ORIE 633 Network Flows October 30, 2007

Lecture 16

Lecturer: David P. Williamson *Scribe: Liaoruo Wang*

1 Efficient algorithms for minimum-cost circulations

1.1 Capacity scaling (cont.)

Last time we started discussing one last algorithm for the minimum-cost circulation problem. This algorithm maintains a 0-optimal pseudoflow, and gradually works towards primal optimality. As we discussed last time, we can modify the graph without affecting the optimal circulation so that we can push any amount of flow from any node to any other node.

The algorithm will maintain a pseudoflow with potentials p, and a parameter Δ ($= U$ initially), where

$$A_f(\Delta) = \{(i,j) \in A_f : u_{ij}^f \geq \Delta\}$$

$$S(\Delta) = \{i \in V : e_i^f \geq \Delta\}$$

$$T(\Delta) = \{i \in V : e_i^f \leq -\Delta\}$$

The idea is to enforce $c_{ij}^p \geq 0$ for all $(i,j) \in A_f(\Delta)$. Then we repeatedly move Δ units of flow from $S(\Delta)$ to $T(\Delta)$ until either $S(\Delta) = \emptyset$ or $T(\Delta) = \emptyset$ (which will imply that there is not too much excess left). Then divide Δ by 2 and repeat.

We showed that the algorithm will eventually reach a feasible, optimal circulation.

Claim 1 *When $\Delta < 1$, f is a feasible circulation and it is optimal.*

At the start of each iteration of the algorithm, we'll saturate any $(i,j) \in A_f(\Delta)$ such that $c_{ij}^p < 0$. Then as we move Δ units of flow from nodes in $S(\Delta)$ to those in $T(\Delta)$ on paths in $A_f(\Delta)$, we need to maintain $c_{ij}^p \geqslant 0$? How can we do this? What we do is we modify the potentials p such that we push flow only on arcs with $c_{ij}^p = 0$.

To do this, pick a start node $k \in S(\Delta)$. Let $\tilde{p}_i$ be the shortest distance from k to i using costs c_{ij}^p and arcs in $A_f(\Delta)$. Then $\tilde{p}_j \leqslant \tilde{p}_i + c_{ij}^p$, $\forall (i,j) \in A_f(\Delta)$. Suppose (i,j) is on the shortest path from k to $l \in T(\Delta)$. Then $\tilde{p}_j = \tilde{p}_i + c_{ij}^p$, since if $\tilde{p}_j < \tilde{p}_i + c_{ij}^p$, there would exist another, shorter path from k to l.

Observe that if we update potentials to $p'_i = p_i + \tilde{p}_i$, then

$$\begin{aligned} c_{ij}^p + \tilde{p}_i - \tilde{p}_j \geqslant 0 \qquad \forall (i,j) \in A_f(\Delta) \quad &\Leftrightarrow \quad c_{ij} + p_i - p_j + \tilde{p}_i - \tilde{p}_j \geqslant 0 \\ &\Leftrightarrow \quad c_{ij} + p'_i - p'_j \geqslant 0 \\ &\Leftrightarrow \quad c_{ij}^{p'} \geqslant 0 \end{aligned}$$

Furthermore, $c_{ij}^{p'} = 0$ for all $(i,j) \in A_f(\Delta)$ on the shortest path from k to l.

We can now give our algorithm.

Capacity Scaling (Edmonds, Karp '72, Orlin '88)

$f \leftarrow 0$; $p \leftarrow 0$; $\Delta \leftarrow U$
While $\Delta \geqslant 1$
  If $c^p_{ij} < 0$ for $(i,j) \in A_f(\Delta)$, saturate (i,j)
  While $S(\Delta) \neq \emptyset$ and $T(\Delta) \neq \emptyset$
    Pick any $k \in S(\Delta)$ and any $l \in T(\Delta)$
    Compute $\tilde{p}_i$= shortest path from k to i using costs c^p_{ij} and arcs in $A_f(\Delta)$
    Update $p_i \leftarrow p_i + \tilde{p}_i$
    Send Δ units of flow from k to l on the shortest path
  $\Delta \leftarrow \Delta/2$

Let $e^+ = \sum_{i:e_i>0} e_i$, $e^- = -\sum_{i:e_i<0} e_i$. Since $\sum_i e_i = 0$, $e^+ = e^-$. We'll show that the amount of excess is bounded, which will then imply the running time of the algorithm.

Lemma 2 *At the start of the inner while loop, $e^+ \leqslant 2\Delta(n+m)$.*

The lemma leads to the the following theorem giving the running time.

Theorem 3 *The algorithm requires $O(m \log U)$ shortest path computations, yielding a running time of $O((m \log U)(m + n \log n))$.*

Proof: We note that the outer loop requires $O(\log U)$ iterations. Since $e^+ \leqslant 2\Delta(n+m)$ and we move Δ units of flow each time, there are at most $O(m)$ iterations of the inner while loop for each outer iteration. Thus the algorithm requires $O(m) \cdot O(\log U) = O(m \log U)$ shortest path computations. □

Proof of Lemma 2: At the end of the inner while loop, either $S(\Delta) = \emptyset$ or $T(\Delta) = \emptyset$, which implies that either $e^+ \leqslant n\Delta$ or $e^- \leqslant n\Delta$. But as we saw above, $e^+ = e^-$, so therefore $e^+ \leqslant n\Delta$. After Δ is halved, $e^+ \leqslant 2n\Delta$.

