

Lecture 13: Matchings

Harvard SEAS - Fall 2024

2024-10-15

1 Announcements

- Midterm on Thursday
- Problem Set 5 distributed tomorrow, due Wed 10/23.
- Embedded EthiCS Module next Tuesday 10/22. You are expected to attend; there will be material on ps6 building on the module.

Recommended Reading: Cormen–Leiserson–Rivest–Stein, Sec. 25.1.

2 Recap

Definition 2.1. For a graph $G = (V, E)$, a *matching* in G is a subset $M \subseteq E$ such that every vertex $v \in V$ is incident to at most one edge in M . Equivalently, no two edges in M share an endpoint.

If a vertex v is incident to an edge in M , we say v is *matched* by M ; otherwise we say it is *unmatched*.

The problem of finding the largest matching in a graph is called Maximum Matching.

Input : A graph $G = (V, E)$

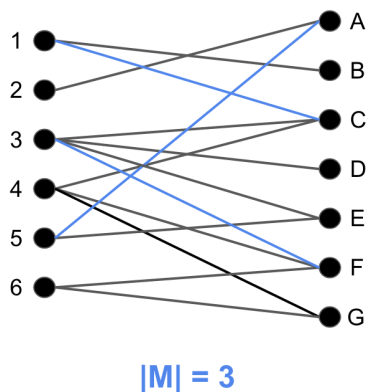
Output : A matching $M \subseteq E$ in G of maximum size

Computational Problem Maximum Matching

Definition 2.2. Let $G = (V, E)$ be a graph, and M be a matching in G . Then:

1. An *alternating walk* W in G with respect to M is a walk $(v_0, v_1, \dots, v_\ell)$ in G such that for every $i = 1, \dots, \ell - 1$, $\{v_{i-1}, v_i\} \in M \Leftrightarrow \{v_i, v_{i+1}\} \in E \setminus M$.
2. An *augmenting path* P in G with respect to M is an alternating walk in which v_0 and v_ℓ are respectively unmatched by M , and in which all of the vertices in the walk are distinct, and $\ell \geq 1$.

Example: Consider the following graph with $M = \{\{1, C\}, \{3, F\}, \{5, A\}\}$.



We check whether the following sequences of vertices constitute an alternating walk or an augmenting path.

Sequence	Alternating Walk	Augmenting Path
2, A, 5		
G, 4		
3, F, 4, C, 3		
1, C, 6		
6, F, 3, C, 1, B		

Theorem 2.3 (Berge's Theorem). *Let $G = (V, E)$ be a graph, and $M \subseteq E$ be a matching. M is not a maximum-size matching in G if and only if G has an augmenting path with respect to M .*

3 A Motivating Application

Kidney Exchange: In the US, there is a nationwide system for arranging kidney transplants, matching patients (who need a kidney) with donors (who are willing to donate a kidney). Each donor can only donate one kidney (they need their other one to survive!) and only to certain patients (due to blood type and HLA type compatibilities). There over 100,000 patients currently on the kidney waiting list in the US, with a little over 25,000 donations happening per year, and patients spending an average of about 3.6 years on the waiting list. Algorithms like what we will cover play a significant role in saving patients through kidney donations.

Q: If our goal is to maximize the number of kidneys donated, how can we model this as a matching problem?

Additional considerations in real-life kidney exchange (to be discussed more in Embedded EthiCS Module next Tue!):

4 Matching vs. Independent Sets

This section was not covered in lecture, and is optional (but recommended) material.

Like IntervalScheduling-Optimization, Maximum Matching can be viewed as a special case of the Independent Set problem we studied last time, i.e. there is a very efficient reduction from Maximum Matching to Independent Set:

Unfortunately, the fastest known algorithm for Independent Set runs in time approximately $O(1.2^n)$. However, as we saw last time for IntervalScheduling-Optimization, special cases of IndependentSet can be solved more quickly. Matching is another example!

