18.415/6.854 Advanced Algorithms	November 1994

Network flows

Lecturer: Michel X. Goemans

In these notes, we study some problems in "Network Flows". For a more comprehensive treatment, the reader is referred to the surveys [12, 1], or to the recent book [2]. Network flow problems arise in a variety of settings; the underlying networks might be transportation networks, communication networks, hydraulic networks, computer chips, or some abstract network. The field was born from applications in the 40's and 50's and has since developed into a strong methodological core with numerous algorithmic issues. The first polynomial time algorithms for network flow problems have been developed in the 70's, and constant progress towards faster and faster algorithms has been made in the 80's. Network flow problems can be formulated as linear programs and, as a result, all the methodology of linear programming can be applied. Duality plays a crucial role, and the simplex algorithm can take advantage of the structure of network flow problems (bases can be nicely characterized).

Some of the basic problems in this area include the single source shortest path problem, the maximum flow problem, and the minimum cost flow problem. First, we shall briefly review each of them and then we shall describe a polynomial time algorithm due to Goldberg and Tarjan [14] for the minimum cost flow problem.

1 Single Source Shortest Path Problem

We are interested in the following problem:
Given

- a directed graph $G = (V, E)$ where V is the set of vertices and E is the set of edges,
- and a length function $l : E \to \mathbb{Z}$,
- a distinguished vertex $s \in V$ (the source vertex),

Find for all $v \in V$ the length $\delta(v)$ of the shortest path from s to v.

This is NP-hard if we allow negative length cycles (i.e. cycles for which the sum of the lengths of its edges is negative). However, if all lengths are nonnegative ($l(u, v) \geq 0$ for all edges $(u, v) \in E$) then a standard algorithm that solves this problem is Dijkstra's algorithm [6] (see also [4]). The implementation of Dijkstra's algorithm is based on the implementation of a priority queue and various implementations of this priority queue lead to different worst-case running times. Using a Fibonacci heap implementation [10] of the priority queue, it can be shown that the algorithm has a total running time of $O(m + n \log n)$ where $m = |E|$ and $n = |V|$. This is the best known strongly polynomial algorithm for the single-source shortest path problem. An algorithm is said to be **strongly polynomial** if

1. It performs a polynomially bounded number of operations in the number of input data (in this case m and n). We allow the operations +,-,*,< and rational division.

2. The sizes of the numbers occurring during the algorithm are polynomially bounded in the size of the input.

There are also single-source shortest path algorithms which may not be strongly polynomial, i.e. algorithms whose running time depends on $L = \max l(u, v) + 1$. These algorithms may achieve a better running time than Dijkstra's algorithm, provided L is not too large. Listed below are four such algorithms:

Dial [5]	$O(m + nL)$
Johnson [16]	$O(m \log \log L)$
Gabow [11]	$O(m \log_d L)$ where $d = \max(2, \lceil m/n \rceil)$
Ahuja, Mehlhorn, Orlin, Tarjan [3]	$O(m + n\sqrt{\log L})$

Observe that all these algorithms except Dial's algorithm are polynomial since the size of the input is at least $\log L$.

If negative lengths are allowed then the problem can still be solved in polynomial time provided that no negative length cycle exists. The algorithm of Bellman-Ford solves this problem in $O(nm)$ time.

We would like to point out that these problems are defined on a directed graph. An undirected shortest path problem can easily be reduced to a directed instance by replacing every edge by bidirected edges. This reduction is fine if all lengths are nonnegative (in which case Dijkstra's algorithm can be used), but does not work if there are edges of negative length. In this case, indeed, a negative length cycle would be created. However, the undirected shortest path problem on an undirected graph with possibly negative edge lengths but no negative length cycle can still be solved in polynomial time. The algorithm is, however, fairly complicated and is based on a reduction of the problem to nonbipartite matching.

2 The Maximum Flow Problem

Given

- a directed graph $G = (V, E)$ where V is the set of vertices and E is the set of edges,

- capacity $u(v, w) \geq 0$ for $(v, w) \in E$,

- source $s \in V$,

- sink $t \in V$,

a flow f is an assignment of values to the edges which satisfies $f(v,w) \leq u(v,w)$ for all edges (v,w) and which satisfies the flow conservation constraints

$$\sum_{w:(v,w)\in E} f(v,w) = \sum_{w:(w,v)\in E} f(w,v)$$

for all vertices v except s and t. The goal is to find a flow such that the net flow $\sum_w f(s,w)$ out of s ($\sum_{v:(s,v)\in E} f(s,v)$) is maximum. One can easily derive that the net flow out of s is equal to the net flow into t, and thus we could maximize this latter quantity as well.

All these constraints are linear constraints and the objective function is linear, so the maximum flow problem (MAX FLOW) is a linear program. We could therefore exploit the structure of the linear program to tailor the simplex algorithm. This has been done and, in fact, although no version of the simplex algorithm is known to run in polynomial time for general linear programs, it is known that it can be made to run in polynomial time (or even in strongly polynomial time) for the maximum flow problem. Goldfarb and Hao [15] have developed a version of the simplex which makes at most nm pivots and run in $O(n^2m)$ time. However, there are some more efficient combinatorial algorithms which exploit the structure of the problem. Most known algorithms are based on the concept of "augmenting paths", introduced by Ford and Fulkerson [8]. There are a variety of algorithms with different running times. The best strongly polynomial running time of a max flow algorithm is $O(nm \log n^2/m)$ (due to Goldberg and Tarjan [13]).

