Topic 6: Minimum Cost Network Flow Problems

F.-Javier Heredia

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Estadística
i Investigació Operativa

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-nd/3.0/ or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.
6.- Minimum Cost Network Flow Problems
(Chap. 9, 10 & 11 Ahuja, Magnanti, Orlin)

- Pseudo-polynomial algorithms (AMO, Chap. 9)
- Polynomial algorithms (AMO, Chap. 10)
- Simplex algorithm (AMO, Chap. 11)
MCNF: Basic algorithms (Chap. 9 AMO)

- Definitions and assumptions.
- Applications.
- Optimality Conditions:
  - Negative Cycle Optimality Conditions.
  - Reduced Cost Optimality Conditions.
  - Complementary Slackness Optimality Conditions.
- Algorithms: (assignments)
  - Cycle-canceling and negative cycle O.C.
  - Successive shortest-path and Reduced cost O.C.
  - Out-of-kilter algorithm and Complementary Slackness O.C.
- The dual-primal network flow algorithm (assignment).
- Relaxation methods.
- Summary.
Minimum Cost Network Flow Problem (MCNFP)

- Given the network $G = (N, A)$ with costs $c$, capacities $u$ and supplies/demands $b$, the **Minimum Cost Network Flow Problem (MCNFP)** is defined as:

\[
\begin{align*}
\text{min} & \quad z = \sum_{(i,j) \in A} c_{ij} x_{ij} \\
\text{s.a.:} & \quad \sum_{\{j: (i,j) \in A\}} x_{ij} - \sum_{\{j: (j,i) \in A\}} x_{ji} = b(i) \quad \forall i \in N \\
& \quad 0 \leq x_{ij} \leq u_{ij} \quad \forall (i,j) \in A
\end{align*}
\]

\[(1a) \quad (1b) \quad (1c)\]
MCNFP: Assumptions

1. All data (cost, supply, demand and capacity) are integral (needed for some proofs, and some running time analysis).
2. The network is directed.
3. The supplies/demands at the nodes satisfy the condition
   \[ \sum_{i \in N} b(i) = 0 \]
   and the minimum cost flow problem has a feasible solution.
4. The network \( G \) contains an uncapacitated directed path between every pair of nodes
   - If necessary artificial uncapacitated arcs \((1, j)\) and \((j, 1)\) \( \forall j \in N \) will be added with infinity costs.
5. All arc costs \( c_{ij} \) are nonnegative.
Artificial Solutions

- To create a feasible solution, add a dummy node $d$.
- Add an arc from $d$ to each demand node, each with a large cost $M$, and large capacity.
- In an optimal solution, arcs with large cost will have a flow of 0.
Residual network

- Residual capacity $r_{ij}$:

\[ c_{ij}, u_{ij} \]

\[ x_{ij} \]

\[ c_{ij}, r_{ij} = u_{ij} - x_{ij} \]

\[ -c_{ij}, r_{ij} = x_{ji} \]

- Residual network $G(x)$: nodes $N$ and arcs $(i, j)$ replaced with arcs $(i, j)$ and $(j, i)$ with positive residual capacity $r_{ij}$
**Augmenting Cycle Theorem**

- **Def:** Augmenting Cycle $W$ in $G$ w.r.t. $x$: any cycle $W$ (not necessarily oriented) s.t. “$x + f(W)$” feasible.
  
  - Prop.: $W$ augmenting cycle in $G$ w.r.t. $x$ $\iff$ $W$ directed cycle in $G(x)$

- **Th. (Augmenting Cycle Theorem.):** Let $x$ and $x^0$ be any two feasible solutions of a network flow problem. Then $x$ equals $x^0$ plus the flow of at most $m$ directed cycles in $G(x^0)$. Furthermore, the cost of $x$ equals the cost of $x^0$ plus the cost of flow on these augmenting cycles.

**Proof:** AMO, page 83

**Interpretation:**

\[
\begin{align*}
  c(x^0) + c(f(W)) &= 35 - 6 = c(x) \\
  G(x^0) 
\end{align*}
\]
Negative cycle optimality conditions

- **Ta (Negative Cycle Optimality Cond.):** A feasible solution $x^*$ is an optimal solution of the minimum cost flow problem if and only if satisfies the negative cycle optimality conditions: namely, the residual network $G(x^*)$ contains no negative cost (directed) cycle.

**Proof:**

$\Rightarrow$: a pos. flow along any neg. cycle of $G(x)$ always improve the o.f.: if $x^*$ optimal, $G(x^*)$ cannot contain a neg. cycle.