5 Maximum Matching Algorithm

Like in a greedy strategy, we will try to grow our matching M on step at a time, building a sequence $M_0 = \emptyset, M_1, M_2, \dots$, with $|M_k| = k$. However, to get M_k from M_{k-1} we will sometimes do more sophisticated operations than just adding an edge. Instead we will rely on Berge's Theorem, which tell us that if our matching is not of maximum size, then there is an augmenting path. We will obtain an algorithm by the following two lemmas:

Lemma 5.1. *Given a graph $G = (V, E)$, a matching M , and an augmenting path P with respect to M , we can construct a matching M' with $|M'| = |M| + 1$ in time $O(n)$.*

Lemma 5.2. *Given a bipartite graph $G = (V, E)$ and a matching M that is not of maximum size, we can find an augmenting path with respect to M in time $O(n + m)$.*

Since a matching can be of size at most $n/2$, repeatedly applying these two lemmas gives us an algorithm that runs in time $(n/2) \cdot (O(n) + O(n + m)) = O(n \cdot (n + m))$. Moreover, by eliminating isolated vertices, in time $O(n)$ we can reduce to the case where $n \leq 2m$, giving us a run time of $O(n) + O(n \cdot (2m + m)) = O(nm)$. Thus we have:

Theorem 5.3. *Maximum Matching can be solved in time $O(mn)$ on bipartite graphs with m edges and n vertices.*

Proof of Lemma 5.1.

□

Example: We repeatedly apply Lemma 5.1 to our example matching given earlier. We first consider the augmenting path $(G, 4)$ and grow our matching accordingly.

Next, we add the augmenting path $(6, F, 3, C, 1, B)$.

Note that $\{3, F\}$ and $\{1, C\}$ are not in M' , since they were part of the original matching M .

In the above example, we gave you the augmenting paths. We still need to show that we can find augmenting paths efficiently (Lemma 5.2). This is the only place we use bipartiteness in the algorithm. The following lemma (whose proof is in the detailed lecture notes) reduces our task to a shortest path problem.

Lemma 5.4. *Let $G = (V_0 \cup V_1, E)$ be bipartite and let M be a matching in G that is not of maximum size. Let U be the vertices that are not matched by M , and $U_0 = V_0 \cap U$ and $U_1 = V_1 \cap U$. Then:*

1. *G has an alternating walk with respect to M that starts in U_0 and ends in U_1 .*
2. *Every shortest alternating walk from U_0 to U_1 is an augmenting path.*

We use this lemma to give the following reduction.

Lemma 5.5. *Finding shortest alternating walks in bipartite graphs reduces to finding shortest paths in directed graphs in time $O(n + m)$.*

Lemma 5.2 follows because Shortest Paths in directed graphs can be found in time $O(n + m)$.

Proof sketch.

□

Example: If we apply this reduction to second-to-last matching (namely $M = \{\{1, C\}, \{3, F\}, \{4, G\}, \{5, A\}\}$) in our example above and we apply BFS to find a shortest path in G' , we explore the vertices of G' as follows:

Putting it all together, we have the following algorithm:

```
1 MaxMatchingAugPaths( $G$ )  
   Input           : A bipartite graph  $G = (V, E)$   
   Output          : A maximum-size matching  $M \subseteq E$   
2 Remove isolated vertices from  $G$ ;  
3 Let  $V_0, V_1$  be the bipartition (i.e. 2-coloring) of  $V$ ;  
4  $M = \emptyset$ ;  
5 repeat  
6   | Let  $U$  be the vertices unmatched by  $M$ ,  $U_0 = V_0 \cap U$ ,  $U_1 = V_1 \cap U$ ;  
7   | Use BFS to find a shortest alternating walk  $P$  that starts in  $U_0$  and ends in  $U_1$ ;  
8   | if  $P \neq \perp$  then augment  $M$  using  $P$  via Lemma 5.1;  
9 until  $P = \perp$ ;  
10 return  $M$ 
```