6.215/6.255J/15.093J/IDS.200J Optimization Methods

Lecture 10: Network Optimization II

October 12, 2021

Today's Lecture

Outline

- Network (linear) optimization: Key facts
- Network simplex method for the uncapacitated min cost flow problem
  - a combinatorial view
  - an algebraic view

Network (linear) optimization

What is so special about it?

- 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 enables more efficient algorithms.

Today: The network simplex method: a tailored simplex algorithm

Network (linear) optimization

An algorithmic comparison

Min-cost flow problem running times:

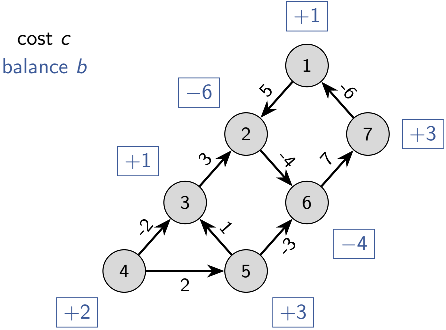
Algorithm	Running Time (sec)	# Iterations
Standard Simplex	334.59	42759
Network Simplex	7.37	23306
Ratio	2.2 %	54 %

Average over 5 random instances with $10,000$ nodes and $25,000$ arcs each.

The Network Simplex Algorithm

Applied to the uncapacitated min-cost flow problem

- Connected directed graph $G = (N, A)$.
- Arc costs $c : A \to \mathcal{Z}$.
- Node balances $b : N \to \mathcal{Z}$.
- Assume $\sum_{i \in N} b_i = 0$.



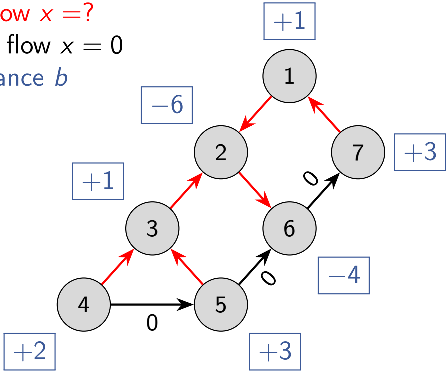
$$
\begin{array}{llll}
\min & \displaystyle\sum_{(i,j) \in A} c_{ij} x_{ij} & & \\
\text{s.t.} & \displaystyle\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = b_i & & \text{for all } i \in N \\
& x_{ij} \geq 0 & & \text{for all } (i,j) \in A
\end{array}
$$

The Network Simplex Algorithm

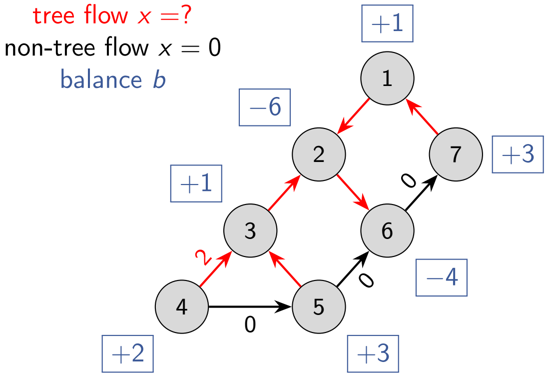
A combinatorial view - tree solutions

- **Definition:** A flow $\boldsymbol{x}$ is called a tree solution if there is a spanning tree of the network (when arc directions are ignored) and every arc not in the spanning tree has flow 0. If resulting flow $\boldsymbol{x} \geq 0$, it is called a feasible tree solution.

