# Lecture 14:

# The Network Simplex Method

AM&O Chapter 11

# The Network Simplex Method

**The Min Cost Flow LP:** Let network $G = (N, A)$ be given, with supplies/ demands $b_i$, $i \in N$, costs $c_{ij}$ (positive or negative), and capacities $u_{ij}$ (possibly $\infty$) $(i, j) \in A$. Recall the description of the Min Cost Flow Problem:

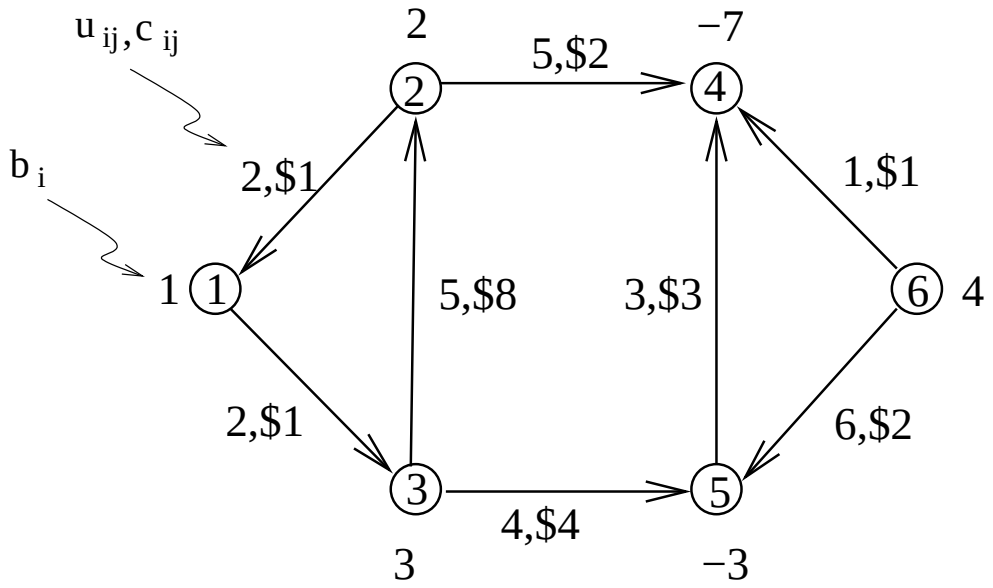
$$\min \; z(x) \; = \sum_{(i,j) \in A} c_{ij} x_{ij}$$

$$(NP) \quad \sum_{j \in \mathcal{A}(i)} x_{ij} - \sum_{j \in \mathcal{B}(i)} x_{ji} \; = \; b_i \;\; i \in N$$

$$0 \leq x_{ij} \leq u_{ij} \quad (i, j) \in A$$

Since this is a **linear program**, it can be solved using the **Simplex Method**. The **Network Simplex Method** is a special implementation of the Simplex Method which makes use of the network structure to significantly streamline the computational effort.

## Example:




## Network LP:

$$
\begin{array}{lrcrcrcrcrcrcrcrcl}
\min\ z = & x_{13} & + & 2x_{21} & + & 2x_{24} & + & 8x_{32} & + & 4x_{35} & + & 3x_{54} & + & x_{64} & + & 2x_{65} & \\
 & & & & & & & & & & & & & & & & \\
\text{Node 1:} & x_{13} & - & x_{21} & & & & & & & & & & & & = & 1 \\
\text{Node 2:} & & & x_{21} & + & x_{24} & - & x_{32} & & & & & & & & = & 2 \\
\text{Node 3:} & -x_{13} & & & & & + & x_{32} & + & x_{35} & & & & & & = & 3 \\
\text{Node 4:} & & & & & -x_{24} & & & & & - & x_{54} & - & x_{64} & & = & -7 \\
\text{Node 5:} & & & & & & & & & -x_{35} & + & x_{54} & & & - & x_{65} & = & -3 \\
\text{Node 6:} & & & & & & & & & & & & & x_{64} & + & x_{65} & = & 4 \\
 & & & & & & & & & & & & & & & & \\
 & x_{13} \le 2 & & x_{21} \le 2 & & x_{24} \le 5 & & x_{32} \le 5 & & x_{35} \le 4 & & x_{54} \le 3 & & x_{64} \le 1 & & x_{65} \le 6 &
\end{array}
$$

# Matrix-Vector Form of $(NP)$

$$\begin{array}{lrcl} \min & cx & & \\ & \mathcal{N}x & = & b \\ 0 & \leq x & \leq & u \end{array}$$

where $\mathcal{N}$ is the node-arc incidence matrix for $G$ and $c$, $b$, and $u$ are the vectors of costs, supplies/demands, and capacities, respectively.