3 Minimum Cost Circulation Problem

In the minimum cost circulation problem, we are given a directed graph $G = (V, E)$. For each arc $(v,w) \in E$, we are given a cost $c(v,w)$, a lower bound $l(v,w)$ and an upper bound $u(v,w)$. Throughout, we assume that $l(.,.), u(.,.)$ and $c(.,.)$ are integral unless mentioned otherwise. We will associate a flow f with each arc of the graph. This flow will be required to satisfy $l(v,w) \leq f(v,w) \leq u(v,w)$ and the cost of the flow on (v,w) is defined to be $c(v,w)f(v,w)$. This is the classical notation. However, in our lectures, we adopt Goldberg and Tarjan's notation [14] in which every directed arc (v,w) is represented by arcs (v,w) and (w,v) (see Figure 1).

This will simplify the proofs later on. In this notation, the flow $f(w,v)$ on (w,v) is assumed to be equal to $-f(v,w)$, i.e. the flow is antisymmetric. Using this antisymmetry assumption, the lower bound on the flow $f(v,w)$ is equivalent to an upper bound of $-l(v,w)$ on $f(w,v)$. Also, the cost $c(w,v)$ on the arc (w,v) is defined to be $-c(v,w)$. This ensures that, if we push some flow on (v,w) and then decide to push it back from w to v, we get a full refund of the cost incurred (i.e. $c(v,w)f(v,w)$). Notice that the total cost of the flow on the arcs (v,w) and (w,v) is equal to

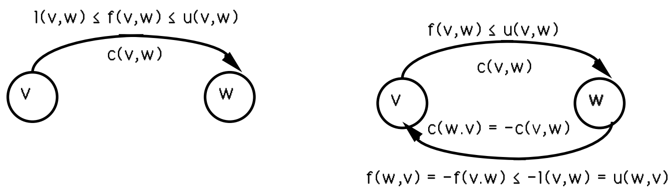

Figure 1: Standard notation vs. Goldberg-Tarjan notation.

$c(v,w)f(v,w) + c(w,v)f(w,v) = c(v,w)f(v,w) - c(v,w)f(w,v) = 2c(v,w)f(v,w) = 2c(w,v)f(w,v)$.

To recapitulate, we are given a bidirected[1] graph $G = (V,E)$, a capacity function $u : E \to \mathbb{Z}$ and a cost function $c : E \to \mathbb{Z}$. The cost function is assumed to be antisymmetric:

$$c(v,w) = -c(w,v) \qquad\qquad \forall (v,w) \in E.$$

A flow is a function $f : E \longrightarrow \mathbb{R}$, which is assumed

1. to be antisymmetric, i.e. $f(v,w) = -f(w,v)$, and
2. to satisfy the capacity constraints: $f(v,w) \leq u(v,w)$ for all $(v,w) \in E$.

The cost of a flow is defined as:

$$c \cdot f = \sum_{(v,w)\in E} c(v,w)u(v,w).$$

A flow is said to be a *circulation* if

$$\sum_w f(v,w) = 0$$

for all $v \in V$. Using the antisymmetry constraints, this is equivalent to saying that the flow out of vertex v minus the flow into v is equal to 0 for all $v \in V$. These conditions are thus the flow conservation constraints. The *minimum cost circulation problem* is the problem of finding a circulation of minimum cost.

A closely related problem to the minimum cost circulation problem is the *minimum cost flow problem.* In this problem, we are also given a supply function $b : V \to \mathbb{Z}$ satisfying $\sum_{v\in V} b(v) = 0$ and the flow is required to satisfy

$$\sum_w f(v,w) = b(v) \tag{1}$$

for all $v \in V$. The goal is to find a flow satisfying (1) of minimum cost. The minimum cost circulation problem is clearly a special case of the minimum cost flow problem (simply take $b(v) = 0$ for all $v \in V$). However, the converse is also true. Consider

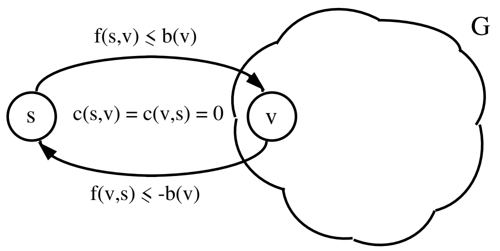

Figure 2: How to convert a minimum cost flow problem into a minimum cost circulation problem.

the following transformation that converts any instance of the minimum cost flow problem into an instance of the minimum cost circulation problem (see Figure 3).

Let $G' = (V', E')$ be the graph obtained by extending G with one extra vertex, say s, linked to all other vertices, i.e. $V' = V \cup \{s\}$ and $E' = E \cup \{(s,v) : v \in V\} \cup \{(v,s) : v \in V\}$. For these new edges, let $c(s,v) = c(v,s) = 0$ and $u(s,v) = b(v) = -u(v,s)$, the other costs and capacities remaining unchanged. The capacities on the bidirected edges incident to s have been chosen in such a way that, for any flow f on this extended graph, we have $f(s,v) = b(v)$. Therefore, any circulation f on G' induces a flow on G satisfying (1) and vice versa. Since this circulation on G' and this flow on G have the same cost, we can solve the minimum cost flow problem by solving a minimum cost circulation problem.

In these notes, we develop a purely combinatorial algorithm to solve the minimum cost circulation problem and we will also show that this problem can be solved in strongly polynomial time. (We'd like to point out that for the minimum cost flow or circulation problem, it is not known whether the simplex method can be adapted to run in strongly polynomial time (contrary to the case for the maximum flow problem).)

In many situations, the circulation is required to be integral. This additional restriction is not restrictive as indicated in the following Theorem — sometimes referred to as the *integrality theorem*.