$\longrightarrow$: tree
tree flow $x =$?
non-tree flow $x = 0$
balance b



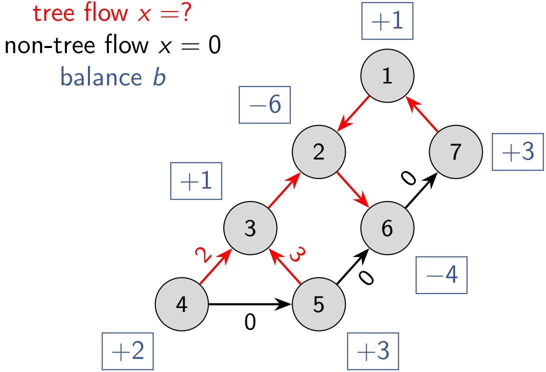
- tree flow computation, I



The Network Simplex Algorithm

A combinatorial view - tree solutions

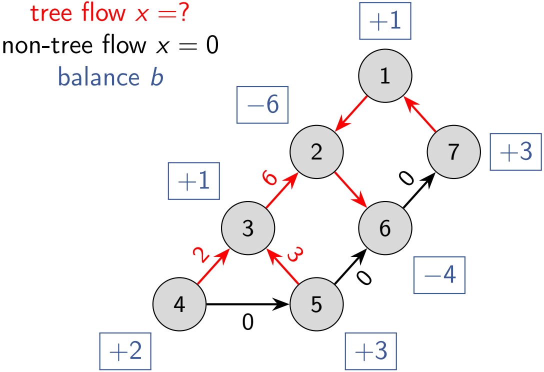
- tree flow computation, II



The Network Simplex Algorithm

A combinatorial view - tree solutions

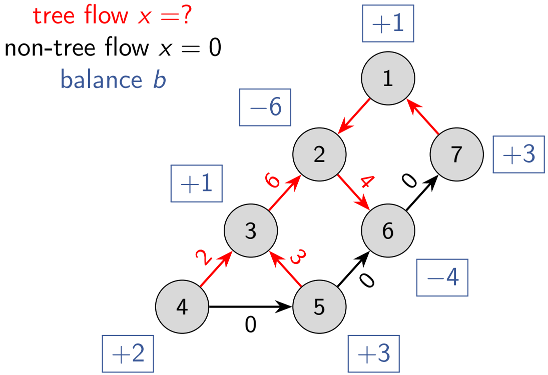
- tree flow computation, III



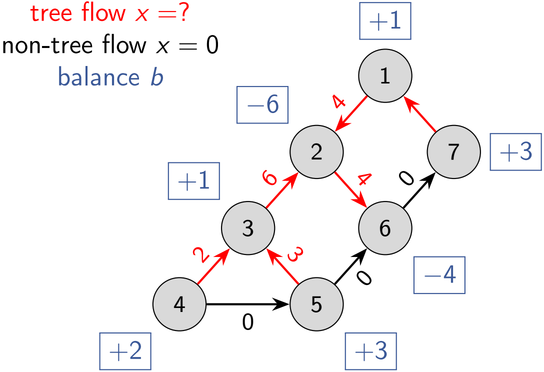
The Network Simplex Algorithm

A combinatorial view - tree solutions

- tree flow computation, IV



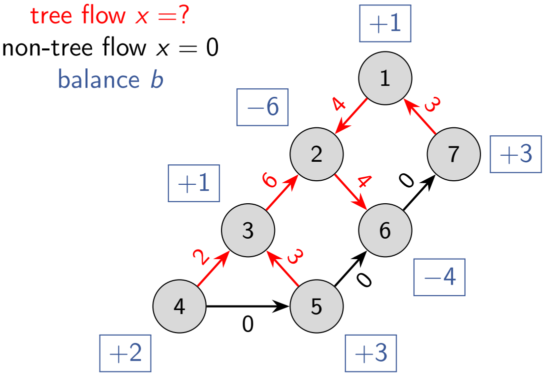
- tree flow computation, V



The Network Simplex Algorithm

A combinatorial view - tree solutions

- tree flow computation, VI

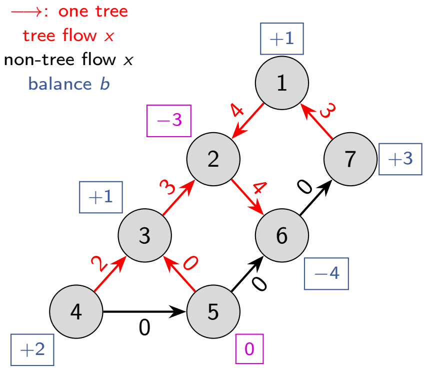


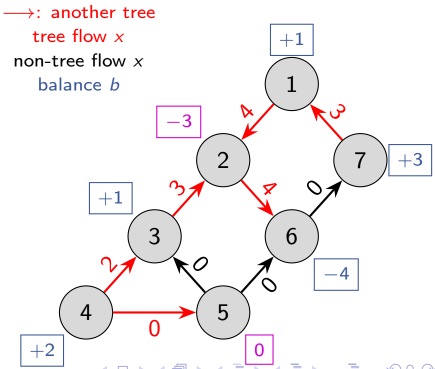
- ⇒ a feasible tree solution

Tree solution vs. Spanning tree

- Given a spanning tree, we obtain a unique tree solution associated with it.
- Every tree solution has at least a corresponding spanning tree (perhaps more).

