Matchings in General Graphs

Jonathan Turner

February 24, 2013

We have studied two approaches to finding matchings in bipartite graphs. This first involved transforming the matching problem into a maximum flow problem. The second used the concept of *augmenting paths*. In this section, we'll see how the augmenting path method can be extended to general graphs. Recall that an augmenting path with respect to a matching M is a path with two unmatched endpoints and with alternate edges in the matching. Given an augmenting path, we can increase the size of the matching by one, by "flipping" the edges in the path; that is, removing the matching edges from M and replacing them with the remaining edges on the path. This is illustrated in Figure 1. Note that this does not create any conflicts with other matching edges and that every vertext that was originally matched remains matched.

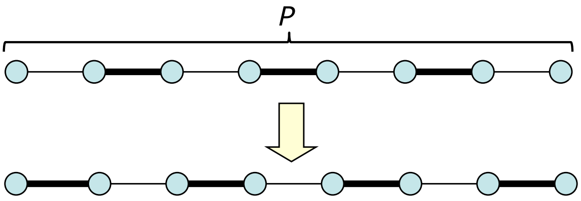

Figure 1: Flipping an augmenting path

To find an augmenting path in a bipartite graph, we build a collection of trees rooted at the unmatched vertices. Each vertex is assigned a *state* that can be *odd*, *even* or *unreached*. Initially the tree roots are even and all other vertices are unreached. Each step in the algorithm examines a previously unexamined edge $\{v, w\}$ with v even. If w is unreached, then it must be incident to a matching edge $\{w, x\}$ In this case, $\{v, w\}$ and $\{w, x\}$ are added to the tree with $p(w) = v$ and $p(x) = w$, the state of w is changed to odd and the state of x to even. If w is odd, we proceed to the next edge, but if w

is even, then the path from the root of v's tree to v, concatenated with the edge $\{v, w\}$ and the path from w to the root of its tree forms an augmenting path.

This last statement is justified in bipartite graphs, by the observation that if two even vertices in the same tree were connected by an edge, then that edge together with the tree paths to their nearest common ancestor would form an odd length cycle. Since bipartite graphs have no odd-length cycles, any edge joining even vertices, must join vertices in different trees. On the other hand, when we extend the algorithm to a general graph, it is possible for an edge to join two even vertices in the same tree, as illustrated in Figure 2. In this diagram, the tree root is at the left, the plus signs

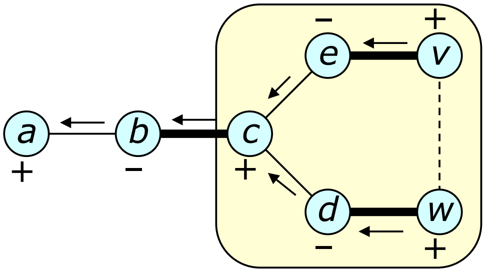

Figure 2: Blossom in a general graph

indicate even vertices, the minus signs indicate odd vertices and the arrows represent the parent pointers. The dashed edge at the right forms an odd length cycle with the tree edges leading back to vertex c. Such a cycle is referred to as a *blossom* and the edge that completes the blossom ($\{v, w\}$ in this case) is referred to as the *bridge* of the blossom. The nearest common ancestor of the vertices in the blossom (c in this case) is called the *base* of the blossom. Note that while all other vertices in a blossom must be incident to a matching edge, the base of a blossom may be unmatched, if it is the root of its tree.

To extend the augmenting path algorithm to general graphs, we replace each blossom we discover with a single vertex and proceed with the path search in this new graph. This strategy is justified by the following theorem.

Theorem 1 *If G' is formed from G by shrinking a blossom b, then G' contains an augmenting path if and only if G does.*

For now, we only show that if G' contains an augmenting path, then G does also. First, note if p is an augmenting path in G' that does not include b, then p is also an augmenting path in G. On the other hand, if b is on p, we can construct an augmenting path in G by expanding the blossom, as illustrated in Figure 3. If b has two incident edges in G', then one must be

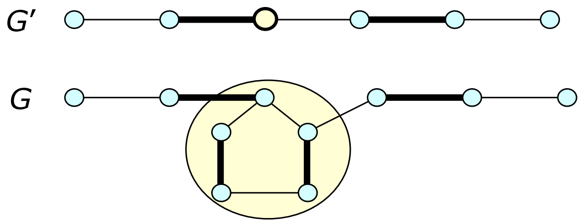

Figure 3: Expanding a blossom

matching edge incident to the base of b, while the other is a non-matching edge incident to some vertex x in b (possibly the base). If x is an even vertex, the augmenting path passes from x up the tree to the base of b. If x is an odd vertex, the augmenting path passes through the bridge of the blossom on its way to the base.

The blossom shrinking algorithm is known as *Edmonds algorithm* and can be formally defined as follows. First, to simplify the description, we treat each original edge $\{x, y\}$ as a pair of directed edges, $[x, y]$ and $[y, x]$. To find an augmenting path, we build a set of trees much as we did in the case of bipartite graphs. In each step we select a previously unexamined edge $[v, w]$ with v even and examine it.