Theorem 1 *If $u(v,w) \in \mathbb{Z}$ for all $(v,w) \in E$ then there exists an optimal circulation (or flow) with $f(v,w) \in \mathbb{Z}$.*

Although there are several ways to prove this result, we will deduce it later in the notes from a simple algorithm for the minimum cost circulation problem. More precisely, we will show that, at every iteration of the algorithm, the current circulation is integral and, hence, it is also integral when the algorithm terminates.

The minimum cost circulation problem has some interesting special cases as described in the next sections. Our strongly polynomial time algorithm for the minimum cost circulation problem will thus lead to strongly polynomial time algorithms

[1] $(v,w) \in E$ implies $(w,v) \in E$.

for these special cases (although more efficient algorithms can be designed for these special cases).

3.1 The Maximum Flow Problem

The maximum flow problem is a special case of the minimum cost circulation problem. Indeed, given an instance of the maximum flow problem, add an edge between s and t (see Figure 3.1) and define $u(t,s) = \infty$, $u(s,t) = 0$, $c(t,s) = -1 = -c(s,t)$ and $c(v,w) = 0$ for all $(v,w) \neq (s,t)$.

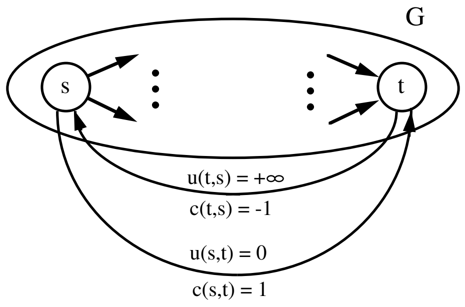

Figure 3: How to transform a maximum flow problem into a minimum cost circulation problem.

The capacities on the bidirected edge (s,t) is such that $f(t,s) \geq 0$, implying that the flow goes from t to s. There is a one-to-one correspondence between circulations in this extended graph and flows in the original graph satisfying all flow conservation constraints in $V \setminus \{s,t\}$. Moreover, the cost of any circulation in this extended graph is exactly equal to minus the net flow out of s (or into t) in the original graph. As a result, the maximum flow problem in G is equivalent to the minimum cost circulation problem in the extended graph.

Using the integrality theorem (Theorem 1), we obtain that the flow of maximum value can be assumed to be integral whenever the capacities are integral.

3.2 Bipartite Matching

The maximum cardinality matching problem on a bipartite graph $G = (A, B, E)$ (A and B denotes the bipartition of the vertex set) is the problem of finding the largest number of disjoint edges. This problem is a special case of the maximum flow problem and, hence, of the minimum cost circulation problem. To transform the maximum cardinality bipartite matching problem into a maximum flow problem (see Figure 3.2), we

1. direct all the edges from A to B,

2. add a source vertex s, a sink vertex t,

3. add the edges (s, a) for all vertices $a \in A$ and the edges (b, t) for all vertices $b \in B$ and

4. define the capacity of all existing edges to be 1 and the capacity of their reverse edges to be 0 (in other words, the flow on the existing edges have a lower bound of 0).

By the integrality theorem, we know that the flow on any existing edge can be assumed to be either 0 or 1. Therefore, to any flow f, there corresponds a matching $M = \{(v, w) \in E : f(v, w) = 1\}$ whose cardinality is precisely equal to the net amount of flow out of vertex s.

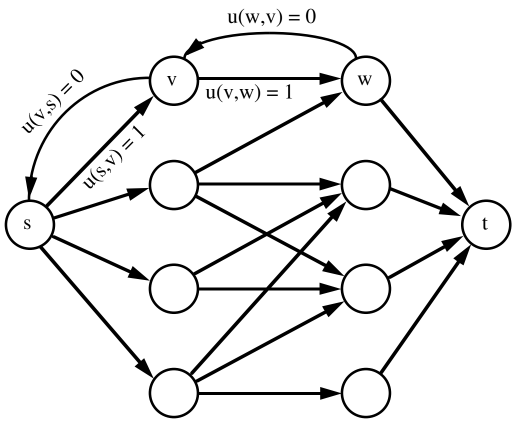

Figure 4: Maximum cardinality bipartite matching is a special case of maximum-flow.

It is also easy to construct from a matching M a flow of value $|M|$. As a result, any integral flow of maximum value will correspond to a matching of maximum cardinality.

In fact, the minimum weighted bipartite matching problem is also a special case of the minimum cost circulation problem. We can modify the above transformation in the following way. Define the cost of any edge of the original graph to be its original cost and the cost of any new edge to be 0. Now, we can model three versions of the minimum weighted bipartite matching problem by appropriately defining the capacities on the edges (t, s) and (s, t):

1. If $u(t, s) = n$ and $u(s, t) = -n$ where $n = |A| = |B|$, we get the minimum weighted perfect (a *perfect* matching is a matching that covers all the vertices) matching.

2. If $u(t, s) = n$ and $u(s, t) = 0$, we obtain the minimum weighted matching.

3. If $u(t, s) = k$ and $u(s, t) = -k$, we obtain the minimum weighted matching of size k.

3.3 Shortest paths

The single source shortest path problem is also a special case of the minimum cost flow problem. Indeed, by setting $l(v,w) = 0$ and $u(v,w) = 1$ for every edge (and letting their cost be the original cost), and introducing an edge from t to s with $u(t,s) = l(t,s) = 1$ and $c(t,s) = 0$, we obtain an equivalent instance of the minimum cost circulation problem.

4 Some Important Notions

We now go back to the minimum cost circulation problem, and before describing a polynomial time algorithm for it, we present some useful tools.