$\Leftarrow$: Assume $x^*$ feasible and $G(x^*)$ without neg. cycle. Then, by the Augmenting Cycle Th., for any feasible flow $x \neq x^*$ holds

$$c(x) = c(x^*) + c(f(W)) \geq c(x^*)$$

and thus, $x^*$ is optimal
Reduced cost optimality conditions (1/2)

- **Potentials nodes:** \( \pi(i) \quad \forall i \in N \)

- **Reduced costs:** \( c_{ij}^\pi = c_{ij} - \pi(i) + \pi(j) \quad \forall (i, j) \in A \)
  
  - Red. cost in the red. Net. \( G(x) \): \( c_{ji}^\pi = -c_{ij} \quad \forall (i, j) \in G(x) \)

- **Properties:**
  
  a) For any directed path \( P \) from node \( k \) to node \( l \),
  
  \[
  \sum_{(i,j) \in P} c_{ij}^\pi = \sum_{(i,j) \in P} c_{ij} - \pi(k) + \pi(l)
  \]

  b) For any directed cycle \( W \),
  
  \[
  \sum_{(i,j) \in W} c_{ij}^\pi = \sum_{(i,j) \in W} c_{ij}
  \]

- **Consequences of properties (a) and (b):**
  
  i. (a) \( \Rightarrow \) \( P \) is the shortest path w.r.t. \( c \) \( \Leftrightarrow \) \( P \) is the shortest path w.r.t. \( c^\pi \)
  
  ii. (b) \( \Rightarrow \) \( W \) is a neg. cycle w.r.t. \( c \) \( \Leftrightarrow \) \( W \) is a neg. cycle w.r.t. \( c^\pi \)
**Reduced cost optimality conditions (2/2)**

- **Th. (Reduced Cost Optimality Cond.):** A feasible solution $x^*$ is an optimal solution of the minimum cost flow problem if and only if some set of node potentials $\pi$ satisfy the following reduced cost optimality conditions:

  $$c^\pi_{ij} \geq 0 \quad \forall (i, j) \in G(x^*) \quad (1)$$

**Proof:**

- We will see the equivalence with the Neg. Cycle. Opt. Cond (NCOC).
- **RCOC $\Rightarrow$ NCOC:**
  - $x^*$ optimal w.r.t. The RCOC $\Rightarrow$ $\sum_{(i,j) \in W} c^\pi_{ij} \geq 0 \quad \forall$ directed cycle $W \in G(x^*)$
  - Prop. (b) $\Rightarrow$ $\sum_{(i,j) \in W} C_{ij}^\pi = \sum_{(i,j) \in W} c_{ij} \geq 0 \quad \Rightarrow G(x^*)$ contains no negative cycle.

- **RCOC $\Leftarrow$ NCOC:**
  - $x^*$ NCOC $\Rightarrow G(x^*)$ contains no negative cycles.
  - Let $d(\cdot)$ be the shortest path labels of the label correcting algorithm from node 1 over $G(x^*)$ and let $\pi = -d$. Then, by the convergence theorem of the label-correcting alg. (1) holds true.

**Remark:** relation between the RCOC over $G(x^*)$ and the simplex optimality conditions for saturated arcs $x_{ij} = u_{ij}$
• **Th. (Complementary Slackness Optimality Conditions):** A feasible solution $x^*$ is an optimal solution of the minimum cost flow problems if and only if for some set of node potentials $\pi$, the reduced costs and flow values satisfy the following complementary slackness optimality conditions for every arc $(i, j) \in A$:

\[
\begin{align*}
&\text{If } c_{ij}^\pi > 0 \quad , \quad \text{then } x_{ij}^* = 0 \\
&\text{If } 0 < x_{ij} < u_{ij} \quad , \quad \text{then } c_{ij}^\pi = 0 \\
&\text{If } c_{ij}^\pi < 0 \quad , \quad \text{then } x_{ij}^* = u_{ij}
\end{align*}
\]

**Proof:** AMO, pag. 310 (quite technical!!)

• Relation with the optimality condition of the simplex algorithm
Complementary Slackness optimality conditions (CSOC)

If $c_{ij}^\pi > 0$, then $x_{ij}^* = 0$

If $0 < x_{ij} < u_{ij}$, then $c_{ij}^\pi = 0$

If $c_{ij}^\pi < 0$, then $x_{ij}^* = u_{ij}$

Kilter Diagram for arc $(i, j)$
Complementary Slackness optimality conditions (CSOC)

\[
(i, j) \quad \equiv \quad (x_{ij}, c_{ij}^\pi)
\]

\[
(i, j)^{B-D-E} \rightarrow \text{in} - \text{kilter}
\]

\[
(i, j)^{A-C} \rightarrow \text{out} - \text{of} - \text{kilter}
\]
Kilter Number \((k(i, j))\)

- **Definition,**
  - The \(k(i, j)\) is a magnitude of the change in \(x(i, j)\) required to make the arc \((i, j)\) an in-kilter arc while keeping \(C(i, j, \pi)\) fixed.

