Edmonds' blossom algorithm

Zdeněk Dvořák

February 24, 2024

Our goal in this lecture will be to describe a polynomial-time algorithm to find a largest matching in a graph.

1 Augmenting paths

Let M be a (not necessarily perfect) matching in a graph G. A path $P = v_1v_2\dots v_m$ in G is *M-alternating* if v_1 is not covered by the matching and for every *even* $i < m$, the edge v_iv_{i+1} of the path is contained in M. I.e., P alternates between the edges not in the matching and the edges in the matching.

An alternating path P is *M-augmenting* if $m \geq 2$ and v_m is not covered by the matching. Note that this implies that m is even. Let

$$M' = (M - E(P)) \cup \{v_1v_2, v_3v_4, \dots, v_{m-1}v_m\}.$$

That is, we exchange the matching and the non-matching edges on P. Observe that M' is also a matching in G, and that $|E(M')| = |E(M)| + 1$. We say M' is obtained from M by *switching on P*.

Clearly, if G contains an M-augmenting path, then M is not the largest matching in G. Interestingly, the converse implication holds as well.

Lemma 1. *Let M be a matching in a graph G. Then M is the largest matching in G if and only if there exists no M-augmenting path in G. Moreover, given a matching M' in G larger than M, we can find an M-augmenting path in G in time $O(|V(G)|)$.*

Proof. As we have already argued, if G contains an M-augmenting path then M is not the largest matching in G.

Conversely, suppose that M is not the largest matching in G; we need to show that then G contains an M-augmenting path. Let M' be a matching in G with more than $|E(M)|$ edges, and consider the subgraph $F = (M \cup M') -$

$(M' \cap M')$ of G. Note that $|E(F \cap M')| - |E(F \cap M)| = |E(M')| - |E(M)| > 0$. Each vertex of F has degree at most two, and in case that a vertex of F has degree two, then it is incident with an edge of M and an edge of M'. Hence, each component K of F satisfies one of the following conditions:

- K is a cycle (of even length) in which edges of M and M' alternate, and thus $|E(K \cap M')| = |E(K \cap M)|$, or
- K is a path in which edges of M and M' alternate, and
  - K has one end in M and the other end in M', and thus K has even length and $|E(K \cap M')| = |E(K \cap M)|$, or
  - K has both ends in M, and thus K has odd length and $|E(K \cap M')| < |E(K \cap M)|$, or
  - K has both ends in M', and thus K has odd length and $|E(K \cap M')| > |E(K \cap M)|$.

Since $|E(F \cap M')| > |E(F \cap M)|$, we conclude that F has a component K of the last type, i.e., $K = v_1 v_2 \dots v_m$ is a path in which edges of M and M' alternate and such that $v_1 v_2, v_{m-1} v_m \in E(M')$. Since v_1 and v_m have degree one in F, they are not covered by M, and thus K is an M-augmenting path in G.

Note also that if we are given the matching M', we can find the components of F and pick K in time $O(|V(G)|)$. □

This gives us the following basic outline of an algorithm to find a largest matching in a graph G:

- Let M be an empty matching in G.
- While there exists an M-augmenting path P in G:
  - Replace M by the matching obtained from M by switching on P.

For the resulting matching M, there exists no M-augmenting path, and thus M is a largest matching in G by Lemma 1. Of course, in order to implement this algorithm, we need to be able to find an M-augmenting path efficiently (without explicitly knowing the larger matching M'), if one exists.

A reader with an experience in algorithmic design might feel that at this point we are basically done—cannot we just find an augmenting path by some variation of a graph search (DFS, BFS, ...)? However, this is not as straightforward as it might seem. It matters whether the search reaches a vertex through an edge in the matching M or through a non-matching edge,

which forces us to do the search on an auxiliary graph where each vertex of G appears twice (once for the possibility of reaching it through the matching edge, once for the possibility of reaching it through a non-matching edge). However, then we may end up with an "augmenting walk" (in which some vertices repeat twice) rather than an augmenting path. Suppose for example that we end up with an alternating walk u, v, w, x, y, w, v, z, where u and z are not covered by M and the edges vw and xy belong to M. There is no way how to turn this walk into an augmenting path!

2 Blossoms

The key observation is that we only can get stuck in this kind of situation where there is an alternating path ending in an odd cycle. An *M-blossom* consists of an M-alternating path $v_1v_2 \dots v_m$ with m odd (the *stem* of the blossom) and an odd cycle $C = v_m u_1 u_2 \dots u_k$ disjoint from $\{v_1, \dots, v_m\}$ (the *head* of the blossom) such that $u_1u_2, u_3u_4, \dots, u_{k-1}u_k \in E(M)$. The M-blossom is *simple* if $m = 1$, i.e., the stem consists of only a single vertex v_1 (not convered by M). The following lemma shows that if we are interested in the existence of an M-augmenting path, then we can only focus on simple blossoms.