4.1 Residual Graph

Given a minimum cost circulation problem and a circulation f, we define *the residual graph* $G_f = (V, E_f)$ with respect to f by $E_f = \{(v,w) : f(v,w) < u(v,w)\}$. For example, if $u(v,w) = 5$, $u(w,v) = -1$ and $f(v,w) = 3$ (hence, $f(w,v) = -3$ by antisymmetry) then both (v,w) and (w,v) will be present in E_f. However, if $f(v,w) = 1$ (i.e. $f(w,v) = -1$), only (v,w) will be in E_f. With respect to f, we define *the residual capacity* of the edge (v,w) by

$$u_f(v,w) = u(v,w) - f(v,w).$$

Notice that the edges of the residual graph have a positive residual capacity.

4.2 Potentials

We associate with each vertex v a vertex potential $p(v)$. The potential of a vertex can be interpreted as the dual variable corresponding to the flow conservation constraints in the linear programming formulation of the problem. The *reduced cost* of the edge (v,w) is then defined as $c_p(v,w) := c(v,w) + p(v) - p(w)$. Note that the reduced costs are still antisymmetric i.e. $c_p(w,v) = c(w,v) + p(w) - p(v) = -c(v,w) - p(v) + p(w) = -c_p(v,w)$. Note also that the *cost*

$$c(\Gamma) := \sum_{(v,w)\in\Gamma} c(v,w)$$

of a directed cycle Γ is equal to its reduced cost

$$c_p(\Gamma) = \sum_{(v,w)\in\Gamma} c_p(v,w)$$

since the vertex potential of any vertex v on the cycle is added and subtracted exactly once. More generally, we have the following result.

Theorem 2 *For any $p : V \to \mathbb{Z}$ and any circulation f, we have $c \cdot f = c_p \cdot f$.*

Proof: By definition,

$$\begin{aligned} c_p \cdot f = \sum_{(v,w)\in E} c_p(v,w)f(v,w) &= \sum_{(v,w)\in E} c(v,w)f(v,w) + \sum_{(v,w)\in E} p(v)f(v,w) \\ &\quad - \sum_{(v,w)\in E} p(w)f(v,w) \\ &= c \cdot f + \sum_{v\in V} p(v) \sum_{w:(v,w)\in E} f(v,w) \\ &\quad - \sum_{w\in V} p(w) \sum_{v:(v,w)\in E} f(v,w) \\ &= c \cdot f + 0 - 0 = c \cdot f, \end{aligned}$$

since by definition of a circulation $\sum_{w:(v,w)\in E} f(v,w) = 0$. □

5 When is a circulation Optimal?

The next theorem characterizes a circulation of minimum cost.

Theorem 3 *For a circulation f, the following are equivalent:*

1. *f is of minimum cost,*
2. *there are no negative (reduced) cost cycles in the residual graph,*
3. *there exist potentials p such that $c_p(v,w) \geq 0$ for $(v,w) \in E_f$.*

This is essentially strong duality, but we will not refer to linear programming in the proof. **Proof:**

- $(\neg 2) \Rightarrow (\neg 1)$.
  Let Γ be a negative cost cycle in E_f. Let

  $$\delta = \min_{(v,w)\in\Gamma} u_f(v,w) > 0.$$

  By *pushing* δ units of flow along Γ, we mean replacing f by $\tilde{f}$ where

  $$\tilde{f}(v,w) = \begin{cases} f(v,w) + \delta & (v,w) \in \Gamma \\ f(v,w) - \delta & (w,v) \in \Gamma \\ f(v,w) & \text{otherwise} \end{cases}.$$

  Notice that, as defined, $\tilde{f}$ also satisfies the antisymmetry constraints and is a circulation. Moreover, $c \cdot \tilde{f} = c \cdot f + \delta \cdot c(\Gamma) < c \cdot f$. This implies that f is not of minimum cost.

- $2 \Rightarrow 3$.
  Let G' be obtained from the residual graph G_f by adding a vertex s linked to all other vertices by edges of cost 0 (the costs of these edges do not matter). Let $p(v)$ be the length of the shortest path from s to v in G' with respect to the costs $c(.,.)$.

  This quantities are well-defined since G_f does not contain any negative cost directed cycle. By definition of the shortest paths, we have $p(w) \leq p(v)+c(v,w)$ for all edges $(v,w) \in E_f$. This implies that $c_p(v,w) \geq 0$ whenever $(v,w) \in E_f$.

- $3 \Rightarrow 1$.
  The proof is by contradiction. Let f^* be a circulation such that $c \cdot f^* < c \cdot f$. Consider $f'(v,w) = f^*(v,w) - f(v,w)$. By definition of the residual capacities, f' is a feasible circulation with respect to $u_f(.,.)$. Its cost is

$$\begin{aligned} c \cdot f' = c_p \cdot f' &= \sum_{(v,w)\in E} c_p(v,w) f'(v,w) \\ &= 2 \sum_{(v,w)\in E: f'(v,w)>0} c_p(v,w) f'(v,w) \\ &\geq 0, \end{aligned}$$

  since $f'(v,w) > 0$ implies that $(v,w) \in E_f$ and, hence, $c_p(v,w) \geq 0$. This contradicts the fact that $c \cdot f' = c \cdot f^* - c \cdot f < 0$.

□

A problem is *well characterized* if its decision version belongs to $NP \cap co{-}NP$. The above theorem gives a good characterization for the minimum cost circulation problem (to be precise, we would also need to show that the potentials can be compactly encoded). It also naturally leads to a simple algorithm, first discovered by Klein [17] also known as the 'Cycle Cancelling Algorithm'.

6 Klein's Cycle Canceling Algorithm

Cycle canceling algorithm (Klein):

1. Let f be any circulation.
2. While G_f contains a negative cycle Γ do
   push $\delta = \min_{(v,w)\in\Gamma} u_f(v,w)$ along Γ.

Recall that in the previous code "push" means that we increase the flow by δ along the well oriented edges of Γ and decrease it by δ along the other edges of Γ.

