Week 5

1 Network Flow

1.1 Max Flow

Given a network with one source node s and one sink node t and capacities on all arcs, find a maximum flow from s to t, i.e., maximize the supply from s, which is equal to the demand satisfied in t. Thus here supply and demand is partly not part of the input; all nodes not equal to s or t have $b_i = 0$. There are no costs on the arcs.

The famous algorithm of Ford and Fulkerson for this problem can be seen as a variation of the negative cost cycle algorithm, if we create an auxiliary arc (t, s) with cost -1 and let the cost on all other arcs be 0. Then, clearly to minimize total cost is equal to maximizing total flow from s to t. Also, given some feasible flow, any unsaturated negative cost cycle must contain arc (t, s) as a forward arc, and the rest of the cycle is an augmenting path from s to t in the algorithm of Ford and Fulkerson with $f_{ij} < u_{ij}$ for forward arcs and $f_{ij} > 0$ for backward arcs. Indeed, Ford and Fulkerson's algorithm inherits all properties of the negative cycle algorithm just proved for the min-cost flow problem. The only difference is that knowing one forward arc on any unsaturated negative cycle, if it exists, allows for a more efficient search for such a cycle in the max-flow problem than in the general problem.

Given a feasible flow f, to find an augmenting path from s to t, as before, the residual graph is constructed. In an exhaustive search the set S of all nodes that are reachable from s in the residual graph are found. $t \in S$ if and only if there exists an augmenting path. The search is done in such a way that as soon as t enters the set S, the path is also given. Along the path an augmentation of f is then performed as before.

There is a famous and beautiful duality result related to the max-flow problem, which proves correct termination if no augmenting path is found: the so-called *max flow - min cut* theorem.

An s-t-cut in a network is a subset S of the nodes $\mathcal{N}$, such that $s \in S$ and $t \notin S$. The capacity of cut S is

$$C(S) = \sum_{\{(i,j)\in\mathcal{A} | i \in S, j \notin S\}} u_{ij}.$$

Theorem 1.1 *The value of a maximum flow is equal to the value of the minimum cut capacity.*

Proof. I only prove the finite case, leaving the infinite case for yourself to read. Let v be the max-flow. $v \leq C(S)$ for every cut S, is evident since any flow has to cross an arc from S to $\mathcal{N} \setminus S$. Given an optimal flow there does not exist any augmenting path. Thus, in the residual graph of this flow t is not reachable from s. Consider the set S of reachable nodes. Then in the residual graph there does not exist an arc from any $i \in S$ to any $j \notin S$, otherwise j would have belonged to S. This means that $f_{ij} = u_{ij}$ for all $\{(i,j) \in \mathcal{A} \mid i \in S, j \notin S\}$ and $f_{ij} = 0$ for all $\{(i,j) \in \mathcal{A} \mid i \notin S, j \in S\}$. Thus, all the flow crossing from S to $\mathcal{N}$ does not return to S, hence must exit through t. Thus $v = C(S)$. $\square$

The complexity of Ford-Fulkerson's search for an augmented path can be implemented to run in $O(|\mathcal{A}|)$ per iteration. In case of integer capacities each iteration increases the s-t-flow by at least 1 (augmenting path relate to unsaturated cycles). Thus, termination will occur within $O(|\mathcal{N}|U)$ iterations, with $U = max_{ij} u_{ij}$.

The following example shows that it can indeed be as bad as this worst-case bound. Consider the network with 4 nodes, s,1,2 and t. We have the following arcs with their capacities: $u(s,1) = u(s,2) = u(1,t) = u(2,t) = U$ and $u(1,2) = 1$ (see for a picture Figure 9-8 in Section 9.2 in [PS]). In the first iteration the residual network is just the given network and it may be that the path found is $(s,1,2,t)$ on which we can augment the flow by 1: $f(s,1) = f(1,2) = f(2,t) = 1$. This gives the residual graph with forward arcs and capacities $u(s,1) = u(2,t) = U - 1$, and backward arcs with capacities $u(1,s) = u(t,2) = u(2,1) = 1$. In the next iteration in R_f the (s,t)-path $(s,2,1,t)$ may be found, again allowing to augment the flow by 1. This continues for $2U$ iterations until we reach the optimum solution $f(s,1) = f(1,t) = f(s,2) = f(2,t) = U$ and $f(1,2) = 0$. Clearly, we could have found this solution in only 2 augmentation steps, if we would have been more lucky.

By choosing among the augmenting paths the one with the smallest number of arcs makes the running time independent of U, and therefore strongly polynomial. I give you the argument below. It is not covered in the book. This would clearly have solved our bad example in just 2 iterations. Let us show that it is not only better for this example.