**Assumptions:**

$(A1)$ $G$ is connected (in the undirected sense)

$(A2)$ $\sum_{i \in N} b_i = 0$

**Fact:** If the second assumption is not satisfied, the equality system $\mathcal{N}x = b$ is inconsistent. If it is satisfied, then at least one row is redundant and can be removed.

It will turn out that the choice of the row to remove is irrelevant, so we will remove the row corresponding to some arbitrary **root node** $s$. Let $\mathcal{N}_s$ be the resulting matrix.

# Spanning Trees

**Spanning tree:** Set of edges which connects every pair of nodes of $G$ and has no cycles (both in the *undirected* sense).

**Properties of Spanning Trees** (from first assignment):

($a$) every spanning tree has exactly $n-1$ arcs;

($b$) every spanning tree has at least two **end** nodes (nodes with only one adjacent arc);

($c$) every pair of nodes in a spanning tree is connected by a **unique** path;

($d$) addition of any arc to a spanning tree results in a graph containing **exactly one cycle.**

# Bases for $\mathcal{N}_s$

A **basis** for $\mathcal{N}_s$ is any matrix $B_{\mathcal{T}}$ consisting of set $\mathcal{T}$ of $n-1$ columns of $\mathcal{N}_s$ so that for *any* choice of $b_i$, $i \in N \backslash s$, there is a **unique solution** to the $(n-1) \times (n-1)$ system

$$(*) \qquad B_{\mathcal{T}} x = b$$

(Note that this is equivalent to saying the $B_{\mathcal{T}}$ is a **nonsingular** matrix.) The corresponding variables are called **basic variables**.

**Basis Lemma:** Let $\mathcal{T}$ be a set of arcs of $G$, and let $B_{\mathcal{T}}$ be the corresponding set of columns of $\mathcal{N}_s$. Then $B_{\mathcal{T}}$ is a basis for $\mathcal{N}_s$ **if and only if** $\mathcal{T}$ is a **spanning tree** for $G$.

# Proof of the Basis Lemma

First suppose that $\mathcal{T}$ is **not** a spanning tree.

**Case 1:** Suppose that $\mathcal{T}$ is not connected, that is there is some node $t$ that is not connected to $s$ by a path. Consider the right-hand-side $b$ with $b_t = -1$ and $b_i = 0$, $i \in N \setminus \{s, t\}$. Then clearly there can be no solution to $(*)$.

**Case 2:** Suppose that $\mathcal{T}$ contains a cycle

$$W: \ v_0, e_1, v_1, \ldots, v_{k-1}, w_k, v_k = v_0$$

with $e_i = (v_{i-1}, v_i)$ or $(v_i, v_{i-1})$. Now let $b_i = 0$ for all $i$. For any $\lambda$, consider the solution $x$ having

$$x_{ij} = \begin{cases} +\lambda & (i,j) \text{ a forward arc of } C \\ -\lambda & (i,j) \text{ a backward arc of } C \\ 0 & (i,j) \text{ not on } C \end{cases}$$

for any $\lambda$ this clearly satisfies $(*)$, and so $(*)$ has more than one solution.

In either case, $(*)$ will not have a unique solution, so that this direction of the lemma is proven.