In Step 1, we will assume that $f = 0$ satisfies the capacity constraints (i.e. $f = 0$ is a circulation). If this is not the case then a circulation can be obtained by solving one maximum flow problem or by modifying the instance so that a circulation can easily be found.

The cycle canceling algorithm can be used to prove Theorem 1. The proof is by induction. Assume that the initial circulation is chosen to be integral. Now, if at iteration k the circulation is integral, then the residual capacities as well as δ are also integral. Therefore, the circulation remains integral throughout the algorithm.

For the maximum flow problem as discussed in Section 3.1, any negative cost directed cycle must consist of a directed path from s to t along with the arc (t, s) since the only negative cost arc is (t, s). Therefore, in this special case, Klein's algorithm reduces to the well-known Ford-Fulkerson's augmenting path algorithm [8].

Ford-Fulkerson's augmenting path algorithm:

1. Start with the zero flow: $f = 0$.
2. While G_f contains a directed path P from s to t do
   push $\delta = \min_{(v,w)\in\Gamma} u_f(v, w)$ along P.

In the next Theorem, we show that the cycle canceling algorithm is correct if the costs and capacities are integral.

Theorem 4 *If $c : E \to \mathbb{Z}$ and $u : E \to \mathbb{Z}$ then Klein's algorithm terminates after $O(mCU)$ iterations where $m = |E|$, C is an upper bound on the absolute value of any cost and U is an upper bound on the absolute value of any capacity. Moreover, the resulting circulation is optimal.*

Proof: Since the costs are integral, any cycle of negative cost has a cost of at most -1. Moreover, if $(v, w) \in G_f$ then $u_f(v, w) \geq 1$ which implies that $\delta \geq 1$. Therefore, at each iteration, the cost of the current circulation decreases by at least 1 unit. On the other hand, since $|c(v, w)| \leq C$ and $|f(v, w)| \leq U$, the absolute value of the cost of the optimal circulation is at most mCU. Therefore, the algorithm terminates after $O(mCU)$ iterations. At that point, the residual graph does not contain any negative cycle and, hence, by Theorem 3, the circulation is optimal. $\Box$

The bound given in Theorem 4 is however not polynomial. In fact, if the negative cycles (or the directed paths in Ford and Fulkerson's algorithm) are not appropriately chosen, the worst-case running time of the algorithm is exponential. In Figure 6, we have an instance of the maximum flow problem in which if we augment alternatively along the paths $s - 1 - 2 - t$ and $s - 2 - 1 - t$, the number of iterations will be $2C$ since at each iteration we push only one additional unit of flow from s to t. If $C = 2^n$, this gives an exponential bound.

Even more surprisingly, the cycle canceling algorithm and the augmenting path algorithm without any specification of which negative directed cycle or which directed st-path to select are not correct if the capacities are irrational. In [9], it is shown that

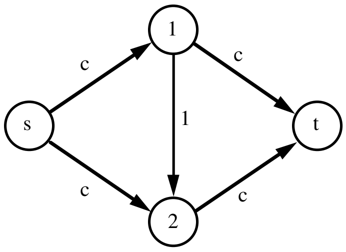

Figure 5: Numbers on the arcs represent the capacities. The reverse arcs have zero capacities.

the augmenting path algorithm can converge to a suboptimal flow if the capacities are irrational and the directed paths are selected in an unfortunate way.

To obtain a *polynomial* running time, we therefore need to specify which negative directed cycle to cancel. If the negative cycle resulting in the maximum cost improvement is selected, the number of iterations becomes polynomial. Unfortunately, finding this cycle is NP-hard. For the maximum flow problem, however, this selection rule reduces to finding the st-path with maximum residual capacity. Such a path can be found in polynomial time (for example, by adapting Dijkstra's algorithm). The resulting algorithm, due to Edmonds and Karp [7], requires $O(m \log U)$ iterations. The time per iteration is $O(m)$ (amortized). Hence we can implement the algorithm with a total running time of $O(m^2 \log U)$ (Tarjan [20]).

For a long time the question of finding a strongly polynomial algorithm (and even its existence) for the minimum cost circulation problem was kept open. In 1985, Éva Tardos [19] devised the first such algorithm. In 1987, Goldberg and Tarjan [14] produced an improved version that we will now present.

7 The Goldberg-Tarjan Algorithm

Define the mean cost of a cycle Γ to be

$$\frac{c(\Gamma)}{|\Gamma|} = \frac{\sum_{(v,w)\in\Gamma} c(v,w)}{|\Gamma|}$$

where $|\Gamma|$ represents the number of arcs in Γ. The minimum mean cost cycle of a graph can be found in strongly polynomial time, namely in $O(nm)$ time, by adapting the Bellman-Ford algorithm for the all pairs shortest path problem. Let

$$\mu(f) = \min_{\text{cycles } \Gamma \text{ in } E_f} \frac{c(\Gamma)}{|\Gamma|}$$

denote the minimum mean cost of all cycles in G_f.

Goldberg-Tarjan algorithm [14]:

1. Let $f = 0$.
2. While $\mu(f) < 0$ do
   push $\delta = \min_{(v,w)\in\Gamma} u_f(v,w)$ along a minimum mean cost cycle Γ of G_f.

The Goldberg-Tarjan algorithm is a cycle canceling algorithm since G_f has a negative directed cycle iff $\mu(f) < 0$.

For the maximum flow problem, this algorithm reduces to the Edmonds-Karp shortest augmenting path algorithm [7] in which the st-path with the fewest number of arcs is selected. The Edmonds-Karp shortest augmenting path algorithm requires $O(m)$ time per augmentation and the number of augmentations is $O(nm)$. This results in a running time of $O(nm^2)$.

