

CO453: Network Design – Winter 2007

Instructor: Chaitanya Swamy

Solutions to Assignment 2

Throughout $G = (V, E)$ with $|V| = n$, $|E| = m$; for a set $A \subseteq V$, $E[A] = \{(u, v) \in E : u, v \in A\}$.

Q1(a) No, T need not be an MST even if each of its edges belongs to some MST. Figure 1 shows a counterexample: T_1 and T_2 are two MSTs, but T is not.

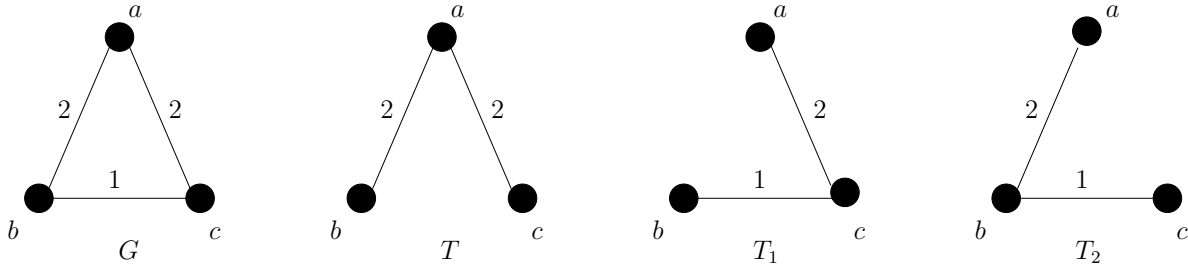


Figure 1: Counterexample for 1a). T_1 and T_2 are two MSTs, but T is not.

(b) Yes, T is a minimum-cost arborescence (MCA) rooted at r . Let y_v denote the minimum-cost edge entering node v . We know that if for each node $v \neq r$ we pick a minimum-cost edge entering v and the edges picked do not form any cycles, then we get an MCA. Thus, since G is acyclic, the edges picked form an MCA, and the MCA has cost $OPT := \sum_{v \neq r} y_v$. We claim that for any MCA T' and any edge $e = (u, v) \in T'$, we must have $c_e = y_v$. This immediately shows that T is an MCA since every edge $e = (u, v) \in T$ lies in some MCA T' and therefore by the claim, $\forall e = (u, v) \in T, c_e = y_v$; hence, $c(T) = \sum_{v \neq r} y_v = OPT$, showing that T is an MCA. To show the claim, suppose $e = (u, v) \in T'$ and $c_e > y_v$. Then,

$$c(T') = \sum_{(u,w) \in T', w \neq v} c_{(u,w)} + c_e > \sum_w y_w + y_v = OPT$$

since $c_{(u,w)} > y_w$. So T' cannot be an MCA giving a contradiction.

Q2(a) Since $c''_e = \max(0, c_e - 2y_v)$ for every edge (u, v) , we have $c_e \leq c''_e + 2y_v$. Therefore if $c''_e = 0$, then $c_e \leq 2y_v$. Also $\sum_{v \neq r} y_v$ is a lower bound on $c(T_{\text{opt}})$. Thus, since T has 0 c'' -cost, we have $c(T) = \sum_{e \in T} c_e \leq 2 \sum_{v \neq r} y_v \leq 2 \cdot c(T_{\text{opt}})$.

(b) Consider the example shown in Figure 2. T is the unique MCA wrt. costs c , and $T'' \neq T$ is the unique MCA wrt. costs c'' .

(c) We have already shown that for every edge $e = (u, v)$, $c_e \leq c''_e + 2y_v$. Also, note that $c''_e \leq c_e - y_v$ (since $0 \leq c_e - y_v$). Therefore, for any arborescence T , we have $c''(T) = \sum_{e \in T} c''_e \leq \sum_{e \in T} c_e - \sum_{v \neq r} y_v = c(T) - \sum_{v \neq r} y_v$, and $c(T) = \sum_{e \in T} c_e \leq \sum_{e \in T} c''_e + 2 \sum_{v \neq r} y_v = c''(T) + 2 \sum_{v \neq r} y_v$.