# Computing a Basic Solution

Now suppose that $\mathcal{T}$ is a spanning tree. To complete the Basis Lemma, we need to show how to compute unique flow values for $x$ on the arcs of $\mathcal{T}$ which will satisfy the flow equations. For any set of flow values $x_{ij}$, $(i,j) \in A$, recall the **imbalance** of $i$ is defined

$$e_i = b_i - \sum_{j \in \mathcal{A}(i)} x_{ij} + \sum_{j \in \mathcal{B}(i)} x_{ji}$$

**Algorithm Compute-Flows**

**Initialize:** Set $e_i = b_i$, $i \in A$ and $\mathcal{S} = \mathcal{T}$.

**while** $\mathcal{S} \neq \emptyset$ **do**

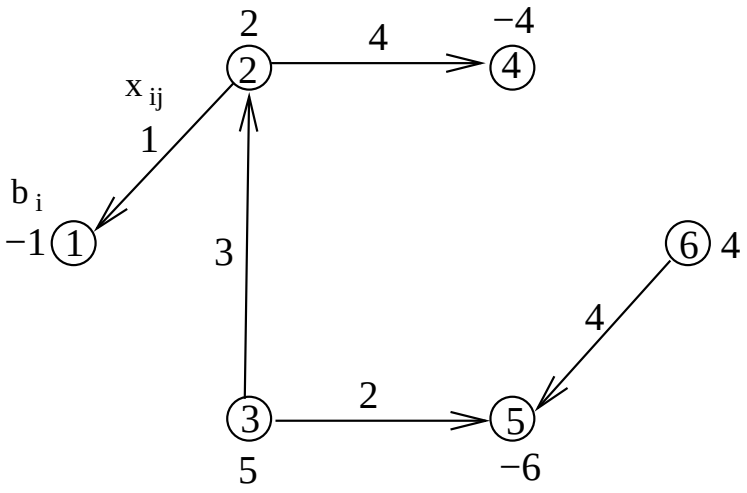
Let $i \neq s$ be an end node of $\mathcal{S}$, with $a = (i,j)$ or $(j,i)$ the associated unique arc of $\mathcal{S}$ adjacent to $i$.

If $a = (i,j)$ assign $x_{ij} = e_i$, and if $a = (j,i)$ assign $x_{ji} = -e_i$.

Remove $a$ and $i$ from $\mathcal{S}$ and add $e_i$ to $e_j$.

**end while**

# Example




**Fact:** Algorithm Compute-Flows correctly computes the unique basic solution associated with spanning tree $\mathcal{T}$.

**Proof:** By construction the given solution will satisfy the node equations (What about $s$?). Further, since the assignment of values to arcs of $\mathcal{T}$ is forced at each stage of the algorithm, then the given solution is in fact unique.

# Bases for $(NP)$

In general, a basis for $(NP)$ is represented by a **triple** $(\mathcal{T}, \mathcal{L}, \mathcal{U})$ of sets of arcs of $G$, where $\mathcal{T}$ is the spanning tree of basic variables, $\mathcal{L}$ is the set of variables at their **lower** bounds, and $\mathcal{U}$ is the set of variables at their **upper** bounds.

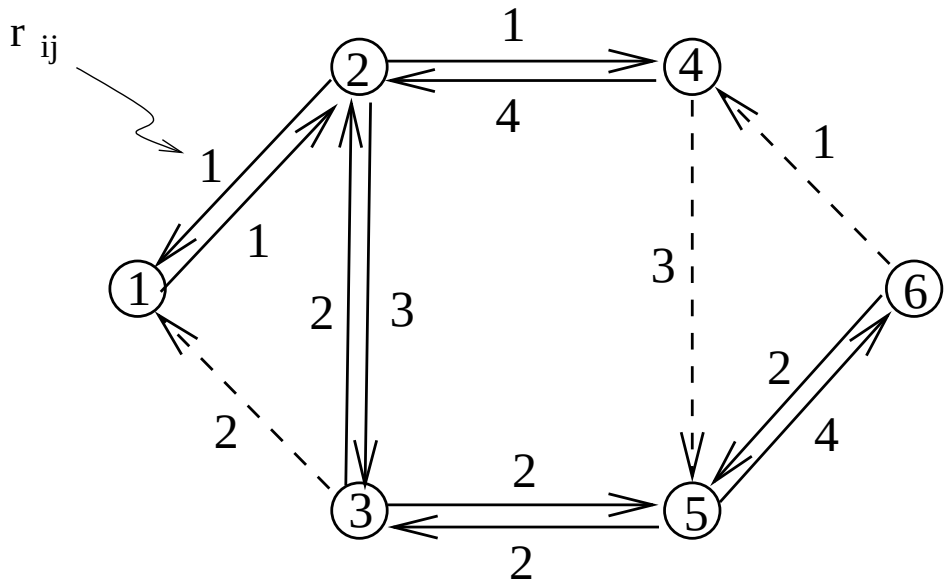
**Steps for finding the basic solution corresponding to** $(\mathcal{T}, \mathcal{L}, \mathcal{U})$:

1. Set $x_{ij} = 0$ for all $(i,j) \in \mathcal{L}$ and $x_{ij} = u_{ij}$ for all $(i,j) \in \mathcal{U}$.
2. Compute the resulting imbalances $e_i$.
3. Use Compute-Flows to determine the values of $x$ on the arcs of the spanning tree $\mathcal{T}$ based on the imbalances computed in (2). This is a basic feasible solution iff the corresponding tree-arc values lie between their lower and upper bounds.

**basic feasible solution:** a basic solution $x$ with $0 \leq x_{ij} \leq u_{ij}$ for all $(i,j) \in \mathcal{T}$.

# The Residual Network for a Basic Solution

The residual network is constructed the standard way. Note that arcs in $\mathcal{L}$ will appear in the **same** direction as in $G$, arcs in $\mathcal{U}$ will appear in the **opposite** direction as they do in $G$, and arcs in $\mathcal{T}$ will (generally) be represented by **two** arcs.




Residual network for starting basis

$\mathcal{T} = \{(2,1),(3,2),(2,4),(3,5),(6,5)\}$,
$\mathcal{U} = \{(1,3),(5,4)\}$, and $\mathcal{L} = \{(6,4)\}$

# Computing Node Potentials

We compute node potentials for $G^{\pi}(x)$ in such way that each arc $(i, j)$ in $\mathcal{T}$ satisfies

$$c_{ij} = \pi_i - \pi_j$$

This can be done by using the following labeling scheme on $\mathcal{T}$:

Label $\pi_s = 0$.

An arc $(i, j)$ is **admissible** if it is in $\mathcal{T}$ and exactly one of its endpoints is labeled. For any admissible arc, set

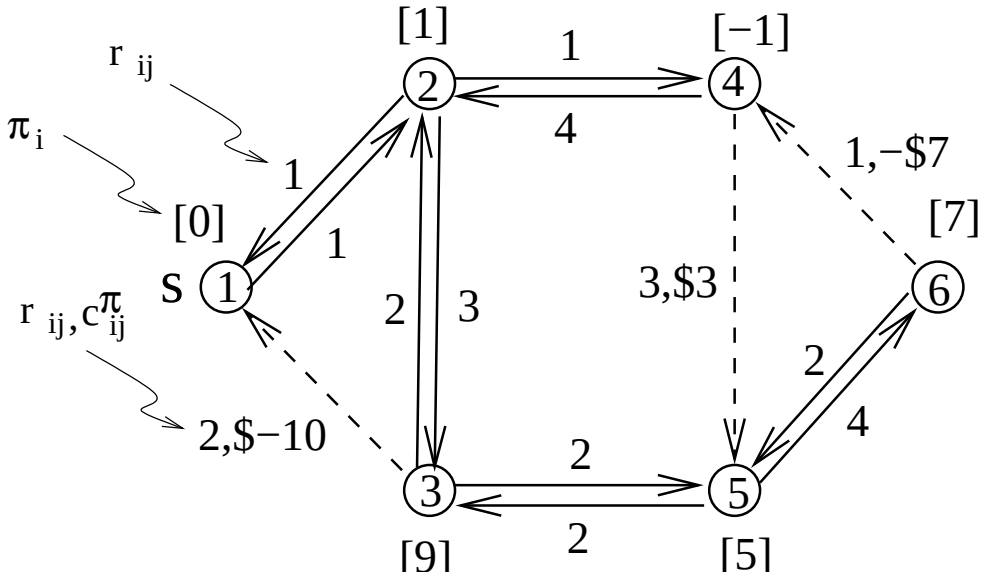
$$\begin{array}{rcll} \pi_j & = & \pi_i - c_{ij} & i \text{ labeled, } j \text{ unlabeled} \\ \pi_i & = & \pi_j + c_{ij} & j \text{ labeled, } i \text{ unlabeled} \end{array}$$

**Fact 1:** The above labeling scheme can be performed in $O(n)$ time using FINDPATH, and uniquely determines the correct values for $\pi$.