Given flow f and corresponding residual graph R. Calling s a level-0 node, we call all nodes reachable in one step from s level-1 nodes, etc. In this way in $O(|\mathcal{A}_R|) = O(|\mathcal{A}|)$ steps we find out that node t is a level-k node, meaning that the shortest path from s to t contains k arcs in R. We keep all vertices and arcs that are on any length-k path from s to t in R and call the resulting restricted graph R_r. Each arc in R_r goes from a level-i node to a level-i+1 node, for some i.

Take one such an (s,t)-path in R_r. Then after augmentation one of the arcs will have disappeared from R_r, either by saturation or by emptying. It could happen that the reverse arc comes in its place, but this goes from a higher-level node to a lower-level node, and hence is not part of the new R_r. Thus after at most $|\mathcal{A}_{R_r}|$ augmentations R_r will be empty, implying that given the flow found at that point there are no (s,t)-paths in the residual network of length at most k.

This implies that in the most straightforward implementation, in which we rebuild R_r only once, but do a path search every iteration, we require $O(|\mathcal{A}_{R_r}||\mathcal{A}_{R_r}|) = O(|\mathcal{A}|^2)$ operations for augmenting along all (s,t)-paths of length k.

Clearly, we will need to consider at most $|\mathcal{N}|$ different lengths of augmenting paths. Thus, overall the algorithm takes $O(|\mathcal{A}|^2|\mathcal{N}|)$ time. This can be slightly improved to $O(|\mathcal{N}|^3)$ time by augmenting over several paths in one round, in any such round saturating all incoming arcs or all outgoing arcs of some node, thereby cutting out the node from R_r. Details you can for example read in C.H. Papadimtriou, K. Steiglitz, *Combinatorial Optimisation: Algorithms and Complexity*, Prentice Hall, 1982.

1.2 Dual ascent and Primal dual

We return to the uncapacitated min-cost flow problem

$$\begin{array}{ll} \min & c^T f \\ \text{s.t.} & Af = b \\ & 0 \leq f. \end{array} \tag{1}$$

To obtain a feasible solution as a start of the simplex method or the negative cycle algorithm we can solve the following max flow problem.
Let $Q_+ = \{i \mid b_i > 0\}$ and $Q_- = \{i \mid b_i < 0\}$. Create the equivalent problem with only one auxiliary source node s and one auxiliary sink node t, and auxiliary arcs (s,i) with capacity b_i for every $i \in Q_+$ and auxiliary arcs (i,t) with capacity $|b_i|$ for $i \in Q_-$. Capacities on all other arcs remain infinite. Call this network G_{st}. Verify for yourself that a maximum flow for this network defines flow on all the original arcs of the network that correspond to a feasible solution to the original min-cost flow problem.

However, we will incorporate computing the maximum flow on subgraphs of the graph G_{st} within a primal-dual algorithm, which maintains dual feasibility and satisfaction of the complementary slack constraints. As often in primal dual algorithms, in various iterations tight constraints in the dual give us arcs on which we may set positive flow (to maintain complementary slackness). This implies, that as soon as we have a feasible solution for the primal problem, then we immediately have an optimal solution. Iteratively the dual solution is improved in a dual ascent way.

Dual ascent methods work on the dual problem

$$\begin{array}{ll} \max & \sum_{i=1}^n p_i b_i \\ \text{s.t.} & p_i \leq c_{ij} + p_j, \ \forall (i,j) \in \mathcal{A}. \end{array} \tag{2}$$

In each iteration there is a dual feasible solution and the algorithm checks wether there exists a direction d with $p + \theta d$ is another feasible solution and $d^T b > 0$. Such a direction is called a *dual ascent direction.* It so happens that for the search for such directions d we can concentrate on those that are of the form

$$d_i = \begin{cases} 1, & \text{if } i \in S \\ 0, & \text{otherwise} \end{cases}$$

for some subset $S \subset \mathcal{N}$. Such a direction, which we write as d^S is called an *elementary direction.* We emphasize that s and t do not have a coordinate in this dual ascent direction.

In the algorithm that we study here we maintain dual feasibility and the complementary slack relations.

$$f_{ij}(c_{ij} - (p_i - p_j)) = 0, \ \forall (i,j) \in \mathcal{A}.$$

We define the set of so-called *balanced* arcs as those arcs (i,j) that satisfy

$$p_i = c_{ij} + p_j$$

According to complementary slackness, these are the only arcs in an optimal *primal* solution that can carry positive flow.

Now consider again the graph G_{st} and delete from G_{st} all arcs that are not balanced, yielding the network $G_{balanced}$, that next to all the auxiliary arcs, i.e. the ones from s and to t, includes only balanced arcs. Solve the max flow problem on this network. If the max flow happens to be $\sum_{\{i|b_i>0\}} b_i$ then we have a feasible flow and a dual solution that satisfy the complementary slackness conditions and therefore it is optimal. In this case, at termination the cut found is just s since all the arcs going out of s are saturated, i.e., used to their full capacity.

