Lecture notes 7: Network flow problems

Vincent Conitzer

1 Introduction

We now consider *network flow* problems. Such problems have some very nice properties. Specifically, we consider the *minimum cost network flow* problem, also known as the *transshipment* problem. In this problem, we are given a directed graph (V, E). The goal is to transport certain amounts of a single commodity from some vertices to others, across the edges of the graph. Specifically, each vertex v has a demand d_v, which is the net amount of the commodity that it wants to receive. d_v can be negative: in this case, the vertex has a net supply rather than a net demand, that is, the vertex is a producer of the commodity. We assume that demand and supply are perfectly matched overall, that is, $\sum_v d_v = 0$. We can transport an unlimited amount across each edge, but associated with each edge e, there is a cost c_e of transporting one unit across that edge. Our goal is to find a flow such that everyone's demand is met (exactly), while incurring the minimum total transportation cost. We can without loss of generality assume that there are edges in both directions between any two vertices (if there is no such edge originally, we can add one with (near) infinite cost).

We can write this as the following linear program:

minimize $\sum_{v,w \in V}^{n} c_{vw} x_{vw}$
subject to
$(\forall v \in V)\ \sum_{u \in V} x_{uv} - \sum_{w \in V} x_{vw} = d_j$
$(\forall v, w \in V)\ x_{vw} \geq 0$

The dual of this problem is:

maximize $\sum_{v \in V} d_v y_v$
subject to
$(\forall v, w \in V)\ y_w - y_v \leq c_{vw}$

This dual can be interpreted as follows: with every vertex v, we associate a *potential* y_v. This can be thought of as the height of vertex v. The constraint then says that for any edge, to move flow along that edge, we must pay a cost that is at least the increase in potential. The objective $\sum_{v \in V} d_v y_v$ is the overall increase in potential if we satisfy all the demands; clearly, we must pay at least this much (and this is the interpretation of weak duality in this context). Strong duality tells us that we can in fact find potentials such that the overall increase in potential is equal to the minimum cost we need to pay to satisfy the demands. Complementary slackness tells us that for optimal solutions to the primal and dual, if an edge has positive flow on it, then the difference in potential between its nodes must be equal to the cost of that edge.

One observation is that we can drop the constraint for any *one* node v^* in the primal without changing the problem: this is because if all the other nodes have their demands met exactly, then so must v^*, because $\sum_v d_v = 0$. In fact, if we sum the constraints for all the other nodes together, we recover the constraint for v^*. Omitting the constraint for v^* in the primal would correspond to setting the variable y_{v^*} to zero in the dual, and it is easy to see that this does not fundamentally change the dual problem, because all that matters in the dual problem is the *differences* between the y_v variables (differences in height). That is, we can always shift the values of all the y_v variables by the same amount without changing feasibility or the objective (because $\sum_v d_v = 0$), and by performing such a shift we can set y_{v^*} to zero.

2 The integrality theorem

While we introduced the minimum cost network flow problem as a linear program, often, we require an integrality constraint on the flows. Surprisingly, such an integrality constraint turns out not to matter. A result known as the *integrality theorem* states that, as long as all the demand data in a minimum cost network flow problem are integers, then there is an optimal solution to the linear program consisting of only integers. In fact, any basic feasible solution consists of only integers, and of course there exists a basic optimal solution (which is what the simplex algorithm will find). That is, we can solve the integer program simply by solving the linear program.

Lemma 1 *A linear program with m constraints has an optimal solution in which at most m variables are set to nonzero values.*

Proof: In an optimal dictionary, only the basic variables can be nonzero, and there can be at most m of these. ∎

Theorem 1 (Integrality) *Any minimum cost network flow problem instance whose demands are all integers has an optimal solution with integer flow on each edge.*

Proof: We will prove this by induction on $|V|$; clearly it is true for $|V| = 0, 1$. Recall that one of the constraints in the primal is redundant, so we really have only $|V| - 1$ constraints in the primal. By Lemma 1, this means that there is an optimal solution in which only $|V| - 1$ edges have positive flow on them. This, in turn, means that there is a vertex v that has at most one edge with positive flow incident to it. (If every vertex had at least two edges with positive flow incident to it, then there would be at least $2|V|$ endpoints of edges with positive flow, and hence at least $|V|$ such edges.)

