1 Introduction

Matching theory is a core topic of both applied and theoretical graph theory, which is full of elegant and deep results. In 1965, Edmonds [1] gave the first efficient algorithm to find maximum matchings in graphs. His result is a milestone of algorithmic graph theory, providing the first efficient algorithm for a graph theoretical problem that could not be formulated as an integrality-preserving linear program.

Our first lecture focus on Edmonds's algorithm, from which we derive the Tutte-Berge Formula. An independent proof of this formula will be given in the next lecture.

In the following section, we explain the notation used, we give a proposition characterising maximum matchings of a given graph and make a remark which will prove to be useful.

2 Notation, a Proposition and a Remark

Given two sets X and Y, the *symmetric difference* of X and Y is the set

$$X \Delta Y := (X \setminus Y) \cup (Y \setminus X)\,.$$

Let G be a graph. The vertex set of G is $V(G)$ and its edge-set is $E(G)$. An *odd component* of G is a connected component with an odd number of vertices. An *even component* is defined analogously. If $X \subseteq V(G)$, then $G[X]$ is the subgraph of G induced by X, and $G - X := G[V(G) \setminus X]$. Similarly, if $M \subseteq E(G)$, we define $G[M]$ to be the subgraph of G spanned by the edges in M, and $G - M := G[E(G) \setminus M]$. Two edges of G are *adjacent* if they share a common vertex. A *matching* of G is a set of edges no two of which are adjacent. Thus, the size of a matching is at most $\lfloor |V(G)|/2 \rfloor$.

A matching M is *maximum* if every matching M' of G satisfies $|M'| \leq |M|$, and it is *perfect* if $|M| = |V(G)|/2$ (in other words, if every vertex of G is incident to an edge of M).

Given a matching M of G, a vertex $v \in V(G)$ is *M-matched* if v is incident to an edge in M. A vertex that is not M-matched is *M-unmatched*. A path $P := v_1, v_2, \ldots, v_k$ is *M-alternating* if its edges alternately belong to M, i.e. $v_{i-1}v_i \in M \Leftrightarrow v_i v_{i+1} \notin M$ for $i \in \{2, \ldots, k-1\}$. The path P is *M-augmenting* if it is M-alternating and both its endvertices are M-unmatched, i.e.

- v_1 and v_k are M-unmatched; and
- for every $i \in \{2, \ldots, k-2\}$, the edge $v_i v_{i+1}$ belongs to M.

In these notations, we may drop the "M-" when there is no risk of confusion.

The following proposition is the main tool used in finding maximum matchings in bipartite graphs, and it is the starting point for the general setting too.

Proposition 1. *Let G be a graph and M a maximum matching of G. Then, M is maximum if and only if G has no M-augmenting path.*

Proof. First, if G has an M-augmenting path P, then set $M' := M \Delta E(P)$. Note that M' is a matching of G with $|M'| > |M|$, and hence M is not a maximum matching of G.

Conversely, assume that M is not a maximum matching of G. Let M' be a maximum matching of G, thus $|M'| > |M|$. We consider the subgraph H of G spanned by the edges in $M \Delta M'$. Every vertex of H has degree 1 or 2; thus each connected component of H is either a path of a cycle. Moreover, every cycle of H is even (why?). Thus, the connected component of H fall into three categories: even cycle, path of even length, path of odd length. Since $|M'| > |M|$, there is a connected component C of H containing more edges in M' than in M. Thus, C is neither an even cycle, nor a path of even length. Consequently, C is a path $P := v_1, v_2, \ldots, v_{2k}$ of odd length. Further, its end-vertices are M-unmatched (by the definition of C), and for every $i \in \{1, \ldots, k-1\}$, the edge $v_{2i} v_{2i+1}$ belongs to M. Hence, P is an M-augmenting path. □

We end this section with the following remark, which gives an upper bound on the size of a maximum matching of a graph. Let M be a matching of a graph G, and let $U \subseteq V(G)$. Then, M induces a matching M' of $G - U$. Let $o(G-U)$ be the number of odd components of $G - U$. Each odd component of $G - U$ contains an M'-unmatched vertex. Such a vertex is M-matched only if it is matched in M to a vertex of U. Thus, at least $o(G-U) - |U|$ vertices of G are M-unmatched. Consequently,

$$|M| \leq \frac{1}{2}(|V(G)| - o(G-U) + |U|)\,.$$

Therefore, the maximum size of a matching of G is at most

$$\min_{U \subseteq V} \frac{1}{2} \cdot (|V(G)| + |U| - o(G-U))\,.$$

As we shall see, there is actually equality between these two quantities. This equality is known as the "Tutte-Berge Formula".

