CMSC 651 Advanced Algorithms	Lecture 17
Lecturer: Samir Khuller	Apr 14, 1998

Notes by Samir Khuller.

1 Shortest Path based Algorithm for Min Cost Flows

Recall that we have a flow network, which is basically a directed graph. Each edge (i, j) has a cost c_{ij} and capacity u_{ij}. This is denoted by a tuple (c_{ij}, u_{ij}) for the edge (i, j). Nodes v have supplies/demands $b(v)$. If $b(v) > 0$ then v is a supply vertex with supply value $b(v)$. If $b(v) < 0$ then v is demand vertex with demand $-b(v)$. As before we assume that the total supply equals the total demand.

Our goal is to find a flow function $f : E \rightarrow \Re$ such that each vertex has a net outflow of $b(v)$ (in other words, supply nodes have a net outflow and demand nodes have a net inflow equal to their $|b(v)|$ value) and the total cost is minimized. As before, the flow on an edge cannot exceed the corresponding capacity. The *cost* of the flow function is defined as $\sum_{e \in E} f(e) \cdot c(e)$.

1.1 Review of Shortest Paths

A shortest path *tree* is a rooted tree with the property that the unique path from the root to any vertex is a shortest path to that vertex. (Notice that Dijkstra's algorithm implicitly computes such a tree while computing shortest paths.)

Shortest Path Verification: Given any rooted tree T, how does one verify that it is a shortest path tree? Suppose we use distances in the tree to compute distance labels $d(i)$ for each vertex i. For each directed edge (i, j) in the graph, we check the condition

$$c_{ij} + d(i) \geq d(j).$$

If this is true for all edges, we claim that T encodes shortest paths. It is easy to see that if for some edge $d(j) > d(i) + c_{ij}$ then we can find a shorter path from s to j by going through vertex i.

Consider any path $P = [s = v_0, v_1, v_2, \ldots, v_k]$ from s to v_k. $c(P) = \sum_{(v_i, v_{i+1})} c_{v_i v_{i+1}} \geq \sum_{(v_i, v_{i+1})}(d(i+1) - d(i)) \geq d(v_k) - d(v_0) \geq d(v_k)$. Hence path P is longer than the path in the tree from s to v_k.

We can define *reduced costs of edges* as follows: $c'_{ij} = c_{ij} + d(i) - d(j)$. It is easy to see that $c'_{ij} \geq 0$ and $c'_{ij} = 0$ for edges on a shortest path.

1.2 Min Cost Flows

Similar to the concept of reduced costs via distance labels, we can define the concept of reduced costs via *node potentials*. We define a function $\pi(i)$ that denotes the potential of vertex i. We define the reduced cost of edges in the flow network as $c^{\pi}_{ij} = c_{ij} + \pi(i) - \pi(j)$.

The proof of the following lemma is trivial.

Lemma 1.1 *For any path P from vertex k to ℓ, we have*

1. $\sum_{(i,j) \in P} c^{\pi}_{ij} = \sum_{(i,j) \in P} c_{ij} + \pi(k) - \pi(\ell)$.
2. *for any directed cycle W,* $\sum_{(i,j) \in W} c^{\pi}_{ij} = \sum_{(i,j) \in W} c_{ij}$.

Lemma 1.2 *(Reduced Cost Optimality) A feasible flow x is optimal if and only if there exist node potentials π such that $c^{\pi}_{ij} \geq 0$ for all edges in G_x.*

Proof:

We recall that a feasible flow x is optimal if and only if there are no negative cost cycles in G_x. If the flow is optimal, we consider G_x and define $\pi(i) = d(i)$ where $d(i)$ is the distance to i from an arbitrary vertex s (since the residual graph has no negative cost cycles, distances are well defined). By the shortest path property all edges have non negative reduced costs.

To prove the converse we see that if such node potentials exist, and we consider the total cost of any cycle W; by the previous lemma, the total length of this cycle is non-negative and the same as the total length of the cycle with the original cost function. Hence the residual graph has no negative cycles and the solution is optimal. $\square$

We define a pseudoflow f, as a flow function that meets the capacity requirement, but may not meet the demand(s). One can view this as a partial solution. The main idea will be to augment flow along shortest paths from a supply vertex to a demand vertex. We will eventually meet the full demand. We start with the zero flow as the pseudoflow, and in each iteration decrease the remaining total demand. In the end, we will have a flow function that satisfies reduced cost optimality.

The algorithm is based on the following two lemmas.

Lemma 1.3 *Let x be a pseudoflow that satisfies reduced cost optimality conditions with respect to node potentials π. Let vector d represent shortest path distances from a vertex s to all other nodes in G_x with respect to cost c^{π}_{ij}. The following properties are valid:*

1. *Pseudoflow x also satisfies reduced cost optimality with respect to node potentials $\pi' = \pi + d$.*
2. *The reduced costs $c^{\pi'}_{ij}$ are zero for all edges on shortest paths from s to nodes in G_x with costs c^{π}_{ij}.*

Proof:

We first show that $c^{\pi'}_{ij} \geq 0$. By definition, $c^{\pi'}_{ij} = c_{ij} + \pi'(i) - \pi'(j) = c_{ij} + \pi(i) + d(i) - \pi(j) - d(j) = c^{\pi}_{ij} + d(i) - d(j) \geq 0$ (by the shortest path property).

The last equation is met with equality for edges on shortest paths, so $c^{\pi'}_{ij} = 0$. $\square$

Lemma 1.4 *Suppose x is a pseudoflow satisfying reduced cost optimality. If we obtain x' from x by sending flow along a shortest path from s to some other demand node k, then x' also satisfies reduced cost optimality.*

Proof:

If we define potentials π' from π such that x satisfies reduced cost optimality with respect to both node potentials. When we saturate a shortest path, the reduced cost of this path is zero and we create edges in the residual graph in the opposite direction with zero cost. These still satisfy reduced cost optimality. $\square$

We can now discuss the algorithm. The main idea is to start with a pseudoflow of zero initially, and $\pi(i) = 0$ for each vertex. Since the initial costs are non-negative, the edges satisfy reduced cost optimality. We compute the new potential π' and saturate flow on a zero cost path from a supply to a demand vertex. (Pick the source s as a supply vertex to compute shortest paths.) We then recompute the new shortest paths, update the potentials and saturate the next zero cost shortest path from a supply to a demand node. In each step we update the excess remaining flow in the supply/demand nodes. The algorithm terminates when we convert our pseudoflow into a real flow.

proc Min cost flow(G);
    $x = 0; \pi = 0; \forall i \;\; e(i) = b(i)$;
    $E = \{i | e(i) > 0\}; D = \{i | e(i) < 0\}$;
    while $E \neq \emptyset$ do
        Select a node $k \in E, \ell \in D$;
        Find shortest distances $d(j)$ from k to nodes in G_x with edge costs c^{π};
        $\forall i \quad \pi(i) = \pi(i) + d(i)$;
        Let P be a shortest path from k to ℓ;
        $\delta = \min(e(k), e(\ell), \min(r_{ij}, (i,j) \in P))$;
        augment δ units of flow along P, update x, G_x, E, D etc.
    end-while
end proc;