CSL851: Algorithmic Graph Theory	Semester I 2013-2014
Lecture 1: July 24	
Lecturer: Naveen Garg	*Scribes: Suyash Roongta*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

Matching in Bipartite Graphs

Definition 1.1 *Given a graph G = (V,E), $M \subseteq E$ is said to be a* **matching** *if M is an independent set of edges, i.e., no two edges of M are incident on the same vertex.*

A single edge set is a trivial matching. We are interested in an algorithm to find the largest matching in a given bipartite graph. The algorithm that we are going to discuss is an iterative algorithm.

1.0.1 Augmenting path

Let M be the current matching. Suppose M is not maximum. Let M' be the maximum matching. Consider the symmetric difference of M and M', i.e., the set of all edges present in exactly one of M and M'.

Let v be any vertex. By the definition of matching, in the symmetric difference, there is at most one edge from M incident on v, and at most one edge from M'. This gives us

$$deg(v) \leq 2$$

This means that we must have a disjoint union of paths and cycles. Let us denote the edges of the matching M by the colour blue and those of the matching M' by the colour red. Observe that two edges of the same colour cannot be incident on a vertex. Hence, the cycles and paths must have alternating red and blue edges.

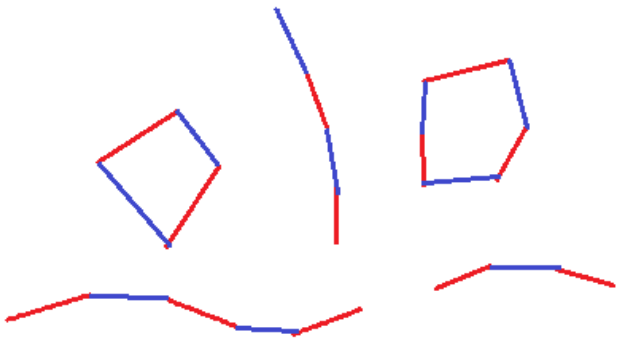

Figure 1.1: Symmetric Difference of M and M'

Lemma 1.2 *If M is not a maximum matching, there must exist an alternating path of odd length starting with a red edge.*

Proof: All the cycles are even in size, because of the alternating edges. Now, suppose all the paths also had even length. Then the number of edges of M in the symmetric difference is equal to that of M'. This implies

$$|M| = |M'|$$

since the edges of M and M' not in the symmetric difference are common to both. Therefore, if M is not maximum, there must be an alternating path with more red edges than blue ones. This will be an odd length alternating path starting with a red edge. ■

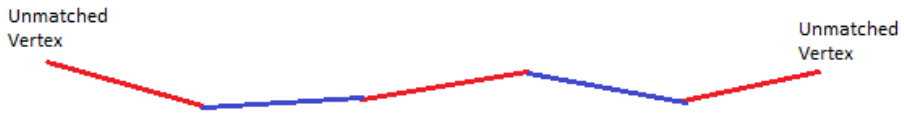

Figure 1.2: Odd length alternating path

We call such a path an **augmenting path**. Note that the vertices on either end of the augmenting path are unmatched in our current matching M. This concept of augmenting paths gives the following general idea of an iterative algorithm to compute the maximum matching in a bipartite graph.

If our current matching M is not maximum, there must exist a path of odd length that starts with an unmatched vertex, takes alternately an edge not in the matching and an edge in the matching and ends up at another unmatched vertex. If we flip the edges along this path, i.e., the edges along this path which were in the matching go out of the matching, and those which weren't are now included in the matching.

Figure 1.3: Augmenting procedure

Note: Flipping will not cause a vertex to have more than two incident edges in the matching, since, the end vertices on the path were unmatched, and the other vertices were already matched, and thus had no edge incident on them apart from the ones in the alternating path.

Following this procedure, we have increased the size of our matching by 1. This is called **augmenting the matching**. We can do this iteratively till we get a maximum matching. The number of times we will have repeat this procedure is at most n where n is the number of vertices in one partition of the bipartite graph.

In the next section, we formally describe the algorithm and argue about its correctness.

1.0.2 Alternating tree algorithm

Given a bipartite graph $G = (U, V, E)$, we start with an empty matching and iteratively augment the matching using the following procedure

(Let us denote the vertices on the U side by the colour red and the vertices on the V side by the colour blue.)

