CS 435 : Linear Optimization — Fall 2006

Lecture 17: Matching in Bipartite Graphs

Lecturer: *Sundar Vishwanathan* — Scribe: *Hardeep Singh Guru*
Computer Science & Engineering — Indian Institute of Technology, Bombay

1 Matching

Definition 1 *A* Matching in a graph is a set of edges, no two of which share an end-point.

Let us look at an example. In the following graph the marked edges form a matching as no two share a common vertex.

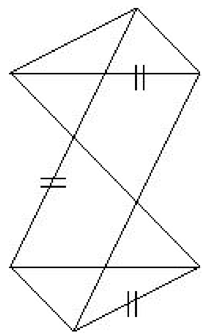

Figure 1: Matching

2 Algorithm

We now look at an algorithm which takes as input a bipartite graph and outputs a matching of maximum size. The basic idea of the algorithm lies in iteratively increasing the size of the matching by one at every stage until a matching of maximum size is obtained.

Let M be the current set of edges in the matching and let M_0 be any maximum matching.

Now, consider the graph induced by $M \oplus M_0$. We can clearly see that this graph would consist of connected components with the degree of each vertex in these being either zero, one or two. The degree would be zero for the end points of the edges which are matched in both M and M_0 or in neither. The degree will be one in the case when there is a matching edge incident on that vertex in only one of the matching and two when two edges incident on the vertex are matched, one in M and the other in M_0.

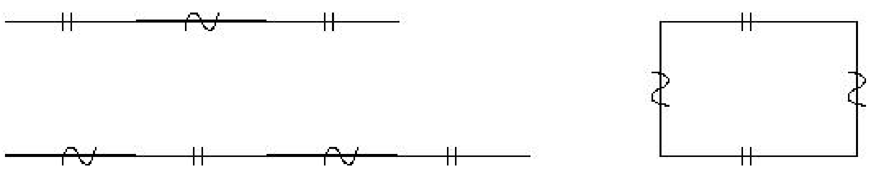

Figure 2: Connected Components

It can also be seen that the connected components of $M \oplus M_0$ are either paths or cycles.

What can we say about the cycles? We can see that the cycles would necessarily need to be of even length.

Hence, we see that the connected components are either paths or cycles of even length.

Now, let us consider a matching M which has size exactly one less the size of the maximum matching M_0.

CLAIM 1 *$M \oplus M_0$ will contain exactly one path of odd length.*

Suppose that there are two odd length paths. They can be either one of the following types.

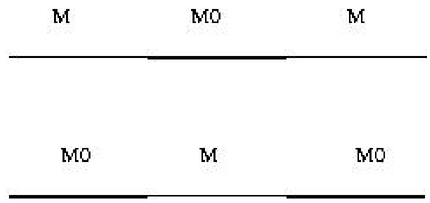

Figure 3: Odd Length Paths

Consider the path of first type where the first edge in the path belongs to M. This cannot be the case since we can otherwise switch the matched edges in M and M_0 which belong to this path and hence increase the size of M_0 by one.

Also, the paths of type two starting with M_0 cannot be more than one in number because then we can switch the matched edges belonging to M and M_0 in all these paths and hence get a matching of size greater then the size of the matching in M_0 which is a contradiction since M_0 has been assumed to be the maximum matching.

Similarly one can prove the following claim.

CLAIM 2 *If M is not a maximum there is a path in the graph which starts and ends at vertices that are unmatched (i.e., do not have any edge of M incident on them) and alternate edges in the path are from M. That is the first, third, ..., last edges are not in M while the second, fourth, ... are in M.*

For a proof note that such a path exists in the graph induced by $M \oplus M_0$. All other connected components in $M \oplus M_0$ have at least as many edges from M as from M_0. Such an path is called an augmenting path. Suppose we find an augmenting path, then we can increase the size of M by "switching" the matched and the unmatched edges. That is remove the matched edges in the path from M and introduce in the unmatched edges on the path in M_0.

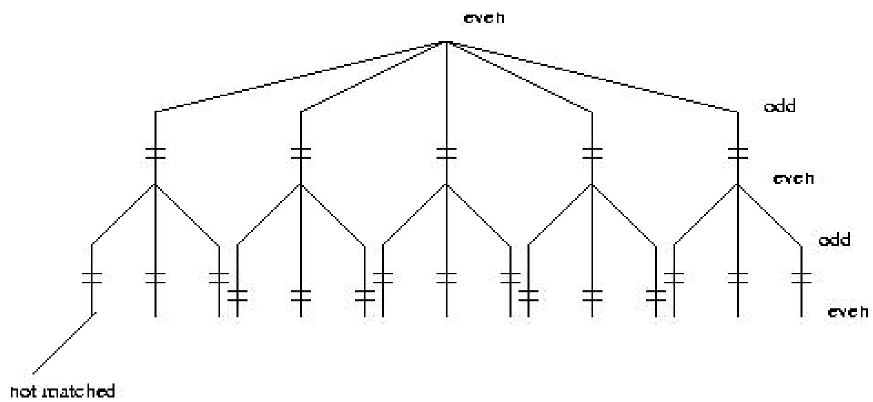

Figure 4: Augmenting Path

The broad outline of our algorithm is now clear. In each iteration, find an augmenting path and increase the size of the matching. We next see how to find such an augmenting path.

We start with a vertex which has no matched edges and label it as “even”. The other end of the edges from this vertex are labeled as “odd” as shown in the figure. At the “odd” level we go through only the matched edges and at “even” level we go through all the edges. Suppose we find an odd vertex that is unmatched, we have found the path that we are looking for which consists of alternating matched and unmatched edges and starts and ends with an unmatched vertex. Hence we can switch the matched and unmatched edges along this path and increase the size of the matching by one. We iteratively repeat this procedure until no augmenting path can be found in the graph. The matching thus obtained will be of maximum size.

Also, we can ignore back edges since edges from an “even” vertex will be only to an “odd” vertex (the graph is bipartite).

Next time we will prove that the algorithm is indeed correct.