Network Simplex Method

Combinatorial Problem Solving (CPS)

Enric Rodríguez-Carbonell

April 26, 2019
Network Programs

- A network program is of the form

\[
\begin{align*}
\text{min} & \quad c^T x \\
Ax &= b \\
\ell &\leq x \leq u,
\end{align*}
\]

where \( c \in \mathbb{R}^m \), \( b \in \mathbb{R}^n \) and \( A \in \{-1, 0, 1\}^{n \times m} \) has the following property:

- each column has exactly one 1 and one \(-1\)
  (and so the remaining coefficients are 0)

- Note that \( n \) is the number of constraints and \( m \) is the number of variables
A network program is of the form

$$\begin{align*}
\min & \quad c^T x \\
Ax & = b \\
\ell \leq x \leq u,
\end{align*}$$

where $c \in \mathbb{R}^m$, $b \in \mathbb{R}^n$ and $A \in \{-1, 0, 1\}^{n \times m}$ has the following property:

- each column has exactly one 1 and one $-1$
- (and so the remaining coefficients are 0)

Example:

$$\begin{align*}
\min & \quad x_1 + x_2 + 3x_3 + 10x_4 \\
\begin{pmatrix}
1 & 0 & 1 & 1 \\
-1 & 1 & 0 & 0 \\
0 & -1 & -1 & -1
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4
\end{pmatrix}
= 
\begin{pmatrix}
5 \\
0 \\
-5
\end{pmatrix}
\end{align*}$$

$$\begin{align*}
0 \leq x_1 \leq 4 & \quad 0 \leq x_3 \leq 4 \\
0 \leq x_2 \leq 2 & \quad 0 \leq x_4 \leq 10
\end{align*}$$
**Minimum Cost Flow Problems**

- Network programs can be seen as **minimum cost flow problems** in a graph.
- We associate a digraph $G = (V, E)$ to the matrix of a network program:
  - Vertices $V$ correspond to rows (constraints).
  - Edges $E$ correspond to columns (variables).
  - A column with a $1$ at row $i$ and a $-1$ at row $k$ gives an edge $(i, k)$. 
Network programs can be seen as **minimum cost flow problems** in a graph.

We associate a digraph $G = (V, E)$ to the matrix of a network program:

- Vertices $V$ correspond to *rows* (constraints).
- Edges $E$ correspond to *columns* (variables).
- A column with a 1 at row $i$ and a $-1$ at row $k$ gives an edge $(i, k)$.

Then we can reinterpret the other elements of the network program:

- Each variable $x_j$ is the flow sent along the $j$-th edge.
- The cost of sending 1 unit of flow is $c_j$.
- Flow cannot exceed capacity $u_j$.
- There must be a minimum flow $\ell_j$ (usually, 0).
- Total production of flow at vertex $i$ is determined by $b_i$.

So solving the network program consists in finding the feasible flow along the graph that minimizes the cost.
Minimum Cost Flow Problems

\[
\begin{align*}
\min & \quad x_1 + x_2 + 3x_3 + 10x_4 \\
\text{subject to} & \quad 
\begin{pmatrix}
1 & 0 & 1 & 1 \\
-1 & 1 & 0 & 0 \\
0 & -1 & -1 & -1
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4
\end{pmatrix}
= 
\begin{pmatrix}
5 \\
0 \\
-5
\end{pmatrix}
\end{align*}
\]

\[
0 \leq x_1 \leq 4 \\
0 \leq x_2 \leq 2 \\
0 \leq x_3 \leq 4 \\
0 \leq x_4 \leq 10
\]
Network Simplex Method

- Network programs satisfy Hoffman & Gale’s conditions. So simplex method is guaranteed to give integer solutions (if $\ell, u, b$ in $\mathbb{Z}$)

- Moreover we can specialize the simplex method for network programs

- This lecture is devoted to this specialization: the network simplex method

- In the first place we need to revisit a bit of graph theory
Vertex-Edge Incidence Matrix

- The vertex-edge incidence matrix of digraph $G = (V, E)$ is a matrix $A$ s.t.:
  - Rows are labelled by vertices
  - Columns are labelled by edges
  - For each $v \in V$ and $e \in E$, coefficient $a_{v,e}$ of $A$ is
    - $1$ if $e = (v, \cdot)$
    - $-1$ if $e = (\cdot, v)$
    - $0$ otherwise

