Math 3322: Graph Theory[1] Mikhail Lavrov

Lecture 21: Bipartite Matchings

October 28, 2021 *Kennesaw State University*

1 The bipartite matching problem

1.1 The setup

Imagine that you're in charge of scheduling classes for the KSU math department. You have 100 sections to schedule, and if you add up the number of classes each professor is willing to teach, that also gets you to 100, so everything seems fine.

Unfortunately, there are other constraints. I would be fine teaching graph theory, but bad at teaching differential equations. My friend in the office next door would be fine teaching differential equations, but bad at teaching graph theory. Neither of us would want to teach a class in a big lecture hall on the Kennesaw campus, because our offices are here on the Marietta campus.

You put all this together into sets of possibilities: maybe "Misha's first class" could be any of

$$\{\text{graph theory}, \text{section 1 of probability}, \text{section 2 of probability}, \text{intro to proofs}, \dots\},$$

and the same for "Misha's second class", and so on for all 100 classes that will be taught. Now your goal is to pick an item from each set so that you end up picking every section that needs to be taught.

1.2 Course scheduling, as a graph

The graph representation of this data is a bipartite graph where the people and their time slots (Misha's first class, Misha's second class, ...) are vertices on one side, while the sections (graph theory, section 1 of probability, ...) are vertices on the other side. There is an edge between "Misha's first class", and every section that Misha's first class could be.

A solution to the class scheduling problem is a subset M of edges: if you say that the first class I'm teaching is graph theory, then that corresponds to putting the edge $\{\text{Misha's first class}, \text{graph theory}\}$ in M. There are two constraints on what M can be:

- Two edges we put in M can't share the same vertex on the first side: "Misha's first class" can only be one thing.
- Two edges we put in M can't share the same vertex on the second side: "graph theory" is only taught once. (If we want multiple sections of a class, we have already represented that by creating multiple vertices for the different sections.)

[1] This document comes from the Math 3322 course webpage: http://facultyweb.kennesaw.edu/mlavrov/courses/3322-fall-2021.php

1.3 The matching problem

In general, a **matching** in a graph G is a set of edges M such that no two edges in M have the same endpoint. If you like, you can also think of this as a subgraph of G in which no vertex has degree more than 1, but we'll stick to the set-of-edges version for the most part; assume that's what these notes mean unless they say otherwise.

We can think about matchings in all kinds of graphs. But we'll start with thinking about matchings in bipartite graphs for two reasons. First, lots of applications (like the course scheduling application) will naturally give us a bipartite graph to look for a matching in. Second, the theory turns out to be simpler for bipartite graphs.

We say that a vertex of G is **covered** by M if it's the endpoint of some edge of M, and **uncovered**[2] otherwise. (In the subgraph interpretation: a vertex is covered by M if it has degree 1 in M, and uncovered if it has degree 0.)

At this point, we can be optimistic or realistic. An optimist asks "is the cup full?" but a realist asks "how much water is in the cup?" Your textbook is optimistic, but we will be realistic in class.

What am I talking about? Well, there's two questions you can ask about matchings in a graph:

1. You can ask "Is there a perfect matching?" A matching is **perfect** if it covers every single vertex.
2. You can ask "How many edges are in a maximum matching?" A **maximum** matching is one with as many edges as possible.

The answer to the second question will also answer the first: a bipartite graph with n vertices on one side and n vertices on the other side has a perfect matching if and only if the maximum matching(s) in that graph have n edges. That's why we'll ask the second question.

1.4 The maximum/maximal distinction

In combinatorial optimization problems like the matching problem, we are trying to find a set S that's as large as possible, subject to some condition. It's common to say that a candidate solution S is:

- **maximum** if it really is as large as you can get; there are no better solutions than S.
- **maximal** if you cannot add anything to S without violating the restriction; there might be better solutions, but you cannot get to them without removing something from S first.

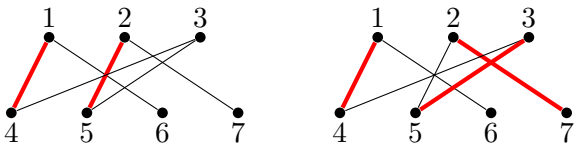
Every maximum solution is also maximal, but the reverse might not hold.