- **Consequently,**
  - **If \((i, j)\) is in-kilter then** \(k(i, j) = 0\).
  - We can measure how far we are from the optimal solution as follows:
    \[
    K = \sum_{(i,j) \in A} k(i, j)
    \]
  - The smaller the value of \(K\), the closer we are from the optimal solution.
Computation of $k(i, j)$'s

- $k(i, j)$ has a close relation with CSOC as follows:
  1) if $c^\pi_{ij} > 0$ then $k_{ij} = |x_{ij}|$
  2) if $c^\pi_{ij} < 0$ then $k_{ij} = |u_{ij} - x_{ij}|$
  3) 3.1) if $c^\pi_{ij} = 0$ & $x_{ij} > u_{ij}$ then $k_{ij} = x_{ij} - u_{ij}$
       3.2) if $c^\pi_{ij} = 0$ & $x_{ij} < 0$ then $k_{ij} = -x_{ij}$

- In order to use $k(i, j)$ in the OFKA and thus, in the $G(x)$, it is redefined as follows:

$$k_{ij} = \begin{cases} 
0 & \text{if } c^\pi_{ij} \geq 0 \\
r_{ij} & \text{if } c^\pi_{ij} < 0
\end{cases} \quad \forall (i, j) \in G(x)$$
Cases

$x(i, j) = -2$, $c(i, j, \pi) > 0 \rightarrow k(i, j) = |-2| = 2$

$x(i, j) = -2$, $c(i, j, \pi) = 0 \rightarrow k(i, j) = |\pi - (-2)| = 2$

$x(i, j) = -2$, $c(i, j, \pi) < 0 \rightarrow k(i, j) = |4 - (-2)| = 6$

$x(i, j) = 3$, $c(i, j, \pi) > 0 \rightarrow k(i, j) = |3| = 3$

$x(i, j) = 3$, $c(i, j, \pi) = 0 \rightarrow k(i, j) = 6 - 4 = 2$

$x(i, j) = 3$, $c(i, j, \pi) < 0 \rightarrow k(i, j) = 2$

$x(i, j) = -2$, $c(i, j, \pi) = 0 \rightarrow k(i, j) = 2$

$x(i, j) = -2$, $c(i, j, \pi) < 0 \rightarrow k(i, j) = 6$
Relation with $G(x)$

- $x_{ij} = 0$
  - $c_{ij}(\pi) \geq 0$
  - $(x$ factible$)$

- $0 < x_{ij} < u_{ij}$
  - $r_{ij} = u_{ij} - x_{ij}$
  - $r_{ji} = x_{ij}$

- $x_{ij} = u_{ij}$
  - $c_{ij}(\pi) \leq 0$
  - $c_{ji}(\pi) \geq 0$

- $x_{ij} = u_{ij}$
  - $c_{ij}(\pi) = -c_{ji}(\pi) \leq 0$

- $x_{ij} = u_{ij}$, $c_{ij}(\pi) = -c_{ji}(\pi) \leq 0$
Out-of-kilter algorithm

algorithm out-of-kilter;
begin

\[ \pi : = 0; \]

establish a feasible flow \( x \) in the network;

define the residual network \( G(x) \) and compute the kilter numbers of arcs;

while the network contains an out-of-kilter arc do
begin

select an out-of-kilter arc \((p, q)\) in \( G(x) \);

define the length of each arc \((i, j)\) in \( G(x) \) as \( \max \{0, C(i, j, \pi)\} \);

let \( d(\cdot) \) denote the shortest path distances from node \( q \) to all other nodes in \( G(x) - \{(q, p)\} \) and let \( P \) denote a shortest path from node \( q \) to node \( p \);

update \( \pi'(i) : = \pi(i) - d(i) \) for all \( i \in N \);

if \( C(p, q, \pi') < 0 \) then
begin

\( W : = P \cup \{(p, q)\}; \)

\( \delta : = \min \{ r(i, j) : (i, j) \in W \}; \)

augment \( \delta \) units of flow along \( W \);

update \( x, G(x) \) and the reduced costs;

end;

end;
end;
Minimum Cost Flow Duality

- Def. (Dual Minimum Cost Flow Problem):

\[
\begin{align*}
\text{max} & \quad w(\pi, \alpha) = \sum_{i \in N} b(i)\pi(i) - \sum_{(i, j) \in A} u_{ij}\alpha_{ij} \\
\text{s.t.:} & \quad \pi(i) - \pi(j) - \alpha_{ij} \leq c_{ij} \quad \forall (i, j) \in A \\
& \quad \alpha_{ij} \geq 0 \quad \forall (i, j) \in A \\
& \quad \pi(j) \text{ unrestricted} \quad \forall j \in N \\
\end{align*}
\]

- It is easy to verify that this is the dual of the standard form of the MCNFP (exercise).
Minimum Cost Flow Duality

• **Th. (Weak Duality Theorem):** Let $z(x)$ denote the objective function of some feasible solution $x$ of the minimum cost flow problem and let $w(\pi, \alpha)$ denote the objective function value of some feasible solution $(\pi, \alpha)$ of its dual. Then $w(\pi, \alpha) \leq z(x)$.

