Matchings in Graphs

Lecturer: *Meena Mahajan* — Meeting: 2
Scribe: *Nitin Saurabh* — 13th January 2010

1 A General Algorithm for Matching

Let $G = (V, E)$ be a general undirected graph. We know from Berge's theorem that a matching M is maximum in G if and only if G has no augmenting paths with respect to M. This immediately suggests the following algorithm for maximum matching in undirected graphs.

```
$M = \emptyset$; $n = |V|$;
**for** $v = 1$ to $n$ **do**
    Find an augmenting path with respect to $M$ starting at $v$
    If success, we get a path $\pi$
    $M \leftarrow M \oplus \pi$
**end for**
Return $M$
```

Remark 1 *Although the above algorithm does a single pass on vertices, scanning them in a fixed order, it does not miss a maximum matching. The correctness of the algorithm follows from the fact that if there is no augmenting path from v with respect to M and π is some augmenting path with respect to M then there is no augmenting path from v with respect to $M \oplus \pi$. We will not prove this here.*

1.1 An algorithm for finding an augmenting path starting at a particular vertex

To find an augmenting path in G starting at v with respect to a matching M, we construct a directed graph H_v from the original graph G, where $V(H_v) = V(G)$ and $E(H_v) = \{(u, v) \mid \exists w \in V(G) : (u, w) \notin M \text{ and } (w, v) \in M\}$. We also define

$$\text{Free} = \{u \mid u \text{ is not matched by } M\}$$

$$\text{Good} = \{u \mid N(u) \cap \text{Free} \setminus \{v\} \neq \emptyset\}$$

Consider the following algorithm to find an augmenting path from v.

```
**if** $v$ is not free **then**
    Return $v$
**else if** $\exists u \in N(v) \cap \text{Free}$ **then**
    Return $(v, u)$
```

```
else
   Construct Free, Good, $H_v$.
   Search in $H_v$ for a simple directed path $\rho$ from $v$ to the set Good. Call such a path a
   pseudo augmenting path.
   if  the walk in $G$ corresponding to $\rho$ is a simple path then
      return this path.
   end if
end if
```

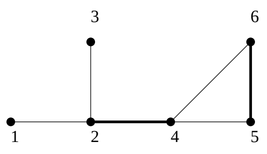
The correctness of the algorithm follows from the following claim:

Claim 2 *There is an augmenting path with respect to M starting at v if and only if there is a directed path ρ from v to the set Good in H_v such that ρ corresponds to a simple path in the original graph.*

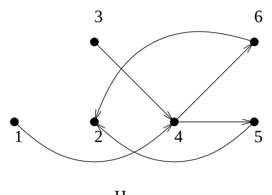
Proof: ($\Rightarrow$) Let τ be an augmenting path in G with respect to M, beginning at v and ending at w. Let u be the neighbour of w on τ. Then u is in the set Good, and the subpath of τ ending at u corresponds to a pseudo-augmenting path in H_v from v to u.

($\Leftarrow$) Let ρ be the directed path from v to some $u \in$ Good in H_v. Since $u \in$ Good, it has a neighbour $w \neq v$ in Free. The simple path in G corresponding to ρ, along with the edge (u, w), gives an augmenting path in G with respect to M. ■

It is easy to find a directed path ρ in H_v from v to Good. But the requirement that the walk underlying ρ be simple in G is what causes difficulties. (To retrieve an augmenting path in G, we may have to scan and discard several directed paths in H_v.) We observe first that this requirement is indeed necessary: Though the existence of an augmenting path in G with respect to M from v implies the existence of a pseudo augmenting path in H_v, the converse is not true. For instance, in the figure below, let $v = 1$. Then $u = 2$ is in Good, because of neighbour $w = 3$. In H_v, there is a directed path from 1 to 2 via 4 and 6, but its pullback in G is the walk 1-2-4-5-6-4-2 wich is not a simple path.



G



H_1

However, despite this problem, we can still use pseudo-augmenting paths, because of the following structure.

