CS 435 : Linear Optimization — Fall 2006

Lecture 20: Algorithm for Non-bipartite Matching

Lecturer: *Sundar Vishwanathan*
Computer Science & Engineering

Scribe: *Lakulish Antani*
Indian Institute of Technology, Bombay

1 Recap

We have seen an algorithm for finding a maximal matching in a bipartite graph. The algorithm is based on a modified BFS, which tries to find an augmenting path with respect to the current matching, if one exists; if no such augmenting path exists, then we conclude that the current matching is maximal. However, the algorithm cannot be used (without modifications) for general non-bipartite graphs.

We have seen that if we encounter an even-even edge during the search, we have an odd-length cycle in the graph. We handle blossoms by "shrinking" them to a single vertex; the only modification required to the previous algorithm is basically to detect and shrink blossoms. This works because of the following theorem (which we have proved):

Theorem 1 *Let $G = (V, E)$ be a graph, and M be a matching in G. Suppose B is a blossom such that its base is unmatched. Consider the graph $G' = (V', E')$ obtained by shrinking B, and the matching M' composed of the matched edges of M which are still present in E'. Then there is an augmenting path in G w.r.t. M if and only if there is an augmenting path in G' w.r.t. M'.*

2 Finding an Augmenting Path

To find an augmenting path for general graphs, we start the (modified) BFS from an unmatched vertex, as before. However, we may encounter an even-even edge between two even vertices v_1 and v_2 (say). These are part of a blossom in the graph, and we first find the base of the blossom. This is easily seen to be the common ancestor v_0 of v_1 and v_2.

Now we need to make sure that v_0 is unmatched for the above theorem to apply. This is clearly true if v_0 is the starting vertex (u). If it is not the starting vertex, then since it has been reached from u, it is matched to its parent in the search graph. Note that there will be an even-length path from v_0 to u. If, on the path p from u to v_0, we change all matched edges to unmatched and vice-versa, we are still left with a valid matching M''. Now it can be shown that there is an augmenting path in G w.r.t M if and only if there is an augmenting path in G w.r.t. M''.

Hence the algorithm is roughly as follows:

```
perform the modified BFS on G as before
u := unmatched vertex from which search begins
if when expanding an even vertex v1, we find an edge to another even vertex v2
    v0 := common ancestor of v1 and v2 in the search graph
    if  v0 = u
        shrink v0
    else
        p := path from u to v0
        change all matched edges in p to unmatched, and vice-versa
        shrink v0
```

In the next lecture, we will see a detailed proof of why the **else** clause above works, and a proof of correctness for the algorithm.

3 Time Complexity

We shall show a generous bound on the time complexity of the algorithm. Suppose the graph is $G = (V, E)$, and that $|V| = n$ and $|E| = m$. Observe that:

- The number of shrinkages performed can clearly be no more than the number of vertices in the graph, i.e. n.
- In the worst case, BFS is performed n times.
- Each BFS takes $O(m)$ time.

So we get a generous bound of $O(n^2m)$ on the complexity.