6.854 Advanced Algorithms

Lecture 16: 10/11/2006 Lecturer: David Karger
Scribe: Kermin Fleming and Chris Crutchfield, based on notes by Wendy Chu and Tudor Leu

Minimum cost maximum flow, Minimum cost circulation, Cost/Capacity scaling

16.0.1 Introduction to minimum cost maximum flow

We previously discussed maximum flow in a network. Today we add one parameter to a flow network, a cost per unit of flow on each edge: $c(v, w) \in \mathbf{R}$, where $(v, w) \in E$.

Definition 1 *The* **cost of a flow** f *is defined as:*

$$c(f) = \sum_{e \in E} f(e) \cdot c(e)$$

A **minimum cost maximum flow** of a network $G = (V, E)$ is a maximum flow with the smallest possible cost. This problem combines maximum flow (getting as much flow as possible from the source to the sink) with shortest path (reaching from the source to the sink with minimum cost).

Note that in a network with costs the residual edges also have costs. Consider an edge (v, w) with capacity $u(v, w)$, cost per unit flow $c(v, w)$ and net flow $f(v, w)$. Then the residual graph has two edges corresponding to (v, w). The first edge is (v, w) with capacity $u(v, w) - f(v, w)$ and cost $c(v, w)$, and second edge is (w, v) with capacity $f(v, w)$ and cost $-c(v, w)$.

Observation 1 *Any flow can be decomposed into paths (some of which can be cycles). We define the cost of a path p as $c(p) = \sum_{e \in p} c(e)$ and express the cost of a flow f as $c(f) = \sum_{p \in P} c(p) f(p)$, where P is the path decomposition of f.*

16.0.2 The min-cost circulation problem

Consider a network without a source or a sink. We can define a flow in this network, as long as it is balanced at every node in that network. This kind of flow is called a **circulation**. The cost of a circulation is defined identically with the cost of a flow.

Observation 2 *Any circulation can be decomposed entirely into cycles. The cost of a circulation f can be expressed as the sum of the costs of all cycles in a decomposition of f.*

A **minimum cost circulation** is a circulation of the smallest possible cost. Note that there is no restriction on the flow through the network. For example, if all costs are positive, the minimum

circulation has no flow on all edges. On the other hand, if there are negative cost cycles in the network, the minimum circulation has negative costs and flow has to exist on the edges of the cycle.

Claim 1 *Finding the minimum cost maximum flow of a network is an equivalent problem with finding the minimum cost circulation.*

Proof: First, we show that min-cost max-flow can be solved using min-cost circulation. Given a network G with a source s and a sink t, add an edge (t, s) to the network such that $u(t, s) = mU$ and $c(t, s) = -(C + 1)n$. The minimum cost circulation in the new graph will use to the maximum the very inexpensive newly added edge. Any path from s to t forms a negative cost cycle together with (t, s), since $-c(t, s)$ is greater than the cost of any such path. This guarantees that we obtain a maximum flow from s to t "included" in the circulation of the new network. Among all maximum flows, this one is also of minimum cost. All the maximum flows use (t, s) at the same capacity, so they use the edge (t, s) at the same cost. This means that the minimum cost circulation has to be minimum cost on the section from s to t, which makes the max-flow also min-cost.

Another reduction from min-cost max-flow to min-cost circulation is to find any maximum flow in the network, regardless of the costs, then find the min-cost circulation in the residual graph. We claim that the resulted flow is a min-cost max-flow. This is because *the difference between two max-flows is a circulation*, and the cost of that difference circulation is the difference between the costs of the two max-flows. Given f, the initial max-flow, and f^*, the resulting maximum flow, $f - f^*$ is a min-cost circulation in the residual network G_f iff f^* is a min-cost max-flow.

The second part of the proof is showing that min-cost circulation reduces to min-cost max-flow. Consider a network G for which we want to find a min-cost circulation. Add a source s and a sink t to the network, without any edges to the rest of the network. The maximum flow in this network is 0, therefore the min-cost max-flow is actually a min-cost circulation.

We conclude then that min-cost max-flow and min-cost circulation are equivalent problems. ■

16.0.3 Optimality criteria

We are interested to find criteria which determine whether a circulation is a min-cost circulation.

