Lecture 19 Matching

Matching refers to a class of problems with many important applications. Assigning instructors to courses or students to seminars with limited enrollment are two examples of matching problems.

Formally, matching problems are expressed as problems on graphs. We will consider four different versions, depending on whether the graph is bipartite or not and whether the graph is weighted or unweighted. The bipartite case is considerably easier, so we will concentrate on that case.

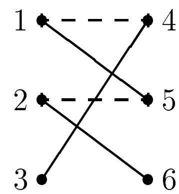
Definition 19.1 Given an undirected graph $G = (V, E)$ with edge weights w, a *matching* is a subset $M \subseteq E$ such that no two edges in M share a vertex. The *maximum-weight matching problem* is to find a matching M such that the sum of the weights of the edges in M is maximum over all possible matchings. If all the weights are 1, then we get the *unweighted matching problem*, which just asks for a matching of maximum cardinality. $\Box$

Definition 19.2 Given a matching M in $G = (V, E)$, an edge $e \in E$ is *matched* if $e \in M$ and *free* if $e \in E - M$. A vertex v is *matched* if v has an incident matched edge, *free* otherwise. $\Box$

Definition 19.3 A *perfect matching* is a matching in which every vertex is matched. $\Box$

Definition 19.4 Given a matching M in $G = (V, E)$, a path (cycle) in G is an *alternating path (cycle)* with respect to the matching M if it is simple (*i.e.*, has no repeated vertices) and consists of alternating matched and free edges. The length of a path or cycle p is the number of edges in p and is denoted $|p|$. An alternating path is an *augmenting path* (with respect to M) if its endpoints are free. $\square$

For example, consider the following graph.



The solid edges form a maximum matching that is also a perfect matching. The dashed edges form a maximal matching that is not maximum (it is maximal because it is not a proper subset of any other matching). With respect to the dashed matching, the edges (1,4) and (2,5) are matched, the edges (1,5), (2,6), and (3,4) are free, the vertices 1,2,4, and 5 are matched, and the vertices 3 and 6 are free. With respect to the dashed matching, the alternating path 3,4,1,5,2,6 is an augmenting path.

Let $\oplus$ be the symmetric difference operator on sets:

$$\begin{aligned} A \oplus B &= (A \cup B) - (A \cap B) \\ &= (A - B) \cup (B - A) \ . \end{aligned}$$

In other words, $A \oplus B$ is the set of elements that are in one of A or B, but not both. If M is a matching and p an augmenting path with respect to M, then considering p as its set of edges, the set $M \oplus p$ is a matching of cardinality $|M| + 1$. Note that $M \oplus p$ agrees with M on edges outside of p, and every edge in p that is matched in M is unmatched in $M \oplus p$ and vice-versa.

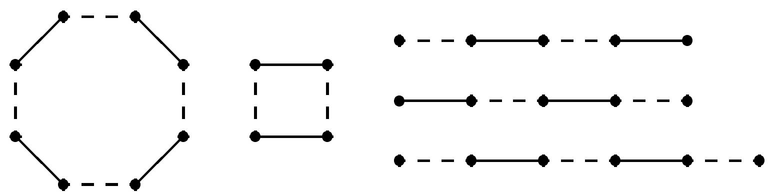
The following early theorem of Berge [10] gives the foundation for an efficient matching algorithm.

Theorem 19.5 (Berge [10]) *A matching M in a graph G is a maximum matching if and only if there is no augmenting path in M.*

This theorem follows immediately from the following enhanced version due to Hopcroft and Karp [51].

Theorem 19.6 (Hopcroft and Karp [51]) *If M is a matching in G, M^* is a maximum matching in G, and $k = |M^*| - |M|$, then with respect to M there is a set of k vertex-disjoint augmenting paths. Moreover, at least one of them has length at most $\frac{n}{k} - 1$, where n is the number of vertices in G.*

Proof. Consider $M \oplus M^*$. No vertex can have more than one incident edge from M or more than one incident edge from M^*, so no vertex can have more than two incident edges from $M \oplus M^*$. The set $M \oplus M^*$ therefore consists of a collection of vertex-disjoint alternating paths and cycles, as illustrated. Here the solid lines indicate edges of M and the dashed lines indicate edges of M^*.



Each odd-length path p has either one more M edge than M^* edge or one more M^* edge than M edge. However, the former is impossible, since then p would be an augmenting path with respect to M^*, thus M^* would not be maximum.

Using the assumption that $|M^*| = |M| + k$,

$$\begin{aligned} |M^* - M| &= |M^*| - |M^* \cap M| \\ &= |M| - |M^* \cap M| + k \\ &= |M - M^*| + k \ . \end{aligned}$$

In other words there are exactly k more M^* edges in $M^* \oplus M$ than M edges. The extra k M^* edges must come from paths of odd length with one more M^* edge than M edge. Cycles and paths of even length have the same number of M as M^* edges, and as we have observed, there are no odd-length paths with one more M than M^* edge. These k odd-length paths with one more M^* than M edge have endpoints that are free with respect to M, therefore are augmenting paths in M.

