EXERCISE 11.1. The nurse scheduling problem can be formulated as the following linear programming problem: minimize $cx$, subject to $Ax \geq b$, $x \geq 0$, where

$$
A = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 70 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 80 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 & 50 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 60 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 40 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 30 & 1
\end{bmatrix}
$$

$$
b = \begin{bmatrix}
70 \\
80 \\
50 \\
60 \\
40 \\
30
\end{bmatrix}
$$

$$
c^T = \begin{bmatrix}
1 \\
1 \\
1 \\
1 \\
1 \\
1
\end{bmatrix}
$$

Notice that in this linear program, the constraint matrix $A$ has consecutive ones in columns. We can use the result of Application 9.6 to formulate this problem as a minimum cost flow problem. Figure S11.1 gives the resulting formulation.

EXERCISE 11.3#. Instead of stating that "Formulate this problem as a minimum cost flow problem," we should say "Show how this problem can be solved using network flow techniques." This exercise has a fairly simple but clever formulation, and uses the observation that the project assignment problem can be decomposed into two independent problems. The first problem matches the students to projects (see Figure S11.3(a) and the second problem matches the projects to supervisions (see Figure S11.3 (b)). In the figure, the cost of an arc $(i, j')$ is the negative of the preference of student $i$ for project $j$; the capacity of the arc $(j', t)$ is the maximum number of students that can work on project $j$; an arc $(j', k'')$ is present whenever faculty $k$ is capable of supervising project $j$; and the capacity of the arc $(k'', t')$ equals the maximum number of projects that the faculty member $k$ can supervise.

Figure S11.47

Figure S11.1
EXERCISE 11.5. Figure S11.5 gives the minimum cost flow formulation of the problem.

![Figure S11.5](image_url)  
Figure S11.5

EXERCISE 11.7. Construct the network as in Application 6.3, and also define the lower and upper bounds of arcs in the same manner. If \( a_{ij} \) is integer, then clearly, \( l_{ij} = u_{ij} = a_{ij} \) and we set \( c_{ij} = 0 \). If \( a_{ij} \) is noninteger, then we define the cost of the arc \((i, j)\) equal to the slope of the error function with respect to the value to which it is rounded, i.e., \( c_{ij} = \left(\frac{a_{ij}}{2} - \left\lfloor a_{ij} \right\rfloor\right)^2 \), which can be simplified to \( c_{ij} = a_{ij}^2 + 2a_{ij}\left\lfloor a_{ij} \right\rfloor - \left\lfloor a_{ij} \right\rfloor^2 \). An optimal circulation in this network yields a corresponding optimal rounding.

EXERCISE 11.9. Suppose that \( T \) is a directed out-tree rooted at node \( s \). Now, no node can have indegree more than one; otherwise there will be more than one path to that node from node \( s \) (because if arcs \((p, r)\) and \((q, r)\) enter node \( r \), then the paths from \((i)\) from \( s \) to \( p \) to \( r \), and \((ii)\) from \( s \) to \( q \) to \( r \), form two distinct paths). Further, every node (except node \( s \)) must have indegree at least 1 because every node is reachable from node \( s \). Hence, every node except node \( s \) has an indegree of exactly one. Node \( s \) must have zero indegree because a spanning tree has \((n-1)\) arcs. To prove the converse result, suppose that every node except node \( s \) has an indegree of 1. Then, the unique path from node \( s \) to any other node must have all its arcs in the same direction (i.e., in the same direction as the path itself), otherwise at least one node in the path will have indegree more than one or node \( s \) will have indegree more than zero. Hence, the tree is a directed out-tree rooted at node \( s \). The equivalent result for a directed intree is the following: A tree is a directed in-tree \( T \) rooted at node \( t \) if and only if every node in \( T \) except node \( t \) has outdegree one.

