Math 482: Linear Programming[1] Mikhail Lavrov

Lecture 28: Minimum-Cost Flow

April 10, 2020 *University of Illinois at Urbana-Champaign*

1 The minimum-cost flow problem

The minimum-cost flow problem is similar to one we looked at in the previous lecture. We still have a network (N, A), and every node $k \in N$ has an associated demand d_k. We still want to find a flow $\mathbf{x} \geq \mathbf{0}$ that assigns a value x_{ij} to every arc $(i, j) \in A$, and these arcs must satisfy

$$\sum_{i:(i,k)\in A} x_{ik} - \sum_{j:(k,j)\in A} x_{kj} = d_k.$$

However, instead of a *capacity* c_{ij} on every arc (i, j), we put a *cost* c_{ij} on every arc (i, j). (Unfortunately, these start with the same letter.) Our goal is to find a feasible flow (one that has $\mathbf{x} \geq 0$ and satisfies all the demands) that minimizes the total cost

$$\sum_{(i,j)\in A} c_{ij}x_{ij}.$$

We assume our network is connected (there aren't two or more pieces with no arcs between them). If the network weren't connected, we could solve each piece as a separate subproblem.

2 The simplex method for minimum-cost flow

The minimum-cost flow is actually solved reasonably well by the simplex method, provided we do it right. We don't want to write down a huge tableau for the problem and apply pivot operations to it.

Instead, we can represent all the information we need for the simplex method in the form of a diagram. Let's go through all the pieces of the simplex method and see what they look like in the case of minimum-cost flow.

2.1 Basic solutions and spanning trees

Our constraints look like $F\mathbf{x} = \mathbf{d}$ for some matrix F with $|N|$ rows and $|A|$ columns; the column corresponding to variable x_{ij} has a 1 in row j and a -1 in row i. (The vector $\mathbf{d}$ is our vector of demands.)

Normally, a basic solution would look like picking a basis $\mathcal{B}$ and setting $\mathbf{x}_\mathcal{B} = F_\mathcal{B}^{-1}\mathbf{d}$. There's one obstacle to this here: our equations are not linearly independent! If we add all $|N|$ equations

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

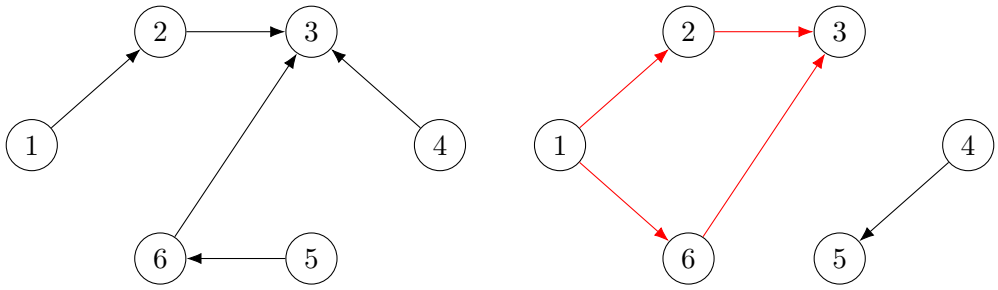
together, then the left-hand sides will cancel (every variable appears once with a positive sign and once with a negative sign) and we'll get

$$0 = \sum_{k \in N} d_k.$$

This tells us that all the demands had better add up to 0: otherwise, there are no feasible solutions. But if they do, then one of our equations is redundant. (When our network is connected, this is the only redundancy.)

We could fix this by deleting any one of the equations. Instead, we'll just keep this in mind and notice that we actually need to pick $|N| - 1$ variables to be our basic variables. (You can think of this as choosing $|N| - 1$ arcs to be "basic arcs".)

Not all choices of $|N| - 1$ arcs will work. (Even if F had full row rank, not all matrices $F_{\mathcal{B}}$ would necessarily be invertible.) We want to choose $|N| - 1$ arcs that do not create any *cycles*. A cycle is a subset of the arcs that form a closed loop, ignoring orientation. In the diagram below, the choice of arcs in the left figure is valid; the choice of arcs in the right figure contains a cycle, shown in red.