In the same way, "minimum" and "minimal" get used when we're trying to pick a set that's as small as possible, subject to some condition.

I will not use the terms "maximal" and "minimal" without bringing up the difference specifically, because the difference is very subtle and easy to miss. However, however you call it, it's important to realize that there **is** a difference, and in particular there is a difference in the matching problem.

[2] You will see "saturated" and "unsaturated", or "matched" and "unmatched", in other sources.

Take the following two matchings in the same graph:



$M_1 = \{14, 25\}$ $M_2 = \{14, 27, 35\}$

We might get matching $M_1 = \{14, 25\}$ if we just pick the first option we find at each step. Vertices $3, 6, 7$ are uncovered, but don't have uncovered neighbors; we can't add any edges to M_1.

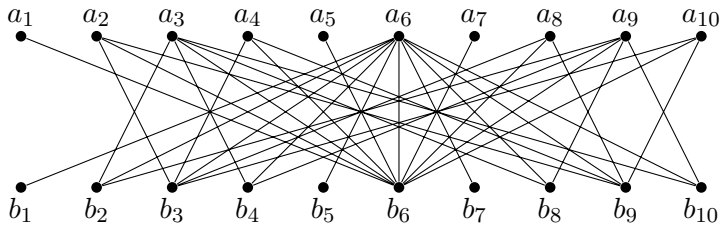
However, if we remove edge 25 from M_1, then instead we can add edges 27 and 35, getting the larger matching M_2. This matching is one possible maximum matching in the graph: there's only one uncovered vertex, so we can't do better. (To have 4 edges in a matching, we'd need at least 8 vertices.)

This tells us that the bipartite matching problem is not straightforward. By contrast, consider the problem of finding a spanning tree in a graph. There, if you try to find the smallest set of edges that connect all the vertices, you'll never get stuck in a suboptimal solution; you'll always end up at $n - 1$ edges for n vertices. That was a much easier problem.

2 Vertex covers

2.1 The multiples-of-six graph

Consider the bipartite graph with vertices $a_1, a_2, \ldots, a_{10}$ on one side, vertices $b_1, b_2, \ldots, b_{10}$ on the other side, and edges defined by the following rule: a_i and b_j are adjacent exactly when the product ij is divisible by 6. Here is a diagram:



What is the largest matching in this graph?

You will not have too much trouble finding a matching with 6 edges. For example, you can take the matching made by three "X" shapes in the graph: $M = \{a_2b_3, a_3b_2, a_5b_6, a_6b_5, a_8b_9, a_9b_8\}$. It seems difficult to find a better matching, but how can we know that it's impossible?

If we try to find other matchings, we encounter a bottleneck. To get a product divisible by 6, we need a multiple of 2 and a multiple of 3, but the multiples of 3 are much harder to come by. Every single edge must include one of the multiples of 3: a_3, a_6, a_9, b_3, b_6, or b_9. However, we can't use any of these vertices twice in a matching. This means we can have at most 6 edges in a matching: one for each of these vertices.

2.2 Vertex covers

This argument generalizes.

Let a **vertex cover** of G be a set of vertices U such that every edge of G has at least one endpoint in U (possibly both). For example, in the graph above, $\{a_3, a_6, a_9, b_3, b_6, b_9\}$ is a vertex cover. It is not the only one:

- The set of *all* the vertices in a graph is always a vertex cover.
- In a bipartite graph like the one above, the set of all vertices on one side is a vertex cover: for example, $\{a_1, a_2, a_3, \ldots, a_{10}\}$ is a vertex cover.
- Every edge needs a multiple of 2, so $\{a_2, a_4, a_6, a_8, a_{10}, b_2, b_4, b_6, b_8, b_{10}\}$ is a vertex cover.

However, $U = \{a_3, a_6, a_9, b_3, b_6, b_9\}$ is the smallest vertex cover.

Vertex covers are important because of the following claim:

Claim 2.1. *If M is a matching and U is a vertex cover in the same graph, then $|M| \leq |U|$.*

Proof. In this proof, we will think of M as a subgraph of G: it will have all the vertices of G, but only the edges that are edges of M. Consider the sum

$$\sum_{u \in U} \deg_M(u).$$

Each term $\deg_M(u)$ of this sum is at most 1, because no vertex can have degree more than 1 in M: that's what being a matching means. So the value of sum is at most $|U|$.