1. Start with an unmatched red vertex v.
2. Consider all the unmatched edges out of it. If one of these leads to an unmatched vertex w, add the edge (v, w) to the matching. Otherwise go to step 3.
3. From the set of blue vertices obtained in the previous step, consider all the matched edges.
4. From the set of red vertices obtained in the previous step, consider all the unmatched edges. If one of these leads to an unmatched vertex, we have found an augmenting path; flip along this path to increase the size of the matching by 1. Otherwise, go back to step 3.

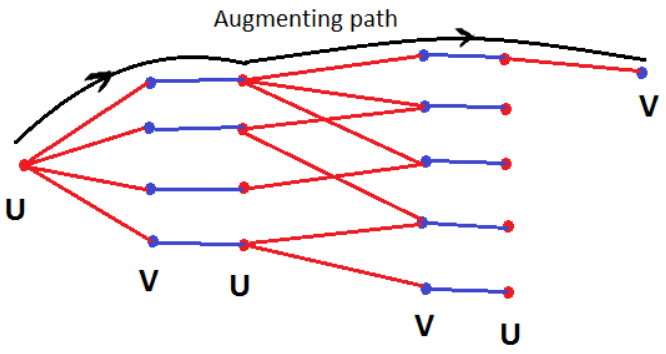

Figure 1.4: Building alternating tree

If we find an augmenting path, we improve our matching by flipping along the augmenting path, and we can start afresh with another unmatched red vertex and repeat the procedure and keep doing that till we get a maximum matching.

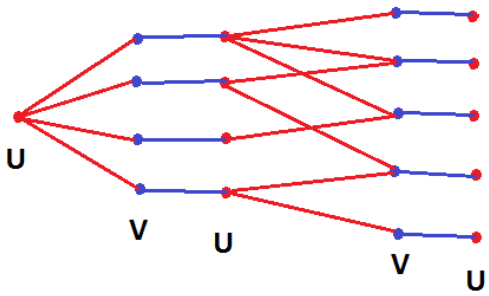

Figure 1.5: No augmenting path found

Suppose, we are not able to find an augmenting path, i.e., at step 4, all the unmatched edges are back edges. Does this imply that our matching is maximum? Let us consider the case when vertex v is the only unmatched red vertex.

Let us denote the set of red vertices in the tree by A and the set of blue vertices by B. Then

$$|A| = |B| + 1$$

since apart from vertex v, each red vertex is matched to exactly one blue vertex (See FIg 1.5). Now, there cannot be an edge from a vertex in A to a vertex in $V \backslash B$. If there was such an edge, by the procedure described above, it would have been included in the tree.

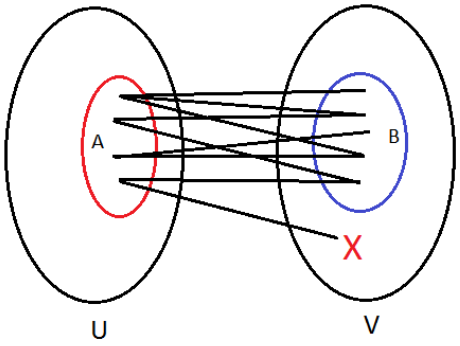

Figure 1.6: All edges starting from A end up in B

This means that all the vertices in A have to be matched to the vertices in B and since the size of B is one less, one vertex of A will remain unmatched in any matching. And hence if v is the only unmatched vertex in U, the current matching is maximum.

Now, suppose v wasn't the only unmatched vertex in U. Then, we throw away all vertices in the alternating tree rooted at v and start building an alternating tree from another unmatched vertex in U. We do this until we exhaust all unmatched vertices. Whenever we find an augmenting path, we improve our matching and start building alternating trees afresh.

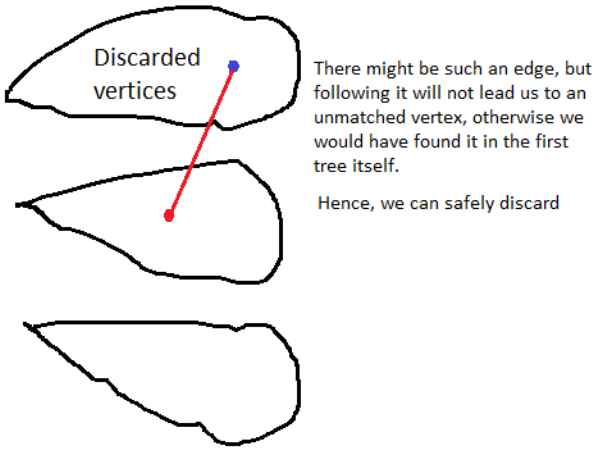

Figure 1.7: Discarding is justified

Suppose we end up with 3 unmatched vertices, for which no augmenting path is found. As before, let us denote the set of vertices of U side in the trees by A and the set of vertices of V side by B. Then, as argued above,