Why are cycles bad? They create redundancy. A basis $\mathcal{B}$ should have the property that for any valid $\mathbf{d}$, the equation $F_{\mathcal{B}}\mathbf{x}_{\mathcal{B}} = \mathbf{d}$ always has a *unique* solution (though maybe not one satisfying $\mathbf{x} \geq \mathbf{0}$). We're using the minimum number of variables possible; if the solution is sometimes not unique, that means that in other cases, it doesn't exist at all.

In the diagram on the left, when solutions exist, they're not unique: if we replace $(x_{12}, x_{23}, x_{63}, x_{16})$ by $(x_{12} + \delta, x_{23} + \delta, x_{63} - \delta, x_{16} - \delta)$, then the excess $\Delta_k(\mathbf{x})$ is unchanged at each node, so we get infinitely many solutions. In general, whenever we have a solution $\mathbf{x}$, we can take a cycle, and (going around the cycle) add δ to every arc we see going forward and subtract δ from every arc we see going backward. This will produce a different solution.

A choice of $|N| - 1$ arcs in the network with no cycles is called a *spanning tree*, and these exactly correspond to basic solutions.

For any spanning tree T and any demand vector $\mathbf{d}$ with sum 0, we can pick x_{ij} for every arc $(i, j) \in T$ so that (setting all other $x_{ij} = 0$) the constraints $F\mathbf{x} = \mathbf{d}$ are satisfied. Here's how:

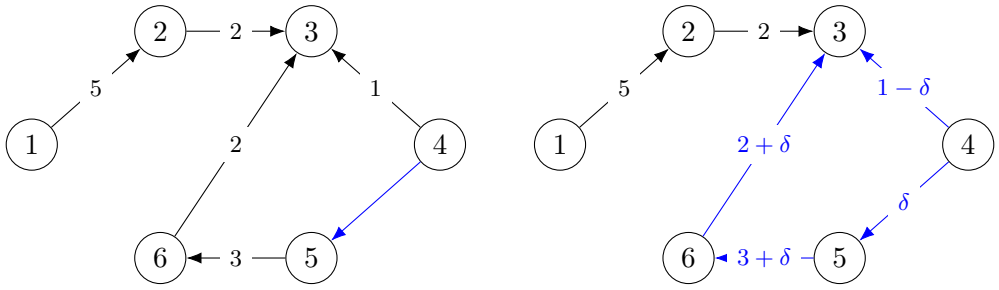
1. Pick a node k such that only one arc of T starting or ending at k has a flow we haven't determined. (At the beginning, pick a node k with only one arc of T starting or ending at k.)
2. Use the demand equation at node k to determine that remaining flow.
3. Repeat steps 1–2 a total of $|N| - 1$ times until all basic variables are known.

This procedure does not necessarily produce a feasible flow, since we might not get $\mathbf{x} \geq \mathbf{0}$. A flow that satisfies $F\mathbf{x} = \mathbf{d}$ but not necessarily $\mathbf{x} \geq \mathbf{0}$ is called a *balanced flow*. This corresponds to the distinction between basic feasible solutions and arbitrary basic solutions.

2.2 Pivoting steps

A pivoting step in the simplex method involves picking a nonbasic variable and bringing it into the basis, replacing some basic variable. In the min-cost flow problem, we pick an arc outside our spanning tree T and add it to the tree, deleting an existing arc to make room.

The key is that any arc we add to T creates a cycle, and we now have the flexibility to change flows along that cycle. For example, if we start with the feasible flow in the diagram on the left and add arc $(4, 5)$ (shown in blue), we have an infinite family of feasible flows with all $|N|$ arcs (shown in the diagram on the right):



Demands: $d_1 = -5,\ d_2 = 3,\ d_3 = 5,\ d_4 = -1,\ d_5 = -3,\ d_6 = 1$

In general, when we find the cycle containing the new arc $(4, 5)$, we should follow the cycle around to determine these signs. Arcs that match the direction of $(4, 5)$ get a $+\delta$; arcs that don't match the direction of $(4, 5)$ get a $-\delta$. This makes sure that the net flow at every node in the cycle is unchanged.

