CS 435 : Linear Optimization — Fall 2008

Lecture 20: Primal-dual algorithm for MST: The Algorithm

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

We continue with the job of designing a primal dual algorithm for MSTs.

Here is the LP formulation.

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

Here is the dual.

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

We will work with the dual. Note that the cost function occurs on the rhs. In the algorithm, we will maintain and update the dual variables, y_Πs. Initially they are all zeroes. At any stage we need to improve the cost function. In general we can do that by increasing some of the ys and decreasing some. To keep things as simple as possible, we will try and do this by only *increasing* one of the ys at a time.

Let us assume that all c_es are strictly positive, so initially, none of the constraints are tight. We wish to increase the cost function. To do so we have to increase some y_Π. Which one? It seems that we gain the most by increasing the one where each vertex is in a separate partition. So, suppose we start to increase this. How much can we increase it by? We see that we can increase this upto the weight of the minimum weight edge. At this point the inequalities corresponding to all edges of minimum weight will be tight.

Let us consider the generic step. So suppose that at some stage we have some y. As per our receipe we need to consider only the inequalities which are equalties. So, let F denote the set of edges which are currently tight (the corresponding inequalities are tight.) We need to increase some y_Π such that for each of these edges the sum of the increases in the ys that the edge crosses is at most zero. This means we can increase a y_Π such that none of the edges of F cross Π. How do we find such a Π? The most natural is to find the current connected components and put each component in one part, to derive our partition.

Here then is the algorithm for MSTs:

1. *Initialization:* We think of all y_Πs to be zero. Note that we cannot explicitly set them.
2. *Iterative Step:* Let E' denote the set of edges which are tight. Find the connected components of the graph $G' = (V, E')$. Increase y_Π till some edge becomes tight, where the parts of Π are the connected components of G'.
3. The previous step terminates when we get one connected component.

First convince yourself that each step can be done in polynomial time. I suggest you take an example graph and see what happens.

Exercise: What can you say about the number of iterations in the worst case? What can you say about the number of non-zero ys at termination? Notice that in each iteration, at least one edge crossing the current partition becomes tight so the number of connected components goes down by at least one.

Exercise: If the edge lengths are unique prove that you get a spanning tree.

Exercise: Prove that the weights of the edges picked are non-decreasing.

What does this algorithm remind you of?

To recap we start with any feasible solution for the dual LP and maintain feasibility in the dual LP. In each iteration, we try to improve the dual solution. For this, we raise the value of the dual variables one by one. If a constraint becomes tight for an edge we do not raise the dual variables of partitions which these edges cross. After each iteration, we take the edges for which the constraints are equalities. If this set of edges yield a connected graph, we are done. A suitable spanning tree will yield a primal feasible solution. We essentially start with zero partition (i.e. an empty tree initially) and add edges as we go along until dual constraint corresponding to some edge e becomes tight. Then the partition is updated by merging the parts containing the end-points of e.

There is one more point: If all edges are of weight one, then we will, in our first step itself, pick all the edges. But you notice that then we can pick any spanning tree here. Hence we need to prune the solution we get. This we do in the reverse order in which we added edges.

The Reverse Delete Step: Consider the partitions which have non-zero weight in reverse order of their appearence. Among the edges remaining which cross the partition pick a spanning set and throw away the rest. The number of edges remaining which cross the partition will be $\Pi - 1$.

Exercise: Prove, by induction on the number of reverse delete steps executed that the final solution will be a spanning tree.

1 Proof of optimality

We can prove that the above algorithm gives an optimum solution by exhibiting a primal and dual solution of the same cost.
Cost of primal is given by:

$$\begin{aligned}
& \sum_e c_e \quad \text{where e is chosen by the algorithm} \\
= & \sum_e \sum_{\substack{\Pi \\ \text{e crosses } \Pi}} y_\Pi \\
= & \sum_\Pi \sum_{\substack{e \\ \text{e crosses } \Pi}} y_\Pi \\
= & \sum_\Pi y_\Pi \sum_{e \text{ crosses } \Pi} 1 \\
= & \sum_\Pi y_\Pi \text{ (number of edges chosen which crosses } \Pi) \\
= & \sum_\Pi y_\Pi(|\Pi| - 1) \qquad\qquad Why?
\end{aligned}$$

This is exactly the cost of the dual. So we have proved that the cost of primal is same as the cost of the dual proving optimality.

2 Some comments and observations

- Optimality can be proved by using complimentary slackness. Try it.
- This LP has an integral solution which is optimal. Why?
- Note that the algorithm is combinatorial. It has nothing to do with LPs except maintaining the ys.
- Indeed once you recognise what the algorithm does you need not maintain the ys explictly and you get Kruskal's algorithm.
- Can you get Prim's by choosing the partitions differently?

3 Exercises

- Design an algorithm with an LP where we consider only those partitions with $|\Pi|$ restricted to 2. How will you manipulate the dual variables so that you end up with Kruskal's algorithm? Try the analysis. Does it work?
- For every positive integer n, find an example where optimum of the LP with Π restricted to two, is less than the weight of the minimum spanning tree.
- Prove that if we add a new constraint

  $$\sum x_e = n - 1,$$

  there exist an integral solution to this new LP.