$$|A| = |B| + 3$$

At least 3 vertices will remain unmatched in any matching, and since we also have 3 unmatched vertices, our matching is maximum.

1.0.3 Size of maximum matching

Definition 1.3 *The neighbourhood of a set S, denoted by $N(S)$ is defined as*

$$N(S) = \{v : u \in S; (u, v) \in E\}$$

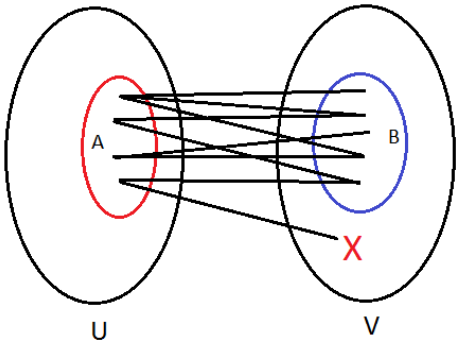

Figure 1.8: Neighbourhood of a set

Theorem 1.4 *A bipartite graph $G = (U, V, E)$ ($|U| = |V| = n$) has a perfect matching, i.e., a matching of size n, iff*

$$\forall S \subseteq V, |N(S)| \geq |S|$$

This theorem is called **Hall's Theorem**. *And a set S s.t. $|N(S)| < |S|$ is called a Hall set.*

Proof:

1. If G has a perfect matching, then there is no Hall set, i.e., $\forall S \subseteq V, |N(S)| \geq |S|$

   Suppose there exists a Hall set, i.e., $\exists S$ s.t. $|N(S)| < |S|$, then by an argument similar to one given above, all vertices of S cannot be matched, and hence there is no perfect matching.

2. If there is no Hall set, i.e., $\forall S \subseteq V, |N(S)| \geq |S|$, then G has a perfect matching

   Suppose G does not have a perfect matching. That means, that if we run our alternating tree algorithm on G, we'll land up with an unmatched vertex v for which we cannot find an augmenting path. Using the alternating tree rooted at v, we can construct a Hall set (set of all U vertices in the tree).

■

Definition 1.5 *The deficiency of a set S, denoted by $def(S)$ is defined as*

$$def(S) = |S| - |N(S)|$$

Theorem 1.6 *Let the size of the maximum matching of a bipartite graph $G = (U, V, E)$ ($|U| = |V| = n$) be denoted by $s(G)$. Then*

$$s(G) = n - \max_{S \subseteq V} def(S)$$

*(***Note:** *The notation $s(G)$ is used for convenience, is not standard.)*

Proof:

1. $s(G) \leq n - \max_{S \subseteq V} def(S)$

   Let the maximum deficiency value be k and let S' be a set with deficiency k. Then, at least k vertices of S' will remain unmatched in any matching. Hence, $s(G) \leq n - k$.

2. $s(G) \geq n - \max_{S \subseteq V} def(S)$

   Suppose $s(G) = n - k$, then in our matching, there must be k unmatched vertices in U for which the alternating tree rooted at these vertices do not contain an augmenting path. We can construct a set A as above, whose deficiency would be k. The maximum deficiency value is then greater than or equal to k, by definition.

$$\begin{aligned} k &\leq \max_{S \subseteq V} def(S) \\ -k &\geq -\max_{S \subseteq V} def(S) \\ n - k &\geq n - \max_{S \subseteq V} def(S) \\ s(G) &\geq n - \max_{S \subseteq V} def(S) \end{aligned}$$

   In fact there cannot be a set with deficiency greater than k, because then the size of the maximum matching would have to be less than n-k, by the first point.

■

1.0.4 Time complexity analysis

The matching is augmented at most n times. The time taken to perform one augmentation (building the alternating tree, finding the augmenting path by building backwards and flipping along it) is O(m) since each edge is considered at most once. So, the total time complexity of the algorithm is O(mn).