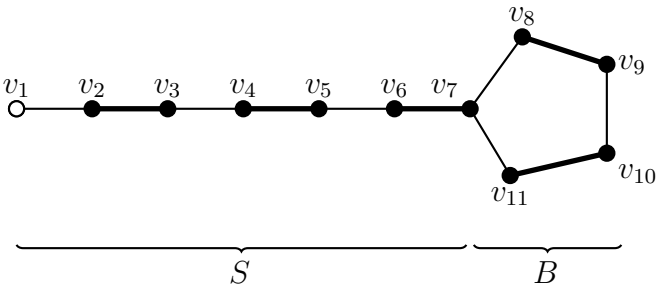

Figure 1: A flower F, composed of a stem S and a blossom B. Bold edges correspond to those in M.

3 Finding Augmenting Paths

We want to design an algorithm to find a maximum matching of a graph. In view of Proposition 1, a natural approach consists in starting from any matching and looking for augmenting paths.

Let M be a matching of a graph G, for instance a single edge. We may assume that G is connected. Let X be the set of M-unmatched vertices, and assume that $X \neq \emptyset$. We want to find an M-alternating path between two vertices of X, so that we can augment M along this path, or conclude that M is maximum if there is no such path. For every M-matched vertex $v \in V(G)$, we let v^* be the neighbour of v in $G[M]$.

While efficiently searching for augmenting paths in bipartite graphs is straightforward, the situation seems more difficult in general graphs. Let us analyse what can go wrong. Let $x, x' \in X$ and suppose that we have found an M-alternating path $P := xw_1w_1^*w_2w_2^* \dots w_kw_k^*x'$ between x and x'. If G were bipartite, P would be M-augmenting.

However, this may not be the case anymore. The reason is that P can contain an odd cycle. For instance, we may have $x = x'$, which prevents us from augmenting M by alternating along P.

Formally, a *flower* F is composed of an alternating path $S := v_1, \dots, v_{2k+1}$ of even length with $v_1 \in X$, and an odd cycle $B := v_{2k+1}, \dots, v_{2k'+1}, v_{2k+1}$ such that $V(S) \cap V(B) = \{v_{2k+1}\}$, and $v_{2i}v_{2i+1} \in M$ for every $i \in \{k+1, k'\}$. The path S is the *stem* of F, and B is the *blossom* of F. The vertex v_{2k+1} is the *root* of the blossom B. See Figure 1.

Note that, by the definition, the two edges of B incident with v_{2k+1} do not belong to M. Moreover, the stem is allowed to be a path of length 0 (just take $k = 0$ in the definition), in which case the flower is *reduced to its blossom*.

The relevance of flowers follows from the next lemma.

Lemma 2. *Let G be a graph with a matching M. Define X to be the set of M-unmatched vertices. Let $v_1, v_k \in X$ and let $P := v_1, \dots, v_k$ be a shortest M-alternating*

path (possibly with repeated vertices and/or edges). Then, either P is an M-augmenting path, or the vertices $v_0, v_1, \ldots, v_s$ yield a flower for some $s < k$.

Proof. Exercise! Hint: "If P has no repeated vertices, then... Otherwise, let s be the largest index such that $v_1, \ldots, v_{s-1}$ are pairwise distinct. Let $i \in \{1, \ldots, s-1\}$ such that $v_i = v_s$. First, realise that s must be even. Next, prove that i is odd by showing that if i is even, then P can be shortened." □

We make a straightforward but useful remark. Let M be a matching and F a flower for M, with stem S and blossom B. The path S is M-alternating. Thus, setting $M' := M \Delta E(P)$, we obtain a matching of size $|M|$, for which we have a flower reduced to its blossom. Moreover, the root of the blossom is M'-unmatched. This is why, in our algorithm, we may always assume that the flowers are reduced to their blossom, which is rooted at an unmatched vertex. This will be used implicitly in the sequel.

Thus, the question is: "what shall we do with a blossom rooted in X?". We shrink it! Formally, if B is a blossom of G rooted at $x \in X$, we define G/B to be the graph obtained by contracting B into a single vertex (and removing loops and parallel edges that may arise). That is, G/B is the graph obtained from G by removing the vertices in $V(B)$, and adding a new vertex b linked to every $v \in V(G) \setminus V(B)$ that is adjacent in G to a vertex of B.

We define M/B to be the set of edges of G/B corresponding to the edges of G in $M \setminus E(B)$. Thus, M/B is a matching of G/B. Moreover, observe that $|M/B| = |M| - \frac{|V(B)|-1}{2}$.

More generally, every matching N of G/B gives rise to a matching of G of size $|N| + \frac{|V(B)|-1}{2}$: first, let M be the set of edges of G corresponding to those in N. Note that most one vertex v of B is M-matched. Therefore, we can add to M the (unique) maximum matching of B in which v is uncovered. Each time we use the expression "*to unshrink a blossom*", we extend the matching we have in this way.