We can't increase x_{45} forever. If we want to keep having a feasible flow, we need all these arcs to remain positive. In this case, the limiting factor is arc $(4, 3)$. It tells us that we need $\delta \leq 1$ to have $x_{43} \geq 0$. When we set $\delta = 1$, x_{43} leaves the basis.

2.3 Reduced costs

In the diagram above, we haven't included the costs on the arcs, but of course, those are a big factor when we want to decide the variables we want to pivot on.

It's easiest to compute the reduced cost of a variable at the moment when we consider bringing it into the basis. In the example above, when we set $x_{45} = \delta$, we are forced to set $x_{56} = 3 + \delta$, $x_{63} = 2 + \delta$, and $x_{43} = 1 - \delta$. This changes the total cost of our flow by

$$\delta c_{45} + \delta c_{56} + \delta c_{63} - \delta c_{43}$$

and so the reduced cost of x_{45} is $c_{45} + c_{56} + c_{63} - c_{43}$. If this is negative, then it makes sense to make x_{45} basic.

As usual, we can repeat such pivoting steps until all reduced costs are nonnegative.

3 A two-phase simplex method for minimum-cost flow

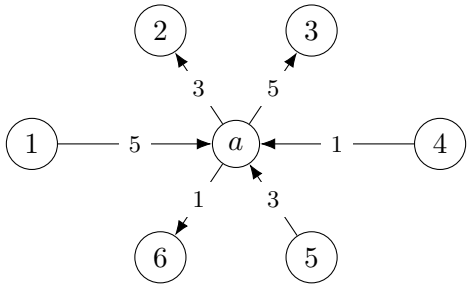
All of this is great, provided we have a starting feasible flow. We've seen that any spanning tree can give us a balanced flow; however, for a feasible flow, we also need $\mathbf{x} \geq \mathbf{0}$, and it's not clear how to ensure that.

Our source for this lecture is Chapter 14 of Vanderbei, which uses the dual two-phase simplex method, but dual pivoting is its own headache. To avoid discussing it, here is a method based on artificial variables instead.

In the first phase, we will add an artificial node a. To make satisfying the demands easy, we add the following artificial arcs:

- For every node $k \in N$ with demand $d_k < 0$, we add an artificial arc (k, a) with cost $c_{ka} = 1$.
- For every node $k \in N$ with demand $d_k > 0$, we add an artificial arc (a, k) with cost $c_{ak} = 1$.

Our initial spanning tree uses only these artificial arcs. The artificial arc in or out of node $k \in N$ will have flow $|d_k|$, which satisfies the demand constraint at k. In our example, we'd start with the spanning tree below:



Demands: $d_1 = -5,\ d_2 = 3,\ d_3 = 5,\ d_4 = -1,\ d_5 = -3,\ d_6 = 1,\ d_a = 0$

We ultimately want to get rid of these artificial arcs. So in the first phase, we ignore the original costs of all arcs; instead, artificial arcs will have cost 1, and arcs in the original network will have cost 0. The total cost of a feasible flow in this network will be minimized when no flow is going in or out of a.

There are $|N| + 1$ nodes in this network, because of node a, and so our spanning tree will have $|N|$ arcs. This means we won't actually be able to force all artificial arcs to leave the spanning tree. However, if we get to the point where only one artificial arc is used, the demand constraint at a will force the flow along that arc to be 0. At that point, node a and all arcs in or out of a can be deleted.

Once that is done, we are left with a spanning tree that corresponds to a basic feasible solution in the original network. At this point, we can go on to solve the original problem (with the original costs on all arcs).