Math 482: Linear Programming[1] Mikhail Lavrov

Lecture 21: The Bipartite Matching Problem

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

1 Bipartite matching

1.1 Definitions

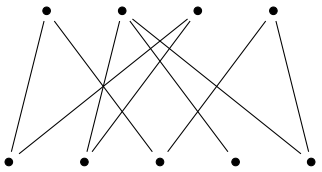
Consider the following example problems:

- You have n employees and m jobs for them. Each employee is qualified to do some, but not all of the jobs. You want to assign employees to jobs.
- A taxi company has a fleet of m taxis; n people request a taxi. Each taxi can only be used for some of the requests, based on proximity. You want to assign a taxi to pick up each person.
- A cloud computing service must distribute n tasks to m servers. Each task can only be done on a subset of the servers, depending on memory available, processor speed, and maybe weird hardware requirements.

All of these are special cases of the bipartite matching problem. In general, to abstract away the details of taxis, jobs, or servers, we call each object in our problem a *vertex*. Vertices come in two types (that's what makes the problem "bipartite").

There is a relation between some vertices of the first type and some vertices of the second type: employees are qualified for jobs, taxis are close to customers, computing tasks can be done on servers. Whenever this relation holds—whatever it is—we connect the two vertices by an *edge*. We say that two vertices are *adjacent* or *neighbors* if there is an edge between them; the two vertices an edge joins are called its *endpoints*.

The abstract structure we consider is called a *bipartite graph*. Formally, a bipartite graph is a triple (X, Y, E) where X and Y are sets of vertices (of two types) and E is a set of edges: E consists of some of the ordered pairs (v, w) where $v \in X$ and $w \in Y$. We often represent these graphs by diagrams such as the one below: vertices are represented by points, and edges by lines joining the points.



A *matching* M in a bipartite graph is a subset of the edges that does not include any vertex (in X or in Y) more than once. Our goal in a bipartite matching problem is to find a matching which is as large as possible.

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

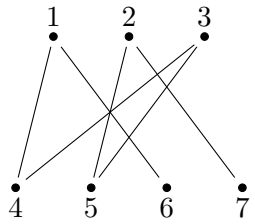
1.2 The bipartite matching linear program

We can describe the bipartite matching problem by a linear program. Let's label the vertices by numbers: X will have vertices $\{1, 2, \ldots, n\}$, and Y will have vertices $\{n+1, n+2, \ldots, n+m\}$

Our linear program will consist of a nonnegative variable x_{ij} for each edge $(i, j) \in E$. We will interpret $x_{ij} = 1$ as "edge (i, j) is part of M" and $x_{ij} = 0$ as "edge (i, j) is not part of M". Then the size of M is just the sum of all the variables.

For each vertex, whether it's in X or in Y, we have a constraint: at most one edge with that vertex as an endpoint can be in M. So if we sum over the variables x_{ij} for all such edges, we should get at most 1. (In particular, each variable should be at most 1.)

Putting this together, we have a linear program. For example, suppose we have the following bipartite graph:



The linear program we get is

$$\begin{aligned}
\text{maximize} \quad & x_{14} + x_{16} + x_{25} + x_{27} + x_{34} + x_{35} \\
\text{subject to} \quad & x_{14} + x_{16} \leq 1 \\
& x_{25} + x_{27} \leq 1 \\
& x_{34} + x_{35} \leq 1 \\
& x_{14} + x_{34} \leq 1 \\
& x_{25} + x_{35} \leq 1 \\
& x_{16} \leq 1 \\
& x_{27} \leq 1 \\
& x_{14}, x_{16}, x_{25}, x_{27}, x_{34}, x_{35} \geq 0
\end{aligned}$$

2 Totally unimodular matrices

2.1 Integer and fractional solutions

There is a potential problem with the linear program we wrote down. What if the optimal solution to it satisfies the constraint $x_{14} + x_{16} \leq 1$ by setting $x_{14} = x_{16} = \frac{1}{2}$? This does not make any sense as a matching: an edge can't be "halfway in" the set M, we either use it or we don't. It especially does not make any sense in applications: a taxi can't split into two half-taxis that drive in different directions to pick up different passengers.

What we want to do is to add a constraint to our linear program saying that every variable is an integer: either 0 or 1, and not $\frac{1}{2}$. Unfortunately, in general, *integer linear programs* with this constraint are very hard to solve: much harder than ordinary linear programs.

We will return to integer programming later on in this course. However, it turns out that the linear program for the bipartite matching problem has an almost magical property: it will always have an optimal solution with integer values for all the variables. So for now, we don't have to worry about integers and fractions: things will just work out for us.