As we see below, it turns out that M/B is a maximum matching of G/B if and only if M is a maximum matching of G. This is the crux of the algorithm.

Theorem 3. *Let M be a matching of graph G and let B be a blossom rooted in X. Then, M is a maximum matching of G if and only if M/B is a maximum matching of G/B.*

Proof. Let b be the vertex of G/B obtained by the contraction of B. Recall that b is M/B-unmatched.

Suppose first that M/B is not a maximum matching of G/B, and let N be a matching of G/B with $|N| > |M/B|$. As we mentioned above, N yields a matching of G of size $|N| + \frac{|V(B)|-1}{2} > |M/B| + \frac{|V(B)|-1}{2} = |M|$. Therefore, M is not a maximum matching of G.

Conversely, suppose that M is not a maximum matching of G, and let $P := v_1, v_2 \ldots, v_{2k}$ be an M-augmenting path in G. If P is disjoint from B, then B yields an

M/B-augmenting path in G/B, and so M/B is not a maximum matching of G/B. So we suppose now that $V(P) \cap V(B) \neq \emptyset$. One of v_1 and v_{2k} is distinct from b, say $v_1 \neq b$. Let i_0 be the smallest index i such that $v_i \in V(B)$. We set $Q := v_1, v_2, \ldots, v_{i_0-1}, b$. Then, Q is an M/B-augmenting path of G/B, as wanted. □

Thus, whenever we find a flower, we first reduce it to a blossom rooted in X, and then we shrink B and use recursion to find a maximum matching M' of G/B. If $|M'| = |M/B|$, then Theorem 3 implies that M is maximum in G. Otherwise, we expand back M' into a matching M'' of G as in the previous proof. Hence, $|M''| > |M|$. Note that M' is *not* necessarily a maximum matching of G! But we could increase the size of the original matching of G. With these tools at hand, we can give a formal description of the algorithm.

4 The Algorithm

The global idea is to build alternating trees from each unmatched vertex, which will also us to efficiently find either an augmenting path—in which case we augment the matching and start again—or a blossom—in which case we shrink it and proceed recursively. The general picture of the algorithm is as follows.

Start with any matching M of G, e.g. $M = \emptyset$ or M is a single edge.
Set X to be the set of M-unmatched vertices.
while G contains an M-alternating path from X to X
  if P is an M-augmenting path, **then** set $M := M \Delta E(P)$ and update X.
  else P contains a blossom B.
    Recursively find a maximum matching M' of M/B.
    if $|M'| = |M/B|$ **then return** M.
    else unshrink M' to obtain a matching of G of size greater than $|M|$.
      Update X.
endwhile

The correctness of this algorithm follows directly from Theorem 3 and Lemma 2.

Let us see how to find an alternating path between vertices of X (which will be either an augmenting path or a flower), if any. Let G be a graph, M a matching of G and X the set of M-unmatched vertices. We start by labelling all the vertices in X by EVEN. The other vertices are UNLABELLED. For each even vertex u, we record its *unmatched parent* $r[u]$. The table r is initialised as $r[x] := x$ for every $x \in X$, the other values being undefined. All each step, the algorithm ensures the following invariant:

- if v is EVEN, then there is an alternating path of even length (possibly 0) from v to $r[v] \in X$; and

- if v is ODD, then there is an alternating path of odd length between v and $r[v] \in X$.

The algorithm processes all the EVEN vertices as follows. Let u be an EVEN vertex. While there is an unmarked edge uw adjacent to u, mark uw and apply the following.

If w is UNLABELLED, then label it ODD. Further, since w is UNLABELLED, we know that $w \notin X$ and hence w^* exists. We label w^* by EVEN. We set $r[w] := r[w^*] := r[u]$.

If w is EVEN, then we consider two cases according to whether $r[w]$ equals $r[u]$. If they belong to different trees, i.e. $r[w] \neq r[u]$, then we have found an M-augmenting path: go from $r[u]$ to u in the tree emanating from $r[u]$, then follow the edge uw and go from w to $r[w]$ in the tree emanating from $r[w]$ (see Figure 3). Otherwise, i.e. $r[w] = r[u]$, we have found a flower (see Figure 4). We reduce it to its blossom B, which is rooted at an EVEN vertex.

If none of the above applies to any edge incident to an EVEN vertex, then the algorithm concludes that the current matching is maximum in the current graph.