• **Th. (Strong Duality Theorem):** Any minimum cost flow problem with optimal solution $x^*$ has a dual minimum cost flow problem with optimal solution $\pi$ satisfying the property that $z(x^*)=w(\pi)$.

• **Th. (Rel. Duality - Comp. Slack. O.C.):** If $x^*$ is an optimal solution of the minimum cost flow problem, and $\pi$ is an optimal solution of the dual minimum cost flow problem, the pair $(x^*, \pi)$ satisfies the complementary slackness optimality conditions.
Relaxation Algorithm (1)

- **Def. Lagrangian relaxed problem** \( LR(\pi) \):

\[
\begin{align*}
\pi & = \min \left\{ \sum_{(i,j) \in A} c_{ij} x_{ij} + \sum_{i \in N} \pi(i) \left( b(i) - \sum_{\{j: (j,i) \in A\}} x_{ij} + \sum_{\{j: (j,i) \in A\}} x_{ji} \right) \mid 0 \leq x_{ij} \leq u_{ij} \, \forall \,(i,j) \in A \right\}
\end{align*}
\]

- **Alternative expressions of** \( LR(\pi) \):

\[
\begin{align*}
\pi & = \min \left\{ \sum_{(i,j) \in A} c_{ij} x_{ij} + \sum_{i \in N} \pi(i) e(i) \mid 0 \leq x_{ij} \leq u_{ij} \, \forall \,(i,j) \in A \right\} \quad (1) \\
\pi & = \min \left\{ \sum_{(i,j) \in A} c^\pi_{ij} x_{ij} + \sum_{i \in N} \pi(i) b(i) \mid 0 \leq x_{ij} \leq u_{ij} \, \forall \,(i,j) \in A \right\} \quad (2)
\end{align*}
\]

- **Solution of problem** \( LR(\pi) \): trivial, from formulation (2).

\[
\begin{align*}
x_{ij}^* & = \begin{cases} 
0 & \text{if } c^\pi_{ij} > 0 \\
u_{ij} & \text{if } c^\pi_{ij} < 0 \\
\text{any } x \in [0, u_{ij}] & \text{if } c^\pi_{ij} = 0
\end{cases}
\end{align*}
\]

**Property 1:** If a pseudoflow \( x \) of the minimum cost flow problem satisfies the **reduced cost optimality conditions** for some \( \pi \), then \( x \) is an optimal solution of \( LR(\pi) \).
Relaxation Algorithm (2)

• **Lemma:**
  
  a) For any node potentials \( \pi \), \( w(\pi) \leq z^* \)
  
  b) For some choice of node potentials \( \pi^* \), \( w(\pi^*) = z^* \)

  **Proof:**
  
  a) If \( x^* \) opt. sol. of the MCNFP \( \Rightarrow \forall \pi \), \( x^* \) feasible for \( LR(\pi) \) and \( w(\pi) \leq z^* \).
  
  b) Let \( \pi^*, x^* \) satisfy the RCOC \( \Rightarrow \) By Prop. 1: \( x^* \) is the opt. solution of \( LR(\pi^*) \Rightarrow w(\pi^*) = z^* \).

• **Rationale of the Relaxation algorithm:**
  
  – Maintains a pair \((\pi, x)\), \( x \) a pseudoflow, s.t. satisfy the RCOC
  
  – It modifies \( \pi \) to \( \pi' \) and \( x \) to \( x' \) so that \( x' \) is an optimal solution of \( LR(\pi') \) and \( w(\pi') > w(\pi) \).
  
  – Keeping \( \pi \) unchanged, it modifies \( x \) to \( x' \) so that \( x' \) is also an optimal solution of \( LR(\pi) \) and the excess of at least one node decreases
Relaxation Algorithm (3)

At some stage of the algorithm:
- \( S \): nodes of a tree \( T \) rooted at node \( s \) s.t.
  - Every node \( s \in S : e(s) \geq 0 \).
  - Every arc \((i,j) \in T : c_{ij}^\pi = 0\)
- \( S = N - S ; (S,S) \) onward arcs of cut \([S, S]\) in \( G(x)\); \((S,S) \) backward arcs of cut \([S, S]\) in \( G(x)\).