(d) We will prove this by induction on $n = |V|$. The claim is that given a graph G and a cost function c , the algorithm returns an arborescence T of cost at most twice the cost of the optimal

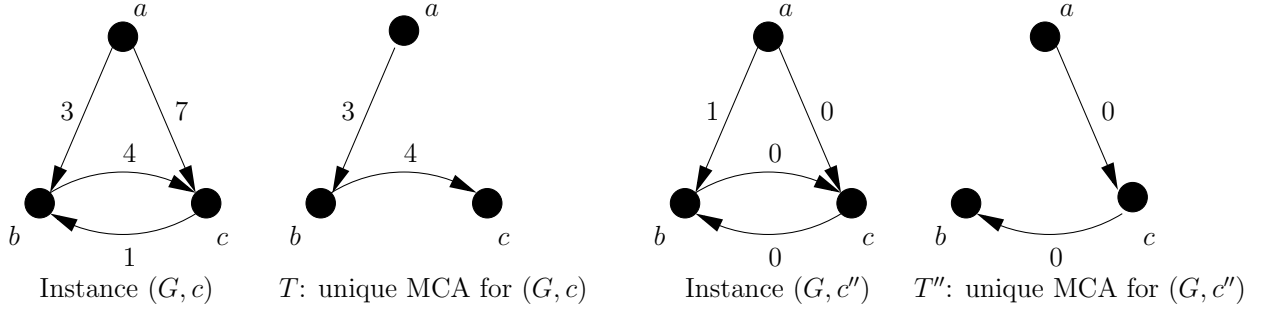


Figure 2: Example for 2b). T and T'' are the unique MCAs wrt. costs c and c'' respectively.

arborescence, i.e., $c(T) \leq 2 \cdot c(T_{\text{opt}})$. The base case, when $n = 1$, is trivially true since we just have the root node. Suppose the claim holds whenever $n \leq k$ and consider $n = k + 1$. Let c'' be the modified cost function. If the algorithm returns an arborescence T of 0 c'' -cost then by part a), $c(T) \leq 2 \cdot c(T_{\text{opt}})$ and we are done. Otherwise let G'' denote the graph obtained by contracting some 0 c'' -cost cycle C , and T'' and T''_{opt} denote respectively the arborescence returned by the algorithm, and an MCA for the instance (G'', c'') . By the induction hypothesis we have, $c''(T'') \leq 2c''(T''_{\text{opt}})$. Our claim involves the original cost function c and the arborescence T_{opt} , which is an MCA for the instance (G, c) , so we need to relate $c''(T''_{\text{opt}})$ and $c(T_{\text{opt}})$. Here the result of part (c) comes handy. The algorithm extends T'' to an arborescence T on G by “opening out” the cycle C , that is, by adding some 0 c'' -cost edges. Now consider the arborescence T_{opt} with cost function c'' . T_{opt} may enter the cycle C more than once. However, as argued in class, we may modify T_{opt} (by deleting some edges and adding edges of 0 c'' -cost) so that it enters C exactly once, and hence induces an arborescence, \hat{T} of no greater c'' -cost in G'' . Thus, we have, $c''(T_{\text{opt}}) \geq c''(\hat{T}) \geq c''(T''_{\text{opt}})$ since T''_{opt} is an MCA in G'' wrt. costs c'' . Putting the various pieces together, we obtain

$$\begin{aligned}
c(T) &\leq c''(T) + 2 \sum_{v \neq r} y_v && \text{(since } c_e \leq c''_e + 2y_v \text{ by 2(c))} \\
&= c''(T'') + 2 \sum_{v \neq r} y_v && \text{(since the edges of } C \text{ added to } T'' \text{ all have 0 } c''\text{-cost)} \\
&\leq 2 \cdot c''(T''_{\text{opt}}) + 2 \sum_{v \neq r} y_v && \text{(Induction Hypothesis)} \\
&\leq 2 \left(c''(T_{\text{opt}}) + \sum_{v \neq r} y_v \right) && \text{(by the above upper bound on } c''(T''_{\text{opt}})) \\
&\leq 2 \cdot c(T_{\text{opt}}). && \text{(since } c''_e + y_v \leq c_e \text{ by 2(c))}
\end{aligned}$$

Q3 First, observe that METRIC FACILITY LOCATION is clearly in NP : the certificate is simply the set of open facilities F' and the assignment of clients to these facilities, and the certifier simply checks if this is a valid solution of cost at most K .

