time. Since the cost of the flow is monotonically decreasing the algorithm would terminate if all the number involved are integers. But we will show that this algorithm performs a polynomial number of iterations in n and m.

It is striking how simple is this algorithm, and the fact that it works in polynomial time. The analysis is somewhat more painful.

19.5. Analysis of the Algorithm

To analyze the above algorithm, let f_i be the flow in the beginning of the ith iteration. Let C_i be the cycle used in the ith iteration. For a flow f, let C_f the minimum average-length cycle of G_f, and let $\mu(\mathsf{f}) = \kappa(\mathsf{C}_\mathsf{f})/|\mathsf{C}_\mathsf{f}|$ denote the average "cost" per edge of C_f.

The following lemma, states that we are making "progress" in each iteration of the algorithm.

$\mathsf{f}, \mathsf{g}, \mathsf{h}, \mathsf{i}$	Flows or circulations
G_f	The residual graph for f
$\mathsf{c}(e)$	The capacity of the flow on e
$\ell(e)$	The lower-bound (i.e., demand) on the flow on e
cost(f)	The overall cost of the flow f
$\kappa(e)$	The cost of sending one unit of flow on e
$\psi(e)$	The reduced cost of e

Figure 19.2: Notation used.

Lemma 19.5.1. *Let* f *be a flow, and let* g *the flow resulting from applying the cycle* $\mathsf{C} = \mathsf{C}_\mathsf{f}$ *to it. Then,* $\mu(\mathsf{g}) \geq \mu(\mathsf{f})$.

Proof: Assume for the sake of contradiction, that $\mu(\mathsf{g}) < \mu(\mathsf{f})$. Namely, we have

$$\frac{\kappa(\mathsf{C}_\mathsf{g})}{\left|\mathsf{C}_\mathsf{g}\right|} < \frac{\kappa(\mathsf{C}_\mathsf{f})}{|\mathsf{C}_\mathsf{f}|}. \tag{19.2}$$

Now, the only difference between G_f and G_g are the edges of C_f. In particular, some edges of C_f might disappear from G_g, as they are being used in g to their full capacity. Also, all the edges in the opposite direction to C_f will be present in G_g.

Now, C_g must use at least one of the new edges in G_g, since otherwise this would contradict the minimality of C_f (i.e., we could use C_g in G_f and get a cheaper average cost cycle than C_f). Let U be the set of new edges of G_g that are being used by C_g and are not present in G_f. Let $U^{-1} = \left\{ e^{-1} \;\middle|\; e \in U \right\}$. Clearly, all the edges of U^{-1} appear in C_f.

Now, consider the cycle $\pi = \mathsf{C}_\mathsf{f} \cup \mathsf{C}_\mathsf{g}$. We have that the average of π is

$$\alpha = \frac{\kappa(\mathsf{C}_\mathsf{f}) + \kappa(\mathsf{C}_\mathsf{g})}{|\mathsf{C}_\mathsf{f}| + \left|\mathsf{C}_\mathsf{g}\right|} < \max\left(\frac{\kappa(\mathsf{C}_\mathsf{g})}{\left|\mathsf{C}_\mathsf{g}\right|}, \frac{\kappa(\mathsf{C}_\mathsf{f})}{|\mathsf{C}_\mathsf{f}|} \right) = \mu(\mathsf{f}),$$

by Eq. (19.2). We can write π is a union of k edge-disjoint cycles $\sigma_1, \ldots, \sigma_k$ and some 2-cycles. A 2-cycle is formed by a pair of edges e and e^{-1} where $e \in U$ and $e^{-1} \in U^{-1}$. Clearly, the cost of these 2-cycles is zero. Thus, since the cycles $\sigma_1, \ldots, \sigma_k$ have no edges in U, it follows that they are all contained in G_f. We have

$$\kappa(\mathsf{C}_\mathsf{f}) + \kappa\Big(\mathsf{C}_\mathsf{g}\Big) = \sum_i \kappa(\sigma_i) + 0.$$

