10: Matchings*

Sariel Har-Peled

July 28, 2006
CS 473g - Algorithms

Version: 0.2

1 Definitions

Definition 1.1 For a graph $G = (V, E)$ a set $M \subseteq E$ of edges is a matching if no pair of edges of $M$ has a common vertex.

A matching is perfect if it covers all the vertices of $G$. For a weight function $w$, which assigns real weight to the edges of $G$, a matching $M$ is a maximal weight matching, if $M$ is a matching and $w(M) = \sum_{e \in M} w(e)$ is maximal.

Definition 1.2 If there is no weight on the edges, we consider the weight of every edge to be one, and in this case, we are trying to compute a maximum size matching.

Problem 1.3 Given a graph $G$ and a weight function on the edges, compute the maximum weight matching in $G$.

2 Unweighted matching in a bipartite graph

We remind the reader that there is a simple way to do a matching in a bipartite graph using network flow. Since this was already covered, we will not repeat it here.

3 Matchings and Alternating Paths

---

*This work is licensed under the Creative Commons Attribution-NonCommercial 2.5 License. To view a copy of this license, visit [http://creativecommons.org/licenses/by-nc/2.5/](http://creativecommons.org/licenses/by-nc/2.5/) or, (b) send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.
Consider a matching \( M \). An edge \( e \in M \) is a matching edge. Naturally, Any edge \( e' \in E(G) \setminus M \) is free. In particular, a vertex \( v \in V(G) \) is matched if it is adjacent to an edge in \( M \). Naturally, a vertex \( v' \) which is not matched is free.

An alternating path is a simple path that its edges are alternately matched and free. An alternating cycle is defined similarly. The length of a path/cycle is the number of edges in it.

For an alternating path/cycle \( \pi \), its weight is

\[
\gamma(\pi, M) = \sum_{e \in \pi \setminus M} w(e) - \sum_{e \in \pi \cap M} w(e).
\]  

(1)

Namely, it is the total weight of the free edges in \( \pi \) minus the weight of the matched edges. This is a natural concept because of the following lemma.

**Lemma 3.1** Let \( M \) be a matching, and let \( \pi \) be an alternating path/cycle with positive weight such that

\[
M' = M \oplus \pi = (M \setminus \pi) \cup (\pi \setminus M)
\]

is a matching, then \( w(M') \) is bigger; namely, \( w(M') > w(M) \).

**Proof:** Just observe that \( w(M') = w(M) + \gamma(\pi, M) \).

**Definition 3.2** An alternating path is augmenting if it starts and ends in a free vertex.

**Observation 3.3** If \( M \) has an augmenting path \( \pi \) then \( M \) is not of maximum size matching (this is for the unweighted case), since \( M \oplus \pi \) is a larger matching.

**Theorem 3.4** Let \( M \) be a matching, and \( T \) be a maximum size matching, and \( k = |T| - |M| \). Then \( M \) has \( k \) vertex disjoint augmenting paths. At least one of length \( \leq n/k - 1 \).

**Proof:** Let \( E' = M \oplus T \), and let \( H = (V, E') \). Clearly, every vertex in \( H \) has at most degree 2 because every vertex is adjacent to at most one edge of \( M \) and one edge of \( T \). Thus, \( H \) is a collection of paths and (even length) cycles. The cycles are of even length since the edges of the cycle are alternating between two matchings (i.e., you can think about the cycle edges as being 2-colorable).

Now, there are \( k \) more edges of \( T \) in \( M \oplus T \) than of \( M \). Every cycle have the same number of edges of \( M \) and \( T \). Thus, a path in \( H \) can have at most one more edge of \( T \) than of \( M \). In such a case, this path is an augmenting path for \( M \). It follows that there are at least \( k \) augmenting paths for \( M \) in \( H \).

As for the claim on the length of the shortest augmenting path, observe that if all these (vertex disjoint) augmenting paths were of length \( \geq n/k \) then the total number of vertices in \( H \) would be at least \( (n/k + 1)k > n \). A contradiction.

