CS 435 : Linear Optimization | Fall 2008

Lecture 20: Primal-dual algorithm for MST

Lecturer: *Sundar Vishwanathan*
Computer Science & Engineering | Indian Institute of Technology, Bombay

Our objective is to design combinatorial algorithms for specific problems using hints from linear programming.

Here are the main steps we will follow.

1. Write an ILP formulation of the problem. Remove the integrality constraint and write the dual. We may or may not need the dual. Depending on the problem we will work either with the primal or the dual. Suppose that the LP formunation we are working with is

$$\begin{aligned} \max\ & c^T x \\ & Ax \leq \mathbf{b} \end{aligned} \tag{1}$$

2. The algorithm we design will be iterative. To begin find an initial feasible solution x_0. After the ith iteration, we will have a feasible solution x_i. Note that this means $Ax_i \leq \mathbf{b}$.
3. Design a combinatorial algorithm for the following problem. Let A' be a subset of the rows of A. For the $i+1$th iteration, A' will be the set of rows of A such that $A'x_i = \mathbf{b}'$. The algorithmic task is to find a y such that $c^T y > 0$ and $A'y \leq 0$. Note: Sometimes, this may lead to unbounded optima. In which case we will restrict $y_i \leq 1$.
4. Update $x_{i+1} = x_i + \lambda y$ such that x_{i+1} is feasible and λ as large as possible.
5. Iterate over this step to get a combinatorial algorithm for the original problem.
6. Analyse running times. LP theory may not help you here.
7. Prove correctness. You may be able to use either the duality theorem or complementary slackness here.

The key design step is finding y. This looks like the original problem with an important difference. The right hand side is now zero. To restate this, the rhs $\mathbf{b}$ is now $\mathbf{0}$. Suppose $\mathbf{b}$ represented the weighted version of the problem; we now have the same problem without the weights which may be easier to solve.

For instance, the unweighted instances of both minimum spanning trees and shortest paths are easier to solve than their weighted counterparts. The algorithm now looks like this:

Find an initial iterate x_0.
Loop over i
Identify rows of A which are equalities. Call this A'. So, $A'x_i = b'$, and the rest are strict inequalities.
Find y such that $A'y \leq 0$, $c^T y > 0$.
Find the maximum $\lambda > 0$ such that $A(x_i + \lambda y) \leq \mathbf{b}$.
Set $x_{i+1} = x_i + \lambda y$.
Endloop.

Motivation. Why did we do this? We have $Ax_i \leq \mathbf{b}$. We need $Ax_{i+1} \leq \mathbf{b}$ such that $c^T x_{i+1} > c^T x_i$. In order to do this we determine the direction of increase which we denote by y. Think of y as $x_{i+1} - x_i$. From the latter inequality we get $c^T y > 0$. For feasibility $Ax_{i+1} = A(x_i + y) \leq \mathbf{b}$. Since for A' we have $A'x_i = \mathbf{b}$ and $A'x_{i+1} \leq \mathbf{b}$, we need $A'y \leq 0$. For the other rows we can scale y to retain feasibility.

We end this lecture with an ILP for minimum spanning trees. Try doing this before you read ahead.

To find an ILP formulation for any problem, we need to do three things:

1. Decide on the variables. To start with, these should define the solution. Sometimes extra variables may be needed, but we defer this discussion. In other words, the assignment to the variables should let us figure out what the solution should be.
2. Decide on the cost function.
3. Decide on the constraints. These should define the space of all possible solutions.

For minimum spanning trees, we have one variable x_e per edge e. The cost is $\Sigma_{e \in E} c_e x_e$. We will assume that $c_e \geq 0$. What about the constraints? The constraints must be such that they ensure that the edges picked yield one connected component. How do we do this?

One way is to use the fact that a graph is connected if and only if for every cut into two parts the number of edges crossing the cut is at least one.

For a technical reason, which we will only hint at, we will use a slight generalisation: a graph is connected *iff* for every partition Π of the vertex set into $|\Pi|$ parts, the number of edges crossing the partition is at least $|\Pi| - 1$. An edge is said to *cross* a partition if its end-points are in different parts.

Here is the ILP formulation.

$$\begin{aligned}
\min \quad & \sum_e c_e x_e & \\
\text{s.t.} \quad & \sum_{e \text{ crosses } \Pi} x_e \geq |\Pi| - 1 & \forall \Pi \\
& x_e \geq 0 & \forall e \in E \\
& x_e \leq 1 & \forall e \in E \\
& x_e \text{integral} &
\end{aligned}$$

Where $c_e \geq 0$ is the weight of the edge $e \in E$. Π denotes a partition of the vertex set V and can be represented as the set of disjoint union of subsets of the vertex set V, i.e. $\Pi \equiv \{p_1, p_2, \ldots, p_k\}, p_i \subset V, \cup p_i = V, p_i \cap p_j = \phi$. $|\Pi| = k$ is the number of parts in Π.

Notice that this LP has an exponential number of constraints. It is to be simply used as a tool for designing algorithms. And as we shall see, this fact will not impede our search for an algorithm.

Let us note that dropping the $x_i \leq 1$ constraints will not change the solution. Why? Argue that changing any larger x_i to 1 will preserve connectivity.

Next, we will drop the integrality constraint.

Now we come to the important part. We would like to get rid of the costs. If these had appeared on the RHS we can go ahead with the other steps, but unfortunately, they appear in the cost function. So, we do the next natural thing: look at the dual. In the dual, the costs appear on the RHS.

Here is the dual.

$$\begin{aligned}
\max \quad & \sum_{\Pi} y_\Pi (|\Pi| - 1) & \\
\text{s.t.} \quad & \sum_{e \text{ crosses } \Pi} y_\Pi \leq c_e & \forall e \in E \\
& y_\Pi \geq 0 & \forall \Pi
\end{aligned}$$

The costs appear on the RHS. We will work with this formulation.