15.081J/6.251J Introduction to Mathematical Programming

Lecture 15: The Network Simplex Algorithm
Network Optimization

Why do we care?

- Networks and associated optimization problems constitute reoccurring structures in many real-world applications.
- The network structure often leads to additional insight and improved understanding.
- Given integer data, the standard models have integer optimal solutions.
- The network structure also enables us to design more efficient algorithms.
## Network Optimization

### A Comparison of Running Times

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Running Time (sec)</th>
<th># Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Simplex</td>
<td>334.59</td>
<td>42759</td>
</tr>
<tr>
<td>Network Simplex</td>
<td>7.37</td>
<td>23306</td>
</tr>
<tr>
<td>Ratio</td>
<td>2.2 %</td>
<td>54 %</td>
</tr>
</tbody>
</table>

Average over 5 random instances with 10,000 nodes and 25,000 arcs each.
The Simplex Algorithm: A Reminder

The Network Simplex Algorithm
A Reminder

The Problem...

\[
\begin{align*}
\min & \quad c'x \\
\text{s.t.} & \quad Ax = b \\
& \quad x \geq 0
\end{align*}
\]
The Simplex Algorithm

1. Start with basis $B = [A_{B(1)}, \ldots, A_{B(m)}]$ and BFS $x$.
2. Compute $\overline{c}_j = c_j - c'_B B^{-1} A_j$.
   - If $\overline{c}_j \geq 0$; $x$ optimal; stop.
   - Select $j$ such that $\overline{c}_j < 0$.
3. Compute $u = B^{-1} A_j$. $\theta^* = \min_{1 \leq i \leq m, u_i > 0} \frac{x_B(i)}{u_i} = \frac{x_B(\ell)}{u_\ell}$.
4. Form a new basis by replacing $A_{B(\ell)}$ with $A_j$.
5. $y_j = \theta^*$; $y_B(i) = x_B(i) - \theta^* u_i$. 

A Reminder

The Algorithm
The Network Simplex Algorithm

The Problem Statement...

Determine a least cost shipment of a commodity through a network in order to satisfy demands at certain nodes from available supplies at other nodes. Arcs have costs associated with them.
The Network Simplex Algorithm

- Network $G = (N, A)$.
- Arc costs $c : A \rightarrow \mathcal{R}$.
- Node balances $b : N \rightarrow \mathcal{R}$.

\[
\begin{align*}
\min & \quad \sum_{(i,j) \in A} c_{ij}x_{ij} \\
\text{s.t.} & \quad \sum_{j : (i,j) \in A} x_{ij} - \sum_{j : (j,i) \in A} x_{ji} = b_i \quad \text{for all } i \in N \\
& \quad x_{ij} \geq 0 \quad \text{for all } (i,j) \in A
\end{align*}
\]
A tree is a graph that is connected and has no cycles.

A spanning tree of a graph $G$ is a subgraph that is a tree and contains all nodes of $G$.

A flow $x$ forms a tree solution with a spanning tree of the network if every non-tree arc has flow 0.
Theorem 1 If the objective function is bounded from below, a min-cost flow problem always has an optimal tree solution.
• Bases and trees.
• Dual variables and node potentials.
• Changing bases and updating trees.
• Optimality testing.
The constraint matrix $A$ of the min-cost flow problem is the node-arc incidence matrix of the underlying network.

\[
\begin{pmatrix}
(1, 2) & (2, 6) & (3, 2) & (4, 3) & (4, 5) & (5, 3) & (5, 6) & (6, 7) & (7, 1) \\
1 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
2 & -1 & +1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & +1 & -1 & 0 & -1 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & +1 & +1 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & -1 & +1 & +1 & 0 & 0 \\
6 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & +1 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & +1
\end{pmatrix}
\]

The rows of $A$ are linearly dependent.
Let $B$ be the submatrix corresponding to the tree:

$$
\begin{array}{cccccc}
(1, 2) & (2, 6) & (3, 2) & (4, 3) & (5, 3) & (7, 1) \\
1 & +1 & 0 & 0 & 0 & 0 & -1 \\
2 & -1 & +1 & -1 & 0 & 0 & 0 \\
3 & 0 & 0 & +1 & -1 & -1 & 0 \\
4 & 0 & 0 & 0 & +1 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & +1 & 0 \\
6 & 0 & -1 & 0 & 0 & 0 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 & +1
\end{array}
$$
Let $B$ be the submatrix corresponding to the tree

\[\begin{align*}
(1,2) & (2,6) & (3,2) & (4,3) & (5,3) & (7,1) \\
4 & 0 & 0 & 0 & 0 & +1 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \\
6 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 & 0 & +1 \\
3 & 0 & 0 & +1 & -1 & -1 & 0 & 0 \\
2 & -1 & +1 & -1 & 0 & 0 & 0 & 0 \\
1 & +1 & 0 & 0 & 0 & 0 & 0 & -1 
\end{align*}\]
Let $B$ be the submatrix corresponding to the tree

