Appendix

```
function UnCenter(Network N, Distance Matrix d)
{   // Current best value on network N.
    F_N := 0
    // Solution set.
    S := ∅
    for all edges e = (v_s, v_t) ∈ E do
        {   // Compute UB1.
            x_UB1 := X(v_s, v_t)
            F_UB1 := F_s^L(x_UB1)
            if F_N > F_UB1 then continue to next edge
            // Compute UB2.
            F_g := ∞, F_h := ∞
            for all nodes v_i ∈ V do
                {   if v_i ≠ v_s and (F_i^L(0) < F_g or (F_i^L(0) = F_g and w_i < w_g)) then
                        {   F_g := F_i^L(0)
                            v_g := v_i
                        }
                    if v_i ≠ v_t and (F_i^R(l_e) < F_h or (F_i^R(l_e) = F_h and w_i < w_h)) then
                        {   F_h := F_i^R(l_e)
                            v_h := v_i
                        }
                }
            x_UB2 := X(v_g, v_h)
            F_UB2 := F_g^L(x_UB2)
            // Try to tighten F_UB2.
            if F_s^L(x_UB2) ≤ F_UB2 then
                {   x_UB2 := X(v_s, v_h)
                    F_UB2 := F_s^L(x_UB2)
                    v_g := v_s
                }
            else if F_s^L(x_UB2) ≤ F_UB2 then
                {   x_UB2 := X(v_g, v_t)
                    F_UB2 := F_t^R(x_UB2)
                    v_h := v_t
                }
```

// F_{UB2} *must be at least as good as* F_{UB1}
if $F_{UB2} \geq F_{UB1}$ **then**
{ $(x_{UB2}, F_{UB2}) := (x_{UB1}, F_{UB1})$
$v_g := v_s$
$v_h := v_t$
}
if $F_N > F_{UB2}$ **then** continue to next edge
// *Compute UB3.*
$F_p := \infty$, $F_q := \infty$
for all nodes $v_i \in V$ **do**
{ **if** $v_i \neq v_s$ **and** ($F_i^L(l_e) < F_p$ **or** ($F_i^L(l_e) = F_p$ **and** $w_i < w_p$)) **then**
{ $F_p := F_i^L(l_e)$
$v_p := v_i$
}

if $v_i \neq v_t$ **and** ($F_i^R(0) < F_q$ **or** ($F_i^R(0) = F_q$ **and** $w_i < w_q$)) **then**
{ $F_q := F_i^R(0)$
$v_q := v_i$
}
}
$x_{UB3} := X(v_p, v_q)$
$F_{UB3} := F_p^L(x_{UB3})$
// *Try to tighten* F_{UB3}.
if $F_s^L(x_{UB3}) \leq F_{UB3}$ **then**
{ $x_{UB3} := X(v_s, v_q)$
$F_{UB3} := F_s^L(x_{UB3})$
$v_p := v_s$
}
else if $F_s^L(x_{UB3}) \leq F_{UB3}$ **then**
{ $x_{UB3} := X(v_p, v_t)$
$F_{UB3} := F_t^R(x_{UB3})$
$v_q := v_t$
}

// F_{UB3} *must be at least as good as* F_{UB1}
if $F_{UB2} \geq F_{UB1}$ **then**
{ $(x_{UB3}, F_{UB3}) := (x_{UB1}, F_{UB1})$
$v_p := v_s$
$v_q := v_t$
}
if $F_N > F_{UB3}$ **then** continue to next edge

```
// Set (x_e, F_e) to the best value found.
if F_UB2 ≤ F_UB3 then
    {   (x_e, F_e) := (x_UB2, F_UB2)
        v_a := v_g
        v_b := v_h
    }
else
    {   (x_e, F_e) := (x_UB3, F_UB3)
        v_a := v_p
        v_b := v_q
    }
// Create set L and R.
if v_a ≠ v_s then   L := L ∪ {v_a}
if v_b ≠ v_t then   R := R ∪ {v_b}
for all nodes v_i ∈ V do
    {   d := d(v_s, v_i) − d(v_t, v_i)
        if d < l_e and F_i^L(x_UB2) < F_UB2 then    L := L ∪ {v_i}
        if −d < l_e and F_i^R(x_UB2) < F_UB2 then   R := R ∪ {v_i}
    }
// Continue till the new value F_e cannot improve the current F_N,
// or until one of the node sets becomes empty.
while F_e ≥ F_N and ( L ≠ ∅ or R ≠ ∅ ) do
    {   // Pair all nodes in L against R, using a max{|L|,|R|} matching
        for all the pair of nodes (v_i ∈ L, v_j ∈ R) in the matching do
            {   x := X(v_i, v_j)
                if F_s^L(x) ≤ F_i^L(x) then  // F_i^L is over F_j^R
                    {   L := L − {v_i}
                        x := X(v_s, v_j)
                        v_i := v_s
                    }
                else if F_s^L(x) ≤ F_i^L(x) then    // F_j^R is over F_i^L
                    {   R := R − {v_j}
                        x := X(v_i, v_t)
                        v_j := v_t
                    }
                // Update (x_e, F_e)
                if F_i^L(x) < F_e then
                    {   x_e := x
                        F_e := F_i^L(x_e)
                        v_a := v_i
                        v_b := v_j
                    }
            }
```

```
                    // Project the value x_e on the lower envelope.
                    // Find the lowest left line.
                    F_a := ∞
                    for all nodes v_i ∈ L do
                        if F_i^L(x_e) < F_a or (F_i^L(x_e) = F_a and w_i < w_a) then
                            {   F_a := F_i^L(x_e)
                                v_a := v_i
                            }
                    // Find the lowest right line.
                    F_b := ∞
                    for all nodes v_i ∈ R do
                        if F_i^R(x_e) < F_b or (F_i^R(x_e) = F_b and w_i < w_b) then
                            {   F_b := F_i^R(x_e)
                                v_b := v_i
                            }
                    x_e := X(v_a, v_b)
                    F_e := F_a^L(x_e)
                    // Delete lines above the new value F_e
                    for all nodes v_i ∈ L do
                        if F_i^L(x_e) ≥ F_e then L := L − {v_i}
                    for all nodes v_i ∈ R do
                        if F_i^R(x_e) ≥ F_e then R := R − {v_i}
                }
            if F_e ≥ F_N then
                {   if F_e > F_N then
                        {   S := ∅
                            F_N := F_e
                        }

                    S := S ∪ {(x_e, e)}
                }
        }

    return (F_N, S)
}
```