Matchings in Graphs

Lecturer: *Meena Mahajan*
Scribe: *Rajesh Chitnis*

Meeting: 3-4
21-28th January 2010

We now consider a fundamentally different algorithm to find a maximum matching. This algorithm, though polynomial time, does not compare well with the blossom-shrinking algorithm in terms of efficiency. However, it reveals, and uses, different insights into how maximum matchings in a graph behave.

1 Some preliminary definitions

Definition 1 *Let $\rho(G)$ denote the size of a maximum matching in G. A vertex $v \in V(G)$ is called* ***critical*** *if $\rho(G) > \rho(G - v)$.*

Definition 2 *Define the partition of the vertex set of G into the following three special subsets:*
$D(G) = \{u \mid u \text{ is not critical } \}$
$A(G) = \{u \mid u \text{ is critical and has a non-critical neighbour } \}$
$C(G) = \{u \mid u \text{ is critical and all neighbours of } u \text{ are critical } \}$
We denote the partition by $\pi(G) = \langle D(G), A(G), C(G)\rangle$. The set $A(G)$ is called the ***Tutte Set*** *of G.*

Definition 3 *Let $\mathcal{L} = \{M_1, \ldots, M_d\}$ be a family of equisized matchings in G. We again define a partition of V into three subset using the family $\mathcal{L}$, as follows.*
$D(\mathcal{L}) = \{u \mid \exists i \in [d] \text{ such that } u \text{ is free in } M_i\}$
$A(\mathcal{L}) = \{u \mid u \notin D(\mathcal{L}) \text{ and } u \text{ has a neighbour in } D(\mathcal{L})\}$
$C(\mathcal{L}) = V(G) \setminus \Big(D(\mathcal{L}) \cup A(\mathcal{L})\Big)$
Let the partition be denoted by $\pi(\mathcal{L}) = \langle D(\mathcal{L}), A(\mathcal{L}), C(\mathcal{L})\rangle$.

Clearly, given G and a family $\mathcal{L}$ of equisized matchings in G, the partition $\pi(\mathcal{L})$ can be computed in time polynomial in $|G|$ and $|\mathcal{L}|$. However, it is not obvious that the partition $\pi(G)$ can also be computed efficiently. If $\mathcal{L}$ is the set $\mathcal{L}_{\max}$ of all maximum matchings, then $\pi(\mathcal{L}) = \pi(\mathcal{L}_{\max}) = \pi(G)$. If $\mathcal{L} \subseteq \mathcal{L}_{\max}$, then $D(G) \subseteq D(\mathcal{L})$; the containment may be proper, in which case the partitions are different. Note that $\mathcal{L}_{\max}$ may be of exponential size. However, it has a polynomial-sized (in fact, of size at most n) subset $\mathcal{L}$ such that $\pi(\mathcal{L}) = \pi(G)$. This is because by definition, for each $u \in D(G)$, there is a maximum matching which leaves u free. Choose any one of these and call it M_u. So if $\mathcal{L} = \{M_u \mid u \in D(G)\}$, then $D(\mathcal{L}) = D(G)$, and then it follows that $\pi(\mathcal{L}) = \pi(G)$. Also, $|\mathcal{L}| \leq |D(G)| \leq n$. The challenge (and the strategy) is find such an $\mathcal{L}$.

2 Structural Algorithm to find Maximum Matching

The strategy to construct a maximum matching in G is as follows: Maintain a family $\mathcal{L}$ of equisized matchings. As long as $\mathcal{L}$ does not satisfy a certain **goodness criterion**, find a next matching which **improves** $\mathcal{L}$ with respect to a different **size criterion**. The goodness criterion is one that is efficiently checkable in the sense that if the family is not good, then a violating matching in it is easily detected, and also is sufficient in the sense that if $\mathcal{L}$ is good, then the matchings in it are maximum. The size criterion is straightforward: either we add to $\mathcal{L}$ another matching M of the same size as the matchings in $\mathcal{L}$ but leaving a vertex not in $D(\mathcal{L})$ free, resulting in $D(\mathcal{L})$ growing, or we find a matching M larger than those in $\mathcal{L}$, in which case we reset $\mathcal{L}$ to be $\{M\}$.

We describe the algorithm below, and the goodness criterion and the NextMatch subroutine are described after that. The algorithm returns a maximum cardinality matching along with the partition $\pi(G)$. We show later that the set $A(G)$ in the partition is a witness set.