It is impossible for all of these paths to have length greater than $\frac{n}{k} - 1$, because then we would have more than n vertices. Therefore at least one of the paths has length less than or equal to $\frac{n}{k} - 1$. □

19.1 Weighted Matchings

Definition 19.7 Let M be a matching in a graph G with edge weights w. For any set A of edges, define

$$w(A) \ = \ \sum_{e \in A} w(e) \ .$$

B

Define the *incremental weight* $\Delta(\cancel{p})$ of a set B of edges to be the total weight of the unmatched edges in B minus the total weight of the matched edges in B:

$$\Delta(B) \ = \ w(B - M) - w(B \cap M) \ .$$

□

If p is an augmenting path with respect to M, then $\Delta(p)$ is the net change in the weight of the matching after augmenting by p:

$$w(M \oplus p) \quad = \quad w(M) + \Delta(p) \ . \tag{25}$$

Here is a good heuristic to use when selecting augmenting paths for maximum weight matching:

Always use an augmenting path of maximum incremental weight.

Lemma 19.8 *If M is a matching of size k that is of maximum weight among all matchings of size k, and if p is an augmenting path with respect to M of maximum incremental weight, then $M \oplus p$ is a matching of size $k+1$ that is of maximum weight among all matchings of size $k+1$.*

Proof. By (25), it suffices to show that if M' is a matching of maximum weight among all matchings of size $k+1$, then there exists an augmenting path p with respect to M such that

$$\begin{aligned} w(M') &= w(M \oplus p) \\ &= w(M) + \Delta(p) \ . \end{aligned}$$

Consider $M \oplus M'$. As before, this is a set of vertex-disjoint cycles, even-length paths, and odd-length paths. The incremental weight of each cycle must be 0, because otherwise it would be possible to exchange the M and M' edges on this cycle to increase the weight of either M or M', which by assumption is impossible. The even-length paths must have incremental weight 0 for the same reason. Thus only the odd-length paths in $M \oplus M'$ can have nonzero weight.

Each odd-length path has either an extra M edge or an extra M' edge. Since there is one more edge in M' than in M, there must be exactly one more path with an extra M' edge than there are paths with an extra M edge.

Pair each path with an extra M edge with a path with an extra M' edge. This will leave all paths paired except for one path p which has an extra M' edge. The incremental weight of each pair must be 0, because otherwise it would be possible to increase the weight of either M or M' by switching M and M' edges in this pair. Therefore

$$\begin{aligned} \Delta(p) &= \Delta(M \oplus M') \\ &= w(M') - w(M) \ . \end{aligned}$$

The path p is an augmenting path with respect to M, and the matching $M \oplus p$ has $k+1$ edges and weight equal to the weight of M', therefore it too is of maximum weight among all matchings of size $k+1$. $\square$

In the next lecture we will show

Lemma 19.9 *Let M^* be a matching of maximum weight among all matchings and let M be a matching of size k of maximum weight over all matchings of size at most k. If $w(M^*) > w(M)$, then M has an augmenting path with respect to M of positive incremental weight.*

Theorem 19.10 *If one always augments by an augmenting path of maximum incremental weight, then one arrives at a matching of maximum weight after at most $\frac{n}{2}$ steps.*

Lecture 20 More on Matching

Let G be an undirected graph with weight function w. Recall from last lecture that the weight of a matching M in G, denoted $w(M)$, is the sum of the weights of the edges in M, and the incremental weight of a set A of edges, denoted $\Delta(A)$, is the sum of the weights of the unmatched edges in A less the sum of the weights of the matched edges in A. For an augmenting path p, $\Delta(p)$ gives the net change in weight that would be obtained by augmenting by p.

We ended the last lecture by proving the following lemma:

Lemma 20.1 *Let M be a matching of size k of maximum weight among all matchings of size k. If we augment M by an augmenting path of maximum incremental weight, then we obtain a matching of size $k+1$ of maximum weight among all matchings of size $k+1$.*

We also need to know that an augmenting path of positive incremental weight exists. This is established in the following lemma.

Lemma 20.2 *Let M be a matching of size k of maximum weight among all matchings of size at most k and let M^* be a matching of maximum weight among all matchings in G. If $w(M^*) > w(M)$, then M has an augmenting path of positive incremental weight.*

Proof. Again, consider the symmetric difference $M^* \oplus M$. As argued in the last lecture, this is a set of vertex-disjoint cycles and paths of alternating edges from M and M^*. We pair the odd-length paths as we did in the last lecture, with each pair consisting of one path with one more M than M^* edge and the other with one more M^* than M edge. We are left with a number of odd-length paths.

