CS420+500: Advanced Algorithm Design and Analysis

Lecture: February 3, 2017

Prof. Will Evans *Scribe: Gabriel Henderson*

In this lecture we:

- proved the correctness of the Ford-Fulkerson algorithm, and
- introduced maximum matchings in bipartite graphs.

1 Correctness of Ford-Fulkerson

Recall the following lemma from last class:

Lemma 1. *For any flow f and any cut (S, T), we have* $\text{size}(f) \leq \text{cap}(S, T)$.

We proved this using the following facts:

1. The flow across cut $(S, T) \leq \text{cap}(S, T)$.
2. The flow across cut $(S, T) =$ flow across cut $(S - v, T + v)$.
3. The flow across cut $(\{s\}, V \setminus \{s\}) = \text{size}(f)$.

Theorem 2. *If a residual network G^f has no augmenting path, then f is a maximum size flow.*

Proof. Let S be the set of all vertices reachable by a directed path in G^f. Let $T = V \setminus S$. Then all edges from S to T are saturated. This implies that the flow across cut $(S, T) = \text{cap}(S, T)$. Hence $\text{size}(f) = f(S, T) = \text{cap}(S, T)$ because $f(u, v) = c(u, v)$ for all $u \in S$, $v \in T$. By Lemma 1, any flow has size at most $\text{cap}(S, T) = \text{size}(f)$. Thus f is a maximum size flow. □

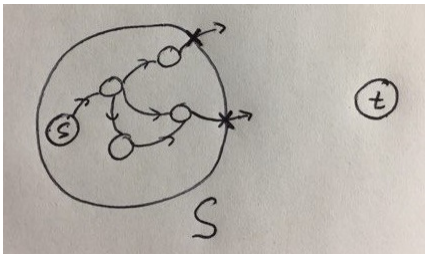

Figure 1: An illustration of how S was chosen.

Theorem 3. *(Max–flow, min–cut). Size of max–flow = capacity of min–capacity cut.*

Proof. The main idea is to use the proof of correctness of Ford-Fulkerson.

If f^* is the maximum size flow and (S^*, T^*) is the minimum capacity cut, we have from Lemma 1 that

$$\text{size}(f^*) \leq \text{cap}(S^*, T^*). \tag{a}$$

From Theorem 2, we have $\text{size}(f^*) = \text{cap}(S, T)$ where (S, T) is the cut defined in the proof of Theorem 2. Since (S^*, T^*) is the minimum capacity cut, we know that

$$\text{size}(f^*) = \text{cap}(S, T) \geq \text{cap}(S^*, T^*). \tag{b}$$

Together (a) and (b) imply the Theorem. □

Theorem 4. *(Integrality). If all capacities are integers, then there exists a max flow such that every edge has integer flow.*

Proof. By induction on the number of augmentations of Ford-Fulkerson. □

2 Maximum Matching in a Bipartite Graph

Definition 5. *A matching in a graph is a set of edges in the graph such that no two edges in the matching have a common endpoint.*

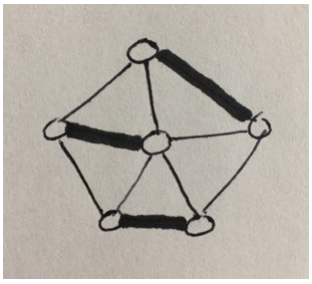

Figure 2: A maximum matching of the 6-wheel. Here the edges in the matching are shaded black.

Definition 6. *A bipartite graph is a graph whose vertices can be partitioned into vertices V_1 and V_2 such that for all edges, one endpoint is in V_1 and the other is in V_2.*

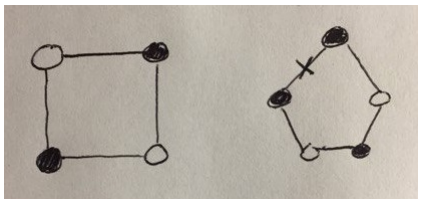

Figure 3: The 4-cycle on the left is a bipartite graph, whereas the 5-cycle on the right is not.

Why do we care about maximum matchings in bipartite graphs? This can be used to solve many problems, such as assigning workers to jobs, etc.

Remark 7. *A solution to the stable marriage problem is a maximum matching.*

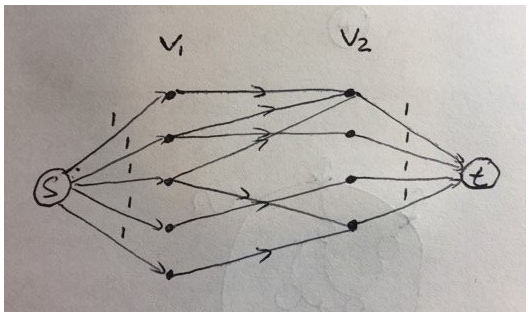

Figure 4: An illustration of assigning flow capacities to the worker–jobs problem.

Remark 8. *Given a graph, one can determine if it is bipartite and find the bipartition in linear time in the size of the graph (number of edges plus number of vertices) using breadth–first search*