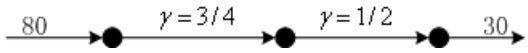
At the beginning of the outer loop, we saturate some arcs, which may also lead to increased excess. Note that any arcs with $u^f_{ij} \geqslant 2\Delta$ already had $c^p_{ij} \geqslant 0$, thus any newly saturated arc had $\Delta \leqslant u^f_{ij} \leqslant 2\Delta$. Therefore the increase in excess at the endpoints i and j is at most 2Δ. This implies that the total change in excess due to the saturating arcs is at most $2m\Delta$.

Therefore, at the start of the inner while loop, $e^+ \leqslant 2\Delta(n+m)$. □

2 Generalized flows

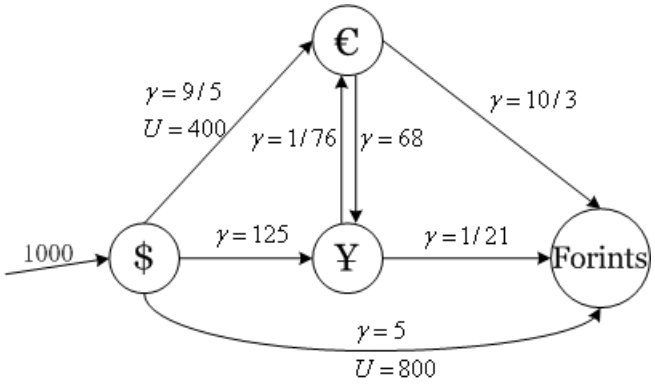
Now we consider a model in which the arcs are "lossy", so the flow sent along an arc is not the same as the flow arriving at the other end, but is scaled by a factor γ. This models leaks, theft, taxes, etc.

In the above graph, if we start with 80 units of flow, we obtain 60 units of flow after following the first arc and 30 units of flow after the second arc. The parameter γ is called the "gain" of the arc.



Gains can also model transformations of the flow. For example, another application of this model is currency conversion. Consider, for instance, the graph below in which we want to convert, say, \$1000 into Hungarian Forints. Besides the gain factor, we can also add capacity constraints to some arcs, for example we can convert at most \$400 directly into Euros. Note that some paths lead better rates than others; for example, the \$ $\rightarrow$ *Euro* $\rightarrow$ *Forint* path gives an exchange rate of 6 *Forints*/\$ as opposed to the direct path for which the rate is just 5.

E.g. Currency conversion



$$\frac{9€}{5\$} = \frac{10 \text{ Forints}}{3 €} = 6\frac{\text{Forints}}{\$}$$

2.1 Definitions

We now turn to a formal definition of the problem. We will study the generalized maximum flow problem. Our input is:

- Directed graph $G = (V, A)$
- A sink $t \in V$
- Capacities $u_{ij} \geqslant 0$, $\forall (i, j) \in A$, $u_{ij} \in \mathrm{Z}^+$
- Gains γ_{ij} (ratios of integers), $\forall (i, j) \in A \Rightarrow (j, i) \in A$ and $\gamma_{ji} = 1/\gamma_{ij}$

We need to give some definitions in order to define the goal of the problem. Essentially we want to maximize the net flow coming into the sink t, while maintaining flow conservation at all other nodes. Note that this is very different from other flow problems in which maintaining flow conservation at all nodes other than some node t would imply that flow is conserved at t as well.

Definition 1 *A* generalized pseudoflow *is* $g : A \to \Re$ *such that*

- $g_{ij} \leqslant u_{ij}$, $\forall (i,j) \in A$ *(capacity)*
- $g_{ij} = -\gamma_{ji} g_{ji}$, $\forall (i,j) \in A$ *(antisymmetry)*

Definition 2 *The* excess *of a pseudoflow* g *at a node* i *is* $e_i^g = -\sum_{j:(i,j)\in A} g_{ij}$.

Definition 3 *A* proper flow g *has* $e_i^g = 0$, $\forall i \neq t$.

We can now state the goal of the generalized maximum flow problem: it is to find a proper flow that maximizes e_t^g. Sometimes we denote $|g| = e_t^g$.

Definition 4 *Given a pseudoflow* g *in* G*, define the* residual graph $G_g = (V, A_g)$*:*

$$\begin{aligned} A_g &= \{(i,j) : g_{ij} < u_{ij}\} \\ u_{ij}^g &= u_{ij} - g_{ij} \end{aligned}$$

Definition 5 *For a path* P*, define the gain of the path as follows:*

$$\gamma(P) = \prod_{(i,j)\in P} \gamma_{ij}$$

Similarly for a cycle C*, the gain of the cycle is*

$$\gamma(C) = \prod_{(i,j)\in C} \gamma_{ij}$$

C *is* flow-generating *if* $\gamma(C) > 1$*, and* flow-absorbing *if* $\gamma(C < 1)$.

Definition 6 *A generalized augmenting path (GAP) is a flow-generating cycle* C *in the residual graph* G_g *together with a (possibly trivial) path* P *from a node on* C *to the sink* t.

It is not hard to see that if there exists a generalized augmenting path, then we can push flow around the flow-generating cycle, generating an excess, which can then be sent down the path to the sink. So if there is a GAP in the residual graph, the flow is clearly not optimal.

2.2 Optimality Conditions

Next time we will prove the following theorem. It will be useful in the proof (and in later algorithms) to introduce a third condition.

Theorem 4 *The following are equivalent:*

1. g *is a maximum proper flow*
2. *There are no GAPs in* G_g