We prove that the problem is NP -complete by reducing the SET COVER problem to it. Let $(U = \{u_1, \dots, u_n\}, \mathcal{S} = \{S_1, \dots, S_m\}, k)$ be an instance of SET COVER. The idea is to map the sets to facilities, and the elements to clients, and set the edge costs in such a way that any “low-cost” facility location solution translates to a set cover of size at most k and vice-versa. We will first define a non-complete graph G' with edge costs c' , and then take (G, c) to be the *metric completion* of

(G', c') ; recall that in the metric completion we create an edge between every pair of nodes and set the cost c_{uv} of an edge to be the shortest path distance in (G', c') between u and v . Thus the costs c will automatically satisfy the triangle inequality. The graph G' will be a complete bipartite graph, where we have set-nodes S_i for each $S_i \in \mathcal{S}$, $i = 1, \dots, m$ on one side and element-nodes u_j for each $u_j \in U$, $j = 1, \dots, n$ on the other side, and there is an edge (u_j, S_i) between every element- and set-node. With some hindsight, we set the cost c'_{u_j, S_i} to k if $u_j \in S_i$, and $3k$ otherwise. The METRIC FACILITY LOCATION instance is as follows: the graph G with edge costs c is the metric completion of (G', c') (note that there is a path between every pair of nodes in G' , so the shortest-path distances are well-defined); the facilities will be the set-nodes and the clients will be the element-nodes, that is, $F = \{S_1, \dots, S_m\}$, $C = \{u_1, \dots, u_n\}$; we set $K = k(n+1)$.

We now have to show that the METRIC FACILITY LOCATION instance has a solution iff the SET COVER instance has a solution. Suppose there is a set cover $\mathcal{S}' \subseteq \mathcal{S}$ of size at most k . Then we take F' to be the facilities corresponding to the sets in \mathcal{S}' , and for every client e_j , we assign it to some facility $S_i \in F'$ where $e_j \in S_i$; such a facility must exist since \mathcal{S}' is a set cover. The cost of this solution is at most $k(n+1)$: $|F'| \leq k$ and if u_j is assigned to $S_i \in F'$, we have $c_{u_j, S_i} = k$ since $u_j \in S_i$. Conversely suppose there exists a metric facility location solution of cost at most $K = k(n+1)$ that “opens” the facilities in $F' \subseteq F$. Observe that we have $c_{u_j, S_i} = k$ if $u_j \in S_i$, and $3k$ otherwise. We claim that (i) every client u_j is assigned to a facility $S_i \in F'$ such that $u_j \in S_i$; and (ii) $|F'| \leq k$. If (i) does not hold then the cost of the solution is at least $3k + (n-1)k = (n+2)k > K$, since there is at least one client $u_{j'}$ for which we incur an assignment cost of $3k$, and the assignment cost incurred for every other client is at least k ; if (ii) does not hold then the total cost incurred is at least $k+1 + nk > K$. Thus, if we take \mathcal{S}' to be the sets corresponding to the facilities in F' , we get a solution to the set cover instance.

Q4(a) We introduce the following notation: given a tree T on $S \cup \{r\}$ rooted at r , we define $\xi_T(S, v)$ to be the cost of the edge connecting v to its parent in T . Let G_S be the graph $(S \cup \{r\}, E[S \cup \{r\}])$, and $G_A = (A \cup \{r\}, E[A \cup \{r\}])$. Let T_S^* and T_A^* be respectively MSTs in G_S and G_A respectively. Thus, $\xi(S, v)$ is simply $\xi_{T_S^*}(S, v)$. We will assume for convenience that all edge costs are distinct; if not, as discussed in assignment 1, we can perturb the edge costs appropriately so that they become distinct, and T_S^* and T_A^* are the unique MSTs in G_S and G_A respectively. Consider any edge $e \in T_A^* \setminus T_S^*$. Adding e to T_S^* creates a cycle C . By the cycle property no edge of $T_S^* \cap C$ is the max-cost edge on C , so every edge of $T_S^* \cap C$ has cost at most c_e . Thus, taking any node $v \in A \cap C$ for which $(u, v) \in C \setminus T_A^*$, where u is v 's parent, we get that $c_{uv} = \xi(S, v) \leq c_e$. We can thus charge every edge of $T_A^* \setminus T_S^*$ to a smaller term $\xi(S, v)$, $v \in A$. This is the basic idea behind the proof of competitiveness, but one has to be careful to not do any double-counting by charging multiple edges of $T_A^* \setminus T_S^*$ to the same term $\xi(S, v)$, and this takes a little care.

