CS 435 : Linear Optimization — Fall 2006

Lecture 20: A BFS based algorithm for matching in general graph (contd.)

Lecturer: *Sundar Vishwanathan* — Scribe: *Rakesh Venkat*
Computer Science & Engineering — Indian Institute of Technology, Bombay

1 Overview

In the previous lecture, we looked at obtaining a new graph G' from the given graph G by shrinking one or more *Blossoms* to a single 'macrovertex' in G. In this lecture, we look at relations between augmenting paths in G' and G. We then develop the algorithm for matching in general graphs based on the modified BFS algorithm used in the bipartite matching.

2 Augmenting paths in G' and G

Let M be a matching in the graph G. Suppose B is a blossom such that the *base is unmatched*(Fig: 1 (a)). Shrink vertices of B to a single vertex to get a new graph G'. Note that the base may also be right at the 'top' of the BFS search tree, as shown in Fig: 1 (b). The matching M' in G' consists of those edges in M that remain in G'.

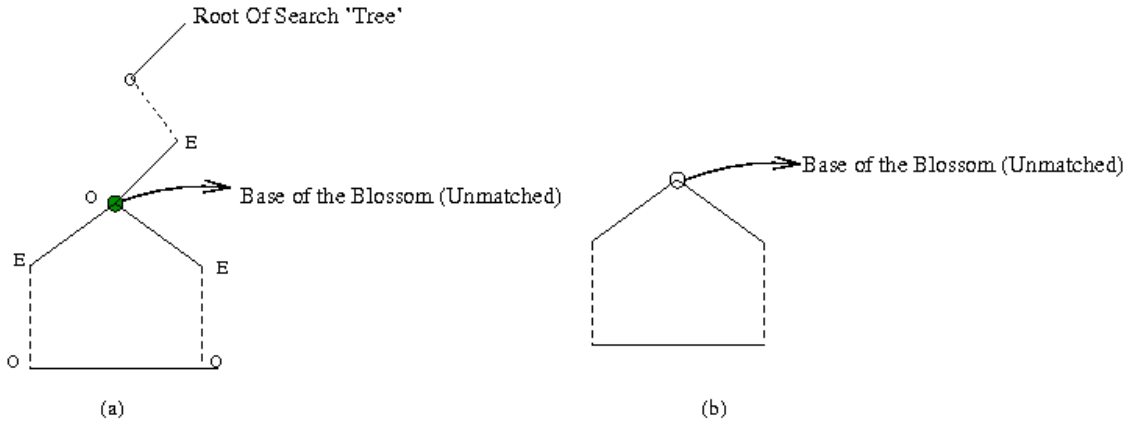

Figure 1: (a) A blossom with an unmatched base (b) A blossom with an unmatched base at the root of the search tree in BFS-modified algo

We start with the following theorem:

Theorem 1 *M' is maximal in G'* iff *M is maximal in G. This is equivalent to proving that* $\exists$*an augmenting path in G* iff $\exists$*an augmenting path in G'.*

Proof: *(This theorem was proved in the previous lecture and was repeated in the class for clarity.)*

Observation 1: The new (shrunk) vertex (say v_0) in the graph G' is unmatched (Fig: 2). This is because all vertices in the blossom B except the base are already matched, and hence can connect to the outer vertices only through unmatched edges. By our assumption, the base is also unmatched, hence v_0 is unmatched.

Part 1: Assume there is an augmenting path P' in G'

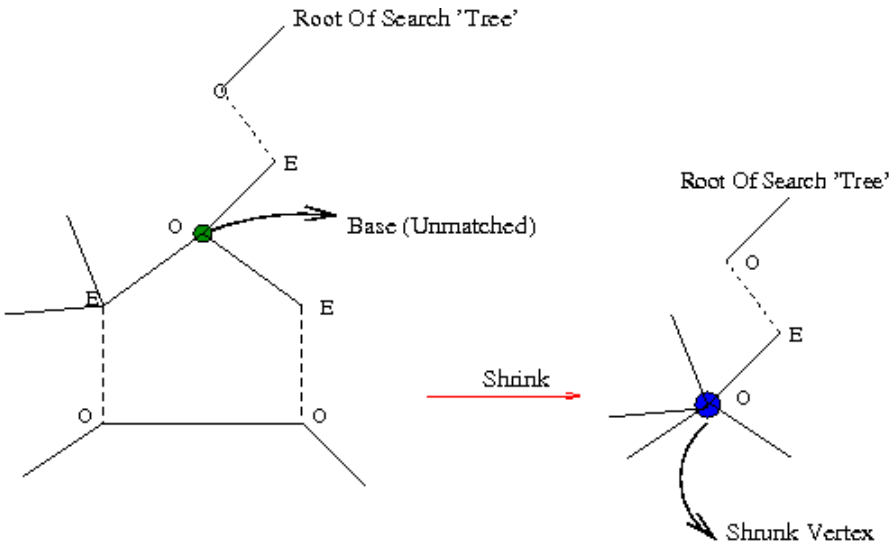

Figure 2: Shrinking a Blossom. The corresponding vertex in the new graph is unmatched.

1. Case (i): If it avoids v_0, then the same augmenting path is present in G too.
2. Case (ii): If it goes through v_0, then as v_0 is unmatched, P' either starts or ends at v_0.
   We can then trace a corresponding augmenting path in P by expanding the blossom, and noting the vertex next to v_0 in P' as shown in Fig: 3. One end of the augmenting path in P is always the base of the blossom. Also, the portion of the path P in the blossom is always of even length.