\[
e(S) = \sum_{i \in S} e(i) \quad ; \quad r(\pi, S) = \sum_{(i,j) \in (S,S)} r_{ij}
\]

\[
S \quad 3 \quad 2 \quad 0 \quad 3 \quad 4 \quad 0 \quad 5 \quad 6 \quad 0 \quad 7 \quad 8
\]

\[
s \quad 8 \quad 1 \quad 3 \quad (0,2) \quad 0 \quad 2 \quad (0,3) \quad 4 \quad (2,4) \quad 2 \quad (3,2)
\]

\[
e(i) \quad (c_{ij}^\pi, r_{ij}) \quad e(j)
\]
**Case 1:** if $e(S) > r(\pi, S)$ then **increase** $w(\pi)$

- Modify $x$ saturating all $(i,j) \in (S,S)$ s.t. $c_{ij}^\pi = 0$ ($w(\pi)$ invariant, due to (2))
- The change in $x$ **reduces** $e(S)$: $e(S) \leftarrow e(S) - r(\pi, S) \geq 0$. 
Relaxation Algorithm (5)

- **Case 1:** if \( e(S) > r(\pi, S) \) then *increase* \( w(\pi) \)
  - Modify \( x \) saturating all \( (i,j) \in (S,S) \) s.t. \( c_{ij}^\pi = 0 \) (\( w(\pi) \) invariant, due to (2))
  - The change in \( x \) reduces \( e(S) : e(S) \leftarrow e(S) - r(\pi, S) \geq 0 \).
  - At this point \( c_{ij}^\pi > 0 \) \( \forall (i,j) \in (S,S) \): \( \alpha := \min \{ c_{ij}^\pi | (i,j) \in (S,S) \} > 0 \) (*)
  - Increase \( \pi(i) : \pi'(i) := \pi(i) + \alpha \ \forall i \in S \)

  \( w(\pi) \exp. (1) : w(\pi') := w(\pi) + [e(S)-r(\pi, S)] \alpha > w(\pi) \)
Relaxation Algorithm (6)

- **Case 1**: if $e(S) > r(\pi, S)$ then **increase** $w(\pi)$
  - Modify $x$ saturating all $(i, j) \in (S, S)$ s.t. $c^{\pi}_{ij} = 0$ ($w(\pi)$ invariant, due to (2))
  - The change in $x$ **reduces** $e(S)$: $e(S) \leftarrow e(S) - r(\pi, S) \geq 0$.
  - At this point $c^{\pi}_{ij} > 0 \forall (i, j) \in (S, S)$: $\alpha := \min\{ c^{\pi}_{ij} | (i, j) \in (S, S) \} > 0$ (***)
  - Increase $\pi(i)$: $\pi'(i) := \pi(i) + \alpha \forall i \in S$
    - $w(\pi)$ exp. (1): $w(\pi') := w(\pi) + [e(S)-r(\pi, S)] \alpha > w(\pi)$
    - The **RCOC is conserved**: $c^{\pi}_{ij} := c^{\pi}_{ij} - \alpha \geq 0 \forall (i, j) \in (S, S)$ due to (***)
      (and the rest of $c^{\pi}_{ij}$ cannot be < 0)
Case 2: if $e(S) \leq r(\pi, S)$:
- $r(\pi, S) \geq e(S) > 0 \Rightarrow \exists (i,j) \in (S, S)$ s.t. $c_{ij}^\pi = 0$
- If $e(j) \geq 0$ then: $S := S + \{j\}$
- If $e(j) < 0$ then: augmenting flow $\delta$ trough $P(s \to j)$
  - $e(S)$ decreases.
  - $w(\pi)$ invariant ($c_{ij}^\pi = 0 \ \forall (i,j) \in P(s \to j)$
Relaxation algorithm: given a MCNFP in standard form.

begin
  \( x := 0 \) and \( \pi := 0 \);
  while the network contains a node \( s \) with \( e(s) > 0 \) do
  begin
    \( S := \{ s \} \);
    if \( e(S) > r(\pi, S) \) then adjust-potencial
    else
      repeat
        select an arc \( (i,j) \in (S,S) \) in the residual network with \( c_{ij}^\pi = 0 \);
        if \( e(j) \geq 0 \) then set \( \text{pred}(j) := i \) and add node \( j \) to \( S \);
      until \( e(j) < 0 \) or \( e(S) > r(\pi, S) \);
  end;
  if \( e(S) > r(\pi, S) \) then adjust-potential
  else adjust-flow;
end;

procedure adjust-potential;
begin
  for every arc \( (i,j) \in (S,S) \) with \( c_{ij}^\pi = 0 \) do send \( r_{ij} \) units of flow on the arc \( (i,j) \);
  compute \( \alpha := \min \{ c_{ij}^\pi : (i,j) \in (S,S) \) and \( r_{ij} > 0 \} \); for every node \( j \in S \) do \( \pi(j) := \pi(j) + \alpha \);
end;

procedure adjust-flow;
begin
  identify directed \( P(s \rightarrow j) \) from \( \text{pred}() \);
  \( \delta := \min \{ e(s), -e(j), \min \{ r_{ij} : (i,j) \in P \} \} \);
  augment \( \delta \) along \( P \);
  update imbalances and residual capac.;
end;

where: \( e(S') = \sum_{i \in S} e(i) \) ; \( r(\pi, S') = \sum_{(i,j) \in (S',S)} r_{ij} \) ; \( c_{ij}^\pi = 0 \)
Relaxation algorithm - convergence