- Given a network program whose matrix is $A$, the vertex-edge incidence matrix of its associated digraph is precisely $A$
Vertex-Edge Incidence Matrix

\[
\begin{pmatrix}
1 & 0 & 1 & 1 \\
-1 & 1 & 0 & 0 \\
0 & -1 & -1 & -1
\end{pmatrix}
\]
Paths and Cycles

- A **path** is a finite sequence \( P = (v_1, e_1, v_2, \ldots, v_K, e_K, v_{K+1}) \) such that either \( e_k = (v_k, v_{k+1}) \) or \( e_k = (v_{k+1}, v_k) \) for all \( 1 \leq k \leq K \).

- Note that paths can invert the orientation of edges.

- The **orientation sequence** of a path \( P \) is \( (O_P(e_1), \ldots, O_P(e_k)) \), where

\[
O_P(e_k) = \begin{cases} 
+1 & \text{if } e_k = (v_k, v_{k+1}) \\
-1 & \text{if } e_k = (v_{k+1}, v_k) \\
0 & \text{otherwise}
\end{cases}
\]

- A **cycle** is a path such that the initial and the final vertices are the same.
(3, 4, 1, 1, 2) is a path with orientation sequence (−1, 1)
Prop. Let $P = (v_1, e_1, v_2, \ldots, v_K, e_K, v_{K+1})$ be a path. Then
\[
\sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = e_{v_1} - e_{v_{K+1}},
\]
where $a_e$ is the column of $e$ in the vertex-edge incidence matrix $A$, and $e_v$ is the $v$-th unit vector, i.e., all zeroes except for a 1 at index $v$. 

Paths and Cycles
**Paths and Cycles**

**Prop.** Let $P = (v_1, e_1, v_2, \ldots, v_K, e_K, v_{K+1})$ be a path. Then

$$\sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = e_{v_1} - e_{v_{K+1}},$$

where $a_e$ is the column of $e$ in the vertex-edge incidence matrix $A$, and $e_v$ is the $v$-th unit vector, i.e., all zeroes except for a 1 at index $v$.

**Proof.** Let $k$ be s.t. $1 \leq k \leq K$. There are two cases:

1. If $e_k = (v_k, v_{k+1})$ then $a_{e_k} = e_{v_k} - e_{v_{k+1}}$ and $O_P(e_k) = 1$
2. If $e_k = (v_{k+1}, v_k)$ then $a_{e_k} = e_{v_{k+1}} - e_{v_k}$ and $O_P(e_k) = -1$

In any case $O_P(e_k) \cdot a_{e_k} = e_{v_k} - e_{v_{k+1}}$. So

$$\sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = (e_{v_1} - e_{v_2}) + (e_{v_2} - e_{v_3}) + \ldots + (e_{v_K} - e_{v_{K+1}}) = e_{v_1} - e_{v_{K+1}}$$
Paths and Cycles

\[ \text{Prop.} \text{ Let } P = (v_1, e_1, v_2, \ldots, v_K, e_K, v_{K+1}) \text{ be a path. Then} \]

\[ \sum_{k=1}^{K} OP(e_k) \cdot a_{e_k} = e_{v_1} - e_{v_{K+1}}, \]

where \( a_e \) is the column of \( e \) in the vertex-edge incidence matrix \( A \), and \( e_v \) is the \( v \)-th unit vector, i.e., all zeroes except for a 1 at index \( v \).

\[ \text{Cor.} \text{ If } C = (v_1, e_1, v_2, \ldots, v_K, e_K, v_{K+1}) \text{ is a cycle, the columns } a_{e_1}, a_{e_2}, \ldots, a_{e_K} \text{ of } A \text{ are linearly dependent.} \]
Prop. Let $P = (v_1, e_1, v_2, \ldots, v_K, e_K, v_{K+1})$ be a path. Then

$$
\sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = e_{v_1} - e_{v_{K+1}},
$$

where $a_e$ is the column of $e$ in the vertex-edge incidence matrix $A$, and $e_v$ is the $v$-th unit vector, i.e., all zeroes except for a 1 at index $v$.

