Math 482: Linear Programming[1] Mikhail Lavrov

Lecture 22: Bipartite Matchings and Vertex Covers

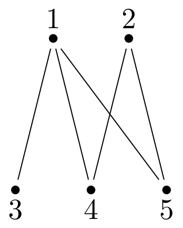
March 25, 2020 *University of Illinois at Urbana-Champaign*

1 Bipartite matching and total unimodularity

Last time, we wrote down a linear program for the bipartite matching problem. Given a bipartite graph (X, Y, E), the linear program is

$$\begin{array}{lll} \underset{\mathbf{x}\in\mathbb{R}^{|E|}}{\text{maximize}} & \displaystyle\sum_{i,j:(i,j)\in E} x_{ij} & \\ \text{subject to} & \displaystyle\sum_{j:(i,j)\in E} x_{ij} \le 1 & \text{for each } i \in X \\ & \displaystyle\sum_{i:(i,j)\in E} x_{ij} \le 1 & \text{for each } j \in Y \\ & \mathbf{x} \ge \mathbf{0} & \end{array}$$

Define the incidence matrix of the graph be the $(|X| + |Y|) \times |E|$ matrix A whose rows correspond to vertices (of either type), whose columns correspond to edges, and where an entry $A_{v,e}$ is 1 if vertex v is an endpoint of edge e, and 0 otherwise. (We sometimes say that "v is *incident* to e".) In terms of this matrix A, the constraints in the linear program for bipartite matching can be written as $A\mathbf{x} \le \mathbf{1}$. Here is an example:



$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

In order to be sure that this linear program will give us meaningful answers, we want to check that the constraint matrix is totally unimodular: that every square submatrix of A has determinant -1, 0, or 1. Then we know, by the theorem that we proved last time, that every basic feasible solution (and therefore the optimal solution we'll find by the simplex method) will have integer values for every variable.

Theorem 1.1. *The incidence matrix of a bipartite graph is always totally unimodular.*

Proof. We prove that $k \times k$ submatrices of the incidence matrix will always have determinant -1, 0, or 1 by induction on k.

[1]This document comes from the Math 482 course webpage: https://faculty.math.illinois.edu/~mlavrov/courses/482-spring-2020.html

When $k = 1$, we'll definitely always get either 0 or 1, because a 1×1 determinant is just the value of the only entry of the 1×1 matrix, and all 1×1 submatrices are either $\begin{bmatrix}0\end{bmatrix}$ or $\begin{bmatrix}1\end{bmatrix}$.

For larger matrices, we consider three possibilities. Here they are, in increasing order of complexity:

First, the submatrix could have a column with only zeroes in it. For example, for the graph above, the submatrix consisting of rows $2, 3, 4$ and the first three columns (corresponding to the edges $(1, 3)$, $(1, 4)$, and $(1, 5)$) is

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

In this case, the determinant always 0. (One way to see this is to use expansion by minors along the all-zero column.)

Second, the submatrix could have a column with only a single 1 in it. For example, from the graph above, the submatrix consisting of the first two rows and the first two columns is

$$\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$$

In this case, expansion by minors along that column reduces the problem of a $k \times k$ determinant to a $(k-1) \times (k-1)$ determinant, up to a ± 1 factor depending on the position of the lonely 1. By induction, the $(k-1) \times (k-1)$ determinant is -1, 0, or 1, and therefore so is the $k \times k$ determinant.

Finally, all columns of the submatrix could have two 1s in them. (But no more than two, because each column of the original matrix corresponds to an edge (i, j), and only has two 1s: in the i^{th} and in the j^{th} positions.) This is true of the entire 5×5 matrix in the example.

In this case, let B be the submatrix. Let $\mathbf{u} \in \mathbb{R}^k$ be the vector such that $u_i = 1$ if the i^{th} row of B corresponds to a vertex in X, and $u_i = -1$ otherwise. Then multiplying $\mathbf{u}^\mathsf{T} B$ will give a vector whose components are $u_i + u_j$ for every edge (i, j) whose column ended up in B. But every edge (i, j) has one endpoint in X and one in Y; therefore $u_i + u_j = 1 + (-1) = 0$, and $\mathbf{u}^\mathsf{T} B = \mathbf{0}^\mathsf{T}$. This means the rows of B are linearly dependent, and therefore $\det(B) = 0$.

We get -1, 0, or 1 in all three cases, and by induction, this happens for submatrices of all sizes. $\square$

2 Vertex covers

When we get a linear program from thinking about some general problem, it is always a good idea to think about the dual of that linear program, and see if it also has an interpretation.

2.1 Duals of combinatorial problems

As a brief aside: taking duals of linear programs can be trickier in this section of the class, where our linear programs are so complicated.