- **Convergence:** The relaxation algorithm terminates with an optimal flow $x^*$. 
  
  **Proof:**
  
  i. The rel. Alg. terminates with $e(s)=0 \ \forall s \in N \Rightarrow x^*$ is a feasible flow.
  
  ii. The pair $(\pi^*, x^*)$ satisfies the RCOC.

- **Finite convergence:** the relaxation algorithm terminates in a finite number of iterations (for problems with integral data).

- **Worst case analysis:** $O(m^2nCU^2)$ (worst than cycle-cancelling, succ. shortest path and primal-dual)
Relaxation algorithm – example

(a) Choose \( S = \{1\} \) and adjust potential of node 1. (b) Choose \( S = \{1\} \) and augment 20 units of flow along arc \((1, 3)\) and then increase the potential of node 1 by one unit.
Relaxation algorithm – example (cont.)

(e) Add node 2 to $S$ and increase potential of every node in $S$ by 4 units.

(d) Add node 4 to $S$ and increase the potential of every node in $S$ by one unit.

(f) Increase the potential of node 3 by $\alpha = 2$ units after choosing $S = \{s\} = \{3\}$.

(g) Add node 5 to $S$ and augment 20 units of flow along the path 3-5.

Figure S9.23
Summary of pseudopolynomial-time MCNF algorithms

The algorithms studied so far are **pseudopolynomial-time algorithms**, that is, its worst-case complexity is bounded by a **polynomial of** $n$, $m$, $C$ and $U$.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Worst-case complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycle cancelling</td>
<td>$O(nm^2CU)$</td>
</tr>
<tr>
<td>Succ. shortest path</td>
<td>$O(n^3U) \leftarrow O(nU\cdot S(n,m,nC))$</td>
</tr>
<tr>
<td>Primal-dual</td>
<td>$O(mn^3\cdot \min{U,C}) \leftarrow O(\min{nU, nC}\cdot {S(n,m,nC)+M(n,m,U)})$</td>
</tr>
<tr>
<td>Out-of-kilter</td>
<td>$O(mn^2U) \leftarrow O(mU\cdot S(n,m,nC))$</td>
</tr>
<tr>
<td>Relaxation</td>
<td>$O(m^2nCU^2)$</td>
</tr>
</tbody>
</table>

$S(n,m,nC)$: running time of the shortest path alg. with $c_{ij} \geq 0$ ($O(n^2)$ for Dijkstra).

$M(n,m,U)$: running time of a max-flow alg. ($O(mn^2)$ for the SAPA)

Instead, **polynomial-time algorithms** have a worst-case complexity that is bounded by a **polynomial of** $n$, $m$, $\log C$ and $\log U$. 
Polynomial MCNFP algorithms

- Capacity Scaling algorithm.
- Cost Scaling Algorithm (assignment)
- Double Scaling Algorithm (assignment)
The Capacity Scaling Algorithm