**Computing Costs:** For each $(i, j) \notin T$ set $c_{ij}^{\pi} = c_{ij} - \pi_i + \pi_j$.

**Fact 2:** $c_{ij}^{\pi} = 0$ for all arcs in $\mathcal{T}$

# The Residual Network $G^{\pi}(x)$ for the Example




**Fact 3:** The value of the reduced cost on nonbasic arc $(i, j)$ is equal to the sum of the costs of the arcs of $G(x)$ in the **unique** cycle $W$ created by adding $(i, j)$ to $\mathcal{T}$, directed consistent with $(i, j)$. Thus nonbasic arcs in $G^{\pi}(x)$ having **negative** reduced costs correspond to **negative cycles** in $G^{\pi}(x)$.

# Performing the Pivot

**Choice of entering variable:** arc $(i, j)$ with negative (heuristically, most negative) reduced cost. If all reduced costs are nonnegative, STOP, current solution is optimal.

**Modifying Flow:** Push flow around the unique directed cycle $W$ induced by $(i, j)$ up to the minimum residual capacity around $W$, modifying $x$ accordingly. If residual capacities around $W$ are all $\infty$, **STOP**, problem is **unbounded**.

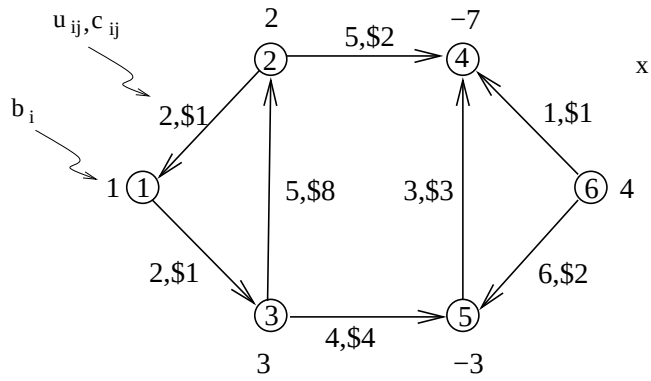
**New basic solution:** Let $(k, l)$ be the arc of $W$ having minimum residual capacity.

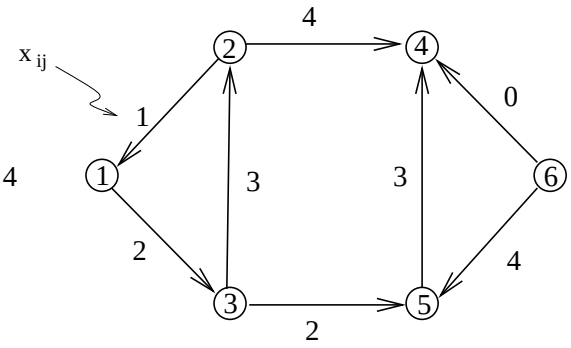
1. Move arc $(i, j)$ into $\mathcal{T}$.
2. If $(k, l)$ is an arc in $G$, then move $(k, l)$ into $\mathcal{U}$
3. If $(l, k)$ is an arc in $G$, then move $(l, k)$ into $\mathcal{L}$

**Note:** $(k, l)$ could actually be the arc $(i, j)$ itself, in which case $\mathcal{T}$ would not change but $(i, j)$ would move from $\mathcal{L}$ to $\mathcal{U}$ or vice versa.

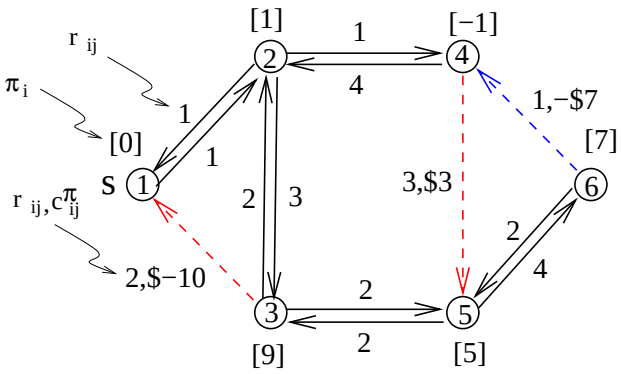
# Example

**Starting basis:** $\mathcal{T} = \{(2,1),(3,2),(2,4),(3,5),(6,5)\}$, $\mathcal{L} = \{(6,4)\}$, $\mathcal{U} = \{(1,3),(5,4)\}$. Arcs in $\mathcal{L}$ are in blue, arcs in $\mathcal{U}$ are in red.