1: $\mathcal{L} = \{\emptyset\}$.
2: Compute $\pi(\mathcal{L}) = \langle D(\mathcal{L}), A(\mathcal{L}), C(\mathcal{L})\rangle$.
3: **while** $\mathcal{L}$ is not good **do**
4: Find $M \in \mathcal{L}$ that is not $\mathcal{L}$-good.
5: $M' = \text{NextMatch}(M, \mathcal{L})$
6: **if** $|M'| = |M| + 1$ **then**
7: Set $\mathcal{L} = \{M'\}$; Update $\pi(\mathcal{L})$.
8: **else**
9: Set $\mathcal{L} = \mathcal{L} \cup \{M'\}$; Update $\pi(\mathcal{L})$.
10: **end if**
11: **end while**
12: Output any $M \in \mathcal{L}$ as a maximum matching.
13: Output $\pi(\mathcal{L})$ as $\pi(G)$.

In the sections below, we describe the goodness criterion, the NextMatch subroutine that gives the improvement with respect to the size criterion, and the proof of correctness. For now, note that the algorithm does not necessarily terminate when it first constructs a maximum matching, because it may not know that the matching is maximum. Only when the family $\mathcal{L}$ is good does it halt.

3 The goodness criterion

Definition 4 *A matching $M \in \mathcal{L}$ is said to be $\mathcal{L}$-good if*

1. *M does not match any vertex of $A(\mathcal{L})$ to a vertex in $A(\mathcal{L}) \cup C(\mathcal{L})$, and*
2. *M is near-perfect on all components of $G[D(\mathcal{L})]$ (the subgraph of G induced by the set $D(G)$).*

The family $\mathcal{L}$ is said to be good if every matching $M \in \mathcal{L}$ is $\mathcal{L}$-good.

Note that if $\mathcal{L}$ is good, then the second condition implies that all components of $G[D(\mathcal{L})]$ are odd.

Clearly, checking if $M \in \mathcal{L}$ is $\mathcal{L}$-good can be done in polynomial time. Hence checking if $\mathcal{L}$ is good, and if it is not, finding an M violating goodness, is possible in time polynomial in $|G|$ and $|\mathcal{L}|$. However, given the guarantees provided by NextMatch, clearly the algorithm never encounters an $|\mathcal{L}|$ of size more than n, and so this checking is possible in time polynomial in $|G|$.

We now show that this measure of goodness is indeed sufficient.

Lemma 5 *Let $\mathcal{L}$ be a family of equisized matchings and let $M \in \mathcal{L}$. If M is $\mathcal{L}$-good, then*

1. *M is a maximum matching, and*
2. *$A(\mathcal{L})$ is a witness set.*

Proof: Since M is $\mathcal{L}$-good, it saturates $A(\mathcal{L})$ and $C(\mathcal{L})$, matching vertices in $A(\mathcal{L})$ with vertices in $D(\mathcal{L})$. So it is perfect on all components of $C(\mathcal{L})$; hence all these components are even. On the other hand, M is near-perfect on all components of $D(\mathcal{L})$, so these are all odd. Hence $o(G \setminus A(\mathcal{L}))$ equals the number p of connected components of $D(\mathcal{L})$. Now, there is exactly one free vertex per component of $D(\mathcal{L})$ not matched to $A(\mathcal{L})$, and there are no other free vertices. So the number of free vertices in M is $(p - |A(\mathcal{L})|) = \Big(o(G \setminus A(\mathcal{L})) - |A(\mathcal{L})|\Big)$. Thus, $A(\mathcal{L})$ is a witness set certifying that M is a maximum matching. ■

Corollary 6 *Let $\mathcal{L}$ be a family of equisized matchings. If one matching from $\mathcal{L}$ is $\mathcal{L}$-good, then every matching of $\mathcal{L}$ is $\mathcal{L}$-good and so $\mathcal{L}$ is good.*

Proof: Let $M \in \mathcal{L}$ be $\mathcal{L}$-good. Let A, C, D denote $A(\mathcal{L}), C(\mathcal{L}), D(\mathcal{L})$ respectively. Then by Lemma 5, M is maximum and A is a witness set. Thus $|M| = \frac{|V|+|A|-o(G\setminus A)}{2} = \frac{|V|+|A|-o(D)}{2}$, as C has perfect matchings and so each component is even.