Thus, there is some non-negative integer constant c, such that

$$\alpha = \frac{\kappa(\mathsf{C}_\mathsf{f}) + \kappa(\mathsf{C}_\mathsf{g})}{|\mathsf{C}_\mathsf{f}| + \left|\mathsf{C}_\mathsf{g}\right|} = \frac{\sum_i \kappa(\sigma_i)}{c + \sum_i |\sigma_i|} \geq \frac{\sum_i \kappa(\sigma_i)}{\sum_i |\sigma_i|},$$

since α is negative (since $\alpha < \mu(\mathsf{f}) < 0$ as otherwise the algorithm would had already terminated). Namely, $\mu(\mathsf{f}) > \left(\sum_i \kappa(\sigma_i)\right) / \left(\sum_i |\sigma_i|\right)$. Which implies that there is a cycle σ_r, such that $\mu(\mathsf{f}) > \kappa(\sigma_r)/|\sigma_r|$ and this cycle is contained in G_f. But this is a contradiction to the minimality of $\mu(\mathsf{f})$. ∎

19.5.1. Reduced cost induced by a circulation

Conceptually, consider the function $\mu(\mathsf{f})$ to be a potential function that increases as the algorithm progresses. To make further progress in our analysis, it would be convenient to consider a reweighting of the edges of G, in such a way that preserves the weights of cycles.

Given a circulation f, we are going to define a different cost function on the edges which is induced by f. To begin with, let $\beta(u \to v) = \kappa(u \to v) - \mu(\mathsf{f})$. Note, that under the cost function α, the cheapest cycle has price 0 in G (since the average cost of an edge in the cheapest average cycle has price zero). Namely, G has no negative cycles under β. Thus, for every vertex $v \in V(G)$, let $d(v)$ denote the length of the shortest walk that ends at v. The function $d(v)$ is a potential in G, by Lemma 19.2.1, and as such

$$d(v) - d(u) \le \beta(u \to v) = \kappa(u \to v) - \mu(\mathsf{f}) . \tag{19.3}$$

Next, let the ***reduced cost*** of $(u \to v)$ (in relation to f) be

$$\psi(u \to v) = \kappa(u \to v) + d(u) - d(v).$$

In particular, Eq. (19.3) implies that

$$\forall\, (u \to v) \in E(\mathsf{G}_\mathsf{f}) \quad \psi(u \to v) = \kappa(u \to v) + d(u) - d(v) \ge \mu(\mathsf{f}) . \tag{19.4}$$

Namely, the reduced cost of any edge $(u \to v)$ is at least $\mu(\mathsf{f})$.

Note that $\psi(v \to u) = \kappa(v \to u) + d(v) - d(u) = -\kappa(u \to v) + d(v) - d(u) = -\psi(u \to v)$ (i.e., it is anti-symmetric). Also, for any cycle C in G, we have that $\kappa(\mathsf{C}) = \psi(C)$, since the contribution of the potential $d(\cdot)$ cancels out.

The idea is that now we think about the algorithm as running with the reduced cost instead of the regular costs. Since the costs of cycles under the original cost and the reduced costs are the same, negative cycles are negative in both costs. The advantage is that the reduced cost is more useful for our purposes.

19.5.2. Bounding the number of iterations

Lemma 19.5.2. *Let f be a flow used in the ith iteration of the algorithm, let g be the flow used in the $(i+m)$th iteration, where m is the number of edges in G. Furthermore, assume that the algorithm performed at least one more iteration on g. Then, $\mu(\mathsf{g}) \ge (1 - 1/n)\mu(\mathsf{f})$.*

Proof: Let $\mathsf{C}_0, \ldots, \mathsf{C}_{m-1}$ be the m cycles used in computing g from f. Let $\psi(\cdot)$ be the reduced cost function induced by f.