Claim 3 *If the full path corresponding to a pseudo augmenting path is not augmenting, then it contains an odd cycle.*

Proof: The full path corresponding to the pseudo augmenting path is not augmenting, which implies that full path is not simple. Hence it contains a cycle. If it is an even length cycle, then the pseudo augmenting path touches a vertex twice, contradicting the fact that a pseudo augmenting path is a simple directed path. ■

We will now see how to use this insight to get an efficient algorithm.

2 Edmonds' Blossom Shrinking Algorithm

A blossom B with respect to M is an odd cycle with maximal number of matched edges. That is, if it contains $2k+1$ vertices, then k edges are matched. Shrinking a blossom B means shrinking all vertices of B in G to a single new vertex, say b, and making it adjacent to all vertices in $V \setminus B$ which are adjacent to any node of B. Shrinking B shrinks the matching M to $M \setminus B$, where we discard all the M-edges on the cycle.

The strategy now is as follows. Repeatedly find directed paths in H from a free vertex to a good vertex (good now means it has at least one free neighbour). If such a path yields an augmenting path in G, augment the matching and start afresh. Otherwise, by Claim 3, the path yields a blossom, and we shrink it (and also shrink the current matching with respect to it) and continue. If no such directed paths are found, report the current matching.

To find directed paths efficiently, the algorithm maintains a forest of scanned edges.

Formally, the algorithm can be stated as follows.

Algorithm

```
1:  $M = \emptyset$, $H = G$, $N = M$
2:  EVEN = $\{v \mid v$ is free in $M\}$
    ODD = $\emptyset$, $E_F = \emptyset$
    Maintain always $V_F$ = EVEN $\cup$ ODD
    Construct forest $F = (V_F, E_F)$
3:  Pick an edge $(u, v)$ such that $u \in$ EVEN and $v \notin$ ODD
    If no such edge exists, Output M
4:  if $v \notin V_F$ then
5:      Put $v$ in ODD, $M(v)$ in EVEN, and $(u,v)$ and $(v, M(v))$ in $E_F$, where $M(v)$ is the
        vertex matched to $v$.
6:  else if $v \in$ EVEN and $u, v$ are connected in $F$ then
7:      A blossom $B$ has been found.
        Shrink $B$ to $b$, and set $H$ to $H \setminus B$, $N$ to $N \setminus B$.
        In $H$, put $b$ in EVEN.
8:      Goto Step 3
9:  else if $v \in$ EVEN and $u, v$ are not connected in $F$ then
10:     We have an augmenting path $\eta$ in $H$ with respect to $N$.
        Retrieve an augmenting path $\rho$ in $G$ with respect to $M$.
11:     Augment $M$.
12:     Goto Step 2.
13: end if
```

Most steps in the algorithm are self-evident. Let us elaborate on some steps.

Step 7: What exactly is the blossom? Let u, v belong to tree T in F. The edge (u, v) creates a fundamental cycle in T, and since both u and v are EVEN vertices (at even distance from the root of T), the cycle is odd. This cycle is the blossom.

Step 10: What is the augmenting path η? Start from the root of the tree in F containing u, come down to u, move to v, go up to the root of the tree containing v.

And how do we retrieve an M-augmenting path ρ in G? The idea is to unshrink the blossoms and matchings shrunk along the way. Consider one stage: H is obtained from G by shrinking B to b, and we have an augmenting path η in H. If η does not visit b, then η is also an M-augmenting path in G. If η visits b, there are two possibilities:

1. η ends in b. This can only happen if the blossom B included the root of the tree in which it was located, since otherwise b is not free in H. Now η in G ends at some vertex at B, from where we can go to the root by proceeding along the appropriate direction which has the matched edge.

2. η visits b along the way. Let η in G reach vertex x on B. The only matched edge leaving B is the one from a vertex y on B towards the root of the tree. From x, we can again choose the appropriate direction along B to maintain M-alternation, reach y, and then continue as in η.

Clearly, this can be done across multiple stages as well.