EXERCISE 11.11. Define some orientation of the cycle \( W \). Suppose that we denote the incidence vector of the cycle \( W \) as \( I(W) \) and the incidence vector of a fundamental cycle defined by a nontree arc \((i, j)\) by \( I(i, j) \). Notice that the incidence vector of a cycle \( W \) is the resulting arc flow vector when unit amount of flow is sent along \( W \). This is also true for the incidence vectors \( I(i, j) \)'s. In view of this result, the problem reduces to expressing the flow vector \( W \) as a sum of the flow vectors \( I(i, j) \)'s. We partition the arcs in the cycle \( W \) into three subsets: (i) \( W^T \), those arcs which are also in \( T \); (ii) \( W^+ \), arcs which do not belong to \( T \) and are in the forward orientation; (iii) \( W^- \), arcs which do not belong to \( T \) and are in the reverse orientation. We claim that

\[
I(W) = \sum_{(i,j) \in W^T} I(i, j) - \sum_{(i,j) \in W^+} I(i, j) + \sum_{(i,j) \in W^-} I(i, j). \tag{i}
\]

Consider any nontree arc \((i, j)\) in \( W^+ \). This arc is not present in any other fundamental cycle defined by arcs in \( W^+ \cup W^- \) \- \{(i, j)\}, hence in the RHS of (i) we have +1 in the row corresponding to the arc \((i, j)\). By definition, arc \((i, j)\) is a forward arc in \( W \), hence the LHS of (i) also contains a +1 in the row corresponding to the arc \((i, j)\). Similar argument shows that for any nontree arc \((i, j)\) in \( W^- \), the LHS as well as RHS of (i) contain -1 in the row.
corresponding to the arc \((i, j)\). For any non-tree arc not in \(W^+ \cup W^-\), both the LHS and the RHS contain a zero. We now need to show that the equality in (i) holds for tree arc too. If we set a flow of +1 on arcs in \(W^+\), a flow of -1 on arcs in \(W^-\), and zero flow on all other non-tree arcs, then to satisfy the mass balance constraints of all the nodes, a unique flow will take place on arcs in the tree \(T\). The flow in \(W_T\) in the cycle \(W\) is such a unique solution. Thus the LHS and RHS of (i) must be the same for the rows corresponding to the arcs in \(T\).

**EXERCISE 11.13.** Figures S11.13(a) and (b) respectively specify the spanning tree solutions for the spanning trees shown in Figures 11.25(a) and 11.25(b). Notice that the solution is infeasible because flows on many arcs violate their flow bounds.

![Figure S11.13](image)

**EXERCISE 11.15.** The optimal node potentials from the given spanning tree are given in Figure S11.15. It can be easily verified that each \(\pi(j)\) equals the length of the tree path from node \(j\) to the root.

![Figure S11.15](image)

**EXERCISE 11.16.** The sequence of pivots (entering and leaving arcs) is shown in Figure S11.16(a) and the optimal spanning tree structure is given in Figure S11.16(b).

<table>
<thead>
<tr>
<th>Pivot no.</th>
<th>Entering arc</th>
<th>Leaving arc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1, 3)</td>
<td>(1, 2)</td>
</tr>
<tr>
<td>2</td>
<td>(2, 4)</td>
<td>(2, 5)</td>
</tr>
<tr>
<td>3</td>
<td>(3, 5)</td>
<td>(4, 5)</td>
</tr>
<tr>
<td>4</td>
<td>(2, 5)</td>
<td>(3, 2)</td>
</tr>
</tbody>
</table>
EXERCISE 11.17. The sequence of pivots performed by the network simplex algorithm with Dantzig’s pivot rule is shown in Figure S11.17. The optimal spanning tree solution is as shown in Figure S11.16(b).

<table>
<thead>
<tr>
<th>Pivot no.</th>
<th>Entering arc</th>
<th>Leaving arc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(2, 4)</td>
<td>(2, 5)</td>
</tr>
<tr>
<td>2</td>
<td>(1, 3)</td>
<td>(1, 2)</td>
</tr>
<tr>
<td>3</td>
<td>(3, 5)</td>
<td>(4, 5)</td>
</tr>
<tr>
<td>4</td>
<td>(2, 5)</td>
<td>(3, 2)</td>
</tr>
</tbody>
</table>

EXERCISE 11.19. The text justifies the procedure compute-flow for uncapacitated networks. To justify the procedure for capacitated networks, observe from Section C.5 in the Appendix C that to determine the flow on spanning tree arcs, we need to solve the following system of equations: \( Bx_B = b - \bigcup u_U = b' \). We next observe that \( b' = b - \bigcup u_U \) can be written as

\[
b'(i) = b(i) - \sum_{j : (i, j) \in U} u_{ij} + \sum_{j : (j, i) \in U} u_{ji}.
\]

Finally, note that the vector \( b' \) computed at the beginning of the procedure compute-flow is precisely the same as given above.

