18.433 Combinatorial Optimization

Matching Algorithms

September 9,14,16 *Lecturer: Santosh Vempala*

Given a graph $G = (V, E)$, a *matching* M is a set of edges with the property that no two of the edges have an endpoint in common. We say that a vertex $v \in V$ is *matched* if v is incident to an edge in the matching. Otherwise the vertex is *unmatched*. A matching is *maximum* if there is no matching of greater cardinality. In particular, a maximum matching is called *perfect* if every vertex of G is matched.

A *bipartite* graph G is a graph in which the vertices of G can be partitioned in two sets A and B with the property that every edge in G has one endpoint in A and one in B. In the case of bipartite graphs, the following theorem characterizes graphs that have a perfect matching. For $U \subseteq A$ denote $N(U)$ the set of vertices that are adjacent to vertices in U.

Theorem 1 (Hall). *A bipartite graph with sets of vertices A, B has a perfect matching iff $|A| = |B|$ and $(\forall U \subseteq A)|N(U)| \geq |U|$.*

Proof. If a bipartite graph has a perfect matching, then it is easy to see that the right hand side is a necessary condition.

We will now prove the reverse implication. First note that the RHS condition implies that $\forall U \subseteq B$ as well, $|N(U)| \geq |U|$. If $|N(U)| < |U|$ for some subset $U \subseteq B$, then $|N(A \setminus N(U))| \leq |B \setminus U| < |A| - |N(U)| = |A \setminus N(U)|$ and $A \setminus N(U)$ violates the condition as well.

We proceed by induction on $|A|$. If $|A|$ is 0 or 1, the claim is true. Now consider two cases:

1. Suppose that $(\forall U \subseteq A, U \neq \emptyset, U \neq A)|N(U)| > |U|$. Consider $e = (u, v)$ and $G' = G - \{u\} - \{v\}$. In G', $\forall U \subseteq A - \{u\}$

$$|N_{G'}(U)| \geq |N(U)| - 1 \geq |U|.$$

So G' has a matching M of $A \setminus \{u\}$ into $B \setminus \{v\}$. Then $M_1 = M \cup \{e\}$ is a matching of A into B in G.

2. Now suppose the opposite to the previous case: there exists $A' \subset A$ nonempty such that $|N(A')| = |A'|$. Let G_1 be the graph induced by $A' \cup N(A')$. Let G_2 be the graph induced by $G - A' - N(A')$.

In G_1, $(\forall U \subseteq A')$, $N_G(U) = N_{G_1}(U)$, and $|N_{G_1}(U)| \geq |U|$. Thus, G_1 has a matching M_1 of A' into $N(A')$.

In G_2, $\forall U \subseteq B \setminus N(A')$, we have $N_G(U) = N_{G_2}(U)$ since there are no edges from $B \setminus N(A')$ to A'. Thus, $|N_{G_2}(U)| = |N_G(U)| \geq |U|$ and G_2 has a matching M_2 of $A \setminus A'$ into $B \setminus N(A')$. Now $M_1 \cup M_2$ is a perfect matching of G. □

Our goal in these lectures is to develop a fast algorithm for finding a matching of maximum cardinality in a given graph. Throughout this course, by "fast" we mean polynomial-time, i.e. the running time of the algorithm should be bounded by a fixed polynomial in the size of the input graph. The size of a graph is determined by number of vertices in the graph, denoted by n, and by the number of edges, denoted by m.

Now take a matching M with respect to the graph G. If every vertex of G is matched by M then M is a perfect matching and hence is a maximum matching of cardinality $\frac{n}{2}$. Should M not be perfect, then we would like to either find another matching of greater cardinality than M, i.e. *augment* M, or conclude that M is already maximum. One way to augment M is the following: find a path P in the graph that starts at an unmatched vertex and consists alternately of edges not in M and edges in M (i.e. unmatched edges and matched edges) and ends at an unmatched vertex. Then consider the set of edges M' obtained by deleting the edges M has in common with the path and adding the rest of the edges on the path, i.e. the symmetric difference of M and P, denoted by $M \oplus P$. It is easy to verify that M' is also a matching, and moreover it has one more edge than M. Such an alternating path P is called an *augmenting* path. This observation motivates the following "algorithm".

The Matching Algorithm

```
{
    1.  Start with any matching.
    2.  Find an augmenting path with respect to the current matching.
    3.  Augment the current matching.
    4.  Repeat the above two steps as long as possible.
}
```

When the algorithm terminates, we have a matching M with no augmenting paths. What do we do now? Our first lemma tells us that at this point M must be maximum.