Let us prove that this is correct, and next compute the overall running time. Suppose that none of the cases above applies any more for any EVEN vertex. We want to show that the obtained matching M' is maximum in the current graph G'. To see this, first note that there are no edges between EVEN vertices, and no edges between an EVEN vertex and an UNLABELLED vertex. Moreover, every UNLABELLED vertex is M'-matched to another UNLABELLED vertex. Now, let U be the set of ODD vertices. Each EVEN vertex itself is an odd component of $G'-U$. So, $o(G'-U) = |W|$, where W is the set of EVEN vertices. Moreover, $|M'| = |U| + \frac{1}{2}(|V(G')| - |U| - |W|)$, since every UNLABELLED vertex is M'-matched and every ODD vertex is M'-matched to an EVEN vertex. Therefore,

$$|M'| = \frac{1}{2}(|V(G') + |U| - |W|) = \frac{1}{2}(|V(G')| + |U| - o(G' - U))\,,$$

so M' is a maximum matching of G' by the remark at the end of Section 2.

The running time of the algorithm is $O\left(|E(G)| \cdot |V(G)|^2\right)$. Indeed, updating X, finding an alternating path (breadth-first search) and shrinking a blossom can all be done in $O(|E(G)|)$. After at most $O(|V(G)|)$ shrinkings, either the algorithm ends, or the size of the original matching M is increased. Since $|M|$ can be increased at most $O(|V(G)|)$ times, the conclusion follows.

The fastest known algorithm [2] to compute a maximum matching in general graphs runs in time $O\left(|V(G)|^{1/2} \cdot |E(G)|\right)$.

A careful analysis of the output algorithm allows us to obtain a theorem, known as the Tutte-Berge formula. This will be dealt with in the next lecture.

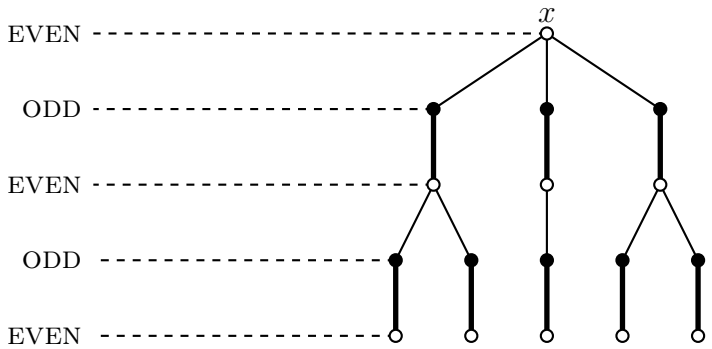

Figure 2: An alternating tree emanating from an unmatched vertex x. Bold edges are those in M, and EVEN vertices are white while ODD vertices are black.

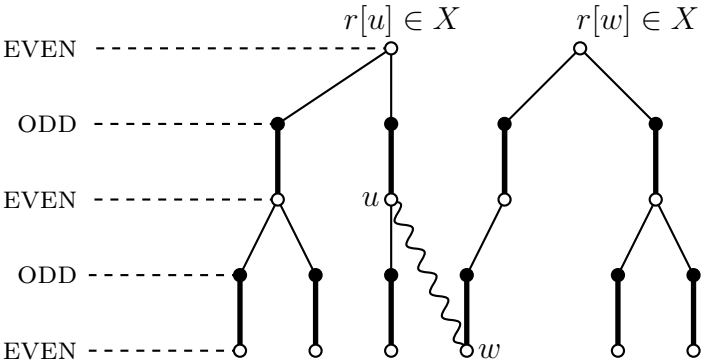

Figure 3: The squiggly edge is the edge uw being processed. Both u and w are EVEN, and $r[u] \neq r[w]$. Therefore, we have found an M-augmenting path between $r[u]$ and $r[w]$.

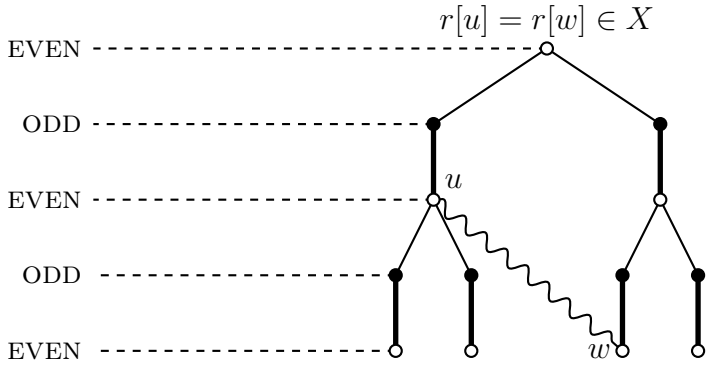

Figure 4: The squiggly edge is the edge uv being processed. Both u and w are EVEN, and $r[u] = r[w]$. Therefore, we have found a flower F. Here, the flower is reduced to its blossom (rooted at $r[u]$), but the F could have a stem (how?). Then, the algorithm first reduces F to its blossom and then shrink it.

References

[1] J. Edmonds. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, 1965.

[2] P. A. Peterson and M. C. Loui. The general maximum matching algorithm of Micali and Vazirani. *Algorithmica*, 3(4):511–533, 1988.