COL751 - Lecture 23

1 Maximum matching in general graphs

In this lecture we will study the first polynomial time algorithm to compute maximum matching in general graphs that was proposed by Jack Edmonds in 1965.

We define the notion of alternating tree (similar to one that was used in bipartite graphs). Given a matching M, an M-alternating tree T is a tree with the following properties:

- The root comprises of one or more unmatched vertices, and all other vertices in T are matched.
- The vertices at odd level have exactly one child, and the corresponding edge from parent to child lies in M.

Let M be a matching, $S = (x_1, \ldots, x_k)$ be set comprising of all unmatched (free) vertices, and T be an *M-alternating tree* rooted at S. Such a tree can be computed in $O(m)$ time using the following algorithm.

```
Initialize $T$ as a tree with singleton node $S$ as root;
foreach $v \in S$ do set $visited(v) = True$;
foreach $v \in V \setminus S$ do set $visited(v) = False$;
$Q$ = a queue data structure initialized with vertices in $S$;
while $Q$ is non-empty do
    $y = Pop(Q)$;
    foreach $z \in N(y)$ do
        if visited(z)=False then
            Add $z$ as child of $y$ and mark $z$ as visited;
            Add $M(z)$ as child of $z$ and mark $M(z)$ as visited;
            Insert $M(z)$ at the end of queue;
        end
    end
end
```

Algorithm 1: Compute-Alternating-M-tree(G, S)

Let V_{odd} and v_{even} be respectively the vertices at odd and even depth in tree T. We can have the following three cases:

1. There exists no edge in $G[V_{even}]$.
2. There exists an edge (y, z) in $G[V_{even}]$, such that $root(y) \neq root(z)$.
3. There exists an edge (y, z) in $G[V_{even}]$, such that $root(y) = root(z)$.

We will separately analyze the three cases described above.

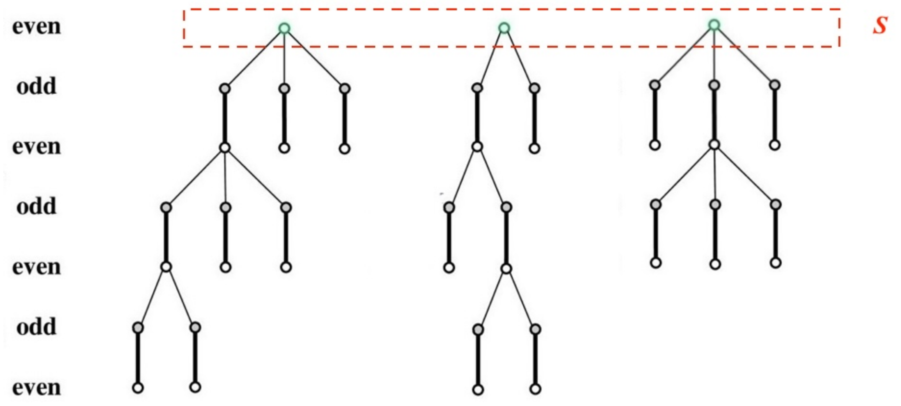

Figure 1: Alternating tree rooted at set S comprising of free vertices in G.

1.1 Case 1

If there is no edge in $G[V_{even}]$, then M is a maximum matching due to the following lemma.

Lemma 1. *Let M be a matching in G and R be a subset of vertices satisfying that* (i) *$G \setminus R$ has zero edges, and* (ii) *$|M| = |R|$. Then M is a maximum matching in G.*

Proof: By pigeonhole principle any matching of size $|R|+1$ in G must have at least one edge with both endpoints in $V \setminus R$. This is not possible as graph $G \setminus R$ has zero edges. Thus, M must be a maximum matching in G. □

In the above lemma we can set R as V_{odd}. Clearly, we have $|M| = |V_{odd}|$, and $G \setminus V_{odd}$ has zero edges.

1.2 Case 2

If there is an edge (y, z) in $G[V_{even}]$ satisfying $root(y) \neq root(z)$, then we get following augmenting path in G:

$$P = \ \text{TREEPATH}\big(root(y), y\big) \ \circ \ (y, z) \ \circ \ \text{TREEPATH}\big(z, root(z)\big).$$

By updating M as $M \oplus P$, we increase the size of matching.

1.3 Case 3

If there is an edge (y, z) in $G[V_{even}]$ satisfying $root(y) = root(z)$, then we have following odd length cycle C in G whose $\frac{|C|-1}{2}$ edges are matched:

$$C = \ \text{TREEPATH}\big(LCA(y, z), y\big) \ \circ \ (y, z) \ \circ \ \text{TREEPATH}\big(z, LCA(y, z)\big).$$

In this case we first toggle edges on path connecting root and $LCA(y, z)$ to ensure that C contains a free vertex under M.