Lemma 2. *A matching M is maximum iff it has no augmenting paths.*

Proof. We have seen that if M contains an augmenting path then it is not a maximum

matching.

So consider the converse. Assume that M does not contain an augmenting path. We will show that M is a maximum matching. In order to prove this we take some maximum matching M^* and show that $|M| = |M^*|$. Consider $M \oplus M^*$, the symmetric difference of M and M^*. Recall that this is the collection of edges that are in M but not M^* and vice versa. Since M and M^* both induce subgraphs of maximum degree one, it follows that $M \oplus M^*$ induces a subgraph of maximum degree two. Note that such a subgraph may consist only of disjoint paths and/or cycles. In addition, observe that since M and M^* are matchings these paths and cycles contain edges that are alternately in M and M^*.

Consider first the cycles in our induced graph. All such cycles must contain an even number of edges, otherwise there must be some vertex that is adjacent to two edges in either M or M^*, contradicting the definition of a matching. Thus, these cycles contain an equal number of edges from M and M^*.

Consider now the induced paths. Suppose we have a path P that contains an odd number of edges. Hence, either P contains one more edge from M than M^* or one more edge from M^* than M. In the former case note that P is then an augmenting path in G with respect to M^*, contradicting the maximality of M^*. In the latter case P is then an augmenting path in G with respect to M, contradicting our initial assumption. Hence all our induced paths contain an even number of edges and thus contain an equal number of edges from M and M^*.

So the paths and cycles induced by $M \oplus M^*$ contain an equal number of edges from M and M^*. Finally consider the edges that are not induced by $M \oplus M^*$. These edges are either in both M and M^* or in neither of them. It follows that M and M^* are of equal cardinality and hence M is a maximum matching. □

How long does our algorithm take? In each iteration of steps 2 and 3 we increase the size of the matching by one. Thus we can repeat steps 2 and 3 at most $\frac{n}{2}$ times. So we are left with the question of how long it takes to find an augmenting path. Actually, first we must figure out *how* to find an augmenting path. It turns out that this will be much easier to do for *bipartite* graphs, which we will consider first.

1 Bipartite graphs

Take a bipartite graph, with a matching M, and let $A^U \subseteq A$ and $B^U \subseteq B$ be the vertices unmatched by M. We wish to find an augmenting path with respect to M. To do this, we will find the set of vertices S accessible from A^U by alternating paths. If S includes a vertex of B^U then the alternating path to that vertex will be an augmenting path.

The set S is determined by building an alternating *forest* F as follows:

1. Start with all the vertices of A^U as separate components of F.
2. Add edges from vertices of $A \cap V(F)$ to vertices of B without merging any two connected components of F. That is, if a vertex of B is adjacent to more than one component, add it to only one of the components.
3. Then add the edges of M incident to vertices of $B \cap V(F)$.
4. Repeat the above two steps till no more edges can added to F.

If we find a vertex of B^U in the forest, then this gives us an augmenting path. If not, by the next lemma, the matching M is a maximum matching.

Lemma 3. *M is maximum iff no vertex of B^U is in F.*

Proof. If F includes a vertex v of B^U then the path from v to the vertex of A^U in the component containing v is an alternating path with unmatched vertices at its ends, i.e. an augmenting path. Hence M is not maximum.

Conversely, suppose that no vertex of B^U is included in F. In order to prove our result we introduce the notion of a a *vertex cover*. This is a set of vertices such that every edge is incident to at least one vertex in the set We will show that G has a vertex cover of size equal to the current matching. Since the size of any vertex cover, is at least the size of the maximum matching (one endpoint from each edge in the matching must be chosen in any vertex cover) this would prove that the matching M is maximum.

Let $X = A - V(F)$ and $Y = B \cap V(F)$. Then we claim that $X \cup Y$ is a vertex cover. Clearly, M meets every vertex of $X \cup Y$. Since M is a matching, no edge of M is incident to two vertices of $X \cup Y$. Now, given a matched vertex $a \in V(F)$, let (a, b) be the matching edge. From the description of F it follows that b must also be in $V(F)$. As a result, every edge of M meets at least one vertex of $X \cup Y$ and so $|M| = |X \cup Y|$.