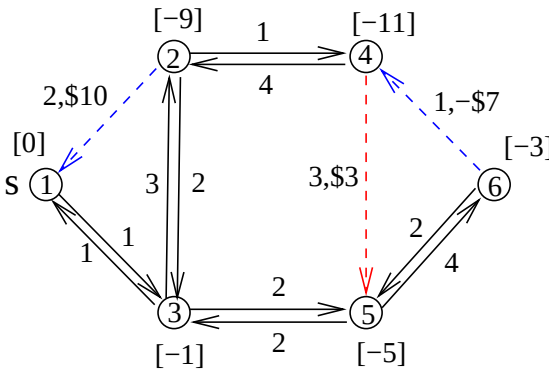


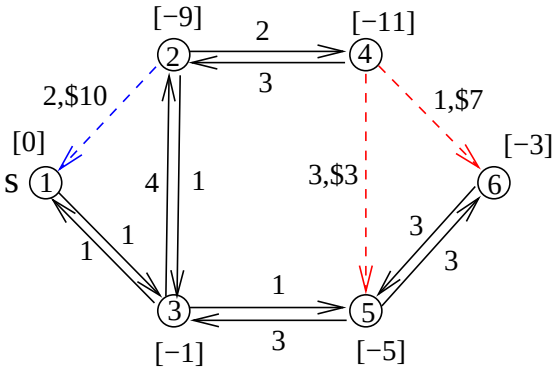

Initial Flow



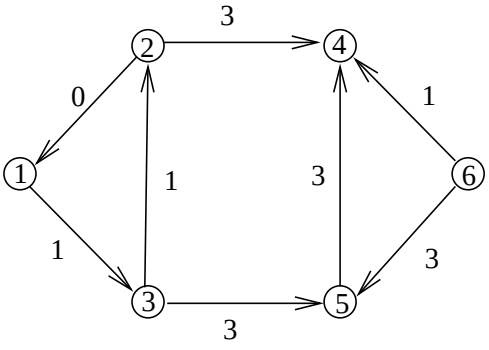

Initial Residual Network




After First Pivot




After Second Pivot




Final Flow

# The Phase II Network Simplex Method

**Initialize:** Start with basis $(\mathcal{T}, \mathcal{L}, \mathcal{U})$ corresponding to a **basic feasible solution**. Compute basic flow values, node potentials, and reduced costs, and form residual network $G^{\pi}(x)$.

**while** there are nonbasic arcs with negative reduced costs in $G^{\pi}(x)$, perform a **pivot**:

($i$) Choose arc $(i, j)$ with $c_{ij}^{\pi} < 0$.

($ii$) Find the associated directed cycle $W$ obtained by adding $(i, j)$ to $\mathcal{T}$, and push flow around $W$ up to the minimum residual capacity.

($iii$) Adjust $(\mathcal{T}, \mathcal{L}, \mathcal{U})$, recompute $x$ and $\pi$, and form new residual network $G^{\pi}(x)$.

# Correctness of Phase II Simplex Method

1. After every pivot, the current basis is **feasible**.

2. If there are no nonbasic arcs with nonnegative reduced costs, then the current solution is **optimal**.

3. Pushing flow around the directed cycle $W$ induced by adding a negative-reduced-cost nonbasic arc to $\mathcal{T}$ will always result in a **strictly smaller cost flow** in $G$.