Let $k = |T_A^* \setminus T_S^*|$. We will obtain a sequence of trees $T_0 = T_S^*, T_1, T_2, \dots, T_k \supseteq T_A^*$, where for every $i = 1, \dots, k$, tree T_i is obtained from T_{i-1} via an edge-exchange operation and will satisfy

$$(*) \quad \sum_{v \in A} \xi_{T_{i-1}}(S, v) \leq \sum_{v \in A} \xi_{T_i}(S, v).$$

Since $T_k \supseteq T_A^*$, we also have $\sum_{v \in A} \xi_{T_k}(S, v) = \text{MST}(A)$ because for every $v \in A$, the unique v - r in T_k is the same as the v - r path in T_A^* , hence, $\xi_{T_k}(S, v) = \xi_{T_A^*}(S, v)$. Thus, we obtain that

$$\sum_{v \in A} \xi(S, v) = \sum_{v \in A} \xi_{T_0}(S, v) \leq \sum_{v \in A} \xi_{T_1}(S, v) \leq \dots \leq \sum_{v \in A} \xi_{T_k}(S, v) = \text{MST}(A).$$

Let $T_A^* \setminus T_S^* = \{e_1, \dots, e_k\}$ where the edges are sorted by increasing cost. The tree T_i will contain edges e_1, \dots, e_i of T_A^* , and will be a subset $T_S^* \cup T_A^*$. Assume inductively that we have constructed trees T_0, T_1, \dots, T_i that satisfy this property and property (*). The base case, $T_0 = T_S^*$, vacuously satisfies these properties (we take $T_{-1} = T_0$). We now proceed to construct T_{i+1} . Adding edge $e_{i+1} = (a, b)$ to T_i creates a cycle C . As argued earlier, every edge $e \in T_S^* \cap C$ satisfies $c_e < \max_{e' \in C} c_{e'}$. Since the edges of T_A^* on cycle C all have lower cost than e_{i+1} , this implies that $c_e < c_{ab}$ for every edge $e \in T_S^* \cap C$. Let w be the least common ancestor of a and b in T_i (i.e., w is the last node, starting from the root r , that lies on both the r - a and r - b paths). Suppose without loss of generality that the w - a portion of the cycle C contains an edge not in T_A^* . Choose $v \in A \cap C$ to be the first node on the w - a path in T_i , starting from a , whose parent-edge (u, v) (when T_i is rooted at r) is not in T_A^* . Set $T_{i+1} \leftarrow T_i \cup \{(a, b)\} \setminus \{(u, v)\}$. It is clear that $T_{i+1} \subseteq T_S^* \cup T_A^*$ and contains edges e_1, \dots, e_{i+1} . We have to prove that it satisfies (*). Observe that the only nodes for which $\xi_{T_i}(S, x) \neq \xi_{T_{i+1}}(S, x)$ are nodes that lie on the v - a path in T_i ; the parent of every other node in S is the same in both T_i and T_{i+1} . Let the v - a path in T_i be $x_0 = v, x_1, \dots, x_\ell = a$. Note that all these nodes lie in A since every edge $(x_{j-1}, x_j) \in T_A^*$ for $j = 1, \dots, \ell$. We have: (i) $\xi_{T_i}(S, v) = c_{uv}$ and $\xi_{T_{i+1}}(S, v) = c_{vx_1}$; (ii) $\xi_{T_i}(S, a) = c_{x_{\ell-1}a}$ and $\xi_{T_{i+1}}(S, a) = c_{ab}$; and (iii) for $0 < j < \ell$, $\xi_{T_i}(S, x_j) = c_{x_{j-1}x_j}$ and $\xi_{T_{i+1}}(S, x_j) = c_{x_jx_{j+1}}$. Thus, $\sum_{x \in A} (\xi_{T_i}(S, x) - \xi_{T_{i+1}}(S, x)) = c_{uv} - c_{ab} < 0$, proving that T_{i+1} satisfies (*).

