ORIE 6300 Mathematical Programming I	October 24, 2014

Problem Set 8

Due Date: October 31, 2014

1. Consider the dual simplex algorithm for an uncapacitated network flow problem as described in the recitation on October 8. Suppose you have a basic solution $\bar{f}$ corresponding to some spanning tree, and all the reduced costs $\bar{c}_{(i,j)}$ are nonnegative, but some basic variable, say $f_{(k,\ell)}$, is negative. So as in the dual simplex method, we want to remove (k, ℓ) from the basis (and thus from the spanning tree).

   (a) (3 points) What happens to the spanning tree when (k, ℓ) is removed?

   (b) (7 points) In the dual simplex method, we want to choose some variable $f_{(g,h)}$ to enter the basis such that the entry $\bar{A}_{p,(g,h)}$ is negative. In this case, which arcs have this entry negative, and what is this entry for such arcs?

   (c) (5 points) Which arc is chosen by the minimum ratio test to enter the basis?

2. In this problem, we consider the *maximum multicommodity flow problem*, a variation on the maximum flow problem. In this problem we are given a directed graph G with nodes V and directed arcs A, and k source-sink pairs (s_i, t_i), where $s_i, t_i \in V$ for $i = 1, \ldots, k$. We may send flow only from a source s_i to the corresponding sink t_i. The goal is to send as much flow as possible from the sources s_i to their corresponding sinks t_i. Each arc $a \in A$ has a capacity u_a; we may not send more than u_a total units of flow through arc a.

   We can write the problem as a linear program. Let $\mathcal{P}_i$ be the set of paths in G from s_i to t_i. Our LP will have a variable x_P for each $P \in \mathcal{P}_i$ for each $i = 1, \ldots, k$. Then the maximum multicommodity flow problem can be modelled as the following linear program:

$$\text{Max} \sum_{i=1}^{k} \sum_{P \in \mathcal{P}_i} x_P$$

$$\begin{aligned} \sum_{i=1}^{k} \sum_{P \in \mathcal{P}_i : a \in P} x_P &\leq u_a \qquad \forall a \in A \\ x_P &\geq 0 \qquad \forall i = 1, \ldots k, \forall P \in \mathcal{P}_i \end{aligned}$$

   We will consider another problem given by a mathematical program on the graph G, where there are variables $z_a \geq 0$ for all arcs $a \in A$. Then we define

$$dist_z(v, w) = \min_{P : P \text{ a path from } v \text{ to } w \text{ in } G} \sum_{a \in P} z_a.$$

   The program defining the problem is as follows:

$$\begin{aligned} \text{Min} \sum_{a \in A} u_a z_a & & \\ dist_z(s_i, t_i) & \geq 1 & \forall i = 1, \ldots, k \\ z_a & \geq 0 & \forall a \in A. \end{aligned}$$

(a) (3 points) A multicut is a partitioning of the nodes V into two or more parts such that there is no path from s_i to t_i contained in a single part. The capacity of a multicut is the sum over all arcs a whose endpoints are in different parts of the capacities u_a. Show that any multicut gives a feasible solution to the minimization problem above with value equal to the capacity of the multicut.

(b) (5 points) Use LP duality to argue that the values of the minimization problem above and the maximum multicommodity flow problem are the same.

(c) (5 points) Give an example for $k > 1$ that shows that the value of the maximum multicommodity flow is not equal to the value of the minimum multicut (Hint: there is one such example on a graph of 3 nodes). Note that when $k = 1$ we have the standard maximum flow problem, and in this case we know that the maximum flow and minimum cut values are equal.

(d) (10 points) We wish to solve the maximum multicommodity flow linear program via the simplex method. To put this into our standard form, we negate the objective function and we add a slack variable $s_a \geq 0$ to each inequality to make it an equality. We can start with an initial basic feasible solution in which all the slack variables are in the basis (and hence all x_P are nonbasic and set to zero).

There can be an exponential number of s_i-t_i paths in the number of vertices, so we do not wish to explicitly maintain all the variables x_P. Explain how you can run the simplex method without ever keep track of more than $O(|A|)$ path variables x_P at a time. You may assume that you have a subroutine that finds a path achieving the minimum distance $dist_z(v, w)$ for any $v, w \in V$ and any $z \geq 0$.