Cor. If $C = (v_1, e_1, v_2, \ldots, v_K, e_K, v_{K+1})$ is a cycle, the columns $a_{e_1}, a_{e_2}, \ldots, a_{e_K}$ of $A$ are linearly dependent.

Proof. If $v_1 = v_{K+1}$ then

$$
\sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = e_{v_1} - e_{v_{K+1}} = 0
$$
Path $P = (3, 4, 1, 1, 2)$ has orientation sequence $(-1, 1)$

$$
\sum_{k=1}^{K} OP(e_k) \cdot a_{e_k} = (-1) \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix} + (1) \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix} = e_3 - e_2
$$
Trees

- A graph is
  - **acyclic** if it has no cycles
  - **connected** if for any pair of vertices \( u, v \) there is a path from \( u \) to \( v \)
  - a **tree** if it is acyclic and connected

- **Thm.** For a graph \( T \) with at least one vertex the following are equivalent:
  - \( T \) is a tree
  - For any pair of vertices \( u, v \) there is a **unique** path from \( u \) to \( v \)
  - \( T \) has one less edge than vertices and is connected
  - \( T \) has one less edge than vertices and is acyclic

- A subgraph \( S \) of \( G \) is **spanning** if it covers all vertices in \( G \)

- **Thm.** Every connected graph has a subgraph that is a spanning tree.
Trees

**Thm.** For any $T$ subgraph of $G$ that is a tree with at least two vertices, the columns $\{a_e \mid e \in T\}$ of $A$ are linearly independent.
Thm. For any $T$ subgraph of $G$ that is a tree with at least two vertices, the columns $\{a_e \mid e \in T\}$ of $A$ are linearly independent.

Proof. By contradiction.

Let $T$ be a tree with the minimum number of vertices $N$ such that $\{a_e \mid e \in T\}$ are linearly dependent, i.e., there are $\lambda_e$ not all null s.t.

$$\sum_{e \in T} \lambda_e a_e = 0$$

If $N = 2$ then $T$ would have one edge, say $e$, and $a_e \neq 0$.

So $N > 2$. Let $v$ be a leaf of $T$ and let $e_v$ be the only edge in $T$ that has $v$ as an endpoint. Let $T'$ be the tree obtained from $T$ by removing $e_v$.

From

$$\lambda_{e_v} a_{e_v} + \sum_{e \in T, e \neq e_v} \lambda_e a_e = 0$$

by projecting onto the row of $v$ we have $\lambda_{e_v} = 0$.

Hence the tree $T'$ is a subgraph of $G$ with $N - 1 \geq 2$ vertices whose columns are linearly dependent. Contradiction!
Edges \{4, 1\} induce a subgraph that is a tree, and

\[
\begin{pmatrix}
1 & 1 \\
0 & -1 \\
-1 & 0
\end{pmatrix}
\]

\[
= 2
\]
Thm. If $G$ is a connected graph with $n > 0$ nodes then $\text{rank}(A) = n - 1$
Thm. If $G$ is a connected graph with $n > 0$ nodes then $\text{rank}(A) = n - 1$

Proof. $G$ has a spanning tree $T$, which has $n - 1$ edges.
Its columns are linearly independent, so $\text{rank}(A) \geq n - 1$.
But since adding all rows of $A$ we get $0$, finally $\text{rank}(A) = n - 1$. 
Reformulating Network Programs

- **Thm.** If $G$ is a connected graph with $n > 0$ nodes then $\text{rank}(A) = n - 1$

- Let us assume graphs of network programs are connected, so $m \geq n - 1$ (otherwise, work independently on the connected components)

- So the matrix of a network program has rank $n - 1$. But the simplex method requires to have a full-rank matrix!

- We add an extra variable $w$ with a unit column $e_r$, where $r$ is taken arbitrarily from $\{1, \ldots, n\}$, and such that it is forced to have value $0$:

  \[
  \begin{align*}
  \min & \quad c^T x \\
  \text{subject to} & \quad Ax + e_r w = b \\
  & \quad \ell \leq x \leq u, \\
  & \quad 0 \leq w \leq 0
  \end{align*}
  \]
Thm. If $G$ is a connected graph with $n > 0$ nodes then $\text{rank}(A) = n - 1$

Let us assume graphs of network programs are connected, so $m \geq n - 1$ (otherwise, work independently on the connected components)

So the matrix of a network program has rank $n - 1$. But the simplex method requires to have a full-rank matrix!