- This algorithm is an enhancement of the successive shortest path algorithm.
- It ensures that in each augmentation, sufficiently large flow (at least $\Delta$) is sent from an excess node to a deficit node.
- $\Delta$-residual network $G(x, \Delta)$: subgraph of $G(x)$ consisting of those arcs whose residual capacity is at least $\Delta$.
- The algorithm selects a node $k$ with excess at least $\Delta$, selects a node $l$ with deficit at least $\Delta$, identifies a shortest path in $G(x)$ from node $k$ to node $l$ with residual capacity at least $\Delta$, and augments flow along this path.
- The algorithm terminates when there is no imbalanced node.
The Capacity Scaling Algorithm (contd.)

```plaintext
algorithm  capacity scaling;
begin
    x: = 0 and \( \pi : = 0; \)
    \( \Delta : = 2 \lfloor \log U \rfloor; \)
    while \( \Delta \geq 1 \) do
        begin \{\( \Delta \)-scaling phases}\}
            for every arc \((i, j)\) in the residual network \(G(x)\) do;
                if \( r_{ij} \geq \Delta \) and \( c_{ij}^{\pi} < 0 \) then
                    send \( r_{ij} \) units of flow along \((i, j)\), update \(x\) and the imbalances \(e(i)\) and \(e(j)\);
            end for;
            \( S(\Delta) := \{i \in N : e(i) \geq \Delta\}; \)
            \( T(\Delta) := \{i \in N : e(i) \leq -\Delta\}; \)
            while \( S(\Delta) \neq \emptyset \) and \( T(\Delta) \neq \emptyset \) (and a path \(P\) from \(k\) to \(l\) over \(G(x,\Delta)\)\(\Leftarrow\) Ass.4) do
                Select a node \(k \in S(\Delta)\) and a node \(l \in T(\Delta)\);
                Determine a shortest path distance \(d(.)\) from node \(k\) to all other nodes in the \(\Delta\)-residual network \(G(x,\Delta)\) with respect to the reduced costs \(c_{ij}^{\pi}\);
                Let \(P\) denote a shortest path from node \(k\) to node \(l\) in \(G(x, \Delta)\);
                Update \(\pi := \pi - d\) and \(c_{ij}^{\pi} := c_{ij}^{\pi} + d(i) - d(j)\);
                Augment \(\Delta\) units of flow along the path \(P\);
                Update \(x, S(\Delta), T(\Delta),\) and \(G(x, \Delta)\);
            end while;
            \( \Delta : = \Delta / 2; \)
        end \{\( \Delta \)-scaling phases\};
    end while;
end;
```

The Original Costs and Node Potentials

![Network Flow Diagram]

Original Costs:
- Node 1: 0
- Node 2: 4
- Node 3: 7
- Node 4: 5
- Node 5: 6
- Node 6: 2

Node Potentials:
- Node 2: 0
- Node 3: 4
- Node 4: 0
- Node 5: 1
- Node 6: 2

Minimum Cost Network Flow Problems

MCNFP- 37
The Original Capacities and Supplies/Demands

The figure represents a network flow problem with the following capacities and supplies/demands:

- Capacities:
  - Edge 1-2: 30
  - Edge 2-3: 25
  - Edge 3-4: 20
  - Edge 4-5: 20
  - Edge 1-5: 23

- Supplies/Demands:
  - Node 1: -7
  - Node 2: 10
  - Node 3: -2
  - Node 4: 5
  - Node 5: -19
Set $\Delta = 16$. This begins the $\Delta$-scaling phase.

We send flow from nodes with excess $\geq \Delta$ to nodes with deficit $\geq \Delta$.

We ignore arcs with capacity $\leq \Delta$. 
Select a supply node and find the shortest paths.

The shortest path tree is marked in bold and blue.
Update the Node Potentials and the Reduced Costs

To update a node potential, subtract the shortest path distance.
Send Flow Along a Shortest Path in G(x, 16)

Send flow from node 1 to node 5.

How much flow should be sent?

10
Update the Residual Network

19 units of flow were sent from node 1 to node 5.
This ends the 16-scaling phase.

The $\Delta$-scaling phase continues when $e(i) \geq \Delta$ for some $i$.

$e(j) \leq -\Delta$ for some $j$.

There is a path from $i$ to $j$. 
This begins and ends the 8-scaling phase.

The $\Delta$-scaling phase continues when $e(i) \geq \Delta$ for some $i$. $e(j) \leq -\Delta$ for some $j$. There is a path from $i$ to $j$. 
This begins 4-scaling phase.

What would we do if there were arcs with capacity at least 4 and negative reduced cost?
Select a “large excess” node and find shortest paths.

The shortest path tree is marked in bold and blue.
Update the Node Potentials and the Reduced Costs

To update a node potential, subtract the shortest path distance.
Send Flow Along a Shortest Path in $G(x, 4)$.  

Send flow from node 1 to node 7

How much flow should be sent?
Update the Residual Network

4 units of flow were sent from node 1 to node 3
This ends the 4-scaling phase.

There is no node $j$ with $e(j) \leq -4$. 
Begin the 2-scaling phase

There is no node \( j \) with \( e(j) \leq -4 \).

What would we do if there were arcs with capacity at least 4 and negative reduced cost?
Send flow along a shortest path

Send flow from node 2 to node 4

How much flow should be sent?
Update the Residual Network

2 units of flow were sent from node 2 to node 4
Send Flow Along a Shortest Path

Send flow from node 2 to node 3

How much flow should be sent?
Update the Residual Network

3 units of flow were sent from node 2 to node 3
This ends the 2-scaling phase.

Are we optimal?
Begin the 1-scaling phase.

Saturate any arc whose capacity is at least 1 and with negative reduced cost.

Reduced cost is negative
Update the Residual Network

Send flow from node 3 to node 1.

Note: Node 1 is now a node with deficit.
Update the Residual Network

1 unit of flow was sent from node 3 to node 1.

\[ \Delta := \Delta/2 = \frac{1}{2} \rightarrow \text{END} \]

Is this flow optimal?
The Final Optimal Flow
The Final Optimal Node Potentials and the Reduced Costs

Flow is at upper bound.