8 Analysis of the Goldberg-Tarjan Algorithm

Before analyzing the Goldberg-Tarjan cycle canceling algorithm, we need some definitions.

Definition 1 *A circulation f is ϵ-optimal if there exists p such that $c_p(v,w) \geq -\epsilon$ for all $(v,w) \in E_f$.*

For $\epsilon = 0$, this definition reduces to the condition 3 of Theorem 3, and, therefore, a 0-optimal circulation is a minimum cost circulation.

Definition 2 *$\epsilon(f) =$ minimum ϵ such that f is ϵ-optimal.*

We now show that approximate optimality is sufficient when the costs are integral.

Theorem 5 *If f is a circulation with $\epsilon(f) < \frac{1}{n}$ then f is optimal.*

Proof: $\epsilon(f) < \frac{1}{n}$ implies that, for some potentials p, $c_p(v,w) > -\frac{1}{n}$ for all $(v,w) \in E_f$. Therefore, any cycle Γ of G_f has reduced cost greater than $-|\Gamma|\frac{1}{n} \geq -1$. Since the costs are integral and the cost of any cycle is equal to its reduced cost, we obtain that any directed cycle of G_f has nonnegative cost. Theorem 3 implies that f is optimal. □

The following Theorem shows that the minimum mean cost $\mu(f)$ of all cycles in G_f represents how close the circulation f is from optimality.

Theorem 6 *For any circulation f, $\mu(f) = -\epsilon(f)$.*

Proof:

- $\mu(f) \geq -\epsilon(f)$.
  By definition, there exists p such that $c_p(v,w) \geq -\epsilon(f)$ for all $(v,w) \in E_f$. This implies that $c_p(\Gamma) \geq -\epsilon(f)|\Gamma|$ for any directed cycle Γ of G_f. But, for any directed cycle Γ, $c(\Gamma) = c_p(\Gamma)$. Therefore, dividing by $|\Gamma|$, we obtain that the mean cost of any directed cycle of G_f is at least $-\epsilon(f)$. Hence, $\mu(f) \geq -\epsilon(f)$.

- $-\mu(f) \geq \epsilon(f)$.
  To show that $-\mu(f) \geq \epsilon(f)$, we want to construct a function p such that $c_p(v,w) \geq \mu(f)$ for all $(v,w) \in E_f$. Let $\tilde{c}(v,w) = c(v,w) + (-\mu(f))$ for all $(v,w) \in E_f$. Notice that G_f has no negative cost cycle with respect to $\tilde{c}(.,.)$ since the mean cost of any directed cycle of G_f is increased by $-\mu(f)$. Next, add a new node s to G_f and also arcs from s to v for all $v \in V$. Let $\tilde{c}(s,v)$ be any value, say 0. Let $p(v)$ be the cost with respect to $\tilde{c}(.,.)$ of the shortest path from s to v in this augmented graph. Hence, for all $(v,w) \in E_f$, we have $p(w) \leq p(v) + \tilde{c}(v,w) = p(v) + c(v,w) - \mu(f)$ implying that $c_p(v,w) \geq \mu(f)$.

□

We are now ready to analyze the algorithm. First, we show that, using $\epsilon(f)$ as a measure of near-optimality, the algorithm produces circulations which are closer and closer to optimal.

Theorem 7 *Let f be a circulation and let f' be the circulation obtained by canceling the minimum mean cost cycle Γ in E_f. Then $\epsilon(f) \geq \epsilon(f')$.*

Proof: By definition, there exists p such that

$$c_p(v,w) \geq -\epsilon(f) \tag{2}$$

for all $(v,w) \in E_f$. Moreover, for all $(v,w) \in \Gamma$, we have $c_p(v,w) = -\epsilon(f)$ since, otherwise, its mean cost would not be $-\epsilon(f)$. We claim that, for the same p, (2) holds for all $(v,w) \in E_{f'}$. Indeed, if $(v,w) \in E_{f'} \cap E_f$, (2) certainly holds. If $(v,w) \in E_{f'} \setminus E_f$ then (w,v) certainly belongs to Γ. Hence, $c_p(v,w) = -c_p(w,v) = \epsilon(f) \geq 0$ and (2) is also satisfied. □

Next, we show that $\epsilon(f)$ decreases after a certain number of iterations.

Theorem 8 *Let f be any circulation and let f' be the circulation obtained by performing m iterations of the Golberg-Tarjan algorithm. Then*

$$\epsilon(f') \leq (1 - \frac{1}{n})\epsilon(f).$$

Proof: Let p be such that $c_p(v,w) \geq -\epsilon(f)$ for all $(v,w) \in E_f$. Let Γ_i be the cycle canceled at the ith iteration. Let k be the smallest integer such that there exists $(v,w) \in \Gamma_{k+1}$ with $c_p(v,w) \geq 0$. We know that canceling a cycle removes at least one arc with negative reduced cost from the residual graph and creates only arcs with positive reduced cost. Therefore $k \leq m$. Let f' be the flow obtained after k iterations. By Theorem 6, $-\epsilon(f')$ is equal to the mean cost of Γ_{k+1} which is:

$$\begin{aligned} \frac{\sum_{(v,w)\in\Gamma_{k+1}} c_p(v,w)}{l} &\geq \frac{-(l-1)}{l}\epsilon(f) \\ &= -(1-\frac{1}{l})\epsilon(f) \geq -(1-\frac{1}{n})\epsilon(f), \end{aligned}$$

where $l = |\Gamma_{k+1}|$. Therefore, by Theorem 7, after m iterations, $\epsilon(f)$ decreases by a factor of $(1 - \frac{1}{n})$. □