EXERCISE 11.21. The following method identifies the pivot-cycle \( W \) in \( O(|W|) \) time.

```
procedure identify-cycle;
begin
    unmark all nodes;
    set i := k and j := l;
    while node i and node j are both unmarked do
        begin
            mark the nodes i and j;
            set i := pred(i) and j := pred(j);
        end;
        if node i is marked then apex := i else apex := j;
end;
```

EXERCISE 11.23. We illustrate our methods using the figure shown in Figure S11.23. Let \((T, L, U)\) be the initial spanning tree structure, arc \((k, l)\) is the entering arc, and arc \((p, q)\) is the leaving arc. Dropping the arc \((p, q)\) from the tree \( T \) partitions it into two subtrees \( T_1 \) and \( T_2 \), where \( T_1 \) contains the root node and \( T_2 \) does not. We define the nodes \( k \) and \( q \) such that they belong to the subtree \( T_1 \), as shown in Figure S11.23. The subtree \( T_2 \) is
rooted at node $p$ but when the arc $(k, l)$ is added, then the tree $T_2$ must be realigned so that it is then rooted at node $l$. As a result of this alignment, the predecessor and depth indices will change.

Let path $P$ be the path from node $l$ to node $p$ in the subtree $T_2$. It is easy to observe that the predecessor indices of the nodes lying in the path $P$ will change and for all other nodes they remain unchanged. Further, for nodes in the path $P$, the parent-child relationship is reversed. If node $i$ is a predecessor of node $j$ in the previous tree, then node $j$ is the predecessor of node $i$ in the next tree. This change is easy to make. We start at node $l$, and walk up the tree using the old predecessor indices and reverse them. This can be done in time proportional to the number of nodes in the path $P$.

Next observe that the depth and thread indices of all the nodes in the subtree $T_2$ need to be updated. Once we have the correct predecessor indices of nodes in $T_2$, we use the procedure described in Exercise 11.22 to compute the depth and thread indices of nodes in $T_2$ by treating node $l$ as the root node. It takes the time proportional to the $|T_2|$, which would typically be faster than computing the tree indices of all nodes from scratch.

**Exercise 11.25.** Suppose that the residual network $G(x)$ contains $(i, \text{pred}(i))$ for every node $i \in N - \{1\}$. Then, for every node $p$, let $p - i_1 \cdots i_k - 1$ be the path from $p$ to the root node 1 obtained using the predecessor indices. Notice that by our assumption, the node sequence $p - i_1 \cdots i_k - 1$ defines a directed path from node $p$ to node 1 in the residual network. Hence for any node $p$, we can send a positive amount of flow from it to the root node. Consequently, the tree is strongly feasible. To prove the converse result, suppose that the spanning tree is strongly feasible. For a tree-arc $(i, j)$ if $0 < x_{ij} < u_{ij}$, the residual network contains both the arcs $(i, j)$ and $(j, i)$. For a strongly feasible tree, every zero flow arc $(i, j)$ must be upward pointing and every arc $(i, j)$ at its upper bound must be downward pointing: we find that in both the cases, either the arc $(i, \text{pred}(i))$ or the arc $(j, \text{pred}(j))$ must be present in the residual network $G(x)$ depending upon whether $j = \text{pred}(i)$ or $i = \text{pred}(j)$.

**Exercise 11.27.** Suppose that $(T, L, U)$ remains feasible even after the vector $b$ is perturbed by $\varepsilon$. Let $x$ be the unperturbed value of the flow vector. Then, for any upward pointing arc $(i, j)$ in $T$, the flow on the arc is $x_{ij} + D(i)/n \leq u_{ij}$, and for any downward point arc $(i, j)$ in $T$, $x_{ij} - D(j)/n \geq 0$. Since, $0 < D(i)/n < 1$ for each node $i \in N$, it follows that $x_{ij} < u_{ij}$ for each upward pointing arc $(i, j)$ in $T$ and $x_{ij} > 0$ for each downward pointing arc $(i, j)$ in $T$. Hence $(T, L, U)$ is strongly feasible with respect to the flow $x$. Conversely, if $(T, L, U)$ is strongly feasible with respect to the flow value $x$, then $x_{ij} < u_{ij}$ (i.e., $x_{ij} + 1 \leq u_{ij}$) for each upward pointing arc $(i, j)$ and $x_{ij} > 0$ (i.e., $x_{ij} \geq 1$) for each downward pointing arc $(i, j)$ in $T$. Since $0 < D(i)/n < 1$ for every node $i \in N$, it follows from the above
discussion that \( x_{ij} + D(i)/n \leq u_{ij} \) for every upward pointing arc \((i, j)\) in \( T \) and \( x_{ij} - D(j)/n \geq 0 \) for every downward pointing arc \((i, j)\) in \( T \). It follows that the perturbed flow on the arcs satisfy the flow bound constraints and hence the flow is feasible. Now use this equivalence and the result of Exercise 11.26(b) to show that when implemented to maintain a strongly feasible basis, the network simplex algorithm runs in pseudopolynomial time irrespective of the pivot rule used for selecting entering arcs.