Let v' be the other vertex incident to this edge (if there is no edge with positive flow incident to v at all, which can only happen if $d_v = 0$, then pick v' arbitrarily). This edge between v and v' must satisfy the entire demand of v, and hence carry an integer amount of flow. The remainder of the solution constitutes an optimal solution of the smaller instance with $|V| - 1$ vertices that is obtained by adding the demand of v to that of v', and removing v and all edges incident to it; moreover, we could replace this with any other optimal solution to this smaller instance. But, by the induction assumption, the smaller instance has an optimal solution with only integer flows. ∎

The integrality theorem is an extremely useful tool for showing that problems that are naturally expressed using integer variables can in fact be solved using linear programming. For example, consider the *minimum weighted bipartite matching* problem, also known as the *assignment* problem. In this problem, we are given a bipartite graph with m vertices on the left and m vertices on the right, and for every combination of a vertex v on the left and a vertex w on the right, there is a cost c_{vw} of matching these two together. Our goal is to find a perfect matching of minimum cost, that is, a subset of the edges that covers each vertex once and has minimum cost. We can take the LP relaxation of (the natural integer program formulation of) this problem, allowing, for example, a vertex on the right to be half-matched to one vertex on the left, and half-matched to another vertex on the left. This can be seen as a special case of the minimum cost network flow problem, by giving each of the vertices on the left a demand of -1 and each of the vertices on the right a demand of 1.

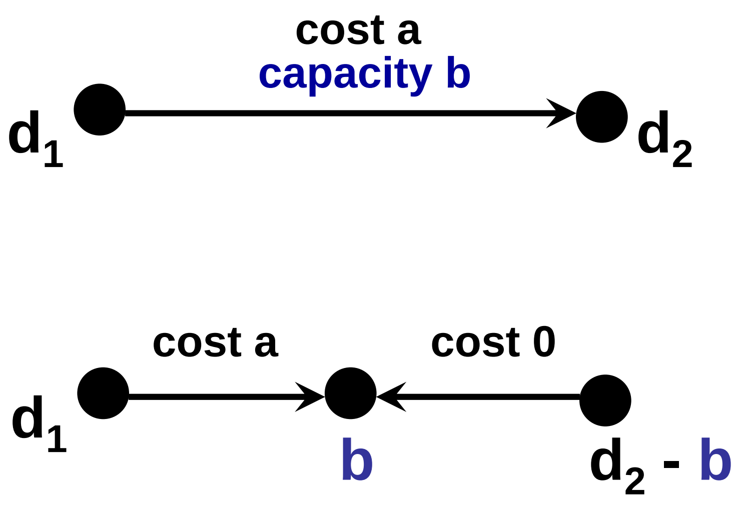

Figure 1: Reducing the problem with capacities to that without capacities.

Thus, by the integrality theorem, any basic feasible solution to this LP relaxation consists of only integers, so we can use the LP relaxation to solve the original assignment problem.

The integrality theorem can also be used in a noncomputational way, to prove mathematical theorems. A nice example is König's theorem, which states that if we have m boys and m girls, each boy knows exactly k girls and each girl knows exactly k boys, then the boys and girls can be arranged into pairs where each knows the other. (Here, when a boy knows a girl, that girl must also know the boy, and vice versa.) This can be seen as a special case of the assignment problem, where there is a cost of 0 for matching a boy and a girl that know each other, and a cost of 1 for matching a boy and a girl that do not know each other. If we consider the LP relaxation of (the natural integer program formulation of) the assignment problem, allowing for partial matchings, it is easy to see that there exists a solution of cost 0: simply match $1/k$ of each boy to each girl he knows, and $1/k$ of each girl to each boy she knows. From the integrality theorem, it follows that there also exists an integer solution with costs 0, which corresponds to an (integer) matching in which everyone knows his or her partner. This proves König's theorem.

3 Adding capacities

We can consider an extended version of the minimum cost network flow problem in which the edges also have capacities, that is, there is a limit on how much can be transported across an edge. In a sense, this does not really affect the problem significantly: we can reduce the extended problem to the regular minimum cost network flow problem (without capacities) using a neat trick, illustrated in Figure 1. The top part of the figure shows an edge with both a capacity and a cost between two vertices with demands. The bottom part of the figure shows an equivalent structure without capacities with which we can replace the original edge. Sending nothing across the original edge corresponds to meeting the middle vertex' demand by sending b flow from the right vertex. Sending the maximum of b flow across the original edge corresponds to meeting the middle vertex' demand by sending b flow from the left vertex. Sending intermediate amounts of flow across the original edge corresponds to meeting the middle vertex' demand by sending some flow from the left and some from the right.

We can replace every edge this way, thereby reducing the problem with capacities to that without capacities. If the input data in the problem with capacities are integers, then so are the data in the transformed problem. Hence, any basic feasible solution in the transformed problem consists of all integers, corresponding to a solution of the original problem consisting of all integers. So, the Integrality Theorem applies to the problem in which we have added capacities as well.

We can now also reduce the maximum flow problem we studied before to the minimum cost network flow problem: put a cost of 0 on every edge, give the target node t a demand that is a clear upper bound for the max flow instance (for example, the sum of the capacities of the edges coming into t), the source node the negative of that, and all other nodes a demand of 0; and add another edge from s to t with infinite capacity and cost 1. An optimal solution will send as little flow through the costly edge as possible, and therefore as much flow through the original graph as possible.