**Result:** The number of pivots performed in the Phase II Simplex Algorithm is at most $2mCU$.

# Degenerate Pivots, Cycling, and Finiteness of NSA

The problem with the Simplex Method as given above is that there may be **no directed cycle** $W$ induced by adding the negative-reduced-cost arc $(i, j)$ to $\mathcal{T}$. This will occur if the basis $(\mathcal{T}, \mathcal{L}, \mathcal{U})$ is **degenerate**, that is, the computed value $x_{ij}$ of a **basic** arc $(i, j)$ may have $x_{ij} = 0$ or $x_{ij} = u_{ij}$. This means that one of the two arcs corresponding to $(i, j)$ may be **missing** in $G^{\pi}(x)$ and thus might prevent the directed cycle $W$ from existing.

We could adjust for this by making **degenerate pivots**, that is, adding an arc with **zero residual capacity** along $W$ in the appropriate direction. This means that we will be pushing **zero flow** around $W$, but at least the basis changes and (hopefully) after a finite number of pivots we will be able to find a "nondegenerate" cycle $W$.

## A "Bland's Rule" for Breaking Cycles

Let $s$ be an arbitrarily chosen root node, from which all basis trees are oriented.

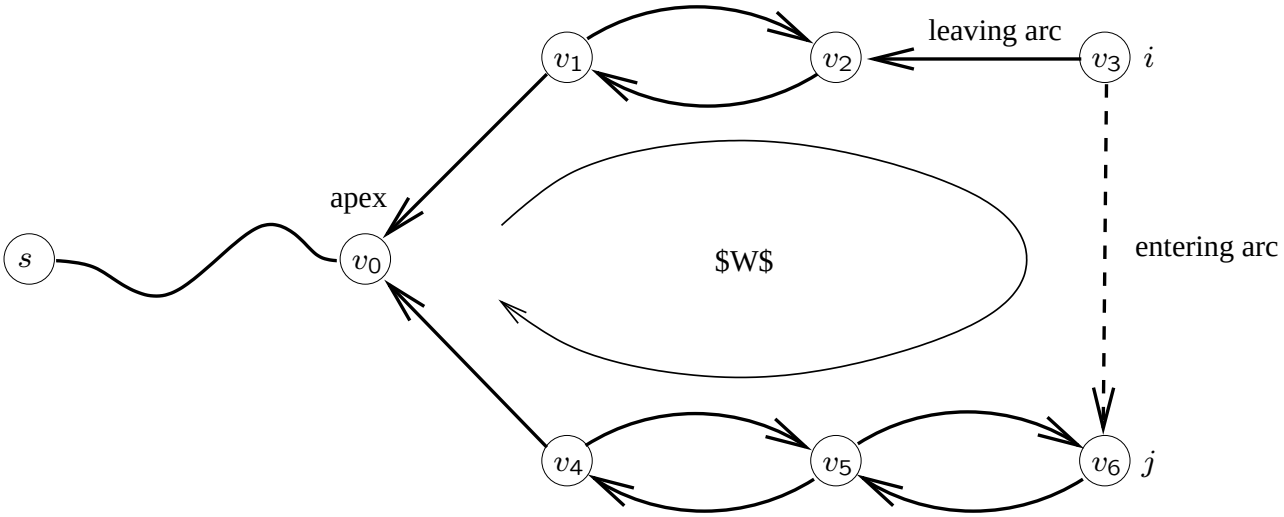
**Strongly feasible basis:** A feasible basis for which any degenerate tree arc in $G^\pi(x)$ — that is, tree arc whose oppositely-directed partner is missing — always points **toward** the root $s$.

We will maintain strongly feasible bases throughout the algorithm. Let $\mathcal{T}$ be the strongly feasible basis, with associated flow $x$ and reduced cost values $c^\pi_{ij}$ in $G^\pi(x)$.

**Entering arc:** Arc $(i, j)$ in $G^\pi(x)$ with $c^\pi_{ij} < 0$.

Now let $W$ be the cycle in $G^\pi(x)$ obtained by adding $(i, j)$ to $\mathcal{T}$, oriented in the direction of the arc $(i, j)$. Then the only backward arcs in $W$ will be the degenerate arcs. The **apex** of $W$ will be the first common vertex in $(i, s)$ and $(j, s)$ paths in $\mathcal{T}$.

**Leaving arc:** If there are backward arcs, then choose the last backward arc encountered in a traversal of $W$ from its apex.