All that is left to show is that $X \cup Y$ is a cover of the graph. Suppose not. Then there is an edge (a, b) with $a \in A$ and $b \in B$ that is not covered. Hence we have $a \in V(F)$ and $b \notin V(F)$. It follows that (a, b) is not a matching edge. In addition, $b \notin B^U$ otherwise it would have been added to $V(F)$. So b is matched, say by the edge (a', b), where $a' \neq a$. But this implies that F can be extended by adding the path aba' contradicting the assumption that F is maximal. □

From the proof of the lemma we may derive the following theorem.

Theorem 4. *(König) The size of a maximum matching in a bipartite graph is equal to the size of a minimum vertex cover of the graph.* □

We say that A has a matching into B if the maximum matching is of size $|A|$. In addition, denote by $\Gamma(X)$ is the set of neighbors of $X \subseteq V$. The classical theorem of Frobenius and Hall then follows from König's theorem (and is actually equivalent to it).

Theorem 5. *(Frobenius-Hall) A has a matching into B iff for every subset X of A, $X \leq |\Gamma(X)|$.*

Proof. Clearly if there is a subset X of A such that $X > |\Gamma(X)|$, then there can be no matching of cardinality $|A|$. Conversely, assume that $X \leq |\Gamma(X)|$ for all $X \subseteq A$. We will show that the minimum vertex cover is of cardinality $|A|$, from which the theorem will follow. We may assume that each vertex is incident to at least one edge and that $|A| \leq |B|$. Note that the vertices of A form a vertex cover of cardinality $|A|$. Suppose we have a vertex cover $X \cup Y$, where $X \subseteq A$ and $Y \subseteq B$. Observe that $\Gamma(A - X) \subseteq Y$. Thus $|A - X| \leq |\Gamma(A - X)| \leq |Y|$ and hence $|X \cup Y| \geq |A|$ as desired. □

Theorem 6. *A maximum matching can be found in a bipartite graph in $O(m\sqrt{n})$ time.*

Proof. It is easy to see that the time spent in finding an augmenting path is $O(m)$ and the total number of augmentations is at most $\frac{n}{2}$. So the total time is $O(mn)$. To improve upon this analysis observe that the algorithm for finding augmenting paths might find more than one path. In this case let us augment on a maximal set of disjoint augmenting paths. With this modification we can show that the number of phases (where a phase is the construction of the alternating forest) is $O(\sqrt{n})$. The key observation, which is left as an exercise, is the following:

Observation 7. *The length of the shortest augmenting path increases in each phase.*

Given this observation, after $\sqrt{n}$ phases, the augmenting paths all have length at least $2\sqrt{n}+1$. Now consider *any* optimal matching M^* and the symmetric difference of M and M^*. If M is not maximum then there must be some augmenting paths with respect to M in the symmetric difference. Since each of these has length at least $2\sqrt{n}+1$ there can only be $O(\sqrt{n})$ such paths in all (the total number of vertices is n). Thus $|M^*|-|M|<\sqrt{n}$ and hence the algorithm will terminate in at most $\sqrt{n}$ more phases. □

2 General graphs

It is not hard to see that the algorithm from the previous section does not apply to general graphs. The main problem is caused by odd cycles with a maximal number of matching edges, i.e. cycles of length $2k+1$ which contain k matching edges. Such cycles are called *blossoms*, an example of which is shown in Figure 1, where the matching edges are shown in bold.

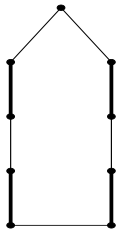

Figure 1: A Blossom.

The next lemma that shows us a way to deal with blossoms is the central idea in Edmonds' algorithm for finding a maximum matching in general graphs.

Lemma 8. *(Cycle Shrinking) Let M be a matching of G and B be a blossom. Further, assume that B is vertex-disjoint from (i.e. has no vertices in common with) the rest of M. Consider the graph G' obtained by contracting B to a single vertex. Then the matching M' of G' induced by M is maximum in G' iff M is maximum in G.*

Proof. First suppose that M' is not maximum in G'. From Lemma 2 it follows that G' contains an augmenting path P' with respect to M'. Suppose that P' does not intersect the blossom B in G, then P' is also an augmenting path in G and hence M is not maximum. So P' intersects B in G. In particular, the contracted blossom B must be an end vertex of the path P' in G' since B is vertex-disjoint from M'. Let P' meet B at the vertex v, and let

u be the unmatched vertex in the blossom. Let P'' be the path from v to u in the blossom that begins with the matching edge incident to v. It is easy to see that $P = P' \cup P''$ is then an augmenting path in G and so, again, M is not a maximum matching.