We add an extra variable $w$ with a unit column $e_r$, where $r$ is taken arbitrarily from $\{1, \ldots, n\}$, and such that it is forced to have value 0:

$$\begin{align*}
\min \ c^T x \\
Ax + e_r w &= b \\
\ell \leq x \leq u, \\
0 \leq w \leq 0
\end{align*}$$

We associate to such a reformulated network program a rooted graph with root vertex $r$ and root edge $w$ ("going nowhere")
Reformulating Network Programs

Here we choose as a root vertex $r = 2$

\[
\min \ x_1 + x_2 + 3x_3 + 10x_4
\]

\[
\begin{pmatrix}
  1 & 0 & 1 & 1 & 0 \\
  -1 & 1 & 0 & 0 & 1 \\
  0 & -1 & -1 & -1 & 0
\end{pmatrix}
\begin{pmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  x_4 \\
  w
\end{pmatrix} =
\begin{pmatrix}
  5 \\
  0 \\
  -5
\end{pmatrix}
\]

0 ≤ $x_1$ ≤ 4
0 ≤ $x_2$ ≤ 2
0 ≤ $x_3$ ≤ 4
0 ≤ $x_4$ ≤ 10
0 ≤ $w$ ≤ 0
Characterization of Bases

Thm. Let $A$ be the matrix of a rooted graph $G$ with root vertex $r$.
If $T$ is a spanning tree for $G$ then $B = e_r \cup \{a_e \mid e \in T\}$ is basis of $(A \mid e_r)$.
Characterization of Bases

**Thm.** Let $A$ be the matrix of a rooted graph $G$ with root vertex $r$. If $T$ is a spanning tree for $G$ then $B = e_r \cup \{a_e | e \in T\}$ is basis of $(A | e_r)$

**Proof.** Let $n$ be the number of vertices of $G$. As $T$ is a spanning tree, $T$ has $n - 1$ edges. Hence $B = e_r \cup \{a_e | e \in T\}$ has $n$ columns.

Let us prove that $B$ spans $\mathbb{R}^n$, i.e., that for any $1 \leq i \leq n$ we can write $e_i$ as linear combination of columns of $B$.

Two cases:

- If $i = r$: trivial
- If $i \neq r$, let $P = (v_1 = i, e_1, v_2, \ldots, v_K, e_K, v_{K+1} = r)$ be a path in $T$ from vertex $i$ to vertex $r$. As

  \[ \sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = e_i - e_r \]

  we have

  \[ e_r + \sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = e_i \]

  Altogether $B$ is a basis for $(A | e_r)$.
Characterization of Bases

**Thm.** Let $A$ be the matrix of a rooted graph $G$ with root vertex $r$. If $T$ is a spanning tree for $G$ then $B = e_r \cup \{a_e \mid e \in T\}$ is basis of $(A \mid e_r)$.

*Proof.* Let $n$ be the number of vertices of $G$. As $T$ is a spanning tree, $T$ has $n - 1$ edges. Hence $B = e_r \cup \{a_e \mid e \in T\}$ has $n$ columns.

Let us prove that $B$ spans $\mathbb{R}^n$, i.e., that for any $1 \leq i \leq n$ we can write $e_i$ as linear combination of columns of $B$.

Two cases:

- **If** $i = r$: trivial
- **If** $i \neq r$, let $P = (v_1 = i, e_1, v_2, \ldots, v_K, e_K, v_{K+1} = r)$ be a path in $T$ from vertex $i$ to vertex $r$. As

$$\sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = e_i - e_r$$

we have

$$e_r + \sum_{k=1}^{K} O_P(e_k) \cdot a_{e_k} = e_i$$

Altogether $B$ is a basis for $(A \mid e_r)$.

**Cor.** $\text{rank}(A \mid e_r) = n$
Characterization of Bases

- **Thm.** Let $A$ be the matrix of a rooted graph $G$ with root vertex $r$. If $B$ is basis of $(A | e_r)$ then $e_r \in B$ and ${e \mid a_e \in B}$ is spanning tree of $G$.
Thm. Let $A$ be the matrix of a rooted graph $G$ with root vertex $r$. If $B$ is basis of $(A|e_r)$ then $e_r \in B$ and $\{e|a_e \in B\}$ is spanning tree of $G$.