Theorem 1 *A circulation is* **optimal (min-cost)** *iff there are no negative cost cycles in the residual network.*

Proof: First, suppose that a circulation f is not optimal, and let $f*$ be an optimal circulation in a network G. We will show that G_f has a negative cost cycle. The difference $f^* - f$ is a circulation, therefore it has a cycle decomposition. Because the cost of f^* is smaller than the cost of f, $f^* - f$ is a circulation of negative cost, and it is also feasible in G_f. At least one of the cycles in the decomposition has to be negative, therefore G_f contains a negative cost cycle.

To prove the other implication, suppose a residual network G_f has a negative cycle. Then f is not a min-cost circulation, because that cycle can be added to f, forming a new circulation of smaller cost. ■

The optimality criteria is checked by looking for negative cost cycles. This can be done with the Bellman-Ford shortest-path algorithm, which can handle negative cost edges (unlike Dijkstra's algorithm), but runs in $O(mn)$.

Price function

We can analyze the optimality of a circulation using a price function. Think of the flow units as widgets that are given away at the source and they are paid for at the source. There is a market for widgets at intermediate vertices.

We can define then a price function p for the vertices of the network. At the source, $p(s) = 0$. Consider an edge (v, w) which has residual capacity. The price $p(w)$ is *feasible* if $p(w) \leq p(w) + c(v, w)$.

Definition 2 *The* **reduced cost** *of an edge* (v, w) *is* $c_p(v, w) = c(v, w) + p(v) - p(w)$.

We can think of the reduced cost as the cost of buying a widget at v, shipping it to w and selling it there. Note that if $c_p(v, w)$ is positive, we would therefore not ship the item from v to w.

Using this definition, we can say that a price function is feasible for a residual graph if no residual edge has a negative reduced cost.

Observation 3 *Prices do not affect the value or structure of a minimum cost circulation.*

As we discussed before, a circulation is decomposed into cycles. Cycle costs do not change if we compute them as the sum of reduced costs of the edges, since the price terms around the cycle cancel out.

Claim 2 *A circulation is optimal iff there is a feasible price function in the residual graph*

Proof:
1) If there is a feasible price function for the residual graph, then the circulation is optimal.
If there is a feasible price function, then no residual edge has negative reduced cost. Then there is no negative cost cycle in the residual graph, therefore the circulation is optimal.

2) If the circulation has minimum cost, then there is a feasible price function for the residual graph. Add a source s' to the residual graph, along with edges of cost 0 to all other vertices. Compute shortest paths $d(v)$ from s' to each vertex v. The distances may be negative, but they are finite, since there are no negative cost cycles (the circulation is optimal).

We claim that we can use the distances as prices. Since d is the shortest-path distance function, $d(w) \leq d(v) + c(v, w)$ if (v, w) is in the residual graph G_f, so d is a feasible price function. ■

16.1 Algorithms

16.1.1 Cycle canceling

To find the minimum cost circulation, look for negative cycles and saturate them until done. Negative cycles can be found with the Bellman-Ford algorithm in $O(mn)$ time.

The number of iterations is less than the cost of the min-cost circulation (taken with a plus sign), because each negative cycle decreases the cost by at least one unit. The minimum cost of a circulation is bounded by $-mUC$, so the total time of this algorithm is $O(m^2nUC)$, a pseudo-polynomial bound. By using a scaling algorithm for shortest paths, we can obtain a running time of $O(m^2\sqrt{n}CU\log C)$.

16.1.2 Shortest Augmenting Path for Unit Capacity Graphs

The shortest augmenting path algorithm for solving the MCF problem is the natural extension of the SAP algorithm for the max flow problem. Note that here the shortest path is defined by edge cost, not edge capacity.

For the unit capacity graph case, we assume that all arcs have unit capacity and that there are no negative cost arcs. Therefore, the value of any flow in the cycle must be less than or equal to n. Given that each augmenting path increases the value of the flow by 1, at most n augmentation steps will suffice in finding the MCF.

Shortest augmenting paths can be found using any single-source shortest path algorithm. We can use Dijkstra's algorithm since there are no negative-cost edges in the graph. Each path calculation takes $O(m\log n)$ time, for a total runtime of $O(nm\log n)$.

Two questions arise:

- what if augmentations create negative cost edges?
- how do we know the result is a MCF?

We answer both of these questions with the following claim.

