Matchings in Bipartite Graphs

Jonathan Turner

January 26, 2013

A *matching* in a graph $G = (V, E)$ is a subset of the edges with no two edges incident to the same vertex. Figure 1 illustrates matchings in two graphs, one with edge weights, one without. In the weighted version of the problem, the weight of a matching is defined to be the sum of the weights of its edges. In the unweighted version of the problem, the objective is to find

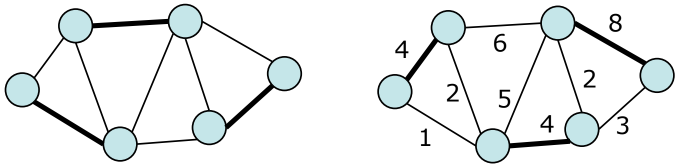

Figure 1: The matching problem in graphs

a matching with the largest number of edges, while in the weighted version, the objective is to find a matching with the largest total weight.

If we think of the graph as representing some kind of compatibility relationship, then a matching is simply a collection of compatible pairs, and if the weights are used to specify different "degrees of compatibility" a maximum weight matching is a set of maximally compatible pairs.

Matching problems arise frequently in the context of *bipartite graphs*, and we'll see that for these graphs there are specialized algorithms that are particularly efficient. Let $G = (V, E)$ be a bipartite graph where V can be divided into two subsets V_1, V_2 such that all edges join vertices in V_1 to vertices in V_2. We can convert G into a flow graph by directing all the edges from V_1 to V_2, adding a source vertex s with an edge to every vertex in V_1, and a sink vertex t with an edge from every vertex in V_2. This is illustrated in Figure 2. If all edges are assigned a capacity of 1, then any integer flow on the flow graph corresponds to a matching in the original graph. We can use the shortest augmenting path algorithm to find such a flow and for

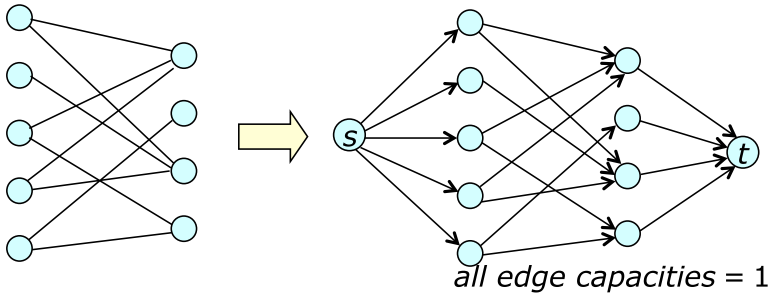

Figure 2: Converting matching problem to a max flow problem

these graphs, the time required to find a maximum flow is $O(mn)$, since the number of augmenting path steps is cannot exceed n in this case. We'll see later that there is a different max flow algorithm can reduce the running time to $O(m\sqrt{n})$.

We can convert a weigted matching problem into a min-cost flow problem in much the same way. The only difference is that we assign edge costs, based on the original edge weights. Specifically, the edges incident to the source and sink are assigned 0 cost, while each edge (u, v) from V_1 to V_2 is assigned $\mathit{cost}(u, v) = -\mathit{weight}(u, v)$. This is illustrated in Figure 3. Note

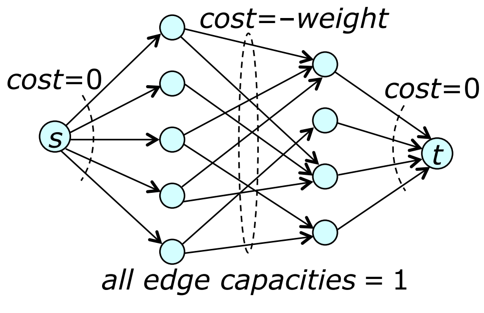

Figure 3: Converting weighted matching to a min-cost flow problem

that because the flow graph is acyclic, there are no negative weight cycles in the flow graph, even though there may be many negative length edges.

To obtain the maximum weight matching, we need to find the flow with the most negative cost. This can be done using the minimum cost augmenting path algorithm. We halt the algorithm, as we soon as we find an augmenting path with non-negative cost. Since the number of augmenting path steps icannot exceed n, the running time is $O(mn + n^2 \log n)$.