Proof. Let $n$ be the number of vertices of $G$ as usual.

Since \( \text{rank}(A) = n - 1 \) and \( \text{rank}(A|e_r) = n \) we have that $e_r \in B$. So the graph $T$ induced by $\{e|a_e \in B\}$ has $n - 1$ edges.

Moreover, by linear independence, $T$ cannot contain cycles. Hence $T$ has at least $(n - 1) + 1 = n$ vertices. But $G$ has $n$ vertices. Thus $T$ has exactly $n$ vertices, and so is spanning.

Since $T$ has one less edge than vertex and is acyclic, it must be a tree.

All in all, $T$ is a spanning tree.
Characterization of Bases

\[
B = \begin{pmatrix}
1 & 1 & 0 \\
-1 & 0 & 1 \\
0 & -1 & 0
\end{pmatrix}
\]
Characterization of Bases

\[ B = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix} \]
Specializing the Simplex Method

■ Where do we use the basis inverse in the simplex method?
Specializing the Simplex Method

1. Initialization: Find an initial feasible basis $B$
   Compute $B^{-1}, \beta = B^{-1}b, z = c_T^B\beta$

2. Pricing: Compute $\pi^T = c_B^T B^{-1}$ and $d_j = c_j - \pi^T a_j$.
   If for all $j \in R, d_j \geq 0$ then return OPTIMAL
   Else let $q$ be such that $d_q < 0$. Compute $\alpha_q = B^{-1}a_q$

3. Ratio test: Compute $I = \{i \mid 1 \leq i \leq m, \alpha_i^q > 0\}$.
   If $I = \emptyset$ then return UNBOUNDED
   Else compute $\theta = \min_{i \in I} (\frac{\beta_i}{\alpha_i^q})$ and $p$ such that $\theta = \frac{\beta_p}{\alpha_p^q}$

4. Update:
   \[
   \bar{B} = B - \{k_p\} \cup \{q\} \\
   \bar{\beta}_p = \theta, \quad \bar{\beta}_i = \beta_i - \theta \alpha_i^q \quad \text{if} \quad i \neq p \\
   \bar{B} = B + (a_q - a_{k_p})e_p^T \\
   \bar{z} = z + \theta d_q
   \]
   Go to 2.
Specializing the Simplex Method

- Where do we use the basis inverse in the simplex method?
  - In pricing: we compute the multipliers  \( \pi^T = c^T_B B^{-1} \)
  - In ratio test: we compute the \( q \)-th column of the tableau  \( \alpha_q = B^{-1} a_q \)
  - In initialization: we compute the initial basic solution  \( \beta = B^{-1} b \)

- Equivalently:
  - In pricing: we solve the equation  \( y^T B = c^T_B \) (and then set  \( \pi = y \) )
  - In ratio test: we solve the equation  \( Bx = a_q \) (and then set  \( \alpha_q = x \) )
  - In initialization: we solve the equation  \( Bx = b \) (and then set  \( \beta = x \) )

- These equations can be efficiently solved with the graph representation
- So the network simplex method doesn’t require to maintain basis inverses
Solving $y^T B = c^T$

- Let $A$ be the matrix of a rooted graph $G$ with root vertex $r$. Let $B$ be a basis for $(A | e_r)$.
- We know that $e_r \in B$ and $T = \{e \mid a_e \in B\}$ is a spanning tree for $G$.

- In the system of equations $y^T B = c^T$:
  - each column (= edge) of $B$ corresponds to one equation
  - each row (= vertex) of $B$ corresponds to one variable