Claim 3 *Under the SAP algorithm, there will never be a negative reduced-cost cycle in the residual graph.*

Proof: (by induction). We want to show that one SAP doesn't introduce negative cycles in G_f. Initially there are no negative cost cycles. Feasible prices can be computed by using shortest path distances from s. After finding the shortest s-t path, it has reduced cost 0. Every arc on the path has reduced length 0. This demonstrates that the triangle inequality property is tight on shortest path edges. When we augment along the path, therefore, the residual backwards arcs we create are of reduced cost 0. Therefore in the new G_f, the price function is still feasible. Furthermore, there are:

- no residual negative reduced cost arcs
- no negative reduced cost cycles
- no negative cost cycles

■

Proof of this claim also proves the correctness of the algorithm, since it will also apply to the residual graph at the time the algorithm terminates.

The SAP algorithm we present suffers from two limitations. It is applicable only to unit capacity graphs, and it cannot handle graphs with negative cost cycles.

16.1.3 Capacity Scaling

We can extend the SAP algorithm to general-capacity networks by scaling. During each scaling phase, we roll in one bit of precision, for a total of $O(\log U)$ phases.

At the end of each phase we have an MCF and a feasible price function. After rolling in the next bit, though, we can introduce residual capaicty on negative reduced cost arcs. This will cause the price function no longer to be feasible. We can correct this problem by sending flow along the negative arcs. This introduces flow excesses (of one unit) at some nodes and deficits (of one unit) at others. We use an MCF to send the excesses back to deficits.

Since each arc can create at most one unit of excess, total excess is at most m units and m SAPs will suffice in returning all excesses to deficits. Using Dijkstra's for finding SAPs as before, runtime per phase is $O(m^2 \log n)$. The total runtime of the algorithm is $O(m^2 \log n \log U)$.

16.2 Cost Scaling

An alternative method of solving for MCF in a general network is by scaling by costs, rather than capacities. This is useful for graphs with integral costs, since all cycles will have integer costs. The idea is to allow for slightly negative cost arcs and continuously improve on the price function. We introduce the idea of *ϵ-optimality*:

Definition 3 *A price function p is* ϵ-optimal *if for all residual arcs (i,j), $c_p(i,j) \geq -\epsilon$.*

We start with a max flow and a zero price function, which will be C-optimal. During each scaling phase, we go from an ϵ-optimal max flow to an $(\epsilon/2)$-optimal max flow. When can we terminate the algorithm?

Claim 4 *A $\frac{1}{n+1}$-optimal max flow is optimal.*

Proof: We start with the observation that the least negative cycle cost is -1 in a integral-cost graph. All cycles in the residual network cost at least $-\frac{n}{n+1}$, which is strictly larger than -1. Therefore the reduced cost of any residual cycle is at least $-\frac{n}{n+1}$, and a $\frac{1}{n+1}$-optimal max flow is optimal. ■

This implies that $O(\log_n C)$ scaling phases are required to obtain an optimal flow.

To get an $(\epsilon/2)$-optimal max flow from an ϵ-optimal max flow, we first saturate all negative-cost residual arcs. This makes all residual arcs have non-negative reduced cost, but introduces excesses and deficits into the network. We then use MCF to push the excesses back to the deficits, without allowing any edge costs to drop below $\epsilon/2$.

Using dynamic trees, the runtime of this algorithm is $O(mn \log n \log C)$.

16.3 State of the Art

The double-scaling algorithm combines cost- and capacity-scaling introduced here. It has the runtime of $O(mn \log C \log \log U)$.

Tardos' minimum mean-cost cycles algorithm ('85) is a strongly polynomial algorithm for MCF. The algorithm proceeds by finding the negative cycles in which the average cost per edge is most strongly negative. Thus short cycles of a particular negativity are preferred over long ones. The algorithm uses a cost scaling technique from the ideas of ϵ-optimality. After every m negative-cycle saturations, an edge becomes "frozen," meaning its flow value never changes again. The minimum mean-cost cycle algorithm has time bound $O(m^2 \text{ polylog } m)$. An algorithm due to Trajan and Goldberg cancels minimum average weight cycles, which can be found in polynomial time, to obtain a MCF after a polynomial number of cycles have been cancelled. This algorithm has runtime $O(n^2 m^3 (\log n))$.

16.4 References

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

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