Lemma 2. *Let M be a matching in a graph G and let $Q = v_1v_2 \dots v_m$ be an M-alternating path in G with m odd. Let M' be the matching in G obtained from by M by switching on Q. Then v_m is not covered by M' and $|E(M')| = |E(M)|$. Moreover, G contains an M-augmenting path if and only if it contains an M'-augmenting path, and an M'-augmenting path in G can be turned into an M-augmenting path in time $O(|V(G)|)$.*

Proof. The facts that v_m is not covered by M' and that $|E(M')| = |E(M)|$ are simple observations. In particular, M is a largest matching in G if and only of M' is a largest matching in G, and by Lemma 1, G contains an M-augmenting path if and only if G contains an M'-augmenting path.

Finally, suppose that P is an M'-augmenting path in M', and let M'' be the matching obtained from M' by switching on P. We have $|E(M'')| > |E(M')| = |E(M)|$. By Lemma 1 applied to M and M'', we can find an M-augmenting path in G in time $O(|V(G)|)$. □

Suppose C is a simple M-blossom. Let G/C denote the graph obtained from G by contracting all vertices of C to a single vertex c (suppresing the loops and the parallel edges that might arise), and let M/C denote the matching $M - E(C)$ in G/C. Note that c is not covered by M/C. This

operation again behaves well with respect to the existence of an augmenting path.

Lemma 3. *Let M be a matching in a graph G and let $C = v_1u_1 \dots u_k$ be a simple M-blossom with stem v_1 in G. Then G contains an M-augmenting path if and only if G/C contains an (M/C)-augmenting path. Moreover, an (M/C)-augmenting path R in G/C can be turned into an M-augmenting path in G in time $O(|V(G)|)$.*

Proof. Let c be the vertex of G/C obtained by contracting c.

Suppose first that G contains an M-augmenting path P. If P is vertex-disjoint from C, then P is also an (M/C)-augmenting path in G/C. Otherwise, note that since v_1 is the only vertex of C not covered by M, P has an end $x \notin V(C)$. Let P' be the shortest subpath of P from x to a vertex of C. Note that the last edge of P' is not contained in M, since v_1 is not covered by M and every other vertex of C is incident with an edge of M contained in C. Therefore, the path in G/C obtained from P' by replacing y by c is (M/C)-augmenting.

Conversely, suppose that R is an (M/C)-augmenting path in G/C. If $c \notin V(R)$, then R is also an M-augmenting path of G. Otherwise, since c is not covered by M/C, the path R starts in c. Let u be the second vertex of R and let v be a vertex of C adjacent to u. Since the cycle C has odd length, one of the M-alternating paths from c to v in C has even length; let R' be this path, and note that either $R' = v_1$ or R' ends with an edge of M. Therefore, $R' + vu + (R - c)$ is an M-augmenting path in G. Clearly, the construction of this path from R can be performed in time $O(|V(G)|)$. □

Suppose we have an algorithm $\mathcal{B}$ that given a graph G and a matching M in G decides that G does not contain an M-augmenting path, or returns an M-augmenting path in G, or returns an M-blossom in G. By Lemmas 2 and 3, we can turn this algorithm to an algorithm $\mathcal{A}$ to return an M-augmenting path (or decide that no such M-augmenting path exists) as follows:

- Run the algorithm $\mathcal{B}$ for G and M.
  - If the outcome is an M-augmenting path or the conclusion that no M-augmenting path in G exists, return the same.
  - Otherwise, $\mathcal{B}$ returns an M-blossom with stem Q and head C.
    - Let M' be obtained from M by switching on Q, so that C is a simple M'-blossom.
    - Run the algorithm $\mathcal{A}$ recursively for G/C and M'/C.

- If the outcome is that G/C does not contain an (M'/C)-augmenting path, then return that G does not contain an M-augmenting path (this is correct by Lemmas 2 and 3).
- Otherwise, let R be the returned (M'/C)-augmenting path in G/C. Turn R into an M'-augmenting path in G using the algorithm from Lemma 3, then to an M-augmenting path in G using the algorithm from Lemma 2, and return it.

For the time complexity, note that $|V(G/C)| \leq |V(G)| - 2$, and thus the depth of the recursion is at most $|V(G)|/2$. At each level of the recursion, we perform one call of $\mathcal{B}$, then contract C, then recurse, then perform postprocessing in time $O(|V(G)|)$. A straightforward way of performing the contraction of C takes time $O(|E(G)|)$, though this can be improved with a more clever implementation. Overall, the time complexity of this algorithm is $O(|V(G)||E(G)|)$ plus $O(|V(G)|)$ times the complexity of the algorithm $\mathcal{B}$.