Now assume that M is not a maximum matching in G. We will show that M' is not a maximum matching in G'. So take an augmenting path P in G. We may assume that P intersects the blossom B, otherwise P is an augmenting path in G'. Note that since B contains only one unmatched vertex, it follows that at least one of the endpoints of P, say w, lies outside B. Let P' be the path created by starting at w and following P until it intersects the blossom. Observe that P' is an augmenting path in G' and the result follows. □

To find an augmenting path in a general graph, we will modify the procedure for bipartite graphs, so that it also detects blossoms. If it does, we shrink the blossom and restart on the new graph. Any augmenting path found on the new graph can be easily translated to an augmenting path in the original graph. Further, by the previous lemma, if the matching is maximum in the new graph, then it is also maximum in the original graph.

Here is a formal description of the algorithm. Let M be a matching of G and let U be the subset of unmatched vertices (if every vertex is matched then the matching is maximum). We construct a forest F so that it has one connected component for each vertex of U. As before extend F by alternately adding unmatched and matched edges. Then the edges of M that are added to F will be at an odd distance from U. Also, vertices that are at an odd distance from U will have degree two (with one unmatched edge and one matched edge). Let us call such vertices *inner* vertices and the rest *outer* vertices. The vertices of U are all outer vertices.

Now consider the neighborhood of outer vertices. One of the following four possibilities must arise.

1. If we find an outer vertex x incident to a vertex y not in F, then we can add the edges (x, y) and (y, z) to F where (y, z) is an edge of M.

2. If two outer vertices belonging to different components are adjacent, then the roots of these components have an augmenting path between them.

3. If two outer vertices x, y in the same component are adjacent, then let C be the cycle formed by the edge (x, y) along with the path from x to y in F. Let P be the path connecting C to the root of the component. First, we can switch the edges of P to obtain a matching M_1 of the same size as P. Then C satisfies the condition of the

cycle shrinking lemma. So we shrink C to a single vertex and get a new graph G'. Now the goal is to find an augmenting path in G'.

4. If every outer vertex only has inner vertices as neighbors, then M is already maximum. Too see this suppose F has p inner vertices and q outer vertices. Then $q - p = |U|$ since each matched outer vertex is matched with an inner vertex and vice versa. Now if we delete all the inner vertices of F from G, then the outer vertices will all be isolated components. But this means that any matching of G has to miss at least $q - p$ of them, and hence $q - p$ vertices of G. Since M misses exactly $q - p$ vertices, it must be maximum.

Thus, from the description of the algorithm, we obtain the lemma below.

Lemma 9. *At each step of the algorithm, we either increase the size of F, or decrease the size of G or find an augmenting path or stop with a maximum matching.* □

Theorem 10. *A maximum matching can be found in $O(n^4)$ time.*

Proof. Clearly our algorithm makes less than n augmentations. In addition, we can shrink at most n blossoms before finding an augmenting path. Finding an augmenting path or a blossom takes $O(m)$ time since in growing a forest we examine each edge at most once. Hence our overall running time is $O(mn^2) = O(n^4)$. □

The following theorem can be derived from Edmonds' algorithm.

Theorem 11 (Tutte). *A graph G has a perfect matching iff for any subset of vertices X, the number of odd-sized components of the graph $G \setminus X$ obtained by deleting X from G is at most $|X|$.* □

Proof. The necessity of the right hand condition is clear: if there exists a set of vertices X such that $G - X$ has more that $|X|$ odd-sized components, then there aren't enough vertices in X to match all the odd-sized components, because odd-sized components need an external vertex to be matched and can only be matched with vertices in X.

For the sufficiency, consider the forest in Edmonds' algorithm at the last step. Denote by X the set of inner vertices, $p = |X|$. Note that the vertices of X haven't been shrunk, because shrunk vertices have to be unmatched. If we consider $G - X$, then we get the outer vertices as isolated components (that is how the algorithm terminates: outer vertices only have inner vertices as neighbors). Some of these may correspond to shrunk odd components in the original graph. As in the description of the algorithm, call q the number of outer

vertices, so that $q - p$ is the number of unmatched vertices. Because of our hypothesis (applied to the set X), we have at most $|X|$ odd components in the original graph, that is, we have at most p outer vertices. In other words, $q = p$ and all vertices are matched. □

In his paper (called "Paths, Trees and Flowers") describing this algorithm, Edmonds also defined the notion of polynomial-time algorithms. In the decades since, this notion has come to play a fundamental role in complexity theory.