CSC 373 H 1 Y — Summer 2007

University of Toronto — St. George Campus

Lecture Summary for Week 7

This summary is not a replacement for the lecture. If you miss a class, please arrange with a friend to take note for you.

3.6 Ford-Fulkerson Algorithm (cont'd)

Our task now is to prove that when all capacities are integers, the Ford-Fulkerson algorithm terminates and is correct. We prove the following theorem:

Theorem: Assume that all capacities are integers, then in each iteration of the While loop all flows on the edges are integers.

Proof: We prove the Theorem by induction (on the iteration).

Base case We start with a 0-flow. At this time all flows on the edges are 0, so the statement is true.

Induction step Consider the iteration $(i+1)$. By the induction hypothesis for iteration i, the flows on the edges before we start the $(i+1)$ iteration are integers, so the residual network has integer capacities. Therefore the bottleneck of the augmenting path is an integer. As a result, the increase in flow value is an integer; therefore the new flows are all integers. QED

Notice that the flow value always increases after each iteration, and since it can take only integer values, the while-loop must terminate when the flow get the maximum value. Therefore the algorithm terminates. When it terminates, there is no augmenting path in the residual network, so the flow obtained must have maximum value. Below we analyze the running time of the algorithm.

Corollary Suppose that all capacities of the network are integers. Let $m = |E|$, and

$$C = \sum_{e \text{ out of } s} c(e)$$

Then

a) The Ford-Fulkerson algorithm runs in time $\mathcal{O}(|E|C)$.

b) In the output of the Ford-Fulkerson algorithm, all flows on the edges are integers.

Proof (a) Since the value of the flow is an integer and increases in each iteration, it must increase by at least one after each iteration. Since C is at least the min cut capacity, the maximum flow value is bounded above by C. So there can be at most C iterations of the while-loop.

Each iteration can be done in time $\mathcal{O}(|E|)$ (looking for an st-path can be done by breadth-first-search or depth-first-search, which take time $\mathcal{O}(|E|+|V|)$—here $|E| \geq |V|$). Thus the total running time of Ford-Fulkerson algorithm is $\mathcal{O}(|E|C)$.

Part (b) follows from the Theorem above. QED.

If the augmenting path in step 3 is found using a BFS, it is also a shortest path from s to t (i.e., an st-path with smallest number of edges). In this case, it can be proved that there are at most $\mathcal{O}(|E||V|)$ iterations. Thus the total running time of the algorithm is $\mathcal{O}(|E|^2|V|)$. The particular

implementation of Ford-Fulkerson algorithm where the augmenting path is also the shortest st-path is known as Edmonds-Karp algorithm.

A number of other algorithm for flow network are proved to have better upper bound on the running time. These include Preflow-Push Maximum-Flow algorithm (running time $\mathcal{O}(|V|^3)$).

In many cases, the upper bound $\mathcal{O}(C|E|)$ given in the Corollary is better than other upper bounds above. For example, for the Bipartite Matching problem, $C = n$, and we have an upper bound $\mathcal{O}(nm)$ where m is the number of edges in the bipartite graph. The upper bound given by Edmonds-Karp algorithm is $\mathcal{O}(m^2 n)$, which is worse than $\mathcal{O}(nm)$.

3.7 Project Selection [K& T 7.11]

The problem is as follows:

A company needs to choose a number of projects from a set P of projects. Each project $i \in P$ is associated with a revenue p_i which can be positive (meaning the project is profitable) or negative (meaning the project needs some expense). Certain projects are prerequisites for some others. For example, if project j is a prerequisite for project i, then project j must be selected if project i is to be selected.

A set $A \subseteq P$ is feasible if the prerequisites of all every project in A are also in A. The profit of a set A of project is the total revenue of all projects in A:

$$profit(A) = \sum_{i \in A} p_i$$

The company wishes to select a feasible set of projects with maximum profit.

To solve the problem, consider the graph $G = (P, E)$ whose set of vertices is P, and whose set of edges

$$E = \{(i, j) \ : \ j \text{ is a prerequisite of } \ i\}$$

Notice that a set A is feasible if and only if, as a set of vertices in G, there is no edge going out of A.

The above remark is very important in designing a flow network for this problem. In particular, we will design a flow network so that a min cut will give us a feasible set of maximum profit. To make sure that only feasible sets are possible candidates for min cut, we will assign large capacity for edges in G. (Thus if a set A has an out-going edge, it cannot form a min cut.)

In more details, consider the flow network obtained by adding a source s and target t to G. We connect s to all i such that $p_i > 0$, the capacity of such edge (s, i) is p_i. Also, if $p_i < 0$ we add edge (i, t) with capacity $-p_i$.

Now the cut $\{s\}, \{t\} \cup P)$ has capacity

$$C = \sum_{p_i > 0} p_i$$

Assign for each edge in G the capacity $C + 1$. By the above remark, no edge of G can cross a min cut.

Lemma: If $(\{s\} \cup A, \{t\} \cup B)$ is a cut with capacity $\leq C$, then A is a feasible set.

Proof: The proof follows from the discussion above: Since the capacity of the cut is less than the capacity of every edge in G, no edge of G goes from $\{s\} \cup A$ to $\{t\} \cup B$. In other words, no project in B is a prerequisite of any project in A. So A is a feasible set. QED

Lemma: If A is a feasible set, then the cut $(\{s\} \cup A, \{t\} \cup B)$ has capacity

$$C - \sum_{i \in A} p_i$$

Proof: There are no edges going from A to B, so the only edges crossing the cut are from s to B and from A to t. Therefore the capacity of the cut is

$$\begin{aligned} &\sum_{i \in B, p_i > 0} p_i + \sum_{j \in A, p_j < 0} (-p_j) \\ &= (\sum_{pi > 0} p_i - \sum i \in A, p_i > 0 p_i) - \sum_{j \in A, p_j < 0} p_j \\ &= C - \sum_{i \in A} p_i \end{aligned}$$

QED

The two Lemmas above show that if $(\{s\} \cup A, \{t\} \cup B)$ is a min cut of the flow network, then A is a feasible set with maximum total profit.

The algorithm: So to find a feasible set with maximum total profit, we do the following:

- Set up the flow network as above.
- Run a max flow algorithm to find a maximum flow in the network.
- Use the maximum flow to find a min cut (as in the proof of the Max Flow-Min Cut Theorem).

Here we can use the upper bound given by the Edmonds-Karp algorithm to find the max flow: $\mathcal{O}(|V||E|^2)$. The min cut (as found in the proof of the Max Flow-Min Cut Theorem) can be found by breadth-first-search, which takes time $\mathcal{O}(|V| + |E|)$. In total, the running time is $\mathcal{O}(|V||E|^2)$.