- Each equation either involves 1 variable (column $e_r$) or 2 (otherwise)
Solving $y^T B = c^T$

\[ B = \begin{pmatrix}
1 & 4 & 5 \\
1 & 1 & 0 \\
-1 & 0 & 1 \\
0 & -1 & 0 \\
\end{pmatrix} \begin{pmatrix}
1 \\
2 \\
3 \\
\end{pmatrix} \]
Let us solve $y^T B = c^T$, where $y^T = (y_1 \ y_2 \ y_3)$ and $c^T = (1 \ 10 \ 0)$

$$
\begin{pmatrix}
    y_1 & y_2 & y_3
\end{pmatrix}
\begin{pmatrix}
    1 & 1 & 0 \\
    -1 & 0 & 1 \\
    0 & -1 & 0
\end{pmatrix}
= 
\begin{pmatrix}
    y_1 - y_2 & y_1 - y_3 & y_2
\end{pmatrix}
$$

$$
\begin{cases}
    y_1 - y_2 = 1 \implies 1 \\
    y_1 - y_3 = 10 \implies 4 \\
    y_2 = 0 \implies 5
\end{cases}
$$

Note that by doing a preorder traversal from root node 2 we can solve the equations.
Let us solve $y^T B = c^T$, where $y^T = (y_1, y_2, y_3)$ and $c^T = (1, 10, 0)$.

\[
\begin{pmatrix} y_1 & y_2 & y_3 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} = \begin{pmatrix} y_1 - y_2 & y_1 - y_3 & y_2 \end{pmatrix}
\]

\[
\begin{cases}
y_1 - y_2 &= 1 \implies y_1 = y_2 + 1 = 1 \\
y_1 - y_3 &= 10 \implies y_3 = y_1 - 10 = -9 \\
y_2 &= 0 \implies y_2 = 0
\end{cases}
\]
Solving  $y^T B = c^T$

- Let us take the root vertex $r$ as the root of $T$. Let $w$ be the root edge.
- To solve $y^T B = c^T$ call $solve(\bot, T)$, where

$$solve(\text{Vertex } p, \text{ Tree } S) \{ \hspace{1em} \text{// } p \text{ is the parent of the root of } S \\
\text{Vertex } v = \text{root}(S);
\text{if } (v == r) \hspace{0.5em} y[r] = c[w];
\text{else if } ((p, v) \in E) \hspace{0.5em} y[v] = y[p] - c[(p, v)];
\text{else} \hspace{0.5em} y[v] = y[p] + c[(v, p)];
\text{solve}(v, S. \text{ left }());
\text{solve}(v, S. \text{ right }()); \}$$

It is a preorder traversal of $T$.

At each recursive call (except 1st one) we handle a new equation ($= column = edge$) with 2 vars $y_p$ and $y_v$ in which one is already assigned ($y_p$) and the other is not ($y_v$).
Solving $y^T B = c^T$

- Let us take the root vertex $r$ as the root of $T$. Let $w$ be the root edge.
- To solve $y^T B = c^T$ call `solve(⊥, T)`, where

  ```
  solve(Vertex p, Tree S) { // p is the parent of the root of S
    Vertex v = root(S);
    if (v == r) y[r] = c[w];
    else y[v] = y[p] + c[(v, p)];
    solve(v, S.left ());
    solve(v, S.right ());
  }
  ```

If $v = r$ then the equation is $y^T e_r = c_w$, i.e., $y_r = c_w$. 
Solving $y^T B = c^T$

- Let us take the root vertex $r$ as the root of $T$. Let $w$ be the root edge.
- To solve $y^T B = c^T$ call $solve(\bot, T)$, where

$$solve(\text{Vertex } p, \text{ Tree } S) \{ \text{ // } p \text{ is the parent of the root of } S$$

$$\text{Vertex } v = \text{root}(S);$$

if $(v == r)$ $y[r] = c[w];$
else if $((p, v) \in E)$ $y[v] = y[p] - c[(p, v)];$
else
$$y[v] = y[p] + c[(v, p)];$$
$$solve(v, S.\text{left}());$$
$$solve(v, S.\text{right}());$$

\}

If $e = (p, v) \in E$ then the equation is

$$y^T (e_p - e_v) = y_p - y_v = c_e,$$

i.e., $y_v = y_p - c_e$. 