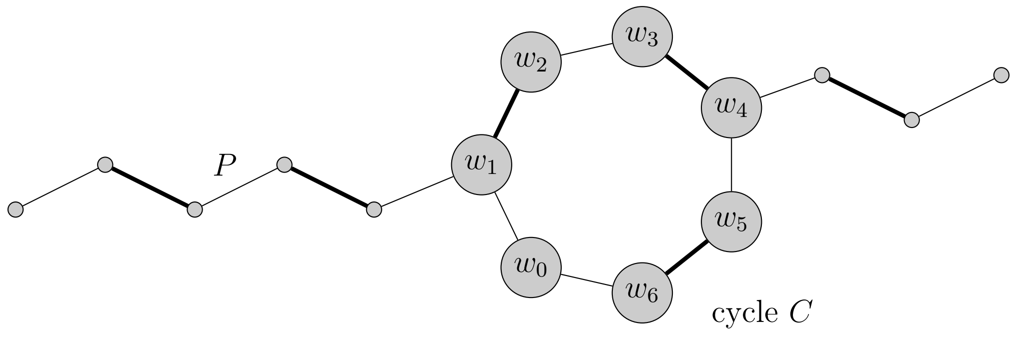

Figure 2: Depiction of an augmenting path P in G passing through a cycle C of length seven containing one free vertex and 3 matched edges. Edges in M are shown in thick.

Lemma 2. *Let M be a matching in $G = (V, E)$ and C be an odd length cycle containing $\frac{|C|-1}{2}$ edges of M and one free vertex. Then there exists an M-augmenting path in G if and only if there exists an M/C augmenting path in G/C.*

Furthermore, given an augmenting path in graph G/C one can compute an M-augmenting path in G in $O(|E|)$ time.

Proof: We will first prove that an M-augmenting path in G gives an M/C-augmenting path in G/C. Let P be an M-augmenting path in G. If $P \cap C = \emptyset$, then P is also an M/C-augmenting path in G/C. If $P \cap C \neq \emptyset$, then let P_0 be segment of P till first vertex in cycle C. The path P_0/C will be be an augmenting path in G/C as the vertex corresponding to cycle C in graph G/C is unmatched under M/C.

We now prove the second part. Let Q be augmenting path in G/C. If endpoints of Q doesn't contain C, then Q is also an M-augmenting path in G. Otherwise, Q can be appended / prepended with an appropriate segment of C to obtain an M-augmenting path in G. This takes $O(|E|)$ time. $\square$

Homework Prove that Lemma 2 fails if C doesn't contain a free vertex.

On combining cases 2 and 3, we obtain the following lemma.

Lemma 3. *For any n vertices, m edges graph G and a matching M of G, we can compute an M-augmenting path, if it exists, in $O(mn)$ time.*

As we need to compute augmenting paths at most $n/2$ times, we have the following result.

Theorem 1 (Jack Edmonds, 1965). *For any n vertices, m edges graph $G = (V, E)$ we can compute a maximum matching in $O(mn^2)$ time.*

2 Tutte-Berge Formula

We show in this section how the structure of Edmonds' matching algorithm can give an algorithmic proof of the Tutte-Berge Formula.

For any n vertex graph G, let $def(G)$ be defined as $n - 2|M_{opt}|$.

Theorem 2 (Tutte-Berge Formula)**.** *For any graph $G = (V, E)$, we have*

$$def(G) \;=\; \max_{R \subseteq V} \; oc(G \setminus R) - |R|, \tag{1}$$

where, $oc(H)$ denotes the number of connected-components in H having odd number of vertices.

Proof: Consider an $R \subseteq V$. We first prove that $def(G) \geqslant oc(G \setminus R) - |R|$. Let $C_1, \ldots, C_k$ be connected-components in $G \setminus R$ having odd number of vertices, where $k = oc(G \setminus R)$. Let M_0 be edges in M not incident to vertices in R, and M_R be edges in M incident to vertices in R. Then under M_0, each component C_i will have at least one unmatched vertex, say v_i. Now if we add edges of M_R to M_0, then at most $|R|$ vertices out of $v_1, \ldots, v_k$ will get matched. This proves that $def(G) \geqslant k - |R| = oc(G \setminus R) - |R|$.

In order to prove equality relation in Eq 1, we will show in Lecture 24 how to compute a subset R satisfying $def(G) = oc(G \setminus R) - |R|$. $\square$