Matching Definitions, Theorems, Algorithms - version 1.4

Given a set A, a subset $B \subseteq A$, and a property P, we say that B is **maximal** with property B iff B has property P, while for all elements $e \in A \setminus B$, $B \cup \{e\}$ does not have property P.

A **simple graph** can have at most one edge between two given vertices. **Multigraphs** can have several distinct edges between the same two vertices; multigraphs are usually not relevant when discussing matching. For a graph/multigraph, we consistently use $n = |V|$ and $m = |E|$.

Given a (multi)graph $G = (V.E)$ and a set of vertices $A \subseteq V$, the subgraph of G **induced** by A has vertex set A and edge set all the edges of E with both endpoints in A. We also use $G \setminus A$ to denote the graph subgraph of G induced by $V \setminus A$.

Given an undirected graph $G = (V, E)$, a **matching** is a subset $M \subseteq E$ such that no two edges in M share a vertex.

Definition 1 Given a matching M in $G = (V, E)$, an edge $e \in E$ is *matched* if $e \in M$, and *free* if $e \in E \setminus M$. A vertex v is *matched* if v has an incident matched edge, and *free* otherwise.

Definition 2 A *perfect matching* is a matching in which every vertex is matched.

Definition 3 Given a matching M in $G = (V, E)$, a path (cycle) in G is an *alternating path (cycle)* with respect to the matching M if it is simple (has no repeated vertices) and consists of alternating matched and free edges. An alternating path is an *augmenting path* (with respect to M) if its endpoints are free.

Theorem 1 *Given a graph $G = (V, E)$ and a matching $M \subseteq E$, M is a maximum matching iff there is no augmenting path in G with respect to M.*

Definition 4 A graph $G = (V, E)$ is bipartite iff V can be partitioned in A and B such that every edge of E has one endpoint in A and one endpoint in B.

Fact 2 *A graph is bipartite iff it does not have any odd cycle.*

Definition 5 If $G = (V, E)$ is an undirected graph, a **vertex cover** of G is a subset of V where every edge of G is adjacent to one node in this subset. The minimum vertex cover problem asks for the size of the smallest vertex cover. An **edge cover** of G is a subset subset of E where every vertex of G is adjacent to one edge in this subset. The minimum edge cover problem asks for the size of the smallest edge cover.

Fact 3 *For any graph $G = (V, E)$, any $M \subseteq E$ matching in G and Q vertex cover in G, $|M| \leq |Q|$.*

Theorem 4 *In a bipartite graph, the size of a maximum matching equals the size of the minimum vertex cover.*

Theorem 5 *In a bipartite graph $G = (V, E)$, with V partitioned into A and B, there is a matching with every vertex of A matched if and only if for all $X \subseteq A$, $|\Gamma(X)| \geq |X|$, where $\Gamma(X) = \{v \in B \mid \exists x \in X \;,\; xv \in E\}$.*

MAXIMUM MATCHING ALGORITHM (EDMONDS) $(G = (V, E))$

$M \leftarrow \emptyset$
while M has augmenting path P **do**
    $M \leftarrow M \oplus P$

The algorithms for computing a maximum matching uses *S-Trees* to search for augmenting paths in a graph G with respect to matching M.

We can prove that the algorithm below either finds an augmenting path or finds a flower F, composed of an odd cycle B of G, called the *blossom* of F, with exactly one vertex v free with respect to $E(G[B]) \cap M$ (v is called the *base* of the blossom), and either v is free with respect to M or there is an alternating path called the *stem* of F, from x to an free vertex u (the *root* of the flower).

S-TREES CONSTRUCTION $(G = (V, E), M)$

$Q \leftarrow \{v \in V \mid v \text{ free}\}$
Each vertex x in Q is a root of a S-Tree ($p(x) \leftarrow NULL$), and is labeled **even**.
while $Q \neq \emptyset$ **do**
  Pick $x \in Q$
  if all edges incident to x are investigated **then**
    $Q \leftarrow Q \setminus \{x\}$
  else
    let xy be an edge incident to x; mark it investigated
    if y unlabeled // y must be matched!
      label y **odd**; $p(y) \leftarrow x$;
      let z be such that $yz \in M$; label z **even**; $p(z) \leftarrow y$; $Q \leftarrow Q \cup \{z\}$
    else if y **even**
      follow p-pointers from x and y to get
       augmenting path (different roots) or odd cycle (same root)

Upon discovering a flower F with blossom B, Edmonds' algorithm constructs a new graph $G' = (V', E')$ and a matching M' in G' as follows: $V' = (V \setminus B) \cup \{b\}$, where b is a new vertex, for every edge $xy \in E$ with $\{x, y\} \cap B = \emptyset$, add xy to E', and if $x \in B$ and $y \notin B$, add by to E'. Put an edge in M' if it comes from an edge in M, and then remove free parallel edges.