Example: The following tree solution flow x has two possible spanning trees:





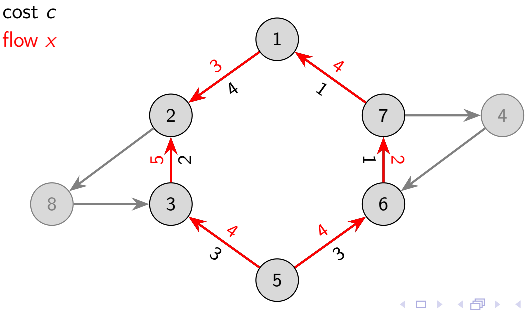
The Network Simplex Algorithm

A combinatorial view - tree solutions

Theorem

If the objective function is bounded from below, a min-cost flow problem always has an optimal tree solution.

- consider a flow solution which is not a tree ⇒ there exists a circulation
- change the flow only on the cycle
- circulating ...



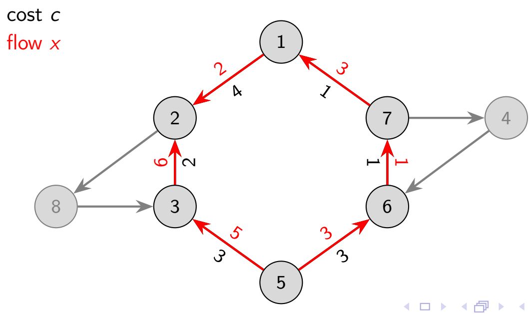
The Network Simplex Algorithm

A combinatorial view - tree solutions

Theorem

If the objective function is bounded from below, a min-cost flow problem always has an optimal tree solution.

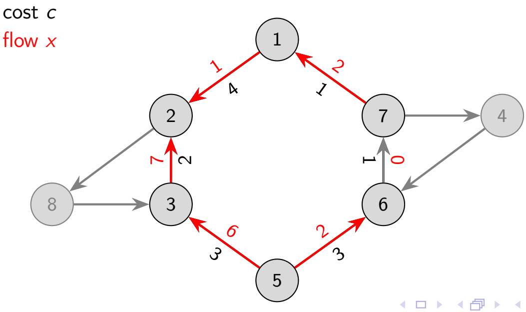
- consider a flow solution which is not a tree $\Rightarrow$ there exists a circulation
- change the flow only on the cycle
- circulating one unit of flow clockwise on the cycle $\Rightarrow$ cost decreases by 4



Theorem

If the objective function is bounded from below, a min-cost flow problem always has an optimal tree solution.

- consider a flow solution which is not a tree $\Rightarrow$ there exists a circulation
- change the flow only on the cycle
- circulating two units of flow clockwise on the cycle $\Rightarrow$ cost decreases by 8



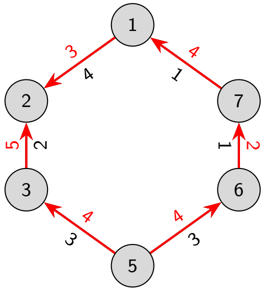
The Network Simplex Algorithm

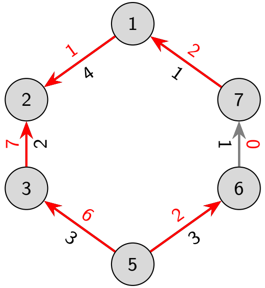
A combinatorial view - tree solutions

Theorem

If the objective function is bounded from below, a min-cost flow problem always has an optimal tree solution.

cost c
flow x





circulating two units of flow clockwise on the cycle $\Rightarrow$ cost decreases by 8

The Network Simplex Algorithm

A combinatorial view - tree solutions and node potentials

Optimality conditions:

Theorem

A feasible tree solution associated with a tree T is optimal if, for some choice of node potentials p_i,

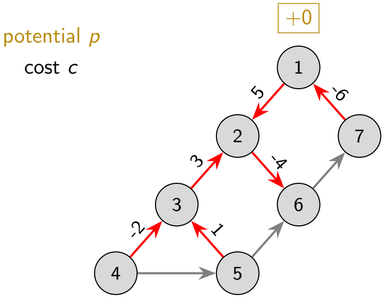
- a. $\bar{c}_{ij} = c_{ij} - p_i + p_j = 0$ *for all* $(i,j) \in T$,
- b. $\bar{c}_{ij} = c_{ij} - p_i + p_j \geq 0$ *for all* $(i,j) \in A \setminus T$.

Proof sketch:

- $\min \sum_{(i,j)\in A} c_{ij}x_{ij}$ is equivalent to $\min \sum_{(i,j)\in A} \bar{c}_{ij}x_{ij}$.
- $\min \sum_{(i,j)\in A} \bar{c}_{ij}x_{ij}$ is equivalent to $\min \sum_{(i,j)\in A\setminus T} \bar{c}_{ij}x_{ij}$.
- ... conclude (how?)

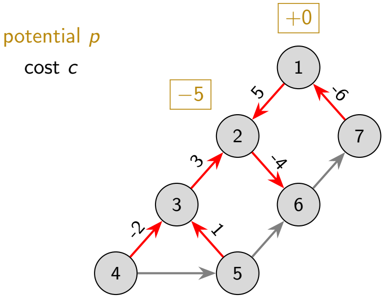
Computing node potentials:

- the potentials associated to a spanning tree T can be computed easily from $c_{ij} = p_i - p_j$ for all $(i, j) \in T$:
- graphical way of computing, starting with the "root node" of the tree (in the example below node 1) setting its potential to 0 (in the example $p_1 = 0$):



Computing node potentials:

- the potentials associated to a spanning tree T can be computed easily from $c_{ij} = p_i - p_j$ for all $(i, j) \in T$:
- graphical way of computing, starting with the "root node" of the tree (in the example below node 1) setting its potential to 0 (in the example $p_1 = 0$):

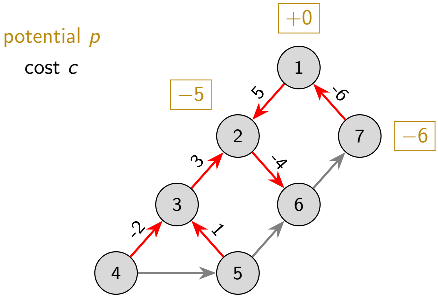


The Network Simplex Algorithm

A combinatorial view - tree solutions and node potentials

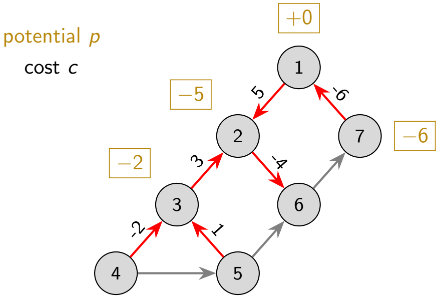
Computing node potentials:

- the potentials associated to a spanning tree T can be computed easily from $c_{ij} = p_i - p_j$ for all $(i, j) \in T$:
- graphical way of computing, starting with the "root node" of the tree (in the example below node 1) setting its potential to 0 (in the example $p_1 = 0$):



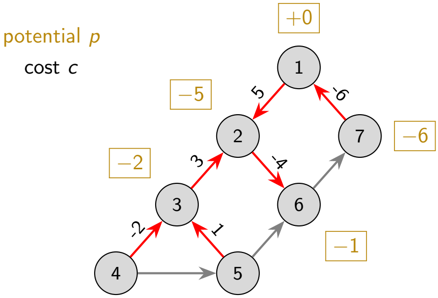
Computing node potentials:

- the potentials associated to a spanning tree T can be computed easily from $c_{ij} = p_i - p_j$ for all $(i, j) \in T$:
- graphical way of computing, starting with the "root node" of the tree (in the example below node 1) setting its potential to 0 (in the example $p_1 = 0$):



Computing node potentials:

- the potentials associated to a spanning tree T can be computed easily from $c_{ij} = p_i - p_j$ for all $(i,j) \in T$:
- graphical way of computing, starting with the "root node" of the tree (in the example below node 1) setting its potential to 0 (in the example $p_1 = 0$):

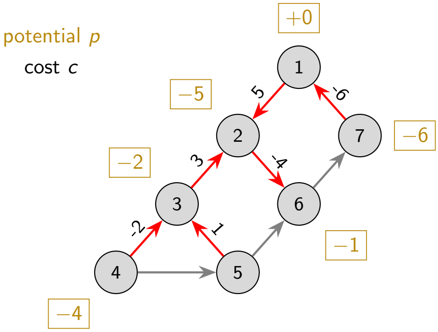


The Network Simplex Algorithm

A combinatorial view - tree solutions and node potentials

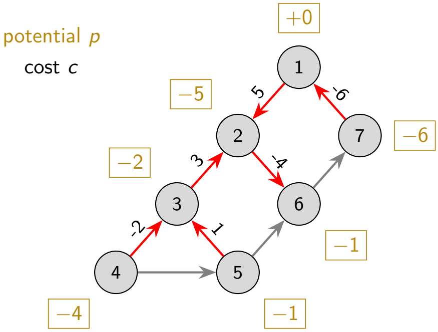
Computing node potentials:

- the potentials associated to a spanning tree T can be computed easily from $c_{ij} = p_i - p_j$ for all $(i, j) \in T$:
- graphical way of computing, starting with the “root node" of the tree (in the example below node 1) setting its potential to 0 (in the example $p_1 = 0$):



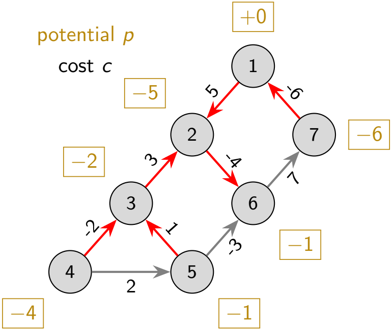
Computing node potentials:

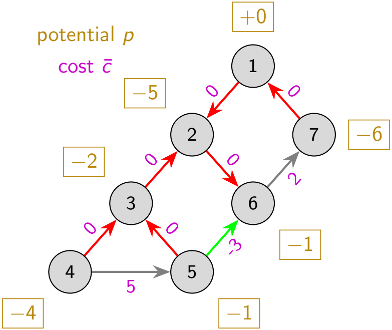
- the potentials associated to a spanning tree T can be computed easily from $c_{ij} = p_i - p_j$ for all $(i, j) \in T$:
- graphical way of computing, starting with the "root node" of the tree (in the example below node 1) setting its potential to 0 (in the example $p_1 = 0$):



Computing modified costs:

- the modified costs $\overline{c}_{ij} = c_{ij} - p_i + p_j$ for all (i, j) can then be easily inferred:





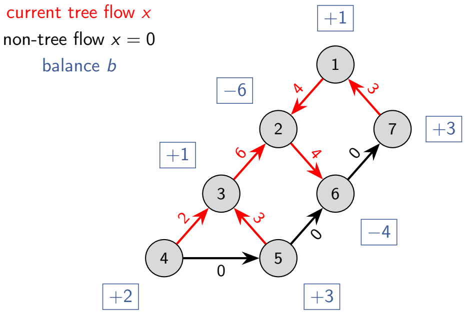
- note that $\overline{c}_{56} < 0$ so the current tree solution is not optimal

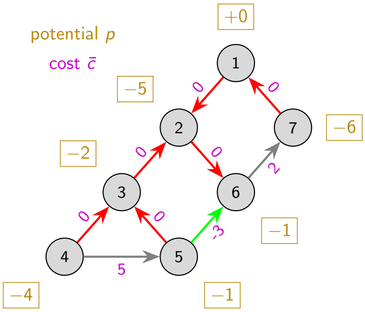
The Network Simplex Algorithm

A combinatorial view - tree solutions

Updating the tree solution:

- Bring the arc $(5,6)$ with modified cost $\overline{c}_{56} = -3$ into a new tree solution and remove one of the arcs from the current tree solution in the created cycle.
- How? By increasing x_{56} from 0 as much as possible while circulating flow on the created cycle (here 3 units counter-clockwise, bringing x_{53} to zero).