If a cycle has only negative ***reduced cost*** edges, then after it is applied to the flow, one of these edges disappear from the residual graph, and the reverse edge (with positive reduced cost) appears in the residual graph. As such, if all the edges of these cycles have negative reduced costs, then G_g has no negative reduced cost edge, and as such $\mu(\mathsf{g}) \ge 0$. But the algorithm stops as soon as the average cost cycle becomes positive. A contradiction to our assumption that the algorithm performs at least another iteration.

Let C_h be the first cycle in this sequence, such that it contains an edge e', such that its reduced cost is positive; that is $\psi(e') \ge 0$. Note, that C_h has most n edges. We have that

$$\kappa(\mathsf{C}_h) = \psi(\mathsf{C}_h) = \sum_{e \in \mathsf{C}_h} \psi(e) = \psi(e') + \sum_{e \in \mathsf{C}_h, e \ne e'} \psi(e) \ge 0 + (|\mathsf{C}_h| - 1)\,\mu(\mathsf{f}) ,$$

by Eq. (19.4). Namely, the average cost of C_h is

$$0 > \mu(\mathsf{f}_h) = \frac{\kappa(\mathsf{C}_h)}{|\mathsf{C}_h|} \ge \frac{|\mathsf{C}_h| - 1}{|\mathsf{C}_h|}\mu(\mathsf{f}) \ge \left(1 - \frac{1}{n}\right)\mu(\mathsf{f}) .$$

The claim now easily follows from Lemma 19.5.1. ∎

To bound the running time of the algorithm, we will argue that after sufficient number of iterations edges start disappearing from the residual network and never show up again in the residual network. Since there are only $2m$ possible edges, this would imply the termination of the algorithm.

Observation 19.5.3. *We have that* $(1-1/n)^n \leq (\exp(-1/n))^n \leq 1/e$, *since* $1 - x \leq e^{-x}$, *for all* $x \geq 0$, *as can be easily verified.*

Lemma 19.5.4. *Let* f *be the circulation maintained by the algorithm at iteration* ρ. *Then there exists an edge* e *in the residual network* G_f *such that it never appears in the residual networks of circulations maintained by the algorithm, for iterations larger than* $\rho + t$, *where* $t = 2nm\lceil \ln n \rceil$.

Proof: Let g be the flow used by the algorithm at iteration $\rho + t$. We define the reduced cost over the edges of G, as induced by the flow g. Namely,

$$\psi(u \to v) = \kappa(u \to v) + d(u) - d(v),$$

where $d(u)$ is the length of the shortest walk ending at u where the weight of edge $(u \to w)$ is $\kappa(u \to w) - \mu(\mathsf{g})$.

flow	in iteration
f	ρ
g	$\rho + t$
h	$\rho + t + \tau$

Now, conceptually, we are running the algorithm using this reduced cost function over the edges, and consider the minimum average cost cycle at iteration ρ with cost $\alpha = \mu(\mathsf{f})$. There must be an edge $\mathsf{e} \in E(\mathsf{G}_\mathsf{f})$, such that $\psi(\mathsf{e}) \leq \alpha$. (Note, that α is a negative quantity, as otherwise the algorithm would have terminated at iteration ρ.)

We have that, at iteration $\rho + t$, it holds

$$\mu(\mathsf{g}) \geq \alpha * \left(1 - \frac{1}{n}\right)^t \geq \alpha * \exp(-2m\lceil \ln n \rceil) \geq \frac{\alpha}{2n}, \tag{19.5}$$

by Lemma 19.5.2 and Observation 19.5.3 and since $\alpha < 0$. On the other hand, by Eq. (19.4), we know that for all the edges f in $E\left(\mathsf{G}_\mathsf{g}\right)$, it holds $\psi(\mathsf{f}) \geq \mu(\mathsf{g}) \geq \alpha/2n$. As such, e can not be an edge of G_g since $\psi(\mathsf{e}) \leq \alpha$. Namely, it must be that $\mathsf{g}(\mathsf{e}) = \mathsf{c}(\mathsf{e})$.