![Diagram](image-url)
Solving \( y^T B = c^T \)

- Let us take the root vertex \( r \) as the root of \( T \). Let \( w \) be the root edge.
- To solve \( y^T B = c^T \) call \( solve(\bot, T) \), where

\[
\text{solve}(\text{Vertex } p, \text{ Tree } S) \begin{cases}
// p \text{ is the parent of the root of } S \\
\text{Vertex } v = \text{root}(S); \\
\text{if } (v == r) \ y[r] = c[w]; \\
\text{else if } ((p, v) \in E) \ y[v] = y[p] - c[(p, v)]; \\
\text{else} \ y[v] = y[p] + c[(v, p)]; \\
\text{solve}(v, S.\text{left}()); \\
\text{solve}(v, S.\text{right}()); \\
\end{cases}
\]

If \( e = (v, p) \in E \) then the equation is

\[
y^T (e_v - e_p) = y_v - y_p = c_e,
\]

i.e., \( y_v = y_p + c_e \).
Solving $Bx = c$

■ Let $A$ be the matrix of a rooted graph $G$ with root vertex $r$.

Let $B$ be a basis for $(A|e_r)$.

■ We know that $e_r \in B$ and $T = \{ e \mid a_e \in B \}$ is a spanning tree for $G$.

■ For any $1 \leq i \leq n$ there is a path $P_i$ from $i$ to $r$, i.e., $P_i = (v_1 = i, e_1, \ldots, e_K, v_{K+1} = r)$ in $T$. But recall that

$$e_i = e_r + \sum_{k=1}^{K} O_{P_i}(e_k) \cdot a_{e_k}$$

■ Let us assume $B$ is of the form $(a_{k_1}, a_{k_2}, \ldots, a_{k_{n-1}}, e_r)$. Then

$$e_i = e_r + \sum_{j=1}^{n-1} O_{P_i}(k_j) \cdot a_{k_j}$$

as edges $k_j$ not in $P_i$ will have a 0 coefficient by definition of $O_{P_i}$. So

$$c = \sum_{i=1}^{n} c_i e_i = (\sum_{i=1}^{n} c_i) \cdot e_r + \sum_{j=1}^{n-1} (\sum_{i=1}^{n} c_i O_{P_i}(k_j)) \cdot a_{k_j}$$

Let $x_n = \sum_{i=1}^{n} c_i$, $x_j = \sum_{i=1}^{n} c_i O_{P_i}(k_j)$ for $1 \leq j < n$. Then $Bx = c$!

■ Solving $Bx = c$ amounts to traverse $T$ keeping track of edge orientation
Solving $Bx = c$

$B = \begin{pmatrix}
1 & 1 & 0 \\
-1 & 0 & 1 \\
0 & -1 & 0 \\
\end{pmatrix}$

1 4 5
Solving $Bx = c$

Let us solve $Bx = c$, where $x^T = (x_1 \ x_4 \ x_5)^T$, and $c^T = (c_1 \ c_2 \ c_3)^T = (0 \ 1 \ -1)^T = e_2^T - e_3^T$

There is no need to consider the path $P_1$ from 1 to 2, as $c_1 = 0$. Moreover $P_2 = (2)$, and hence $O_{P_3} (\cdot) = 0$.

Path from 3 to 2: $P_3 = (3, 4, 1, 1, 2)$ with orientation sequence $(-1, 1)$.

- $x_1 = c_3 \cdot O_{P_3} (1) = (-1) \cdot 1 = -1$
- $x_4 = c_3 \cdot O_{P_3} (4) = (-1) \cdot (-1) = 1$
- $x_5 = c_1 + c_2 + c_3 = 0 + 1 + (-1) = 0$
Solving $Bx = c$

- Let $A$ be the matrix of rooted graph $G$ with root vertex $r$.
  Let $B$ be a basis for $(A|e_r)$.

- We know that $e_r \in B$ and $T = \{e | a_e \in B\}$ is a spanning tree for $G$.

- In the ratio test, $c$ will be one of the columns of $A$.

- If $c$ is of the form $e_i - e_j$,
  let $P$ be the path in $T$ going from vertex $i$ to vertex $j$.
  Then recall that
  \[
  \sum_{e \in P} O_P(e) \cdot a_e = e_i - e_j
  \]

- Hence the orientation sequence gives us already the solution.
Solving $Bx = c$

$$B = \begin{pmatrix} 1 & 4 & 5 \\ -1 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}$$

$B = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}$
Solving $Bx = c$

Let us solve $Bx = c$, where $x^T = (x_1, x_4, x_5)$, and $c^T = (c_1, c_2, c_3) = (0, 1, -1) = e_2^T - e_3^T$.