**Exercise 11.29.** The first three iterations of the algorithm are illustrated in Figure S11.29.

**Exercise 11.31.** Since the assignment problem can be formulated as a minimum cost flow problem in which all arc capacities are of unit value, (1) and (2) follow as a direct consequence of Definition 1 of the strongly feasible spanning tree given in Section 11.6. Since any supply node \( i \) has only outgoing arcs and has unit supply/demand, it is obvious that every downward pointing arc will be the unique arc with flow equal to 1 emanating from node \( i \).

**Exercise 11.33.** Figure S11.33 illustrates various iterations of the parametric network simplex algorithm when it is applied to the minimum cost flow problem in Figure 11.25. The parametric network simplex algorithm, when applied to this problem solves it in three iterations. In the first iteration, 10 units of flow are augmented along the path 1-2-5-6; in the second iteration 5 units of flow are augmented along the path 1-2-4-6; and in the third iteration, 5 units of flow are augmented along the path 1-3-5-2-4-6.
EXERCISE 11.35. The parametric network simplex algorithm is a special case of the right-hand side parametrization of a linear program. Here the right-hand side of the mass balance constraints for node \( s \) is a parameter \( \lambda \), the right-hand side of the mass balance constraints for node \( t \) is \(-\lambda\), and all other right-hand entries are zero. As in the network simplex algorithm, the basis is a spanning tree. Augmenting maximum flow from source to sink in the spanning tree corresponds to identifying the characteristic interval of the current basis. Dropping the blocked arc defines an \( s-t \) cut, this cut corresponds to a row in the simplex tableau. All eligible arcs have a -1 entry in the simplex tableau. The \( \theta_{ij} \) is the negative of the ratio of the reduced cost \( c_{ij}^* \) with the updated input-output coefficient, i.e., \( \theta = -c_{ij}^*/a_{ij}^* \). Identifying the minimum \( \theta_{ij} \) corresponds to the minimum ratio test. Finally, changing the spanning tree corresponds to changing the basis.

EXERCISE 11.37. We define the set \( S \) as in Section 11.9. The set of eligible arcs in empty, when each arc in \( (S, \bar{S}) \), except the arc \((p, q)\), is at its upper bound and each arc in \( (\bar{S}, S) \) is at its lower bound (see Figure S11.37). Summing the mass balance constraints of nodes in \( S \) would yield

\[
\sum_{i \in S} b(i) = b(S) = \sum_{\{(i,j)\in(S, \bar{S})\}} x_{ij} - \sum_{\{(j,i)\in(\bar{S}, S)\}} x_{ij}.
\]

Using the facts that each arc \((i, j) \in (S, \bar{S}) - \{(p, q)\}\) has \( x_{ij} = u_{ij} \); (ii) \( x_{pq} > u_{pq} \) and (iii) \( x_{ij} = 0 \) for each arc \((i, j) \in (\bar{S}, S)\), we conclude that \( b(S) > \sum_{\{(i,j)\in(S, \bar{S})\}} u_{ij} = u[S, \bar{S}] \). In other words, \( b[S] - u[S, \bar{S}] > 0 \), which in view of Theorem 6.12 would imply that the minimum cost flow problem is infeasible.

EXERCISE 11.39. Figure S11.39(a) shows the initial spanning tree structure.
(1) When $c_{23}$ increases from 0 to 6, then there is just one pivot. The arc (3, 4) enters and the arc (3, 4) leaves. Figure S11.39(b) shows the modified optimal spanning tree.

(2) When $c_{78}$ decreases from 9 to 2, then the algorithm performs one pivot operation only. The arc (7, 8) enters and the arc (5, 6) leaves. Figure S11.39(c) shows the modified optimal spanning tree.

(3) When $b(2)$ decreases to 15 and $b(8)$ increased to -8, then the algorithm performs one pivot operation in which the arc (2, 6) enters and the arc (5, 6) leaves. The final optimal spanning tree structure is illustrated in Figure S11.39(d).