- If w is unreached, it must be incident to a matching edge. Let x be the other endpoint of that matching edge. We extend the tree by making $p(w) = v$ and $p(x) = w$. We also change the state of w to odd and the state of x to even.

- If w is even and is in a different tree from v, then the edge, together with the two tree paths leading to the tree roots forms an augmenting path, and the algorithm terminates.

- If w is even and in the same tree as v, we form a new blossom b. If the nearest common ancestor of v and w is u, then the blossom consists of the vertices on the tree paths from v to u and from w to u. In the new graph that results from shrinking the blossom, the state of b is even.

A vertex is called *shrunken* if at some point in the algorithm, it has been included in a blossom. Also note that all steps in the description above should be interpretes as taking place in the "current graph", which may include one or more shrunken blossoms. In addition, note that when we shrink a blossom, the edges with one endpoint in the blossom and one endpoint outside the blossom become edges in the new graph, and these edges all have an even endpoint in the new graph (the blossom).

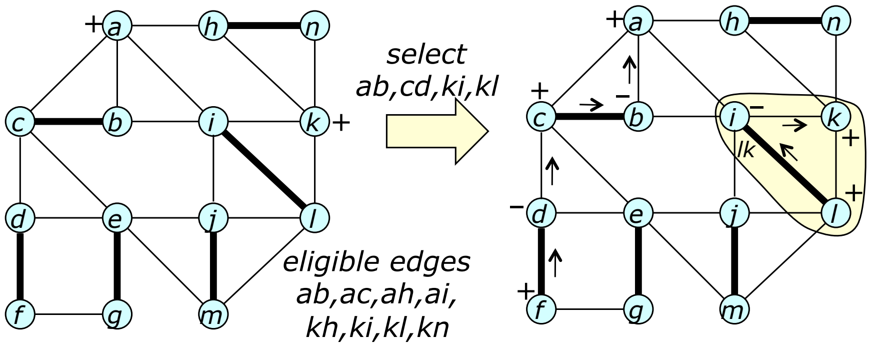

Figure 4: Example of Edmond's algorithm

Let's look at an example of Edmond's algorithm in action. The left side of Figure 4 shows a graph with two unmatched vertices a and k. At the start of the algorithm, these are the only even vertices and so the only edges that are "eligible"; to be considered by the algorithm are the edges incident to a and k. If the first step examines the edge $[a, b]$, the tree rooted at a is extended to b and c. This makes c even and its incident edges eligible for consideration. If the next few steps select edges $[c, d]$, $[k, i]$ and $[k, l]$, the tree rooted at a is extended further, while a blossom is formed in the tree rooted at k. Note that in this blossom, k is the base of the blossom and the edge joining k and l is the bridge.

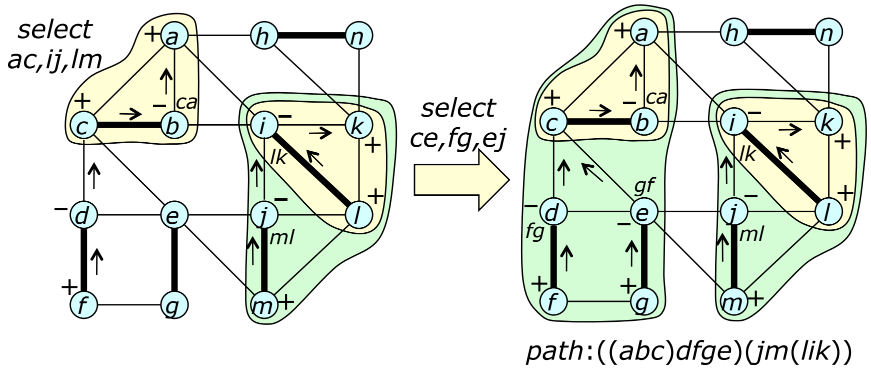

Figure 5: Continuing the example

Continuing the example in Figure 5, if the next three steps examine edges $[a, c]$, $[i, j]$, and $[l, m]$, we form two additional blossoms, as shown on the left side of the figure. Again, take note of the base and bridge of each of these blossoms. If we next examine edges $[c, e]$ and $[f, g]$, we form yet another blossom in the tree rooted at a. Now, note that at this point, vertices e and j are both contained within blossoms, so the edge $[e, j]$ joins two even vertices in the current graph, even though both e and j are odd. If the next step selects the edge $[e, j]$, we have a single edge joining two unmatched blossoms. That edge is an augmenting path in the current graph. If we expand the blossoms, we get the path $((abc)dfge)(jmlik)$ where the parentheses identify blossom boundaries. Note that in order to extend an augmenting path in the current graph to an expanded blossom, we need to be prepared to go around the blossom in either direction (straight up the tree, or across the bridge). The parent pointers can be used to go up the tree, but we also need the bridge information in order to traverse a blossom in the other direction.