1. Determine an initial feasible tree T. Compute flow x and node potentials p associated with T.
2. Calculate $\overline{c}_{ij} = c_{ij} - p_i + p_j$ for $(i, j) \notin T$.
    - If $\overline{c} \geq 0$, x optimal; stop.
    - Select (i, j) with $\overline{c}_{ij} < 0$.
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, ℓ).
4. $T := (T \setminus (k, \ell)) \cup (i, j)$. Repeat.

Min-Cost Flow

Integrality

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).

Min-Cost Flow

The algebraic view

- Bases and tree solutions.
- Dual variables and node potentials.
- Changing bases and updating tree solutions.
- Optimality testing.

The Simplex Algorithm

A reminder

Assume a linear optimization problem in standard form with an optimal solution:

$$\begin{array}{lrcl} \min & \boldsymbol{c}^T\boldsymbol{x} & & \\ \text{s.t.} & \boldsymbol{A}\boldsymbol{x} & = & \boldsymbol{b} \\ & \boldsymbol{x} & \geq & 0 \end{array}$$

The algorithm:

1. Start with a feasible basis $\boldsymbol{B} = [\boldsymbol{A}_{B(1)}, \ldots, \boldsymbol{A}_{B(m)}]$ and a BFS $\boldsymbol{x}_B = \boldsymbol{B}^{-1}\boldsymbol{b}$.
2. Compute $\boldsymbol{p}^T = \boldsymbol{c}_B^T\boldsymbol{B}^{-1}$, $\overline{c}_j = c_j - \boldsymbol{p}^T\boldsymbol{A}_j$ for all j.
   - If $\overline{c}_j \geq 0$ for all j; $\boldsymbol{x}$ optimal; Stop; Else select $j : \overline{c}_j < 0$.
3. Compute search direction $\boldsymbol{u} = \boldsymbol{B}^{-1}\boldsymbol{A}_j$.
   - If $\boldsymbol{u} \leq 0 \Rightarrow$ cost unbounded; Stop.
4. $\theta^* = \min_{1 \leq i \leq m, u_i > 0} \dfrac{x_{B(i)}}{u_i} \doteq \dfrac{x_{B(\ell)}}{u_\ell}$
5. Form a new basis $\overline{\boldsymbol{B}}$ by replacing $\boldsymbol{A}_{B(\ell)}$ with $\boldsymbol{A}_j$.
6. Values of new basic variables: $y_j = \theta^*$, $y_{B(i)} = x_{B(i)} - \theta^* u_i$, $i \neq \ell$.

Compact formulation of the uncapacitated min-cost flow problem

Let $\boldsymbol{x} = (x_{ij})_{(i,j)\in A}$ the flow through the network.

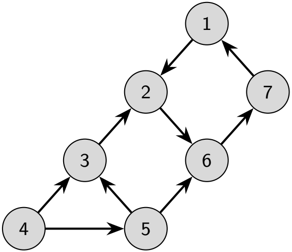
Compact linear optimization formulation:

$$\begin{aligned} \min \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq 0 \end{aligned}$$

where $\boldsymbol{A}$ is node-arc incidence matrix of the graph G.

The Network Simplex Algorithm

Compact formulation - node-arc incidence matrix



Node-arc incidence matrix $\boldsymbol{A}$:

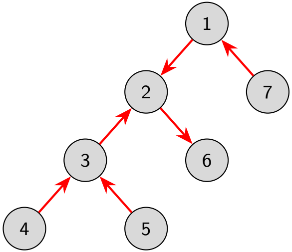
$$
\begin{array}{c|ccccccccc}
 & (1,2) & (2,6) & (3,2) & (4,3) & (4,5) & (5,3) & (5,6) & (6,7) & (7,1) \\
1 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
2 & -1 & +1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & +1 & -1 & 0 & -1 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & +1 & +1 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & -1 & +1 & +1 & 0 & 0 \\
6 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & +1 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & +1
\end{array}
$$

Rows of $\boldsymbol{A}$ are linearly dependent ... it has rank $n-1$ ... can ignore the last row, say