**Theorem 3.5** Let \( M \) be a matching of maximum weight among matchings of size \( |M| \). Let \( \pi \) be an augmenting path for \( M \) of maximum weight, and let \( T \) be the matching formed by augmenting \( M \) using \( \pi \). Then \( T \) is of maximum weight among matchings of size \( |M| + 1 \).

**Proof:** Let \( S \) be a matching of maximum weight among all matchings with \( |M| + 1 \) edges. And consider \( H = (V, M \oplus S) \).

Consider a cycle \( \sigma \) in \( H \). The weight \( \gamma(\sigma, M) \) (see Eq. (1)) must be zero. Indeed, if \( \gamma(\sigma, M) > 0 \) then \( M \oplus \sigma \) is a matching of the same size as \( M \) which is heavier than \( M \). A contradiction to the definition of \( M \) as the maximum weight such matching.
Similarly, if \( \gamma(\sigma, M) < 0 \) than \( \gamma(\sigma, S) = -\gamma(\sigma, M) \) and as such \( S \oplus \sigma \) is heavier than \( S \). A contradiction.

By the same argumentation, if \( \sigma \) is a path of even length in the graph \( H \) then \( \gamma(\sigma, M) = 0 \) by the same argumentation.

Let \( U_S \) be all the odd length paths in \( H \) that have one edge more in \( S \) than in \( M \), and similarly, let \( U_M \) be the odd length paths in \( H \) that have one edge more of \( M \) than an edge of \( S \).

We know that \( |U_S| - |U_M| = 1 \) since \( S \) has one more edge than \( M \). Now, consider a path \( \pi \in U_S \) and a path \( \pi' \in U_M \). It must be that \( \gamma(\pi, M) + \gamma(\pi', M) = 0 \). Indeed, if \( \gamma(\pi, M) + \gamma(\pi', M) > 0 \) then \( M \oplus \pi \oplus \pi' \) would have bigger weight than \( M \) while having the same number of edges. Similarly, if \( \gamma(\pi, M) + \gamma(\pi', M) < 0 \) (compared to \( M \)) then \( S \oplus \pi \oplus \pi' \) would have the same number of edges as \( S \) while being a heavier matching. A contradiction.

Thus, \( \gamma(\pi, M) + \gamma(\pi', M) = 0 \). Thus, we can pair up the paths in \( U_S \) to paths in \( U_M \), and the total weight of such a pair is zero, by the above argumentation. There is only one path \( \mu \in U_S \) which is not paired, and it must be that \( \gamma(\mu, M) = w(S) - w(M) \) (since everything else in \( H \) has zero weight as we apply it to \( M \) to get \( S \)).

This establishes the claim that we can augment \( M \) with a single path to get a maximum weight matching of cardinality \( |M| + 1 \). Clearly, this path must be the heaviest augmenting path that exists for \( M \). Otherwise, there would be a heavier augmenting path \( \sigma' \) for \( M \) such that \( w(M \oplus \sigma') > w(S) \). A contradiction to the maximality of \( S \).

The above theorem imply that if we always augment along the maximum weight augmenting path, than we would get the maximum weight matching in the end.

### 4 Maximum Weight Matchings in A Bipartite Graph

Let \( G = (L \cup R, E) \) be the given bipartite graph, with \( w : E \rightarrow \mathbb{R} \) be the non-negative weight function. Given a matching \( M \) we define the graph \( G_M \) to be the directed graph, where if \( rl \in M \), \( l \in L \) and \( r \in R \) then we add \((r \rightarrow l)\) to \( E(G_M) \) with weight \( w((r \rightarrow l)) = w(rl) \) Similarly, if \( rl \in E \setminus M \) then add the edge \((l \rightarrow r) \in E(G_M) \) to \( G_M \) and set \( w((l \rightarrow r)) = -w(rl) \).

Namely, we direct all the matching edges from right to left, and assign them their weight, and we direct all other edges from left to right, with their negated weight. Let \( G_M \) denote the resulting graph.

An augmenting path \( \pi \) in \( G \) must have an odd number of edges. Since \( G \) is bipartite, \( \pi \) must have one endpoint on the left side and one endpoint on the right side. Observe, that a path \( \pi \) in \( G_M \) has weight \( \alpha(\pi) = -\gamma(\pi, M) \).