(b) Suppose the modified algorithm runs for k iterations. Fix a set $S \subseteq V \setminus \{r\}$. A node/vertex will always refer to a node/vertex in S . The proof will proceed by induction on the number of iterations, but one needs to be careful in defining the induction hypothesis. Suppose e is the min-cost edge in the original instance (\bar{G}, \bar{c}) in the first iteration, and node v gets contracted along with nodes in $B_v \subseteq S \subseteq V$ to form a node of \bar{G}' in the second iteration: the cost share $\xi(S, v)$ of v is *not equal* to $(\text{cost share of } B_v) / |B_v| + y_v$. Thus, assuming inductively that the cost shares of the nodes in \bar{G}' (which correspond to subsets of S) recover the \bar{c}' -cost of the arborescence returned by the algorithm for (\bar{G}', \bar{c}') is not a strong enough inductive hypothesis.

Let G^i and \bar{c}^i denote respectively the contracted graph and (the symmetric) cost function, in iteration i , $i = 1, \dots, k$, where iteration 1 is the starting iteration. Let T^i denote the arborescence constructed by the algorithm for the instance (G^i, \bar{c}^i) . Let y_B^i denote the y_B value in iteration i ; if e is the min \bar{c}^i -cost edge in G^i , then y_B^i is equal to \bar{c}_e^i if $B \subseteq V$ is a supernode of G^i , and 0 otherwise.

Although in the modified algorithm, y_r could be positive (to maintain symmetric costs), we still set $y_B = 0$ for the supernode containing r , since the root r is not supposed to pay. Also note that in the modified algorithm the cycle C found in an iteration might include a supernode containing the root r : to “open out” this cycle we keep all edges of C except the one entering the supernode containing r . (In the earlier algorithm we implicitly always had $y_r = 0$ since y_v is only defined for $v \neq r$; so there we always had $y_B = 0$ if $r \in B$. Also, we only consider cycles C that do not contain r .) These were not explicitly mentioned in the problem statement (which has now been modified in the assignment).

We define $\xi^i(S, v) = \sum_{j=i}^k \sum_{B: v \in B} \frac{y_B^j}{|B|}$. Clearly $y_B = \sum_{i=1}^k y_B^i$, and hence, $\xi^1(S, v) = \sum_{B: v \in B} \frac{y_B}{|B|} = \xi(S, v)$. We show by induction on $k - i$ that $\sum_{v \in S} \xi^i(S, v) = \bar{c}^i(T^i)$ for every $i = 1, \dots, k$. Thus, $\sum_{v \in S} \xi(S, v) = \bar{c}^1(T^1) = c(T^1)$, and since T^1 is an MCA in the directed instance (\bar{G}, \bar{c}) it is the MST on S in the original instance. The base case, $i = k$ is trivial since there must be a 0 \bar{c}^k -cost arborescence rooted at r , and $y_B^k = 0$ for all B . For the induction step, suppose the claim is true for iteration $i + 1$. Consider iteration i . Let e be the min \bar{c}^i -cost edge and C be the cycle contracted.

Thus, $\bar{c}_{e'}^{i+1} = \bar{c}_{e'}^i - \bar{c}_e^i$ for any edge e' in \bar{G}^{i+1} . The algorithm returns an arborescence T^{i+1} for $(\bar{G}^{i+1}, \bar{c}^{i+1})$. We obtain an arborescence T^i for (\bar{G}^i, \bar{c}^i) from T^{i+1} by adding some 0 \bar{c}^{i+1} -cost edges from the cycle C , so $\bar{c}^{i+1}(T^i) = \bar{c}^{i+1}(T^{i+1})$. Let n^i be the number of nodes in G^i , each of which corresponds to a supernode $B \subseteq S$. So,