Each edge of M has at least one endpoint in U, because that's what being a vertex cover means, contributing at least 1 to $\deg_M$ of that endpoint. So the value of the sum is at least $|M|$.

Therefore $|M| \leq |U|$. □

Therefore vertex covers are a way to get upper bounds on the maximum size of a matching in a graph. For example, a simple observation: if G is a bipartite graph with bipartition (A, B), then the maximum number of edges in a matching is at most $\min\{|A|, |B|\}$. That's because both A and B are vertex covers.

We can try to find a **minimum vertex cover** (one with as few vertices as possible) to get the best upper bound. However, this upper bound could still be really bad. For example, the largest matching in the complete graph K_{100} has only 50 edges: at that point, you've used up all the vertices! However, the smallest vertex cover in K_{100} has 99 vertices: you have to use all but one of the vertices, because if you skip two vertices, you don't cover the edge between them.

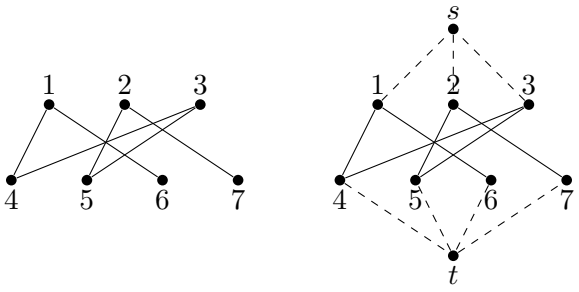
3 König's theorem

The idea of vertex covers, finally, is why the bipartite matching problem is special. Specifically, it's special because of the following theorem:

Theorem 3.1 (König)**.** *In a bipartite graph G, the number of edges in a largest matching is equal to the number of vertices in a smallest vertex cover.*

Proof. The matching-to-vertex-cover relationship is a lot like the relationship between $s-t$ cuts and internally disjoint $s-t$ paths in a graph we get from Menger's theorem! And, indeed, we can use Menger's theorem to prove this theorem, saving us a lot of work.

Take a bipartite graph G with bipartition (A, B) and modify it: add a vertex s adjacent to all of A, and a vertex t adjacent to all of B. Call the result H. For example, if G is the graph below on the left, then H is the graph on the right:



We will show that vertex covers in G correspond to $s-t$ cuts in H, and matchings in G correspond to sets of internally disjoint $s-t$ paths in H. Then, Menger's theorem finishes off the proof.

Vertex covers in G and $s-t$ cuts in H are one and the same:

- If U is a vertex cover in G, then $H-U$ cannot have an $s-t$ path: every $s-t$ path must at some point take an edge from A to B, but every such edge has an endpoint in U, and so it is deleted in $H-U$: U must be an $s-t$ cut.
- Conversely, if U is an $s-t$ cut in H, and ab (with $a \in A, b \in B$) is any edge of G, then (s, a, b, t) is an $s-t$ path in H, so it must be destroyed in $H-U$. Therefore U must contain at least one of a or b, which is exactly what it means to be a vertex cover.

The correspondence between matchings and internally disjoint $s-t$ paths is slightly less well-behaved, because paths can do weird things and don't have to restrict themselves to a single edge of G. In the example above, $(s, 1, 4, 3, 5, t)$ is one such "weird" path. However, every $s-t$ path must contain *at least* one edge from G (to get from A to B).

If we have a collection of k internally disjoint $s-t$ paths, then pick one such edge from every $s-t$ path. We get k edges in G, which must form a matching: they won't share any endpoints, because the paths did not share vertices other than s and t.

Now it's time to finish the proof. Let U be a minimum vertex cover in G. Because vertex covers in G and $s-t$ cuts in H are one and the same, $|U| = \kappa(s,t)$. By Menger's theorem, there is a collection of $|U|$ internally disjoint $s-t$ paths, and we can use it to get a matching M in G of size $|U|$.

By Claim 2.1, every matching in G has size at most $|U|$, so the matching M we found is a maximum matching, proving the theorem. □