Discrete Mathematics 2017: Lecture V

Contents

1 Maximum matchings

Definition 1.1 *Given an undirected graph $G = (V, E)$ a* matching *M of G is a subset of the edges E, such that no vertex in V is incident to more than one edge in M.*

Definition 1.2 *A matching M of G is said to be* maximal *if there exists no other matching M' for which $M \subset M'$. A matching M is* maximum *if for any other matching M', $|M| \geq |M'|$. A matching M is said* perfect *if every vertex in V is incident to one edge in M.*

Intuitively, a matching is maximal if it is not possible to add any other edge to the existing set of edges. The *size* of a matching M is the value of $|M|$.

Definition 1.3 *Given a matching M of G, a vertex v is* matched *if it is an endpoint of an edge in M, otherwise we say v is* free.

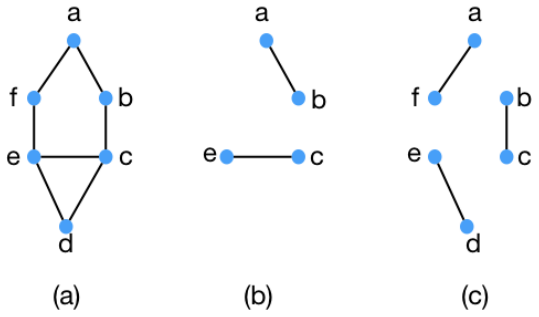

Figure 1: (a) A graph G, (b) a maximal matching of G but not a maximum matching of G, note that vertices f and d are free while the other vertices are matched(c) a maximum matching of G which happens also to be also a perfect matching for G.

Exercise 1.1 *For a graph $G = (V, E)$ on n vertices (i.e. $|V| = n$) what is the size of a perfect matching?* Ans.

Exercise 1.2 *Given a graph $G = (V, E)$ on n vertices write an algorithm that finds a maximal matching in G.* Ans.

Finding a maximum matching in a bipartite graph We show now how to find a maximum matching in a bipartite graph.

Definition 1.4 *Given a matching M of G, a path is* alternating *if its edges alternate between M and $E - M$. An alternating path is said* augmenting *if both endpoints (i.e. the first and the last vertex of the path) are free.*

Figure 2: The path $f - a - b - c - e - d$ is an alternating path that is also augmenting. The vertices f and e are free.

For example consider the graph in Figure1.(a) and the matching in Figure1.(b), the path $f - a - b - c - e - d$ is an alternating path that is also augmenting (see Figure7). Note that an augmenting path has one less edge in M than in $E - M$, hence replacing the edges of M by the edges of $E - M$ we increment the size of the matching by one. The following theorem is due to Berge and is the main result we are going to use.

Theorem 1.1 (Berge's Theorem) *A matching is M is maximum if and only if it has no augmenting path.*

Proof: Clearly if there is an augmenting path then M is not maximum as we can exchange the edges in the augmenting path and obtain a matching M' that has size one more than M.
In the other direction assume there is no augmenting path and suppose on the contrary the matching M is not a maximum matching. Then this means there exists a matching M' that is maximum, with $|M'| > |M|$. Focus only on the subgraph of G, denoted by H, containing only the edges that belong only to M and M' but not to both of them. In other words, $H = M\delta M'$ (recall that $M\delta M' = (M \cup M') - (M \cap M')$). Observe that the degree of each vertex in H is at most 2 and hence H is a collection of cycles and paths. Note also that each cycle in H has an even number of edges, this because each vertex in the cycle must be incident to exactly two edges and neccessary one of them belongs to M and the other to M'. Hence for every cycle, the number of edges that belong to M are equal to the number of edges that belong to M'. However, we said that M' has more edges than M, then there must neccessary be a path where this happens. It is not difficult to see that this is an augmenting path, contradicting our initial supposition that there are no augmenting paths. Thus, we reached a contradiction. This concludes the proof. □

This results gives rise to the following simple algorithm: Find a maximal matching, if there is an augmenting path then exchange the edges in the path and obtain a matching that is of larger size. Continue until there is no more augmenting paths. The main issue now is how to find augmenting paths. The first idea is the

following: Given a maximal matching find a vertex that is free and then using this vertex as a root, perform a DFS. Keep track of the distances of each vertex encountered from the root while running the DFS. We try to alternate between edges on M and edges in $E-M$ in the following way: when reaching a new vertex found is at an odd distance from the root then choose an edge $e \in M$ to continue the DFS, otherwise choose an edge in $E - M$. If we reach a free vertex that is at an odd distance from the root we have found an augmenting path (see Figure7).
This approach does not always work. Indeed, in some cases even when there is an augmenting path, the algorithm may fail to find it, as shown by the next exercise.

Exercise 1.3 *Find a counterexample of a graph where the previous algorithm does not work. Ans.*

In fact as shown intuitively by the exercise above, the problem are the odd cycles. In presence of odd cycles there is a free vertex that is incident to two edges that do not belong to the matching, then only one of them will belong to the maximum matching, hence if we traverse the graph in the wrong way we may miss the augmenting path. The previous algorithm works however for bipartite graphs.
We recall that in a bipartite graph it is possible to partition the vertices in two sets L, R such that every edge $\{u, v\} \in E$, $u \in L$ and $v \in R$.