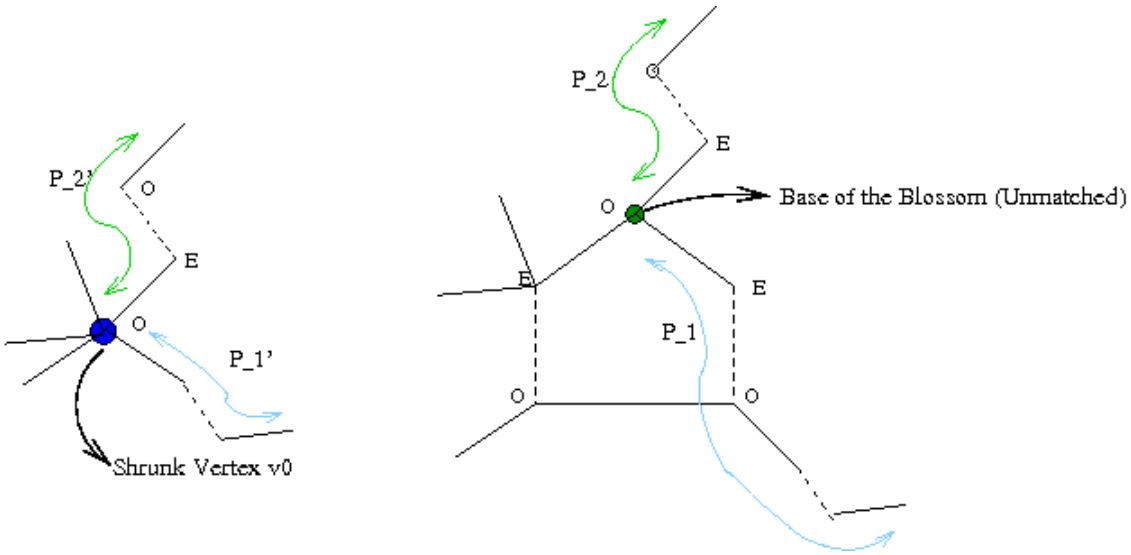

Figure 3: Mapping augmenting paths in G to G' and vice versa. The curved lines of the same colour in the two graphs indicate corresponding augmenting paths.

Part 2: Assume there is an augmenting path P in G

1. Case (i): If it doesnt hit a blossom (B) , we are done, the same augmenting path works in G'.
2. Case (ii): If P goes through the Blossom, the last edge in the path before it hits the blossom must be unmatched.

Thus, no path in G' can go past v_0 in accordance with *Observation 1.*. The part of the path P from the start till it hits the blossom is thus an augmenting path P' in G'. (Again, see Fig: 3).

□

3 Adapting the BFS-based algorithm to the general graph

We run the modified BFS algorithm as in the bipartite case, but with a small modification. In this algorithm, at any point of time, we have a *tree* starting from the current root vertex, and a matching on this.
The only place where we encounter a problem is when we hit a blossom. We can identify a blossom when we follow an edge to an already explored even vertex from an even vertex we are currently at. (See Fig:4). Also note that in such a case, the first explored vertex in the blossom is the base (the lowest common ancestor of the mentioned two even vertices in the tree). The base has to be matched if we are to *enter* the blossom, as in the figure. The algorithm would continue normally as in the bipartite case if we entered the blossom through any other point but the base.

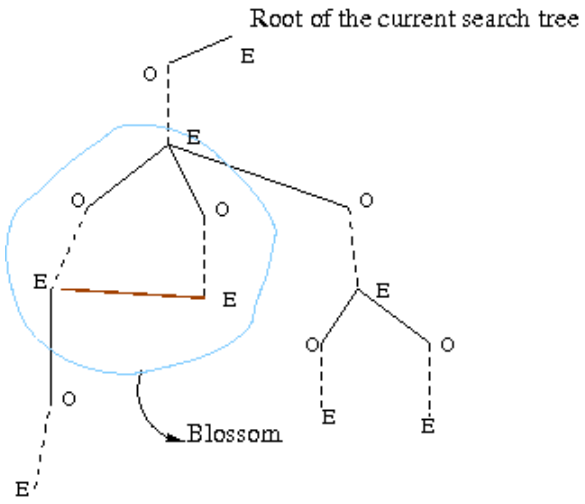

Figure 4: Discovering a blossom in the BFS-based algo in the general graph. On encountering the Brown edge shown, we infer a blossom, and proceed to shrink it, switching the matched-unmatched edges from the root to the base.

On encountering the above case, we do the following:

1. Switch the state of edges on the path till the base from matched-unmatched and vice versa. This preserves the matching size and augmenting paths. (in the next lecture this step is justified in more detail). Call the new graph G_1.

2. Now, the blossom satisfies the conditions of the Theorem above. Shrink the blossom B in G_1 to a single vertex. Call the new graph G'.

   By the previous theorem, augmenting paths in G_1 are preserved in G' and vice versa, and we can also map the paths(and hence matching) appropriately.

Now we continue as usual. We see that in the tree maintained in the general case (as opposed to the bipartite case), the vertices at each level may be either (a) normal vertices or (b) shrunk blossoms.

On finding an augmenting path in the new graph, find the corresponding augmenting path in G_1 (and hence G) by unshrinking the shrunk blossoms one by one in reverse order of shrinking as explained in the theorem above (Part 1), and update the matching.

The algorithm will terminate when there are no remaining augmenting paths in the graph.

Runtime

We shrink blossoms atmost $|V|$ times. Till we hit a blossom, the length of the path is atmost $|E|$. To discover all augmenting paths, we have to run the algorithm from each vertex in turn. So a loose upper bound is $O(|V|^2|E|)$. The algorithm can be implemented in $O(|E|\sqrt{|V|})$.