3 Finding an augmenting path or a blossom

We can now use a simple variation of BFS to obtain the algorithm $\mathcal{B}$. Let L_0 be the set of vertices of G that are not covered by M; we can assume that $L_0 \neq \emptyset$, as otherwise M is a perfect matching and there obviously does not exist an M-augmenting path. Let us now proceed as follows for $i = 0, 1, \ldots$, defining layers L_i so that for $i > 0$, each vertex $u \in L_i$ has a neighbor $p(u) \in L_{i-1}$. Moreover, the edge $up(u)$ will belong to M if and only if i is even. Thus, there exists an alternating path $P_u = up(u)p^2(u)\ldots p^i(u)$ ending in a vertex $p^i(u) \in L_0$ not covered by M. Finally, we are going to have $|L_i| = |L_{i+1}|$ for every odd i, implying that M forms a perfect matching between L_i and L_{i+1}:

- Let us first consider the case that i is even.
  - Suppose that there is an edge $uv \in E(G[L_i])$. Recall that if $i > 0$, the edge $up(u)$ belongs to M, and thus $uv \notin E(M)$. If the paths P_u and P_v are disjoint, then their union with uv is an M-augmenting path, which we return. Otherwise, $(P_u \cup P_v) + uv$ is a blossom, which we again return.
  - Otherwise, let L_{i+1} consist of the vertices $z \in V(G) \setminus (L_1 \cup \ldots \cup L_i)$ which have a neighbor z' in L_i (over an edge not belonging to M, since either z' is not covered by M, or $z'p(z') \in E(M)$). For

each such vertex z, choose a neighbor $z' \in L_i$ arbitrarily and let $p(z) = z'$.

- Next, consider the case that i is odd.
  - If L_i is empty, then the algorithm stops and returns that G does not contain any M-augmenting path.
  - Otherwise, suppose that there is an edge $uv \in E(G[L_i] \cap M)$. If the paths P_u and P_v are disjoint, then their union with uv is an M-augmenting path, which we return. Otherwise, $(P_u \cup P_v) + uv$ is a blossom, which we again return.
  - Otherwise, consider a vertex $v \in L_i$ and let uv be the incident edge of M. Note that $u \notin L_j$ for any $j < i$, since M forms a perfect matching on $L_1 \cup L_2 \cup \ldots \cup L_{i-1}$ and L_0 consists of vertices not covered by M. Let $L_{i+1} = \{u : v \in L_i, uv \in E(M)\}$, and for each $uv \in E(M)$ with $v \in L_i$, let $p(u) = v$. Note that this ensures that $|L_{i+1}| = |L_i|$ and M forms a perfect matching between L_i and L_{i+1}, as required.

Suppose that this algorithm ends without finding an M-augmenting path or an M-blossom, and let m be maximum such that the layer L_m is non-empty; clearly m is even, since $|L_i| = |L_{i+1}|$ for every odd i. Note that $L_0 \cup L_2 \cup \ldots \cup L_m$ is an independent set in G: For each even i, the set L_i is independent (as otherwise the algorithm would return an M-augmenting path or an M-blossom), and all edges from L_i to $V(G) \setminus (L_1 \cup \ldots \cup L_i)$ belong to L_{i+1}. Let $S = L_1 \cup L_3 \cup \ldots \cup L_{m-1}$; since $|L_i| = |L_{i+1}|$ for every odd i, we have $|L_0 \cup L_2 \cup \ldots \cup L_m| = |S| + |L_0|$. Thus, $G - S$ is an independent set of size $|S| + |L_0|$, and this implies that every matching in G leaves at least $|L_0|$ vertices uncovered. Since M leaves exactly $|L_0|$ vertices uncovered, M is a largest matching in G, and thus there indeed is no M-augmenting path in G.

4 Putting things together

Note that the algorithm $\mathcal{B}$ is a straightforward variation on BFS and can be implemented in time $O(|E(G)|)$. This results in the algorithm $\mathcal{A}$ having $O(|V(G)||E(G)|)$ time complexity. Finally, the algorithm from the end of Section 1 runs the algorithm $\mathcal{A}$ $O(|V(G)|)$ times (more precisely, $\beta(G) + 1$ times). Hence, we obtain an algorithm to find a largest matching in a general graph in time $O(|V(G)||E(G)|\beta(G)) \leq O(|V(G)|^2|E(G)|)$. Note that there

exist more efficient (but substantially more complicated) algorithms for the problem.

One might wonder whether we do not overcomplicate things in Section 2 by finding an (M/C)-augmenting path in G/C and turning it to an M-augmenting path in G. Could we instead find a largest matching in G/C and turn it into a largest matching in G, instead? This does not work (at least not easily), as indicated by the following exercise.

Exercise 4. *Find a graph G, a matching M in G, and a simple M-blossom C such that G has a perfect matching, but G/C does not.*