Min-Cost Flow

The algebraic view - bases vs. tree solutions

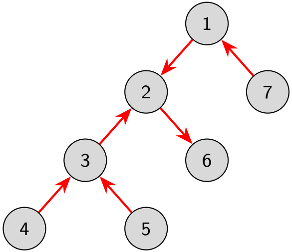
$\longrightarrow$ tree solution



Let $\boldsymbol{B}$ be the corresponding matrix. It is a basis:

$$
\begin{array}{c|cccccc}
 & (1,2) & (2,6) & (3,2) & (4,3) & (5,3) & (7,1) \\
1 & +1 & 0 & 0 & 0 & 0 & -1 \\
2 & -1 & +1 & -1 & 0 & 0 & 0 \\
3 & 0 & 0 & +1 & -1 & -1 & 0 \\
4 & 0 & 0 & 0 & +1 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & +1 & 0 \\
6 & 0 & -1 & 0 & 0 & 0 & 0 \\
\hline
7 & 0 & 0 & 0 & 0 & 0 & +1
\end{array}
$$

$\longrightarrow$ tree solution



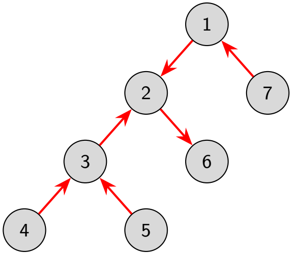
Let $\boldsymbol{B}$ be the corresponding basic matrix, permuting rows:

$$
\begin{array}{c|cccccc}
 & (1,2) & (2,6) & (3,2) & (4,3) & (5,3) & (7,1) \\
4 & 0 & 0 & 0 & +1 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & +1 & 0 \\
6 & 0 & -1 & 0 & 0 & 0 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 & +1 \\
3 & 0 & 0 & +1 & -1 & -1 & 0 \\
2 & -1 & +1 & -1 & 0 & 0 & 0 \\
\hline
1 & +1 & 0 & 0 & 0 & 0 & -1
\end{array}
$$

Min-Cost Flow

The algebraic view - bases vs. tree solutions

$\longrightarrow$ tree solution



Let $\boldsymbol{B}$ be the corresponding basic matrix, permuting columns:

$$\begin{array}{c|cccccc} & (4,3) & (5,3) & (2,6) & (7,1) & (3,2) & (1,2) \\ 4 & +1 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & +1 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & -1 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & +1 & 0 & 0 \\ 3 & -1 & -1 & 0 & 0 & +1 & 0 \\ 2 & 0 & 0 & +1 & 0 & -1 & -1 \\ \hline 1 & 0 & 0 & 0 & -1 & 0 & +1 \end{array}$$

$\Rightarrow$ lower triangular matrix! ... can solve $\boldsymbol{B}^{-1}\boldsymbol{b}$ efficiently using back substitution.

Min-Cost Flow

The algebraic view - bases vs. tree solutions

In conclusion:

Theorem

Every tree solution defines a basis, and conversely, one can show that every basis defines a tree solution.

Min-Cost Flow

The algebraic view - dual variables vs. node potentials

Remember, the simplex algorithm computes the dual variables $\boldsymbol{p}$ as the solution to $\boldsymbol{p}^T \boldsymbol{B} = \boldsymbol{c}_B^T$.

$$[p_4,\ p_5,\ p_6,\ p_7,\ p_3,\ p_2] \begin{bmatrix} +1 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & 0 \\ -1 & -1 & 0 & 0 & +1 & 0 \\ 0 & 0 & +1 & 0 & -1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} c_{43}, & c_{53}, & c_{26}, & c_{71}, & c_{32}, & c_{12} \end{bmatrix}$$

Hence, $p_2 = -c_{12}$, $p_3 = c_{32} + p_2$, $p_7 = c_{71}$, ...

Remember, the simplex algorithm computes the reduced costs $\overline{\boldsymbol{c}}$ as $\overline{c}_{ij} = c_{ij} - \boldsymbol{p}^T \boldsymbol{A}_{ij}$.

	(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	−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

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

Note: The optimality conditions on the modified costs given by our previous theorem (repeated below) is nothing else than the complementary slackness conditions for linear optimization

Min-Cost Flow

Summary

- 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.