\[
\begin{array}{cccccccc}
(4, 3) & (5, 3) & (2, 6) & (7, 1) & (3, 2) & (1, 2) \\
4 & +1 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & +1 & 0 & 0 & 0 & 0 & 0 \\
6 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
7 & 0 & 0 & 0 & +1 & 0 & 0 & 0 \\
3 & -1 & -1 & 0 & 0 & +1 & 0 & 0 \\
2 & 0 & 0 & +1 & 0 & -1 & -1 & 0 \\
1 & 0 & 0 & 0 & -1 & 0 & +1 & 0 \\
\end{array}
\]
Corollary 1

(a) *The matrix* $A$ *has rank* $n - 1$.
(b) *Every tree solution is a basic solution.*
Theorem 2 Every tree defines a basis and, conversely, every basis defines a tree.

Suppose the graph defined by a basis contains a cycle $1 - 2 - 3 - 4 - 5 - 6$:

\[
\begin{align*}
(1, 2) & (2, 3) (4, 3) (5, 4) (5, 6) (1, 6) \\
1 & +1 0 0 0 0 +1 \\
2 & -1 +1 0 0 0 0 \\
3 & 0 -1 -1 0 0 0 \\
4 & 0 0 +1 -1 0 0 \\
5 & 0 0 0 +1 +1 0 \\
6 & 0 0 0 0 -1 -1
\end{align*}
\]
Remember, the simplex algorithm computes the dual variables $p$ as the solution to $p' B = c'_B$.

$$
\begin{pmatrix}
+1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & +1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & +1 & 0 & 0 & 0 \\
-1 & -1 & 0 & 0 & +1 & 0 & 0 \\
0 & 0 & +1 & 0 & -1 & -1 & 0
\end{pmatrix}
$$

$$(p_4, p_5, p_6, p_7, p_3, p_2) = (c_{43}, c_{53}, c_{26}, c_{71}, c_{32}, c_{12})$$

Hence, $p_2 = -c_{12}$, $p_3 = c_{32} + p_2$, $p_7 = c_{71}$, ...
Remember, the simplex algorithm computes the reduced costs $\overline{c}$ as $\overline{c}_{ij} = c_{ij} - p'A_{ij}$.

<table>
<thead>
<tr>
<th></th>
<th>(1, 2)</th>
<th>(2, 6)</th>
<th>(3, 2)</th>
<th>(4, 3)</th>
<th>(4, 5)</th>
<th>(5, 3)</th>
<th>(5, 6)</th>
<th>(6, 7)</th>
<th>(7, 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>−1</td>
<td>+1</td>
<td>−1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>+1</td>
<td>−1</td>
<td>0</td>
<td>−1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+1</td>
<td>+1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>−1</td>
<td>+1</td>
<td>+1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>−1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>−1</td>
<td>+1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>−1</td>
</tr>
</tbody>
</table>

Therefore, $\overline{c}_{ij} = c_{ij} - p_i + p_j$. 
The Network Simplex Algorithm

**Theorem 3** A (feasible) tree $T$ is optimal if, for some choice of node potentials $p_i$,

(a) $\overline{c}_{ij} = c_{ij} - p_i + p_j = 0$ for all $(i, j) \in T$,

(b) $\overline{c}_{ij} = c_{ij} - p_i + p_j \geq 0$ for all $(i, j) \in A \setminus T$. 
The Network Simplex Algorithm

Overview of the Algorithm

1. Determine an initial feasible tree \( T \). Compute flow \( x \) and node potentials \( p \) associated with \( T \).

2. Calculate \( \bar{c}_{ij} = c_{ij} - p_i + p_j \) for \( (i, j) \notin T \).
   - If \( \bar{c} \geq 0 \), \( x \) optimal; stop.
   - Select \( (i, j) \) with \( \bar{c}_{ij} < 0 \).

3. Add \( (i, j) \) to \( T \) creating a unique cycle \( C \). Send a maximum flow around \( C \) while maintaining feasibility. Suppose the exiting arc is \( (k, \ell) \).

4. \( T := (T \setminus (k, \ell)) \cup (i, j) \).
Our reasoning has two important and far-reaching implications:

- There always exists an integer optimal flow (if node balances $b_i$ are integer).
- There always exist optimal integer node potentials (if arc costs $c_{ij}$ are integer).
The network simplex algorithm is extremely fast in practice.

Relying on network data structures, rather than matrix algebra, causes the speedups. It leads to simple rules for selecting the entering and exiting variables.

Running time per pivot:
- arcs scanned to identify an entering arc,
- arcs scanned of the basic cycle,
- nodes of the subtree.

A good pivot rule can dramatically reduce running time in practice.