**Proposition:** The pivot rule given above guarantees that the basis trees remain strongly feasible, and that there can be only a finite number of consecutive degenerate pivots.

**Corollary:** The network simplex method, using the above pivot rule, finds an optimal solution in a finite number of pivots.

# Polynomial Network Simplex Implementations

- If all data are integer, then the number of pivots taken by the simplex algorithm using the above rule is $O(nCU)$, which is pseudopolynomial.

- Polynomial-time implementations of the simplex algorithms have been developed for shortest path, max flow, and assignment problems.

- The simplex algorithm can be embedded in a cost-scaling procedure for general min cost flow problems, in which the number of pivots in each scaling phase is polynomial.

- The problem of finding a pivot rule that leads to a polynomial number of pivots in the unscaled version of the simplex method is still an open problem.

# The Phase I Network Simplex Method

To get a starting basis for a Network LP, we form an **artificial Min Cost Flow LP** whose optimal solution will result in a feasible starting basis for $(NP)$. The steps of the Phase I Network Simplex Method are as follows:

1. Choose an arbitrary root node $s$, and for each node $i$ add **artificial arc**

   $(i, s)$ if $b_i \geq 0$.

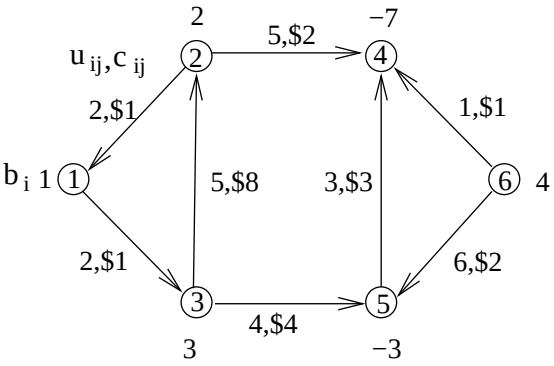
   $(s, i)$ if $b_i < 0$, and

2. Set the initial flow on these artificial arcs to the absolute value of the supply or demand at the node $i$, and set the capacity to $\infty$. Set $\mathcal{T}$ = the set of artificial arcs, $\mathcal{L}$ = the set of original arcs, and $\mathcal{U} = \emptyset$.

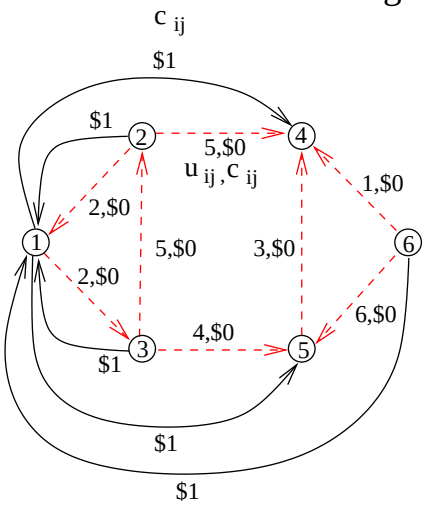
3. Perform the Network Simplex Method on this network, using a cost of 1 on each artificial arc, and a cost of 0 on each original arc.

4. If the optimal solution has artificial arcs with nonzero flow, then the original flow problem is **infeasible**. Otherwise delete all artificial arcs and continue with the Network Simplex Method, using the final basis and arc costs equal to those of the original LP.

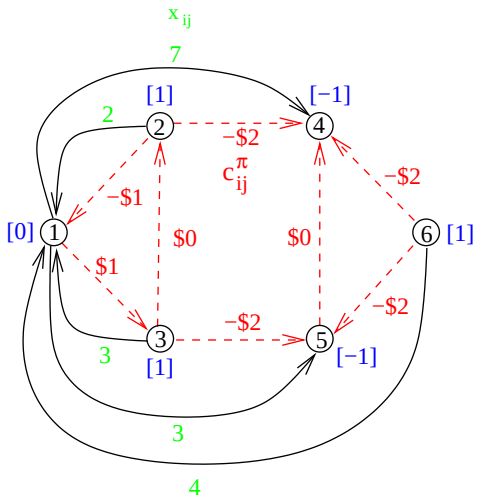
# Example




Original Problem




Phase I Problem




Feasible Solution and Reduced Costs