Algorithm $MaximumMatching((L, R, E))$
Require: A bipartite graph $G = (L \cup R, E)$
Ensure: Returns a maximum matching M of G.

1: $M \leftarrow \emptyset$
2: **while** $\exists$ an augmenting path P **do**
3: $\quad M \leftarrow M \oplus P$;
4: **end while**
5: **return** M;

Figure 3: Algorithm $MaximumMatching(G)$

Observe that at each iteration the size of the matching is increased by 1. Hence, we can have at most $n/2$ iterations. Each iteration starts a DFS that creates a path, thus at most m vertices are visited. Hence, the total time compelxity is $O(nm)$.
This can be improved by augmenting along several augmenting paths simultaneously. The main idea is the following: Start with a matching M, now let U be the set of free vertices. We will construct a forest F_U that has each vertex in U in a component. To do so, we label initially all the vertices of U as *even*. Then we choose a vertex $w \in U$ such that $label(w) = even$ and w is connected to some vertex in M, in other words, there exists and edge $\{w, v\}$ with v in the matching M. Then if v' is the adjacent of v in M we grow F_U as follows: $F_U = F_U \cup \{w, v\} \cup \{v, v'\}$ and we label the new vertices as follows $label(v) = odd$ and $label(v') = even$. We repeat the process by choosing again an even vertex in F_U. See Figure5 that illustrates this process. If it is not possible to grow anymore F_U then if there are edges between two even vertices than we have an augmenting path. Also these vertices must belong to two different components as the graph has no odd cycle.

Algorithm *FindAugmentingPath*$((L, R, E), M)$

Require: A bipartite graph $G = (L \cup R, E)$ and a matching M.

Ensure: Returns an augmenting path P if it exists.

Choose a vertex $v \in L$ such that v is *free*.
Start DFS at v
if current vertex is in L **then**
    follow an edge $e \in M$;
else
    follow an edge $e \notin M$;
end if
if current vertex is in R and is *free* **then**
    an augmenting path P is found;
end if
return P;

Figure 4: Algorithm *FindAugmentingPath*$((L, R, E), M)$

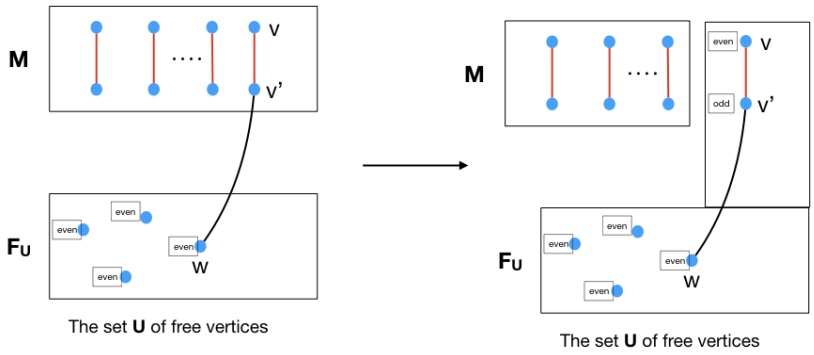

Figure 5: The growing of the forest F_U.

2 Solution of the exercises

Solution 1 *Exercise 1.1 Every vertex should be incident to exactly one edge in the matching, hence the size of a perfect matching is $n/2$. Also note that a perfect matching to exists n must be even.*

Solution 2 *Exercise 1.2*

Solution 3 *Exercise 1.3*

References

[1] J.A. Bondy and U.S.R. Murty, Graph Theory with Applications, North-Holland, 1976. (This book is out of print. You can download their personal copy from the web page: http://book.huihoo.com/pdf/graph-theory-With-applications/pdf/GTWA.pdf)

Algorithm $MaximumMatching2((L, R, E))$

Require: A bipartite graph $G = (L \cup R, E)$

Ensure: Returns a maximum matching M of G.

1: Choose an arbitrary matching M
2: **while** $\exists$ an augmenting path P **do**
3: Grow forest as above and label vertices *even*/*odd*;
4: Find all even - even edges and let $(P_1, \ldots, P_k)$ be the corresponding maximally disjoint set of paths that are augmenting.
5: $M \leftarrow M \oplus P_1 \oplus \ldots P_k$
6: **end while**
7: **return** M;

Figure 6: Algorithm $MaximumMatching2(G)$

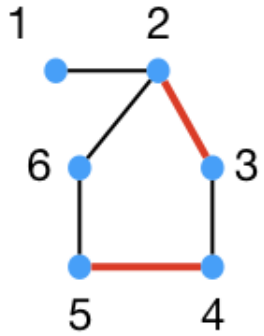

Figure 7: In this example let the maximal matching contain only the edges $\{2, 3\}$ and $\{4, 5\}$, if we start from 6 as the root, if we take the path $6 - 5 - 4 - 3 - 2 - 1$ we find an augmenting path, however if we take the path $6 - 2 - 3 - 4 - 5$ we fail to find one.

[2] R. Diestel, Graph Theory, 4th edition, Springer, 2010. See electronic edition at: http://diestel-graph-theory.com/