Now we need to prove that the algorithm is indeed correct; that it terminates with a maximum matching. This will follow from Lemma 4 and Lemma 5 below.

Lemma 4 *When the algorithm terminates, N is a maximum matching in H.*

Lemma 5 *Let N be a matching in H and B be a blossom. $N \setminus B$ is maximum in $H \setminus B$ if and only if N is maximum in H.*

To prove Lemma 4, we need a test for when a matching is maximum. Such a test can be devised based on odd components. For any set of vertices U, let $\circ(G \setminus U)$ denote the number of odd components of $(G \setminus U)$. Clearly, for every $U \subseteq V$ and for every matching M, M restricted to $(G \setminus U)$ must leave at least $\circ(G \setminus U)$ vertices free, one per odd component in $G \setminus U$. These vertices can only be matched, if at all, to vertices in U. So the following holds:

$$\begin{aligned}\text{Number of vertices free w.r.t. } M = |V| - 2 \times |M| &\geq \circ(G \setminus U) - |U| \\ \text{and hence } |M| &\leq \frac{|V| + |U| - \circ(G \setminus U)}{2}\end{aligned}$$

If the above relation is in fact an equality, then M is a maximum matching, and U certifies this fact. Such a subset U is therefore called a **witness set** for M.

Thus to certify that a matching M is maximum in a graph G, it suffices to find a witness set certifying this. Now, to establish Lemma 4, we show exactly this.

Claim 6 *The set ODD constructed by the algorithm at termination is a witness set certifying that N is maximum in H.*

Proof: When the algorithm terminates, the vertex set V is partitioned into three sets: EVEN, ODD and REST. The set EVEN is an independent set, because if there is an edge between any two vertices in EVEN, then either Step 6 or Step 9 of the algorithm is true, contradicting the fact that algorithm has terminated. Step 5 of the algorithm ensures that for every vertex in ODD, its matched vertex is put in EVEN. It also ensures that the vertices in EVEN are matched only to the vertices in ODD. Step 3 ensures that there are no edges connecting EVEN and REST. So, on deleting ODD, each vertex of EVEN forms a connected component by itself. And all vertices of REST are matched by M within REST, so on deleting ODD, the components within REST are all of even size. Hence

$$\begin{aligned} \circ(H \setminus ODD) = |\text{EVEN}| &= |\text{ODD}| + |\text{Free vertices}| \\ |\text{ODD}| + |V(H)| - \circ(H \setminus \text{ODD}) &= |V| - |\text{Free vertices}| = 2|N| \\ \frac{|\text{ODD}| + |V(H)| - \circ(H \setminus \text{ODD})}{2} &= |N| \end{aligned}$$

Thus, the set ODD is a witness set certifying that N is maximum in H. ■

Now we need to show that a maximum matching in the graph after shrinking blossoms implies a maximum matching in the original graph.

Proof of Lemma 5: We will prove the contrapositive; namely, $N \setminus B$ is not maximum in $H \setminus B$ if and only if N is not maximum in H.

($\Rightarrow$) Suppose that $N \setminus B$ is not maximum; then it has an augmenting path η. Step 10 of the algorithm (see the discussion after the algorithm pseudo-code) describes how to retrieve from η an N-augmenting path ρ in H. So N is not maximum.

($\Leftarrow$) Assume now that N is not maximum in H.

Let x be the vertex on B such that both edges of B incident on x are unmatched. By construction, N has an alternating even length path τ from a free vertex y to x. (τ could be ϵ, if x itself is free.) Then $N' = N \oplus \tau$ is a matching of the same size as N, and so N' is not maximum in H either. So there exists an N'-augmenting path ρ in H.