So, assume that at a later iteration, say $\rho + t + \tau$, the edge e reappeared in the residual graph. Let h be the flow at the $(\rho + t + \tau)$th iteration, and let G_h be the residual graph. It must be that $\mathsf{h}(\mathsf{e}) < \mathsf{c}(\mathsf{e}) = \mathsf{g}(\mathsf{e})$.

Now, consider the circulation $\mathsf{i} = \mathsf{g} - \mathsf{h}$. It has a positive flow on the edge e, since $\mathsf{i}(\mathsf{e}) = \mathsf{g}(\mathsf{e}) - \mathsf{h}(\mathsf{e}) > 0$. In particular, there is a directed cycle C of positive flow of i in G_i that includes e, as implied by Lemma 19.3.3. Note, that Lemma 19.3.2 implies that C is also a cycle of G_h.

Now, the edges of C^{-1} are present in G_g. To see that, observe that for every edge $\mathsf{g} \in \mathsf{C}$, we have that $0 < \mathsf{i}(\mathsf{g}) = \mathsf{g}(\mathsf{g}) - \mathsf{h}(\mathsf{g}) \leq \mathsf{g}(\mathsf{g}) - \ell(\mathsf{g})$. Namely, $\mathsf{g}(\mathsf{g}) > \ell(\mathsf{g})$ and as such $\mathsf{g}^{-1} \in E\left(\mathsf{G}_\mathsf{g}\right)$. As such, by Eq. (19.4), we have $\psi\left(\mathsf{g}^{-1}\right) \geq \mu(\mathsf{g})$. This implies

$$\forall \mathsf{g} \in \mathsf{C} \qquad \psi(\mathsf{g}) = -\psi\left(\mathsf{g}^{-1}\right) \leq -\mu(\mathsf{g}) \leq -\frac{\alpha}{2n},$$

by Eq. (19.5). Since C is a cycle of G_h, we have

$$\kappa(\mathsf{C}) = \psi(C) = \psi(\mathsf{e}) + \psi(C \setminus \{\mathsf{e}\}) \leq \alpha + (|\mathsf{C}| - 1) \cdot \left(-\frac{\alpha}{2n}\right) < \frac{\alpha}{2}.$$

Namely, the average cost of the cycle C, which is present in G_h, is $\kappa(\mathsf{C})/|\mathsf{C}| < \alpha/(2n)$.

On the other hand, the minimum average cost cycle in G_h has average price $\mu(\mathsf{h}) \geq \mu(\mathsf{g}) \geq \frac{\alpha}{2n}$, by Lemma 19.5.1. A contradiction, since we found a cycle C in G_h which is cheaper. ∎

We are now ready for the "kill" – since one edge disappears forever every $O(mn \log n)$ iterations, it follows that after $O(m^2 n \log n)$ iterations the algorithm terminates. Every iteration takes $O(mn)$ time, by Corollary 19.1.2. Putting everything together, we get the following.

Theorem 19.5.5. *Given a digraph* G *with* n *vertices and* m *edges, lower bound and upper bound on the flow of each edge, and a cost associated with each edge, then one can compute a valid circulation of minimum-cost in* $O(m^3 n^2 \log n)$ *time.*

19.6. Bibliographical Notes

The minimum average cost cycle algorithm, of Section 19.1, is due to Karp [Kar78].

The description here follows very roughly the description of [Sch04]. The first strongly polynomial time algorithm for minimum-cost circulation is due to Éva Tardos [Tar85]. The algorithm we show is an improved version due to Andrew Goldberg and Robert Tarjan [GT89]. Initial research on this problem can be traced back to the 1940s, so it took almost fifty years to find a satisfactory solution to this problem.

Bibliography

[GT89] A. V. Goldberg and R. E. Tarjan. Finding minimum-cost circulations by canceling negative cycles. *J. Assoc. Comput. Mach.*, 36(4):873–886, 1989.

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

[Sch04] A. Schrijver. *Combinatorial Optimization : Polyhedra and Efficiency (Algorithms and Combinatorics)*. Springer, July 2004.

[Tar85] É. Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3):247–255, 1985.