Let $M' \in \mathcal{L}$; since $|M| = |M'|$, we have $|M'| = \frac{|V|+|A|-o(D)}{2} = \frac{|C|}{2} + |A| + \frac{|D|-o(D)}{2}$. In the matching M', let there be l_1, l_2 edges from A to D and from A to C repectively; clearly $l_1 + l_2 \leq |A|$. Noting that G has no edges between D and C and that M' saturates A and C, we have

$$\begin{aligned}
& \frac{|C|}{2} + |A| + \frac{|D| - o(D)}{2} = |M'| \\
= \; & |M' \cap (A \times D)| + |M' \cap (A \times C)| + |M' \cap G[C]| + |M' \cap G[A]| + |M' \cap G[D]| \\
= \; & l_1 + l_2 + \frac{|C| - l_2}{2} + \frac{|A| - l_1 - l_2}{2} + |M' \cap G[D]| \\
= \; & \frac{|C|}{2} + \frac{|A| + l_1}{2} + |M' \cap G[D]| \\
\leq \; & \frac{|C|}{2} + \frac{|A| + l_1}{2} + \frac{|D| - o(D)}{2}
\end{aligned}$$

Thus we can conclude that $l_1 \geq |A|$ But $l_1 + l_2 \leq |A|$, so it must be the case that $l_1 = |A|$ and $l_2 = 0$. This means that M' matches all vertices of A to D, satisfying the first condition for $\mathcal{L}$-goodness.

So now, since we know that M' has no edges within A or between A and C, we have $|M'| = |M' \cap (A \times D)| + |M' \cap G[C]| + |M' \cap G[D]| = |A| + \frac{|C|}{2} + |M' \cap G[D]|$. But we also know that $M' = \frac{|C|}{2} + |A| + \frac{|D|-o(D)}{2}$. This is possible only if M' is near-perfect on each component of $G[D]$. So M' satisfies the second criterion for $\mathcal{L}$ as well and so M' is $\mathcal{L}$-good. ■

4 The Gallai-Edmonds Structure Theorem

Before giving the details of NextMatch, we note that the results of the previous section give an algorithmic proof of the Gallai-Edmonds structure theorem, stated below. First, a definition.

Definition 7 *A graph H is said to be hypomatchable, or factor-critical, if for every vertex $u \in H$, the graph $H \setminus \{u\}$ has a perfect matching.*

Theorem 8 *For any graph G, the partition $\pi(G) = \langle D(G), A(G), C(G) \rangle$ from Definition 2 satisfies the following properties:*

1. *$A(G)$ is a witness set.*
2. *$G[C(G)]$ has a perfect matching.*
3. *Every maximum matching in G*
   - *is perfect on $G[C(G)]$,*
   - *is near-perfect on each component of $G[D(G)]$, and*
   - *matches vertices in $A(G)$ to distinct components in $G[D(G)]$*
4. *Each component of $G[D]$ is hypomatchable.*

Proof of Theorem 8: The structural algorithm constructs a good family $\mathcal{L}$ with at least one maximum matching M. Lemma 5 shows (1). By the definition of goodness, since M is $\mathcal{L}$-good, (2) follows. To see (3), let M' be any maximum matching in G. If $M' \in \mathcal{L}$, then the proof of shows that M' satisifies (3). However, this proof also applies to a maximum matching M' not in $\mathcal{L}$, since the only way it uses $M' \in \mathcal{L}$ is to conclude that $|M'| = |M|$.

To see (4), let H be a component of $G[D]$, and let u be an arbitrary vertex in H. Since $u \in D$, there is a maximum matching M in G that leaves u free. By (3) above, M is near-perfect on H. Since u was chosen arbitrarily, it follows that H is hypomatchable. ■

5 The NextMatch sub-routine

We now complete the structural algorithm by giving a description of the NextMatch routine. Recall that this routine is applied on a family $\mathcal{L}$ of equisized matchings (say of size k), and a matching M in it that is not good according to Definition 4. It is expected to return either a matching of size $k+1$, or a matching of size k such that some vertex u saturated by every matching in $\mathcal{L}$ is left free by it. In the latter case, we say that the vertex u is **released** into D.

Let A, C, D denote the sets $A(\mathcal{L}), C(\mathcal{L}), D(\mathcal{L})$ respectively.

1. $\exists\ x \in A$ `such that` $y = M(x) \notin D$. `Note that` $x \in A$ `implies` $\exists$ `some neighbour` z `of` x `such that` $z \in D$.