Each cycle and path of even length has incremental weight 0, otherwise the M and M^* edges could be switched to increase the weight of either M or M^*, contradicting the maximality of M or M^*. By the same argument, the incremental weights of the pairs of odd-length paths are 0. Thus we are left with a set of unpaired odd-length paths. Either all these paths have one more M^* edge than M edge or they all have one more M edge than M^* edge (otherwise there would be another pair). The latter is impossible, because then M^* would be a matching of greater weight and smaller cardinality than M, contradicting our assumptions. Thus all these unpaired paths are augmenting paths with respect to M. If we augment by all of them simultaneously, we achieve a maximum matching of weight $w(M^*) > w(M)$; therefore, at least one of them must have positive incremental weight. □

Thus we can construct a maximum-weight matching by beginning with the empty matching and repeatedly performing augmentations using augmenting paths of maximum incremental weight until a maximum matching is achieved. This takes at most $\frac{n}{2}$ augmentations, since the number of matched vertices increases by two each time. We will show below how to obtain augmenting paths efficiently in bipartite graphs.

20.1 Unweighted Bipartite Matching

Now we will see an $O(m\sqrt{n})$ algorithm of Hopcroft and Karp [51] for unweighted matching in bipartite graphs. Micali and Vazirani [80, 105] have given an algorithm of similar complexity for general graphs.

The idea underlying the algorithm of Hopcroft and Karp is similar to Dinic's idea for maximum flow. The algorithm proceeds in phases. In each phase, we find a maximal set of vertex-disjoint minimum-length augmenting paths, and augment by them simultaneously. In other words, we find a set S of augmenting paths with the following properties:

(i) if the minimum-length augmenting path is of length k, then all paths in S are of length k;

(ii) no two paths in S share a vertex;

(iii) if p is any augmenting path of length k not in S, then p shares a vertex with some path in S; *i.e.*, S is a setwise maximal set with the properties (i) and (ii).

We will need the following three lemmas:

Lemma 20.3 *A maximal set S of vertex-disjoint minimum-length augmenting paths can be found in time $O(m)$.*

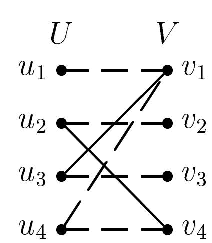
Lemma 20.4 *After each phase, the length of a minimum-length augmenting path increases by at least two.*

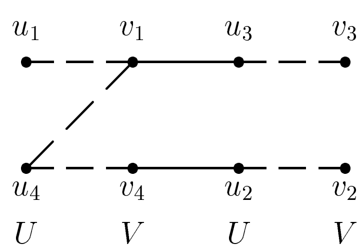
Lemma 20.5 *There are at most $\sqrt{n}$ phases.*

Proof of Lemma 20.3. Let $G = (U, V, E)$ be the undirected bipartite graph we are working in, and let M be a matching in G. We will grow a "Hungarian tree" from G and M. Calling it a tree is somewhat misleading, since the Hungarian tree is really a dag. It is obtained in linear time by a procedure similar to breadth-first search. We start with the free (unmatched) vertices in U at level 0. Starting from an even level $2k$, the vertices at level $2k+1$ are obtained by following *free (unmatched)* edges from vertices at level $2k$. Starting from an odd level $2k+1$, the vertices at level $2k+2$ are obtained by following *matched* edges from vertices at level $2k+1$. Since the graph is bipartite, the even levels contain only vertices in U and the odd levels contain only vertices in V. We do not expand a vertex that has been seen at an earlier level.

We continue building the Hungarian tree and adding more levels until all vertices have been seen at least once before or until we encounter a free vertex at an odd level (say t). In the latter case, every free vertex at level t is in V and is the terminus of an augmenting path of minimum length. Note that free vertices in U can be encountered only at level 0, since vertices at even levels greater than 0 are matched.

Example 20.6 The following figure illustrates a bipartite graph with a partial matching and its Hungarian tree. The solid lines indicate matched edges and the dashed lines free edges.





□

Now we find a maximal set S of vertex-disjoint paths in the Hungarian tree. We will use a technique called *topological erase*, called so because it is reminiscent of the topological sort algorithm we saw in Lecture 1. With each vertex x except those at level 0 we associate an integer counter initially containing the number of edges entering x from the previous level. Starting at a free vertex v at the last level t, we trace a path back until arriving at a free vertex u at level 0. This path is an augmenting path, and we include it in S. We then place all vertices along this path on a deletion queue. As long as the deletion queue is nonempty, we remove a vertex from the queue and delete it and all incident edges from the Hungarian tree. Whenever an edge is deleted, the counter associated with its right endpoint is decremented. If the counter becomes 0, the vertex is placed on the deletion queue (there can be no augmenting path in the Hungarian tree through this vertex, since all incoming edges have been deleted). After the queue becomes empty, if there is still a free vertex v at level t, then there must be a path from v backwards through the Hungarian tree to a free vertex on the first level, so we can repeat the process. We continue as long as there exist free vertices at level t. The entire process takes linear time, since the amount of work is proportional to the number of edges deleted. □