$$\begin{aligned} \bar{c}^i(T^i) &= \bar{c}^{i+1}(T^{i+1}) + (n^i - 1)\bar{c}_e^i \\ &= \sum_{v \in S} \xi^{i+1}(S, v) + \sum_{\text{supernodes } B \text{ of } \bar{G}^i} y_B^i \quad (\text{Induction Hypothesis, } y_B^i = \bar{c}_e^i \text{ for a supernode of } \bar{G}^i) \\ &= \sum_{v \in S} \xi^{i+1}(S, v) + \sum_{B \subseteq V} \sum_{v \in B} \frac{y_B^i}{|B|} \quad (y_B^i = 0 \text{ for a non-supernode } B) \\ &= \sum_{v \in S} \sum_{j=i}^k \sum_{B: v \in B} \frac{y_B^j}{|B|} = \sum_{v \in S} \xi^i(S, v). \end{aligned}$$

This completes the proof of the budget-balance property.

To prove competitiveness, we will show that for every edge $e = (u, v)$, we have that $\sum_{B: v \in B, u \notin B} y_B \leq \bar{c}_e$. Note that y_B is positive only if $B \subseteq S$. We will prove this shortly, but first let us see how this implies competitiveness. Fix a set $A \subseteq S$. Then,

$$\sum_{v \in A} \xi(S, v) = \sum_{v \in A} \sum_{B: v \in B} \frac{y_B}{|B|} = \sum_{B \subseteq S} \sum_{v \in B \cap A} \frac{y_B}{|B|} \leq \sum_{B \subseteq S} y_B.$$

On the other hand, rooting the MST on $A \cup \{r\}$, at r , let $p(v)$ denote the parent of node $v \in A$ (i.e., the node just before v on the r - v path). Then,

$$\text{MST}(A) = \sum_{v \in A} \bar{c}_{p(v)v} \geq \sum_{v \in A} \sum_{B: v \in B, p(v) \notin B} y_B \geq \sum_{B \subseteq A} y_B |\{v \in B : p(v) \notin B\}| \geq \sum_{B \subseteq A} y_B.$$

The second inequality follows since we are only considering a subcollection of all possible sets B , and the third one follows since for any $B \subseteq A$, there is at least one node of B whose parent is not in B (since $r \notin B$).

We prove by induction on $k - i$, that for every iteration i , for every edge $f = (u, v)$ in G^i , we have $\sum_{j \geq i} \sum_{B \supseteq B_v, B_u \cap B = \emptyset} y_B^j \leq \bar{c}_f^i$, where $B_u, B_v \subseteq S$ are respectively the supernodes corresponding to u and v respectively. Clearly taking $i = 1$ yields the desired claim. The base case, $i = k$, is again trivial, since $y_B^k = 0$ for all B . Suppose the claim is true for iteration $i + 1$ and consider iteration i . Let e be the min \bar{c}^i -cost edge and C be the cycle contracted. Fix an edge $f = (u, v)$ in G^i . Observe that $\sum_{j \geq i} \sum_{B \supseteq B_v, B_u \cap B = \emptyset} y_B^j = y_{B_v}^i + \sum_{j \geq i+1} \sum_{B \supseteq B_v, B_u \cap B = \emptyset} y_B^j$. We consider two cases. If both u and v are in cycle C , then in every subsequent iteration $j \geq i + 1$, if B is the supernode containing B_v , then $B \supseteq B_u$; so $\sum_{j \geq i} \sum_{B \supseteq B_v, B_u \cap B = \emptyset} y_B^j = \bar{c}_e^i \leq \bar{c}_f^i$. If at most one of u and v is part of C , then edge f maps to a unique edge f' in G^{i+1} (B_u or B_v may be a subset of another supernode in G^{i+1} and f' would then be incident on that supernode). The induction hypothesis implies that $\sum_{j \geq i+1} \sum_{B \supseteq B_v, B_u \cap B = \emptyset} y_B^j \leq \bar{c}_{f'}^{i+1}$ because if $y_B^j > 0$ for $j \geq i + 1$ for a set considered in the summation, then B must contain the supernode in G^{i+1} containing B_v , and be disjoint from the supernode containing B_u in G^{i+1} . So we get $\sum_{j \geq i} \sum_{B \supseteq B_v, B_u \cap B = \emptyset} y_B^j \leq \bar{c}_e^i + \bar{c}_{f'}^{i+1} = \bar{c}_f^i$.