The reason why this happens has to do with special kinds of matrices called *totally unimodular matrices*.

2.2 Matrices with integer inverses

Recall that a basic solution to the system $A\mathbf{x} = \mathbf{b}$ for a basis $\mathcal{B}$ is given by $\mathbf{x}_\mathcal{B} = A_\mathcal{B}^{-1}\mathbf{b}$ and $\mathbf{x}_\mathcal{N} = \mathbf{0}$. One way we can be certain that $\mathbf{x}_\mathcal{B}$ will only have integer entries in such a basic solution is if both $\mathbf{b}$ and $A_\mathcal{B}^{-1}$ have only integer entries.

For $\mathbf{b}$, that's something we can look for in the original linear program. How can we tell if $A_\mathcal{B}^{-1}$ will have integer entries?

Lemma 2.1. *A square matrix M with integer entries has an inverse M^{-1} also with integer entries if and only if* $\det(M) = \pm 1$.

Proof. On one hand, we have $\det(M)\det(M^{-1}) = \det(I) = 1$. If M and M^{-1} are both integer matrices, then they must have integer determinants. The only ways two integers can multiply together to get 1 are $1 \cdot 1 = 1$ and $-1 \cdot -1 = 1$.

On the other hand, there is a formula for M^{-1} using adjugate matrices. In the 2×2 case, it's

$$M^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{\det(M)} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}.$$

For larger matrices, the formula is $\frac{1}{\det(M)}$ times another matrix whose entries are themselves determinants of some submatrices of M. The exact formula doesn't matter too much. What matters is that the formula exists, and the denominator in that formula is $\det(M)$. So if $\det(M) = \pm 1$, the formula gives an integer answer. □

2.3 Totally unimodular matrices

In the case of a linear program, the condition is a bit more complicated to check, because we don't know which matrix we'll be inverting in advance. So we can ask for the following:

Theorem 2.2. *Given a linear program with feasible region $\{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$, all the basic feasible solutions will definitely be integers if $\mathbf{b}$ is an integer, and every square submatrix of A (formed by taking all the entries in any k rows and any k columns of A) has determinant -1, 0, or 1.*

Such a matrix A is called *totally unimodular*.

Proof. When we find the basic feasible solutions, we first add slack variables, turning $A\mathbf{x} \leq \mathbf{b}$ into $A\mathbf{x} + I\mathbf{s} = \mathbf{b}$. Given a basis $\mathcal{B}$, the basic solution is given by $A_\mathcal{B}^{-1}\mathbf{b}$, but because some of the slack variables can be basic, $A_\mathcal{B}$ is an $n \times n$ submatrix of the extended matrix $\begin{bmatrix} A & I \end{bmatrix}$.

When $A_{\mathcal{B}}$ has k columns from A and $n-k$ columns from I, the columns from I can be used to simplify $\det(A_{\mathcal{B}})$ to the determinant of a $k \times k$ matrix. For example:

$$\det \begin{bmatrix} 1 & 5 & 0 & 0 \\ 2 & 6 & 1 & 0 \\ 3 & 7 & 0 & 0 \\ 4 & 8 & 0 & 1 \end{bmatrix} = -\det \begin{bmatrix} 1 & 5 & 0 \\ 3 & 7 & 0 \\ 4 & 8 & 1 \end{bmatrix} = -\det \begin{bmatrix} 1 & 5 \\ 3 & 7 \end{bmatrix}$$

This is done by expanding by minors along one of the $n-k$ columns of $A_{\mathcal{B}}$ coming from I. Up to a factor of ± 1, the determinant of $A_{\mathcal{B}}$ will be equal to the determinant of the $k \times k$ submatrix where we take the k columns coming from A, and the rows which did not contain any of the 1's from the other $n-k$ columns.

In a totally unimodular matrix, all such $k \times k$ determinants will be -1, 0, or 1. Therefore the determinant $\det(A_{\mathcal{B}})$ will always be -1, 0, or 1.

If the determinant is 0, then $A_{\mathcal{B}}$ is not invertible. We don't need to worry about this case: it happens when $\mathcal{B}$ is not a valid basis, and does not result in a basic feasible solution at all.

If the determinant is -1 or 1, then by Lemma 2.1, $A_{\mathcal{B}}^{-1}$ will have integer entries, and therefore the basic solution $\mathbf{x}_{\mathcal{B}} = A_{\mathcal{B}}^{-1}\mathbf{b}$, $\mathbf{x}_{\mathcal{N}} = \mathbf{0}$ will have integer entries. This is exactly what we wanted. □

In the next lecture, we will show that the matrix for the bipartite matching linear program is totally unimodular. This will reassure us that solving the linear program will actually work to find a bipartite matching.