Theorem 9 *Let* $C = \max_{(v,w)\in E} |c(v,w)|$*. Then the Goldberg-Tarjan algorithm finds a minimum cost circulation after canceling* $nm\log(nC)$ *cycles* ($\log = \log_e$)*.*

Proof: The initial circulation $f = 0$ is certainly C-optimal since, for $p = 0$, we have $c_p(v,w) \geq -C$. Therefore, by Theorem 8, the circulation obtained after $nm \log nC$ iterations is $\epsilon-$optimal where:

$$\epsilon \leq \left(1 - \frac{1}{n}\right)^{n\log(nC)} C < e^{-\log(nC)} C = \frac{C}{nC} = \frac{1}{n},$$

where we have used the fact that $(1-\frac{1}{n})^n < e^{-1}$ for all $n > 0$. The resulting circulation is therefore optimal by Theorem 5. □

The overall running time of the Goldberg-Tarjan algorithm is therefore $O(n^2m^2\log(nC))$ since the minimum mean cost cycle can be obtained in $O(nm)$ time.

9 A Faster Cycle-Canceling Algorithm

We can improve upon the algorithm presented in the previous sections by using a more flexible selection of cycles for canceling and explicitly maintaining potentials to help identify cycles for canceling. The idea is to use the potentials we get from the minimum mean cost cycle to compute the edge costs $c_p(v,w)$ and then push flow along all cycles with only negative cost edges. The algorithm Cancel and Tighten is described below.

Cancel and Tighten:

1. Cancel: As long as there exists a cycle Γ in G_f with $c_p(v,w) < 0, \forall (v,w) \in \Gamma$ push as much flow as possible along Γ.
2. Tighten: Compute a minimum mean cost cycle in G_f and update p.

We now show that the Cancel step results in canceling at most m cycles each iteration and the flow it gives is $(1 - 1/n)\epsilon(f)$ optimal.

Theorem 10 *Let* f *be a circulation and let* f' *be the circulation obtained by performing the Cancel step. Then we cancel at most* m *cycles to get* f' *and*

$$\epsilon(f') \leq (1 - \frac{1}{n})\epsilon(f).$$

Proof: Let p be such that $c_p(v,w) \geq -\epsilon(f)$ for all $(v,w) \in E_f$. Let Γ be any cycle in f' and let l be the length of Γ. We know that canceling a cycle removes at least one arc with negative reduced cost from the residual graph and creates only arcs with positive reduced cost. Therefore we can cancel at most m cycles. Now $G_{f'}$

has no negative cycles therefore every cycle in $G_{f'}$ contains an edge (v,w) such that $c_p(v,w) \geq 0$. Hence the mean cost of Γ is at least:

$$\begin{aligned}\frac{\sum_{(v,w)\in\Gamma} c_p(v,w)}{l} &\geq \frac{-(l-1)}{l}\epsilon(f) \\ &= -(1-\frac{1}{l})\epsilon(f) \geq -(1-\frac{1}{n})\epsilon(f),\end{aligned}$$

□

The above result implies that the Cancel and Tighten procedure finds a minimum cost circulation in at most $n\log(nC)$ iterations (by an analysis which is a replication of Theorem 9). It also takes us $O(n)$ time to find a cycle on the admissible graph. This implies that each Cancel step takes $O(nm)$ steps due to the fact that we cancel at most m cycles and thus a running time of $O(nm)$ for one iteration of the Cancel and Tighten Algorithm. Therefore the overall running time of Cancel and Tighten is $O(n^2 m \log(nC))$ (i.e. an amortized time of $O(n)$ per cycle canceled). We can further improve this by using dynamic trees [14] to get an amortized time of $O(\log n)$ per cycle canceled and this results in an $O(nm\log n\log(nC))$ algorithm.

10 Alternative Analysis: A Strongly Polynomial Bound

In this section, we give another analysis of the algorithm. This analysis has the advantage of showing that the number of iterations is strongly polynomial, i.e. that it is polynomial in n and m and does not depend on C. The first strongly polynomial algorithm for the minimum cost circulation problem is due to Tardos [19].

Definition 3 *An arc $(v,w) \in E$ is $\epsilon-$fixed if $f(v,w)$ is the same for all $\epsilon-$optimal circulations.*

There exists a simple criterion for deciding whether an arc is $\epsilon-$fixed.

Theorem 11 *Let $\epsilon > 0$. Let f be a circulation and p be node potentials such that f is ϵ-optimal with respect to p. If $|c_p(v,w)| \geq 2n\epsilon$ then $(v,w)$is $\epsilon-$fixed.*

Proof: The proof is by contradiction. Let f' be an ϵ-optimal circulation for which $f'(v,w) \neq f(v,w)$. Assume that $|c_p(v,w)| \geq 2n\epsilon$. Without loss of generality, we can assume by antisymmetry that $c_p(v,w) \leq -2n\epsilon$. Hence $(v,w) \notin E_f$, i.e. $f(v,w) = u(v,w)$. This implies that $f'(v,w) < f(v,w)$. Let $E_< = \{(x,y) \in E : f'(x,y) < f(x,y)\}$.