Path from 2 to 3: $P_3 = (2, 1, 1, 4, 3)$ with orientation sequence $(-1, 1)$. So:

- $x_1 = -1$
- $x_4 = 1$
- $x_5 = 0$
Example

Let us apply one iteration of the simplex method to

\[ \min \ x_1 + x_2 + 3x_3 + 10x_4 \]

\[
\begin{pmatrix}
1 & 0 & 1 & 1 & 0 \\
\-1 & 1 & 0 & 0 & 1 \\
0 & \-1 & \-1 & \-1 & 0
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5
\end{pmatrix}
= \begin{pmatrix}
5 \\
0 \\
\-5
\end{pmatrix}
\]

\[
0 \leq x_1 \leq 4 \\
0 \leq x_2 \leq 2 \\
0 \leq x_3 \leq 4 \\
0 \leq x_4 \leq 10 \\
0 \leq x_5 \leq 0
\]
Example

Let us consider the basis $B$ corresponding to variables $(x_1, x_4, x_5)$

Moreover, let us assume that:

- non-basic variable $x_2$ is set to its lower bound $0$
- non-basic variable $x_3$ is set to its upper bound $4$
Example

Let us compute the initial basic solution: $x_B = B^{-1}b - B^{-1}R x_R$

So $x_B = B^{-1}(5e_1 - 5e_3) - B^{-1}a_2 0 - B^{-1}a_3 4 = 5B^{-1}(e_1 - e_3) - 4B^{-1}a_3$

$= B^{-1}(e_1 - e_3)$

The path from 1 to 3 is $P = (1, x_4, 3)$ with orientation sequence (1)

So the only non-zero value for a basic variable is for $x_4$, with value 1

Hence the basis is feasible and its solution is $(x_1, x_2, x_3, x_4, x_5) = (0, 0, 4, 1, 0)$
Example

- \( x_2 \): lower bound 0
- \( x_3 \): upper bound 4

Let us do the pricing, i.e., compute
\[
d_j = c_j - c_T^T B^{-1} a_j = c_j - \pi^T a_j
\]
for each non-basic variable \( x_j \)

The solution to \( \pi^T B = c_B^T \) is \((\pi_1, \pi_2, \pi_3) = (1, 0, -9)\), and so:

- for \( x_2 \): \( d_2 = c_2 - \pi^T (e_2 - e_3) = c_2 - \pi_2 + \pi_3 = -8 \)
- for \( x_3 \): \( d_3 = c_3 - \pi^T (e_1 - e_3) = c_3 - \pi_1 + \pi_3 = -7 \)

Only variable \( x_2 \) is candidate for entering the basis
Example

\[ x_2: \text{ lower bound } 0 \]
\[ x_3: \text{ upper bound } 4 \]
\[ (x_1, x_2, x_3, x_4, x_5) = (0, 0, 4, 1, 0) \]

Let us do the ratio test.
We need to compute \( \alpha_2 = B^{-1}a_2 \), and we get \( \alpha_2^T = (-1, 1, 0) \). Then
\[
\theta = \min(u_q - \ell_q, \min\{\frac{\beta_i - \lambda_i}{\alpha_q^i} \mid \alpha_q^i > 0\}, \min\{\frac{\beta_i - \mu_i}{\alpha_q^i} \mid \alpha_q^i < 0\})
\]
\[
= \min(2, \frac{1-0}{-1}, \frac{0-4}{-1}) = 1
\]
The outgoing basic variable is \( x_4 \).
Example

- Non-basic variable $x_2$ enters the basis
- Basic variable $x_4$ leaves the basis with value 0
- New basis $\bar{B}$ corresponds to $(x_1, x_2, x_5)$
- New basic solution: $\bar{\beta}_p = x_q + \theta$, $\bar{\beta}_i = \beta_i - \theta \alpha_q^i$ if $i \neq p$
  - $\bar{x}_2 = 0 + 1 = 1$
  - $\bar{x}_1 = 0 - 1(-1) = 1$
  - $\bar{x}_5 = 0 - 1(0) = 0$
- The basic solution for the new basis is $(\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4, \bar{x}_5) = (1, 1, 4, 0, 0)$
  And the process continues...