(4) When $u_{23}$ increases to 20 units, then the optimality condition for the arc (2, 3) are not violated. Hence initial spanning tree remains optimal.

**EXERCISE 11.41.** The basis matrix $B$ of the spanning tree in Figure 11.25(a) is illustrated in the following table. Notice that the redundant row (i.e., row 1) has been deleted.
EXERCISE 11.43. The value of $B^{-1}$ shall be computed under the assumption that the rows of $B$ are arranged in increasing order of node numbers and the columns of $B$ are arranged so that the corresponding arcs occur in increasing order of tail nodes, and if tail nodes are the same, then in the increasing order of the head nodes. Hence for the spanning tree shown in Figure 11.25(a), the flow vector $X = [x_{13}, x_{32}, x_{35}, x_{43}, x_{56}, x_{57}]^T$. Therefore, 

$$B^{-1}_2 = [-1, -1, 0, 0, 0, 0]^T, B^{-1}_3 = [-1, 0, 0, 0, 0]^T, B^{-1}_4 = [-1, 0, 1, 0, 0]^T, B^{-1}_5 = [-1, 0, -1, 0, 0]^T, B^{-1}_6 = [-1, 0, -1, 0, -1]^T, B^{-1}_7 = [-1, 0, -1, 0, 0, -1]^T.$$ 

EXERCISE 11.45. If $N$ is a totally unimodular matrix, then $N^T$ must also be totally unimodular, because (i) every square submatrix $A$ of $N$ has a corresponding square submatrix $A^T$ in $N^T$; and (ii) $\det(A) = \det(A^T)$. Further, if $N$ is totally unimodular, then $[N, N]$ must also be totally unimodular because every square submatrix of $[N, N]$ is either a square submatrix of $N$ or it has a repeated column (in which case its determinant is zero). Since $[N, N]$ is totally unimodular, $[N, -N]$ must also be totally unimodular, because every square matrix $A$ of $[N, N]$ has a corresponding square matrix $A'$ in $[N, -N]$ such that some of the columns of $A'$ might differ from the corresponding columns of $A$ in sign. If there are $k$ such columns, then $\det(A') = (-1)^k \det(A).$

EXERCISE 11.47#. Part (a) of the exercise requires a correction: We need to show that if $G$ is a complete graph, then the average cycle length is at most twice the average depth plus one.

(a) A complete graph $G$ has an arc between every pair of nodes, and therefore has $n(n-1)$ arcs. For a given tree $T$, the average cycle length is $\frac{\sum_{(i,j) \in A}(d(i,j)+1)}{n(n-1)}$. Notice that $d(i,j) \leq d(1,i) + d(1,j)$. Hence the average cycle length is at most $\frac{\sum_{(i,j) \in A} (d(1,i) + d(1,j)+1)}{n(n-1)}$. The preceding expression contains $2n(n-1)$ terms and each $d(1,i)$ occurs exactly $2(n-1)$ times (why?). Hence it can be simplified to $2\frac{\sum_{i \in N} d(1,i)+1}{n}$, which is two times the average depth plus one. This result is not true if $G$ is not a complete graph. Figure S11.47 gives a counterexample. For the graph shown in this figure, the sum of the depths is $(0 + 1 + 2 + \ldots + \sqrt{n-1} + 1(n - \sqrt{n})) < 2n$. Hence the average depth of the tree is less than 2. But, the network contains a unique cycle with length $\sqrt{n}$. Clearly, the above result does not hold.

(b) Observe that if node $i$ is an ancestor of node $j$, then node $j$ is descendant of node $i$. Hence each ancestral relationship has a corresponding descendant relationship. Therefore, $\sum_{i \in N} |E(i)| = \sum_{i \in N} |D(i)|$. Next observe that the depth of any node $i$ is $|E(i)|-1$, which implies that the average depth is $\frac{\sum_{i \in N} (|E(i)|-1)}{n} = \frac{\sum_{i \in N} |E(i)|}{n} - 1$. On the other hand, the average subtree size is $\sum_{i \in N} |D(i)|$. The preceding two results imply that the average subtree size is one more than the average depth.
EXERCISE 11.49. (a) The flow $x^0$ denotes the arc flow vector when the right-hand side vector is $b^0$ and we can compute the flow using the procedure compute-flows. The flow $x^*$ denotes the changes in flows on spanning tree arcs as $\lambda$ changes by one unit (while flow on nontree arcs remain fixed at their respective bounds). Therefore, we compute $x^*$ by using the procedure compute-flows and assuming that all nontree arcs are at their lower bounds. Since $x^0$ satisfies $Ax^0 = b^0$, $0 \leq x^0 \leq u$, and $x^*$ satisfies $Ax^* = b^*$, $x^* \geq 0$, it follows that $A(x^0 + \lambda x^*) = (b^0 + \lambda b^*)$, $0 \leq (x^0 + \lambda x^*) \leq u$. This establishes the first part of the exercise. Let $\lambda_{ij}$ denote the largest value of $\lambda$ permitted by the flow bound constraint of arc $(i, j)$. It follows from $0 \leq (x^0_{ij} + \lambda x^*_{ij}) \leq u_{ij}$, that