Claim 12 *There exists a cycle Γ in $(V, E_<)$ that contains (v,w).*

Proof: Since $(v,w) \in E_<$, it is sufficient to prove the existence of a directed path from w to v in $(V, E_<)$. Let $S \subseteq V$ be the nodes reachable from w in $(V, E_<)$. Assume $v \notin S$. By flow conservation, we have

$$\sum_{x \in S, y \notin S} (f(x,y) - f'(x,y)) = \sum_{x \in S} \sum_{y \in V} (f(x,y) - f'(x,y)) = 0.$$

However, $f(v,w) - f'(v,w) > 0$, i.e. $f(w,v) - f'(w,v) < 0$, and by assumption $w \in S$ and $v \notin S$. Therefore, there must exists $x \in S$ and $y \notin S$ such that $f(x,y) - f'(x,y) > 0$, implying that $(x,y) \in E_<$. This contradicts the fact that $y \notin S$. $\square$

By definition of $E_<$, we have that $E_< \subseteq E_{f'}$. Hence, the mean cost of Γ is at least $\mu(f') = -\epsilon(f') = -\epsilon$. On the other hand, the mean cost of Γ is ($l = |\Gamma|$):

$$\begin{aligned} \frac{c(\Gamma)}{l} &= \frac{c_p(\Gamma)}{l} = \frac{1}{l}\left(c_p(v,w) + \sum_{(x,y) \in \Gamma \setminus \{(v,w)\}} c_p(x,y)\right) \\ &\leq \frac{1}{l}(-2n\epsilon + (l-1)\epsilon) < \frac{1}{l}(-l\epsilon) = -\epsilon, \end{aligned}$$

a contradiction. $\square$

Theorem 13 *The Goldberg-Tarjan algorithm terminates after $O(m^2 n \log n)$ iterations.*

Proof: If an arc becomes fixed during the execution of the algorithm, then it will remain fixed since $\epsilon(f)$ does not increase. We claim that, as long as the algorithm has not terminated, one additional arc becomes fixed after $O(mn \log n)$ iterations. Let f be the current circulation and let Γ be the first cycle canceled. After $mn \log(2n)$ iterations, we obtain a circulation f' with

$$\epsilon(f') \leq \left(1 - \frac{1}{n}\right)^{n \log(2n)} \epsilon(f) < e^{-\log(2n)} \epsilon(f) = \frac{\epsilon(f)}{2n}$$

by Theorem 10. Let p' be potentials for which f' satisfies the $\epsilon(f')$-optimality constraints. By definition of Γ,

$$-\epsilon(f) = \frac{c_{p'}(\Gamma)}{|\Gamma|}.$$

Hence,

$$\frac{c_{p'}(\Gamma)}{|\Gamma|} < -2n\epsilon(f').$$

Therefore, there exists $(v,w) \in \Gamma$ such that $|c_{p'}(v,w)| > -2n\epsilon(f')$. By the previous Theorem, (v,w) is $\epsilon(f')-$fixed. Moreover, (v,w) is not $\epsilon(f)-$fixed since canceling Γ increased the flow on (v,w). This proves that, after $mn \log(2n)$ iterations, one additional arc becomes fixed and therefore the algorithm terminates in $m^2 n \log(2n)$ iterations. $\square$

Using the $O(mn)$ algorithm for the minimum mean cost cycle problem, we obtain a $O(m^3n^2 \log n)$ algorithm for the minimum cost circulation problem. Using the Cancel and Tighten improvement we obtain a running time of $O(m^2n^2 \log n)$. And if we implement Cancel and Tighten with the dynamic trees data structure we get a running time of $O(m^2 n \log^2 n)$.

The best known strongly polynomial algorithm for the minimum cost circulation problem is due to Orlin [18] and runs in $O(m \log n(m + n \log n)) = O(m^2 \log n + mn \log^2 n)$ time.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. Some recent advances in network flows. *SIAM Review*, 33:175–219, 1991.

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

[3] R. K. Ahuja, K. Mehlhorn, J. B. Orlin, and R. E. Tarjan. Faster algorithms for the shortest path problem. *Journal of the ACM*, 37:213–223, 1990.

[4] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.

[5] R. Dial. Shortest path forest with topological ordering. *Communications of the ACM*, 12:632–633, 1969.

[6] E. W. Dijkstra. A note on two problems in connection with graphs. *Numer. Math.*, 1:269–271, 1959.

[7] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19:248–264, 1972.

[8] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canad. J. Math.*, 8:399–404, 1956.

[9] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton Univ. Press, Princeton, NJ, 1963.

[10] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization problems. *Journal of the ACM*, 34:569–615, 1987.

[11] H. N. Gabow. Scaling algorithms for network problems. *Journal of Computer and System Sciences*, 31:148–168, 1985.

[12] A. V. Goldberg, E. Tardos, and R. E. Tarjan. Network flow algorithms. In B. Korte, L. Lovasz, H. J. Promel, and A. Schrijver, editors, *Paths, flows, and VLSI-layout*, volume 9 of *Algorithms and Combinatorics*, pages 101–164. Springer-Verlag, 1990.

[13] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. *Journal of the ACM*, 35:921–940, 1988. Preliminary version in Proc. 18th Symposium on Theory of Computing, pages 136–146, 1986.

[14] A. V. Goldberg and R. E. Tarjan. Finding minimum-cost circulations by canceling negative cycles. *Journal of the ACM*, 36:873–886, 1989. Preliminary version in Proceedings of the 20th Annual ACM Symposium on Theory of Computing, pages 388–397, 1987.

[15] D. Goldfarb and J. Hao. A primal simplex algorithm that solves the maximum flow problem in at most nm pivots and $o(n^2m)$ time. *Mathematical Programming*, 47:353–365, 1990.

[16] D. Johnson. A priority queue in which initialization and queue operations take $O(\log\log D)$. *Math. Systems Theory*, 15:295–309, 1982.

[17] M. Klein. A primal method for minimal cost flows with applications to the assignment and transportation problems. *Management Science*, 14:205–220, 1967.

[18] J. B. Orlin. A faster strongly polynomial minimum cost flow algorithm. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 377–387, 1988.

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

[20] R. E. Tarjan. Algorithms for maximum network flow. *Mathematical Programming Study*, 26:1–11, 1986.