	6.854 Advanced Algorithms	
Lecture 10: 10/04/2004		Lecturer: David Karger
	Scribe: Wendy Chang	

Min-Cost Flow Algorithms

10.1 Shortest Augmenting Paths: Unit Capacity Networks

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 1 *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.

10.2 MCF Scaling by Capacity: General Networks

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 $O(m^2 \log n \log U)$.

10.3 MCF Scaling by Cost

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 1 *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 phase, we go from an ϵ-optimal max flow to an $(\epsilon/2)$-optimal max flow. When can we terminate the algorithm?

Claim 2 *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. ■

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 $O(mn \log n \log C)$.

10.4 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 $O(m^2 \text{ polylog } m)$.