   (a) If z is free in M, then y is free in $M' = M + xz - xy$.

   (b) Suppose z is not free in M.
   But $z \in D$ and so $\exists\ N \in \mathcal{L}$ such that z is free in N. Let ρ be the maximal M, N alternating path starting from z. Note that the first edge of ρ is an M-edge as z is free in N.

   If ρ ends in an M-edge, then augment (N, ρ) to get a larger M'.

   Now suppose ρ ends in an N-edge. If ρ avoids the edge xy then it also avoids the vertices x and y (as otherwise it would have to use the edge xy). We switch M on $\rho + xz + xy$ to get M'; this releases y.

   Now the only case left is that ρ ends in an N-edge and uses edge xy. Suppose ρ uses xy through y first; the case for x is analogous. Let ρ' be the suffix of ρ starting from y. Switch M on ρ' to get M'; this releases y.

2. $\exists$ `component` T `of` $G[D]$ `such that` M `is not near-perfect on` T.
   Three cases arise.

   (a) Suppose $M|_T$ is perfect on T.
   Then T must be of even size. Pick any $x \in T$, then $\exists\ N \in \mathcal{L}$ which leaves x free. So $N|_T$ is not perfect, and hence leaves x as well as some y free. Set $M = N$ and go to Case 2(c).

   (b) Suppose $M|_T$ leaves at least two vertices free, but none of these is free in M.
   Let x, y be free in M_T; by assumption, they are not free in M. Now $x \in D$ implies $\exists\ N \in \mathcal{L}$ which leaves x free. Let ρ be the maximal M, N alternating path starting at x. Clearly the first edge of ρ is an M-edge, as x is free in N.

   If ρ ends in an M-edge, then augment (N, ρ) to get a larger M'.

   So now suppose that ρ ends in an N-edge, say at z.

   If $z \notin D$, then switch N on ρ to get M', releasing z.

   So suppose $z \in D$.

   If ρ avoids y, then switch M on ρ to get M''. In $M''|_T$, both x and y are free; furthermore, x is also free in M''. Set $M = M''$ and go to Case 2(c).

Now the only case left is that ρ ends in an N edge at a $z \in D$, and ρ hits y.

If ρ reaches y via an N-edge (u, y), then let ρ' be the suffix of ρ starting at y. Switch M on ρ' to get M''. Both x and y are free in M''_T, and y is free in M'' as well. Set $M = M''$ and go to Case 2(c).

If ρ reaches y via an M-edge (v, y), then let ρ' be the suffix of ρ starting at v. Switch M on ρ' to get M' in which v is free. Since y was free in M_T, we know that $v \notin T$ and so $v \notin D$. So M' releases v into D.

(c) Suppose $M|_T$ leaves some x, y free, and at least one of them (say x) is free in M as well.

If $xy \in E(G)$, then set M' to $M+xy$ if y is free in M, and to $\Big(M-(y, M(y))+xy\Big)$ otherwise, to release $M(y)$.

So now assume xy is not an edge. Let ρ be the shortest xy path in T. Let z be the neighbour of x on ρ. Since $z \in T \subseteq D$, there is an $N \in \mathcal{L}$ that leaves z free; pick any such N. Let η be the maximal M, N alternating path starting at z.

If $\eta = \epsilon$, then $M' = M + xz$ is a larger matching.

So assume $\eta \neq \epsilon$. Then η must start with an M-edge.

If η also ends with an M-edge, then augment (N, η) to get a larger matching M'.

So consider that η ends with an N-edge.

If η does not visit x, then switch M on η and add xz to get a larger matching M'.

So now assume η visits x. Since x is free in M, once η visits x it has to end there. So assume η ends at x. Thus $\eta + xz$ is an odd length cycle X on which both M and N are maximum. On an odd cycle, any vertex can be made free in a maximum matching.

If η goes through y, let $u = M(y)$. By assumption, $u \notin D$. Replace M on X by the matching that leaves u free, tog et M' which releases u.

If η avoids y, switch M on part of X to leave z free; let this matching be M''. Now both z and y are free in in M''_T, and z is free in M'' as well. Repeat Case 2(c) with M'' instead of M and z, y instead of x, y. Note that $d_T(z, y) < d_T(x, y)$; so this kind of recursion will repeat at most n times and will eventually terminate.