(c) Now we prove that the cost-sharing is monotonic. We introduce a notion of time, t . Initially $t = 0$. Suppose in iteration i we choose e as the min \bar{c}^i -cost edge, then the time ‘‘advances’’ by

\bar{c}_e^i , i.e., after iteration i we have $t \leftarrow t + \bar{c}_e^i$. Let t^i be the time at which iteration i ends, with $t_0 = 0$. Observe the following important fact: for any vertex u of \bar{G} and iteration i , the supernode B_u containing u in G^i is exactly *the set of nodes reachable from u using contracted edges*. Suppose this is true for iterations 1 through i . In iteration i , suppose e and \bar{e} be two min-cost edges going in opposite directions. If e, \bar{e} are not incident on (the node in G^i corresponding to the supernode) B_u , then both the supernode containing u and the reachability from u do not change. Otherwise suppose $e = (B_v, B_u)$ (incident on the nodes in G^i corresponding to B_v, B_u). Then, to obtain G^{i+1} , we contract e and \bar{e} and merge the supernodes B_u, B_v , so all nodes in $B_u \cup B_v$ are reachable from u , since all nodes in B_v are reachable from v (induction hypothesis) and v is now reachable from u . We can extend the notion of time to instants between iterations — think of the iterations as running synchronously with a clock with iteration i taking $t^{i+1} - t^i$ time steps. Note that an edge f gets contracted at the time t and iteration i such that $t = t^i = \bar{c}_f$ (this follows from the equality $t^{i-1} + \bar{c}_f^{i-1} = t^i + \bar{c}_f^i$). Let B_t be the set of vertices reachable from u at time t using contracted edges. For $t \in [t^{i-1}, t^i)$, B_t is the supernode containing u in G^i . If u merges into the root r in iteration i_u and time $t_u = t_{i_u}$, we have,

$$(1) \quad \xi(S, u) = \sum_{j=0}^{i_u-1} \frac{y_{B_{t^j}}}{|B_{t^j}|} = \sum_{j=0}^{i_u-1} \frac{t^{j+1} - t^j}{|B_{t^j}|} = \sum_{t=0}^{t_u-1} \frac{1}{|B_t|},$$

since after time t_u , the supernode containing u also contains r and therefore its y -value is 0.

Let R, R' be the executions of the algorithm on the instances with and without node v respectively. Consider any vertex $u \in S, u \neq v$. Let B_t, B'_t be the set of vertices reachable from u at time t using contracted edges in runs R and R' respectively. The crucial fact is that at any time t , $B_t \supseteq B'_t$ since any edge that contracts in run R' will also contract in run R . This follows since we know that an edge f contracts at time $t = \bar{c}_f$. Thus, if $v \in B_t$, then $B_t \supset B'_t$, otherwise $B_t = B'_t$. Let t_u, t'_u denote the times respectively at which u reaches the root in R, R' . It follows that $t_u \leq t'_u$. Plugging this in (1) we get that $\xi(S, u) \leq \xi(S \setminus \{v\}, u)$, so the cost-sharing is cross-monotonic.

Comments: You might have noticed a close similarity between cost shares and the dual variables that arise for the MST and MCA problems. Consider the cost shares defined in part (b). One can write a linear program for the minimum-cost-arborescence problem, and take its dual, and show that these cost shares give an *optimal dual solution*. In fact, we have already proved this above. The first part of the proof of competitiveness shows that the cost shares form a *feasible solution*, and in fact a feasible solution to the dual of the MCA problem on $A \cup \{r\}$ for any subset $A \subseteq S$. The proof of competitiveness is thus really a weak-duality argument in disguise. The proof of budget-balance shows that the value of the dual is equal to the value of a feasible primal solution, and hence, proves that both the primal and dual solutions are optimal respectively. Thus, this yields another proof of correctness for Edmonds' algorithm and the modified algorithm, which are really primal-dual algorithms in disguise. Part (c) gives some further insights about the primal-dual process that gives rise to the modified algorithm. You should think about how one could explicitly describe Edmonds' and the modified algorithms as explicit primal-dual algorithms, analogous to the primal-dual algorithms for Steiner tree and 0-1 proper-function connectivity described in class.