A cleaner proof of the correctness of the Min Cost Arborescence Algorithm. Feb. 1, 2011

Let G be the original graph, and for each node v, let δ_v be the minimum cost of the edges entering v. Now at every node $v \neq r$, subtract δ_v from the cost of every edge entering v, and let G_1 denote the resulting graph. Note that all edge costs are non-negative. We proved that a minimum cost arborescence in G_1 is a min cost arboresence in G. So we consider the problem of finding a min cost arborescence in G_1.

The algorithm then selects exactly one edge of cost zero into each $v \neq r$. Let $\tilde{E}_1$ be the set of those $n-1$ selected edges. If they form an arborescence rooted at r, then return $\tilde{E}_1$ since it has cost zero which is the min possible cost in G_1. If they do not form an arborescence, then we proved that there must be a directed cycle C in $\tilde{E}_1$ which does not contain r. Contract C to a single node c and let G_2 be the resulting graph[1] Then recursively find a minimum cost arborescence in G_2.

The recursion ends because the instances get smaller (fewer nodes and edges). In the extreme case the algorithm creates a graph with one edge from r to a node v. After the cost reduction at that level of the recursion, that single edge has cost zero and so is a min cost arborescence. So assume, inductively that the algorithm finds a min cost arborescence T_2 for G_2.

Let (u,c) be the edge in T_2 directed into node c, and let (u,v) be the corresponding edge in G_1 where $v \in C$. Now, starting with T_2, expand c back to the cycle C minus the edge in C that is directed into v. The result is an arborescence T_1 in G_1. We want to prove that T_1 is a min cost arborescence in G_1. Note that the edge costs in T_1 are their costs in G_1, but since the costs of the expanded edges from C are zero, the cost of T_1 in G_1 is the same as the cost of T_2 in G_2.

Let T_1' be a min cost arborescence in G_1, and consider C as a set of nodes. Unless T_1' already has this form, remove all but one edge in T_1' into the nodes in C, say node v; remove any edge in T_1' between two nodes in C; and then add in all the edges in C except the edge in C into node v. The resulting graph is an arboresence T_1'' rooted at r, and its cost is less than or equal to that of T_1'. But this must be equality since T_1' is a min cost arborescence in G_1. Hence there always is a min cost arboresence in G_1 with the property

[1]Note that in this exposition we only contract a single cycle even if there are several. This works fine and there is no loss of generalization in recursing after contracting just a single cycle.

that only one node in C has an edge into it from a node outside of C, and the edges between nodes strictly inside C have cost zero.

Now we want to prove that T_1 (the arborescence in G_1 created by the algorithm) and T_1'' have the same costs in G_1. Suppose not, so that the cost of T_1'' is strictly less than the cost of T_1 in G_1. By construction, if the nodes of C in T_1 are contracted to a single node, the result is the arborescence T_2 in G_2. Moreover, the cost of T_1 in G_1 is equal to the cost of T_2 in G_2. Now T_1'' also has the property that only one node in C has an edge into it from a node outside of C, and any edge between two nodes in C has cost zero. So consider the directed tree created by contracting the nodes of C in T_1'' to a single node. The result is an arborescence T_2'' in G_2, and its cost is equal to the cost of T_1'' in G_1. So the cost of T_2'' is strictly less than the cost of T_2 in G_2. But that contradicts the assumption that T_2 is a min cost arborescence in G_2.

Therefore the algorithm finds a min cost arborescence in G_1 and hence in G.