

CSC 373 H 1 Y — Summer 2006

University of Toronto — St. George Campus

Lecture Summary for Week 8

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

Tutorial this week: 1. A specific example of a flow network. 2. The *Escape Problem*, based on Exercise 14, chapter 7 in the text.

References for week 8 & week 9 lectures: Sections 7.1-6

Extra reading: Sections 7.8, 7.9, 7.11, 7.12.

The Bipartite Matching Problem

The scenario: In a dancing party, each girl has a list of preferred boys to dance with. A girl is “happy” as long as she is paired up with a boy in her list. The problem is to pair up boys and girls so that as many girls as possible are happy.

Definition A graph is called bipartite if the set of vertices can be partitioned into two disjoint subsets X and Y so that all edges have endpoints in different partitions.

A matching in a graph is just a set of edges where no two edges share a common endpoint. A matching is maximum if it has maximum cardinality.

For the above “dancing party” problem, if we represent each boy by a node in X , and each girl by a node in Y , and connect a point in X to a point in Y just in case the corresponding boy is in the preference list of the girl. Then the dancing party problem is the same as the Bipartite Matching Problem:

Given a bipartite graph, find a maximum matching.

Later we will see how to solve this problem using Flow Network.

Flow Network

Definition A *flow network* is a directed graph $G = (V, E)$ where

- associated with each edge e is a capacity $c(e) > 0$;
- there are two designated nodes in V : s (the source) and t (the sink) (nodes other than s, t are called *internal nodes*);
- there is no edge entering the source s , and no edge leaving the sink t ;
- for every internal node v , there is a path from s to v to t .

A flow f in the flow network G is a real function on the set of edges that satisfies:

1. **Capacity condition** $0 \leq f(e) \leq c(e)$ for every $e \in E$
2. **Conservative condition** the total flow into an internal node is the same as the total flow outgoing from it, i.e.,

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$$

for every internal node v .

Value of a flow The value $\text{Value}(f)$ of a flow f is defined to be the total flow out of the source:

$$\text{Value}(f) = \sum_{e \text{ out of } s} f(e)$$

st -Cut: An st -cut (or just cut) is a partition (A, B) of V where $s \in A$ and $t \in B$.

Capacity of a cut: The capacity of a cut (A, B) is defined to be

$$c(A, B) = \sum_{e \text{ out of } A} c(e)$$

Lemma 1 For any cut (A, B) :

$$\text{Value}(f) = f^{\text{out}}(A) - f^{\text{in}}(A)$$

where

$$f^{\text{out}}(A) = \sum_{e \text{ out of } A} f(e), \quad f^{\text{in}}(A) = \sum_{e \text{ into } A} f(e)$$

Proof

$$\begin{aligned} \text{Value}(f) &= f^{\text{out}}(s) \\ &= f^{\text{out}}(s) - f^{\text{in}}(s) \\ &= \sum_{v \in A} (f^{\text{out}}(v) - f^{\text{in}}(v)) \quad \text{by conservation} \\ &= \left(\sum_{v \in A} f^{\text{out}}(v) \right) - \left(\sum_{v \in A} f^{\text{in}}(v) \right) \\ &= \left(\sum_{e \text{ out of } A} f(e) + \sum_{e \text{ inside } A} f(e) \right) - \left(\sum_{e \text{ into } A} f(e) + \sum_{e \text{ inside } A} f(e) \right) \\ &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e) \\ &= f^{\text{out}}(A) - f^{\text{in}}(A) \quad \text{QED.} \end{aligned}$$

Corollary 1 $\text{Value}(f) = f^{\text{in}}(B) - f^{\text{out}}(B)$.

Corollary 2 $\text{Value}(f) = f^{\text{in}}(t)$.

Corollary 3 $\text{Value}(f) \leq c(A, B)$.

The Maximum Flow Problem

Given a flow network, find a flow of maximum value.

By Corollary 3 above, the value of a maximum flow is bounded above by the capacity of any cut, i.e., it is bounded above by the *min* cut.

When all capacities are integers, the Ford-Fulkerson algorithm (to be presented later) is guaranteed to produce a maximum flow. Furthermore, *all flows on the edges are integers*.

(Note that even when all capacities are integers, there are maximum flows where the flows on the edges are not integers.)

Solving the Bipartite Matching Problem

We follow the following steps:

1. Construct the flow network.
2. Specify the algorithm to find the maximum flow in the network (Ford-Fulkerson algorithm or other algorithms to be mentioned later). Describe the output.
3. Argue that the max flow returned by the algorithm can be used to construct the (optimal) solution to the original problem.
4. Construct the solution to the original problem from the maximum flow.

Running time: The running time is the total running time of steps 1, 2 and 4.

1. Construct the flow network The flow network G' is obtained from the bipartite graph G by

- The nodes in G' consist of the nodes in G and two new vertices: s and t .
- The edges in G' consist of
 - edges in G (with direction from X to Y) with capacity 1
 - edges from s to nodes in X with capacity 1
 - edges from nodes in Y to t with capacity 1

2. Run Ford-Fulkerson algorithm on the network. The output is a flow where on each edge, the flow is either 0 or 1.

3. Relationship between the above maxflow of G' and an optimal solution for the Matching problem in G The value of the above flow is the same as the maximum number of matching in G .

Proof Let f be the flow returned by Ford-Fulkerson algorithm. We show that $Value(f)$ is \leq the maximum number of matching in G by showing that there is a matching in G with $Value(f)$ edges.

The matching is defined to be the set of edges in G where the flow is 1. We show that no two edges in G with flow 1 can share an endpoint. This is because the capacity of the edge (s, v) is 1 for every $v \in X$, and the capacity of (u, t) is 1 for every $u \in Y$.

Now we show that the maximum number of matching in G must be $\leq Value(f)$. Let M be a maximum matching in G . Then we can define a flow f' in G' by letting for each edge (u, v) in M

$$f'(u, v) = 1, \quad f'(s, u) = 1, \quad f'(v, t) = 1$$

Exercise Verify that f' is a flow (check for the capacity condition and conservation condition).

Thus the number of edges in M is the same as $Value(f')$. Since $Value(f') \leq Value(f)$, we have that the number of edges in M is $\leq Value(f)$.

4. Construct a maximum matching in G From the proof above, a maximum matching in G can be obtained by selecting the edges in G with flow 1.

Running time See exercise next week.