In order to prove Lemma 20.4 we will use the following lemma:

Lemma 20.7 *Let p be an augmenting path of minimum length with respect to some matching M, let M' be the matching obtained by augmenting M by p, and let q be an augmenting path in M'. Then*

$$|q| \quad \geq \quad |p| + 2|p \cap q| \ , \tag{26}$$

where $|q|$ and $|p|$ denote the number of edges of q and p, respectively, and $p \cap q$ denotes the set of edges common to p and q.

Proof of Lemma 20.7. If q and p are vertex-disjoint, then q is also an augmenting path with respect to M. Then $|q| \geq |p|$, since p is of minimum length, and (26) holds since the intersection is empty.

Otherwise, consider the symmetric difference $p \oplus q$ of the two paths. We observe the following facts.

(i) All edges in $q - p$ are in M if and only if they are in M'. This is because augmenting M by p only changes the status of edges on p.

(ii) Each time q joins (leaves) p it is immediately after (before) a free edge. This is because each vertex in p already has one adjacent edge in $p \cap M'$.

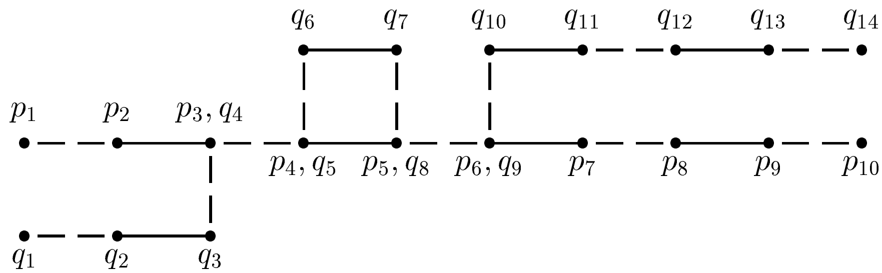
(iii) The endpoints of q are not contained in p, since they are free in M'.

It follows from property (iii) that $p \oplus q$ contains exactly four free vertices with respect to the original matching M, namely the endpoints of p and the endpoints of q. Thus $p \oplus q$, considered with respect to M, consists of exactly two augmenting paths and possibly some disjoint cycles as well. Each of the two paths must be at least as long as p, since p was of minimum length; thus $|p \oplus q| \geq 2|p|$. But

$$|q| + |p| \quad = \quad |p \oplus q| + 2|p \cap q| \quad \geq \quad 2|p| + 2|p \cap q| \ ,$$

from which (26) follows. □

Example 20.8 Lemma 20.7 is illustrated in the following picture.



In this example, the solid lines represent edges in M and the dashed lines represent edges not in M. The path $p_1, \ldots, p_{10}$ is an augmenting path with respect to the matching M, and $q_1, \ldots, q_{14}$ is an augmenting path after augmenting M by $p_1, \ldots, p_{10}$. The paths $p_1, p_2, p_3, q_3, q_2, q_1$ and $q_{14}, \ldots, q_9, p_7, \ldots, p_{10}$ are also augmenting paths with respect to M. The path $q_5, \ldots, q_8$ forms an alternating cycle with respect to M. □

Proof of Lemma 20.4. Suppose that at some phase we augmented M by a maximal set S of vertex-disjoint paths of minimum length k to obtain a new matching M'. Consider any augmenting path q with respect to M'. If q is vertex-disjoint from every path in S, then its length must be greater than k, otherwise S was not maximal. If on the other hand q shares a vertex with $p \in S$, then $p \cap q$ contains at least one edge in M', since every vertex in p is matched in M'. By Lemma 20.7, $|q|$ exceeds $|p|$ by at least two. □

Proof of Lemma 20.5. Let M^* be a maximum matching and let M be the matching obtained after $\frac{1}{2}\sqrt{n}$ phases. The length of any augmenting path with respect to M is at least $\sqrt{n}$. By a lemma from the last lecture, $M^* \oplus M$ contains a set T of exactly $|M^*| - |M|$ vertex-disjoint augmenting paths, and augmenting by all of them gives a maximum matching. But there can be at most $\sqrt{n}$ elements of T, otherwise they would account for more than n vertices. Thus $|M^*| - |M| \leq \sqrt{n}$. Since each phase increases the size of the matching by at least one, at most $\sqrt{n}$ more phases are needed. □

Since each phase requires $O(m)$ time and there are at most $O(\sqrt{n})$ phases, the total running time of the algorithm is $O(m\sqrt{n})$.