$$\lambda_{ij} = \begin{cases} \frac{(u_{ij} - x^0_{ij})}{x^*_{ij}} & \text{if } x^*_{ij} > 0 \\ \frac{-x^0_{ij}}{x^*_{ij}} & \text{if } x^*_{ij} < 0 \\ \infty & \text{otherwise} \end{cases}$$

Then, clearly, $\lambda^* = \min \{\lambda_{ij} : (i, j)\in T\}$.

(b) At $\lambda = \lambda^*$, one of the tree arc, say $(p, q)$, reaches its lower bound or upper bound. We perform a dual pivot with arc $(p, q)$ as the leaving arc. Recall the discussion in Section 11.9 that we choose the entering arc as an arc with the minimum value of $\theta_{ij}$ among the eligible arcs, due to which the new spanning tree structure satisfies the optimality conditions. Further, in this dual pivot, no flow is augmented in the cycle created, so the arc flows remains feasible. Consequently, the new spanning tree structure is also optimal.

(c) The solution is similar to that in Exercise 11.48(c).

EXERCISE 11.51. (a) In the network shown in Figure S11.51, the optimal flow is obtained by augmenting as much flow as possible along the uncapacitated negative cycle 1-2-1 subject to the budget constraint. When $D = 1$, this leads to the solution $x_{12} = x_{21} = 1/2$, which is nonintegral.

(b) Linear Programming Argument. The constraint matrix of the constrained minimum cost flow problem contains the mass balance constraints and the budget constraint. The basic variables for the mass balance constraints define a spanning tree. The basis will contain an additional basic variable corresponding to the budget constraint. This additional basic variable will be a nontree arc and adding it to the spanning tree yields an augmented tree solution. Since linear programming problems always possess an optimal basic feasible solution, the constrained minimum cost flow problem will have an augmented tree solution.

Combinatorial Argument. We will show that if $n$ is an optimal solution of the constrained minimum cost flow problem and is not an augmented tree solution, then we can make it an augmented tree solution using a combinatorial algorithm. Define the free and restricted arcs as in Section 11.2. If the set of free arcs contains a unique cycle, then it is easy to construct an augmented tree solution, similar to the method described in Section
11.2. However, if there are two or more cycles, then we need to do some additional work. Suppose that the set of free arcs contains two cycles $W_1$ and $W_2$. Define arbitrary orientations for the two cycles. Recall that the flow on these two cycles can be sent in any direction without violating the flow bounds. Suppose that if we send unit flow along $W_1$ (or $W_2$), then its impact on $\sum_{(i,j) \in A} d_{ij} x_{ij}$ is $d(W_1)$ (or $d(W_2)$). Define $\theta$ so that $d(W_1) + \theta d(W_2) = 0$. Observe that if we send $y$ units of flow along $W_1$ and $\theta y$ units of flow along $W_2$, then it has no impact on $\sum_{(i,j) \in A} d_{ij} x_{ij}$ and the budget constraint remains satisfied. Notice that these two augmentations do not decrease the cost of flow (i.e., $\sum_{(i,j) \in A} c_{ij} x_{ij}$) because then it would contradict the optimality of the flow $x^*$. Also notice that these flow augmentations cannot increase the cost of flow too, because then by seeing flow in the opposite direction, we would contradict the optimality of $x^*$. Thus, we conclude that these two augmentations preserve the cost of flow. We next determine the flow on arcs in $W_1 \cup W_2$ as a function of $y$, and compute the smallest value of $y$, when one of the arcs in $W_1 \cup W_2$ reaches its lower or upper bound (this arc now ceases to be a free arc). We repeat this process until free arcs contain a unique cycle. Then it is easy to construct an augmented tree solution.