In $H \setminus B$, the matching $N' \setminus B$ is of the same size as the matching $N \setminus B$. So it suffices to show that $N' \setminus B$ is not maximum. We will construct from ρ an augmenting path in $H \setminus B$ with respect to $N' \setminus B$. If ρ completely avoids B, then ρ itself is the desired path, since it is also an augmenting path in $H \setminus B$. Otherwise, note that $N' \setminus B$ leaves the vertex b free. If ρ visits B, then since B has at only one free vertex, but ρ has both endpoints free, at least one endpoint of ρ is not on B. Start from such an endpoint and consider the initial segment η of ρ until the point where it first visits B. Then η in $H \setminus B$ reaches the free vertex b, and so it is an augmenting path. ■

We have thus established that the blossom-shrinking algorithm is indeed correct. Regarding its complexity, here's a rough-and-ready reckoner. There are at most $n/2$ augmentations. Between one augmentation and the next, the algorithm can add at most m edges to the forest, and shrink at most $n/3$ blossoms. Detecting an augmentation or blossom requires $O(m)$ time, since we maintain a forest. So the overall time is at most $O(n(n+m)m)$. A better analysis is possible.

3 Witness sets and odd set covers

In establishing correctness, we used the concept of witness sets. A witness set U for M certifies that M is maximum. It is not at all obvious a priori that witness sets exist. However, Claim 6 shows that for the matching N in the final graph H, ODD forms a witness set. We can show something stronger:

Claim 7 *When the algorithm terminates, the set ODD is a witness set for M in G.*

Proof of Claim 7: Note that all shrunk-blossom vertices are in EVEN in H. If we ushrink the blossoms, each blossom will form an odd component in $G \setminus$ ODD. Consider the effect of unshrinking one blossom at a time. let the blossom be of size $2k+1$. We know from Claim 6 that

$$|N| = \frac{|\text{ODD}| + |V(H)| - \circ(H \setminus \text{ODD})}{2}$$

When we replace b by B, V increases by $2k$, $|N|$ increases by k, and ODD remains the same in size. Also, since B is now an odd component instead of b, $\circ(H \setminus \text{ODD})$ remains the same. So the euality continues to hold.

Repeating this analysis for every blossom, we conclude that ODD is a witness set certifying that M is maximum in G. ■

Thus, a nice consequence of Edmonds' algorithm is that for every graph, witness sets always exist. We have proved the Tutte-Berge theorem:

Theorem 8 (Tutte-Berge's Theorem)

$$\max_M |M| = \min_{U \subseteq V} \frac{|V| + |U| - \circ(G \setminus U)}{2}$$

Proof: For every matching M and for every $U \subseteq V$, the LHS is upper bounded by the RHS. Hence taking maximum on left and minimum on right gives

$$\max_M |M| \leq \min_{U \subseteq V} \frac{|V| + |U| - \circ(G \setminus U)}{2}$$

Claim 7 shows that there is a witness set where equality is achieved. ■

The notion of witness sets can be framed equivalently using odd set covers.

Definition 9 *An* Odd Set Cover *is a family of disjoint subsets of V, $\mathcal{C} = (C_1, C_2, \ldots, C_l)$, where*

1. *each $|C_i|$ is odd, and*
2. *for each edge (u, v), either either $\{u\} \in \mathcal{C}$, or $\{v\} \in \mathcal{C}$, or $\exists C \in \mathcal{C}$ such that $\{u, v\} \subseteq C$.*

That is, the family $\mathcal{C}$ has disjoint odd subsets such that for every edge, either one (or both) endpoint is a singleton set in the family, or both endpoints are contained in a single set in the family.

Definition 10 *The cost of an odd set cover* $\mathcal{C} = (C_1, C_2, \ldots, C_l)$ *is defined to be* $\sum_i \mathit{Cost}(C_i)$, *where for each odd-sized* $C \subseteq V$,

$$\mathit{Cost}(C) = \begin{cases} 1 & \text{if } |C| = 1, \\ \frac{|C|-1}{2} & \text{otherwise} \end{cases}$$

It is easy to see that for any matching M and any odd set cover $\mathcal{C}$, $|M| \leq \mathrm{Cost}(\mathcal{C})$. Theorem 8 implies that this is tight; this is left as an exercise.