Otherwise at termination, the max flow algorithm will give us a cut S, containing a subset of Q_+. The claim is that d^S is a dual ascent direction.

Feasibility. Take into account that there is no coefficient for s in d^S. Take any arc $(i,j) \in \mathcal{A}$ with $i \in S$, $i \neq s$, and $j \notin S$, $j \neq t$. These are the only arcs we have to worry about for feasibility. Because the capacity on (i,j) is infinite (hence it cannot be saturated), this implies that (i,j) simply does not exist in $G_{balanced}$ (otherwise j would have been in S as well). Thus (i,j) is not a balanced arc; i.e. $p_i < c_{ij} + p_j$. Hence, indeed d^S is a feasible direction.

Improvement. Again, because of the infinite capacities of the non-auxiliary arcs, we will argue that the flow from s into nodes in $S \cap Q_+$, $\sum_{i \in S \cap Q_+} f_{si}$, must reach nodes in $S \cap Q_-$. First we argue that there is no flow from $\mathcal{N} \setminus S$ to S. Notice that for any arc $(i,j) \in \mathcal{A}$ with $i \notin S$ and $j \in S$ we must have $f_{ij} = 0$

(again otherwise (j,i) would be an arc in the residual graph and i would have been in S as well).

Thus, because of flow-conservation on all intermediate nodes the flow from s into nodes in $S \cap Q_+$ must reach nodes in $S \cap Q_-$. Since there does not exist an augmenting path, this flow from the nodes in $S \cap Q_-$ to t must saturate all the arcs $\{(j,t) \mid j \in S \cap Q_-\}$. Thus,

$$\sum_{i \in S \cap Q_+} f_{si} = \sum_{j \in S \cap Q_-} |b_j|.$$

Now,

$$(d^S)^T b = \sum_{i \in S \cap Q_+} b_i + \sum_{j \in S \cap Q_-} b_j > \sum_{i \in S \cap Q_+} f_{si} - \sum_{j \in S \cap Q_-} |b_j| = 0.$$

The strict inequality is due to the fact that $S \cap Q_+ \neq \emptyset$ implying that there exists an $i \in S \cap Q_+$ for which the arc (s,i) is not saturated: $f_{si} < b_i$.

What we just showed is

Theorem 1.2 *A dual solution p is optimal or there exists an elementary direction for improvement.*

Given such an improving d^S we reset $p \to p + \theta d^S$ and we choose θ as large as possible, i.e., such that all dual restrictions remain satisfied. This is automatically true for i,j both in S or i,j both not in S. *In particular, this implies that all arcs that were balanced remain balanced.* Also, if $i \notin S$ and $j \in S$ the dual constraint remain satisfied: $p_i \leq c_{ij} + p_j + \theta$. Thus we only need to take care that

$$p_i + \theta \leq c_{ij} + p_j, \ \forall i \in S, \ j \notin S, \ (i,j) \in \mathcal{A}.$$

We therefore choose

$$\theta^* = \min_{i \in S, \ j \notin S, \ (i,j) \in \mathcal{A}} \{c_{ij} + p_j - p_i\}.$$

In doing so, at least one extra arc becomes balanced, and we start a new max flow search in the larger graph. We continue the process until, either a flow augmentation has been found, in which case we augment the flow and restart the search for a new augmenting path, or no elementary dual ascent direction is found, i.e., $S = \{s\}$. This implies that all arcs out of s are saturated. Hence, we have found a primal feasible flow, and we have a dual feasible solution and complementary slackness holds. Therefore we conclude optimality.

This is the description of the so-called primal-dual algorithm. A little thought should make it clear that after a dual update, we don't need to compute a flow from scratch, since the update simply adds some extra arcs with infinite capacity to the network, making extra nodes reachable from s, next to the ones that were already reachable before the update. The search for the extra reachable nodes can therefore be started with the cut from before the update.

Assuming that the total supply (is total demand) is B, there are at most B flow augmentations. In between two consecutive augmentations there can be at most n dual updates (at most n nodes can become reachable), giving in total at most nB dual updates. Straightforward calculation of θ^* in each update would cost $O(m)$ making the total running time $O(mnB)$. Using the shortest path augmentations the running time becomes strongly polynomial.

Material of Week 5 from [B& T]

Chapter 7, 7.5 – 7.7. Sections 7.8 – 7.10 are left to yourselves to read, but will not be part of the exam.

Exercises of Week 5

7.19, 7.20, 7.28, 7.29, and you may try 7.30.

Next time

Chapter 8