Let \( U_L \) be all the unmatched vertices in \( L \) and let \( U_R \) be all the unmatched vertices in \( R \).

Thus, what we are looking for is a path \( \pi \) in \( G_M \) starting \( U_L \) going to \( U_R \) with maximum weight \( \gamma(\pi) \), namely with minimum weight \( \alpha(\pi) \).

**Lemma 4.1** If \( M \) is a maximum weight matching with \( k \) edges in \( G \), than there is no negative cycle in \( G_M \) where \( \alpha(\cdot) \) is the associated weight function.

**Proof**: Assume for the sake of contradiction that there is a cycle \( C \), and observe that \( \gamma(C) = -\alpha(C) > 0 \). Namely, \( M \oplus C \) is a new matching with bigger weight and the same number of edges. A contradiction to the maximality of \( M \).

**The algorithm.** So, we now can find a maximum weight in the bipartite graph \( G \) as follows: Find a maximum weight matching \( M \) with \( k \) edges, compute the maximum weight augmenting path for \( M \), apply it, and repeat till \( M \) is maximal.

Thus, we need to find a minimum weight path in \( G_M \) between \( U_L \) and \( U_R \) (because we flip weights). This is just computing a shortest path in the graph \( G_M \) which does not have negative cycles, and this can just
be done by using the Bellman-Ford algorithm. Indeed, collapse all the vertices of $U_L$ into a single vertex, and all the uncovered vertices of $U_R$ into a single vertex. Let $H_M$ be the resulting graph. Clearly, we are looking for the shortest path between the two vertices corresponding to $U_L$ and $U_R$ in $H_M$ and since this graph has no negative cycles, this can be done using the Bellman-Ford algorithm, which takes $O(nm)$ time. We conclude:

**Lemma 4.2** Given a bipartite graph $G$ and a maximum weight matching $M$ of size $k$ one can find a maximum weight augmenting path for $G$ in $O(nm)$ time, where $n$ is the number of vertices of $G$ and $m$ is the number of edges.

We need to apply this algorithm $n/2$ times at most, as such, we get:

**Theorem 4.3** Given a weight bipartite graph $G$, with $n$ vertices and $m$ edges, one can compute a maximum weight matching in $G$ in $O(n^2m)$ time.

### 4.1 Faster Algorithm

It turns out, in fact, that the graph here is very special, and one can use the Dijkstra algorithm. We omit any further details, and just state the result. The interested student can figure out the details (warning: this is not easy).

**Theorem 4.4** Given a weight bipartite graph $G$, with $n$ vertices and $m$ edges, one can compute a maximum weight matching in $G$ in $O(n(n \log n + m))$ time.

### 5 The Bellman-Ford Algorithm - A Quick Reminder

The Bellman-Ford algorithm computes the shortest path from a single source $s$ in a graph $G$ that has no negative cycles to all the vertices in the graph. Here $G$ has $n$ vertices and $m$ edges. The algorithm works by initializing all distances to the source to be $\infty$ (formally, for all $u \in V(G)$, we set $d[u] \leftarrow \infty$ and $d[s] \leftarrow 0$). Then, it $n$ times scans all the edges, and for every edge $(u \rightarrow v) \in E(G)$ it performs a Relax$(u, v)$ operation. The relax operation checks if $x = d[u] + w((u \rightarrow v)) < d[v]$, and if so, it updates $d[v]$ to $x$, where $d[u]$ denotes the current distance from $s$ to $u$. Since Relax$(u, v)$ operation can be performed in constant time, and we scan all the edges $n$ times, it follows that the overall running time is $O(mn)$.

We claim that in the end of the execution of the algorithm the shortest path length from $s$ to $u$ is $d[u]$, for all $u \in V(G)$. Indeed, every time we scan the edges, we set at least one vertex distance to its final value (which is its shortest path length). More formally, all vertices that their shortest path to $s$ have $i$ edges, are being set to their shortest path length in the $i$th iteration of the algorithm, as can be easily proved by induction. This implies the claim.

Notice, that if we want to detect negative cycles, we can ran Bellman-Ford for an additional iteration. If the distances changes, we know that there is a negative cycle somewhere in the graph.