Remember that we have a dual variable for each primal constraint, and a dual constraint for each primal variable. Keep this pairing of variables and constraints in mind at all times. The following rule tells you what the dual constraints are:

- If x_i is a primal variable and u_j is a dual variable, the coefficient of u_j in the i^{th} dual constraint (the one paired with x_i) is equal to the coefficient of x_i in the j^{th} primal constraint (the one paired with u_j).
- In particular: variable u_j appears in the i^{th} dual constraint if and only if variable x_i appeared in the j^{th} primal constraint.

2.2 The dual of the bipartite matching problem

In the case of the bipartite matching linear program, the dual is a minimization problem with variables $u_1, u_2, \dots, u_{n+m}$: one for each vertex of X or Y. We get a constraint for each edge of the graph, because in the primal program, we had a variable for each edge. Putting everything together, we get

$$\begin{array}{lll} \underset{\mathbf{u}\in\mathbb{R}^{n+m}}{\text{minimize}} & \displaystyle\sum_{i\in X\cup Y} u_i & \\ \text{subject to} & u_i + u_j \ge 1 & \text{for each } (i,j)\in E \\ & \mathbf{u} \ge \mathbf{0} & \end{array}$$

Let's interpret the vector $\mathbf{u}$ as describing a set of vertices S, with $i \in S$ when $u_i = 1$, and $i \notin S$ when $u_i = 0$. Then the objective function, which adds up all the variables u_i, is just $|S|$. The constraints $u_i + u_j \ge 1$ for each edge (i, j) ask that for every edge, one of its endpoints should be included in S.

So the combinatorial interpretation of this dual is that we're looking for the smallest set of vertices possible that "cover" every edge (they include at least one endpoint of every edge). Such a set is called a *vertex cover.*

Even without thinking about linear programs, we can observe a dual relationship between the bipartite matching problem and the vertex cover problem: the size of any matching is at most the number of vertices in any vertex cover. Here's why: suppose that S is a vertex cover, and M is a matching. Then each edge of M must include at least one vertex of S (because every edge includes at least one vertex of S) and no two edges of M can use the same vertex of S (because no two edges of M share an endpoint). So there can be no more edges in M than there are vertices in S.

This is a sort of "weak duality" statement that we can deduce just from combinatorics. However, more is true:

Theorem 2.1. *In any bipartite graph, the number of edges in a maximum matching is equal to the number of vertices in a minimum vertex cover.*

Proof. This follows by putting together two linear programming results: strong duality, and total unimodularity.

Because the incidence matrix is totally unimodular, we know that the combinatorial problems (a maximum matching, a minimum vertex cover) are exactly equivalent to the optimal solutions to the corresponding linear programs. Even though we only checked total unimodularity for the primal program, it follows for free in the dual, because the dual's constraint matrix is the transpose of the primal's constraint matrix.

By strong duality, the optimal objective values of the primal and the dual are equal. □

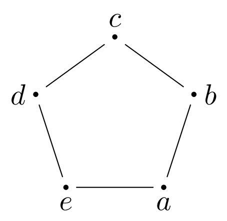
3 Non-bipartite graphs

To prevent you from getting too complacent, here is what happens if a graph is *not* bipartite. (We won't spend too much time thinking about this case, precisely because it's not easily approached using linear programming.)

A general graph is simply a pair (V, E) where V is a set of vertices and E is a set of edges: pairs $\{v, w\} \subseteq V$. We just forget about sorting the vertices into two types, so we lose the "bipartite" qualification; a matching in the graph is still a set of edges sharing no endpoints, and a vertex cover is still a set of vertices covering all the edges.

Our proof that the incidence matrix of a bipartite graph is totally unimodular really did use the bipartiteness: in the last step, the vector $\mathbf{u}$ we constructed would not be possible to find if the graph were not bipartite. So we might be worried that not everything will continue to work for graphs that aren't bipartite.

In fact, there are some simple examples where things *do not* continue to work well. For example, take the graph below:



In this graph,

- The maximum matching contains 2 edges. For example, we could take edges $\{a, b\}$ and $\{c, d\}$. Three edges would have to share at least one endpoint, because there are not 6 vertices in the graph.
- The linear program for the maximum matching has objective value $2.5 > 2$. This is achieved by setting $x_{ab} = x_{bc} = x_{cd} = x_{de} = x_{ae} = \frac{1}{2}$. The constraints are that at every vertex, the variables for the edges meeting there add up to at most 1, and this is true here.
- The linear program for the minimum vertex cover also has objective value 2.5 (which we expected from strong duality). This is achieved by setting $u_a = u_b = u_c = u_d = u_e = \frac{1}{2}$. The constraints are that for every edge, the variables for its endpoints add up to at least 1, and this is true here.
- The minimum vertex cover contains 3 vertices. (Two vertices could cover at most 4 of the 5 edges, since every vertex covers at most 2.) One such vertex cover is $\{a, c, d\}$.

Just as before, the matching problem and the vertex cover problem are weakly dual in graphs: a minimum vertex cover gives an upper bound on the size of a maximum matching. But as this example shows, there is no equivalent of strong duality: the answers to the two problems may be different, when the graph is not bipartite.