Claim 6 *G has an augmenting path with respect to M if and only if G' has an augmenting path with respect to M'. Moreover, given an augmenting path P' for M' in G', we can obtain an augmenting path P for M in G in $O(m + n)$.*

Proof sketch of "only if". Assume there is augmenting path P for G and M, from free vertices x to y. If P does not interesect B, then it is augmenting path for M' in G'. Else, let P_x be the directed path from x to some vertex of B, and P_y the directed path from y to some vertex of B; one of these two paths may consist of only one vertex, if the flower has no stem. If the flower has no stem, the path from P_x and P_y that has more than one vertex is augmenting for M in G'; so we assume from now on the stem is non-empty.

Note that P_x and P_y are vertex disjoint, P_x and P_y each end in a vertex of B using a free edge, and they do not end at the same vertex of B since we could not combine them in an augmenting path.

Call a matched edge e that belongs to both P_x and the stem *cool* iff P_x traverses e downward (on the stem, same way as the path from the root of the flower to the base of the blossom), and the subpath of P_x before e does not intersect the subpath of the stem from e to the base of blossom (this subpath can have only one vertex, the base of the blosom). The path P_y may have its own cool edges, defined as above with P_y instead of P_x.

In a first case, there are no cool edges. One of x, y is not u, the root of the flower; say $x \neq u$. If P_x does not touches any vertex in the stem of the flower, then an augmenting path for G' is given by P_x followed by the stem upwards. If P_x touches the stem, it must use the matched edge in the stem incident to the vertices is touches. As no cool edge exists, then with e being the first edge on P_x and the stem, we get an augmenting path for M' in G' by following P_x up to e and then going up the stem.

In a second case, there exists cool edges, and consider e to be the one closest to the blossom, and switch x with y if needed to have e on P_x (we don't need $x \neq u$ anymore). If P_y does not touch the stem under e, we get an augmenting path for M' in G' by combining the portion on P_x up to e, then the stem down to b (the supervertex), then P_y to y. If P_y touches the stem under e, than its first matched edge on this part of the stem, say e_y, must go upwards on the stem, or else e_y would be cool and lower than e. We get an augmenting path for M' in G' by combining the portion on P_x up to e, then the stem down to e_y, then P_y to y. This finishes the sketch.

Find-Augmenting-Path ($G = (V, E), M$). Returns path P or "No augmenting path"

```
Run S-Trees construction (G, M)
 if one edge xy is found with both endpoints even then
      if x and y in different trees then
                Return P, the path obtained from x, y using p-pointers
      else
            Identify the blossom B, construct G' and M'
            if Find-Augmenting-Path (G', M') returns P' then
                  Construct P from P'; Return P
Return "No augmenting path"
```

It is clear that G' can be constructed from G in $O(|E| + |V|)$. Then the running time for FIND-AUGMENTING-PATH obeys the recurrence $T(n) \geq (n + m) + T(n - 2)$, with the solution $T(n) = O(nm)$. Clever book-keeping can reduce this to $O(m)$, for a Maximum Matching algorithm with complexity $O(mn)$. Best known is $O(m\sqrt{n})$, but I can present this bound only for bipartite graphs.

Edmonds-Gallai decomposition

Theorem 7 *In polynomial time, using Edmonds' Maximum Matching Algorithm, we obtain a set $A \subseteq V$ such that $G \setminus A$ has connected components $B_1, B_2, \ldots, B_k, D_1, D_2, \ldots D_j$ such that:*

- *for each $1 \leq i \leq j$ D_i has a perfect matching,*
- *for each $1 \leq i \leq k$ and each vertex $v \in B_i$, $B_i \setminus \{v\}$ has a perfect matching.*
- *any maximum matching of G matches all the vertices of A to vertices in distinct B_i's; moreover, the matchings above can be quickly found and extended to a maximum matching M of G.*

Note then that

$$|M| = |A| + \sum_{i=1}^{k}(|B_i| - 1)/2 + \sum_{i=1}^{j}|D_i|/2 \tag{1}$$