Flow is at lower bound.
The Capacity Scaling Algorithm (contd.)

- **Convergence:** as in the successive shortest-path algorithm.
- **Running time:**
  - A $\Delta$-scaling phase performs at most $2(n+m) = O(m)$ augmentations:
    - In the $\Delta$-scaling phase the sum of excesses is bounded by $2(n+m)\Delta$ (see text).
    - Each augmentation reduces the excess by at least $\Delta$ units.
  - Each augmentation solves a shortest path problem, $S(n,m,nC)$
  - At most $\log U$ $\Delta$-scaling phases.
  - In the overall: $O(m \log U S(n,m,nC))$
- **Theorem:** the capacity scaling algorithm solves the minimum cost flow problems in $O(m \log U S(n,m,nC))$.
  - Successive shortest-path alg.: $O(n U S(n,m,nC))$ (pseudo-polynomial).

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>U</th>
<th>$m \log U$</th>
<th>n$U$</th>
<th>$nU/m \log U$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>2000</td>
<td>1,0E+01</td>
<td>6,64E+03</td>
<td>1,00E+04</td>
<td>1,51E+00</td>
</tr>
<tr>
<td>1000</td>
<td>2000</td>
<td>1,0E+02</td>
<td>1,33E+04</td>
<td>1,00E+05</td>
<td>7,53E+00</td>
</tr>
<tr>
<td>1000</td>
<td>2000</td>
<td>1,0E+03</td>
<td>1,99E+04</td>
<td>1,00E+06</td>
<td>5,02E+01</td>
</tr>
<tr>
<td>1000</td>
<td>2000</td>
<td>1,0E+04</td>
<td>2,66E+04</td>
<td>1,00E+07</td>
<td>3,76E+02</td>
</tr>
<tr>
<td>1000</td>
<td>2000</td>
<td>1,0E+05</td>
<td>3,32E+04</td>
<td>1,00E+08</td>
<td>3,01E+03</td>
</tr>
<tr>
<td>1000</td>
<td>2000</td>
<td>1,0E+06</td>
<td>3,99E+04</td>
<td>1,00E+09</td>
<td>2,51E+04</td>
</tr>
</tbody>
</table>
MCNF:
Network Simplex Algorithm. (Cap. 11 AMO)

- **Standard form.**
- **Spanning Tree Structure.**
- **Simplex specialization:**
  - Computing the basic feasible solution.
  - Reduced cost calculation.
  - Leaving variable selection.
  - Base change.
  - Simplex Algorithm for Network Flows.
  - Example.
  - Exercises.
Standard form of the Minimum Cost Flow Problem

- Mathematic formulation:

\[
\begin{align*}
\min & \quad z = \sum_{(i,j) \in A} c_{ij} x_{ij} \\
\text{s.a.:} & \quad \sum_{\{j: (i,j) \in A\}} x_{ij} - \sum_{\{j: (j,i) \in A\}} x_{ji} = b_i \quad \forall i \in N \\
& \quad x_{ij} \geq 0 \quad \forall (i, j) \in A
\end{align*}
\] (1.1a, 1.1b, 1.1c)

- Assuming that the network is balanced: \( \sum_{i=1}^{n} b_i = 0 \)

- Matrix notation:

\[
\begin{align*}
\min & \quad c'x \\
\text{s.a.:} & \quad Nx = b \\
& \quad x \geq 0
\end{align*}
\] (1.2a, 1.2b, 1.2c)
Spanning tree and bases.

- **Ta**: "Each spanning tree $T$ of $G$ define a base of the minimum cost flow problem. Conversely, each base of the minimum cost flow problem define a spanning tree of $G."$

![Graph](image)

$$B = \begin{pmatrix}
1 & 1 & -1 & 0 & 0 \\
2 & -1 & 0 & 0 & 1 \\
3 & 0 & 1 & 1 & 0 \\
4 & 0 & 0 & 0 & -1 \\
5 & 0 & 0 & -1 & 0
\end{pmatrix}$$

- **Ta**: "The rows and columns of the $B$ matrix linked to a spanning tree $T$ it can be rearranged so that $B$ is triangular"

Procedure:
- Nodes are arranged in crossbar **order**: 3-5-2-4-1
- The arcs are arranged visiting the nodes according to the crossbar order. Each node is associated with the only arc that relates it with its **predecessor**: (3,1), (3,5), (1,2), (2,4)
Feasible basic solution calculation.

- Procedure:
  1. Select a leaf node.
  2. Assign flow to its associated arc.
  3. Modify the predecessor contribution.
  4. Eliminate the node and the arc treated.

\[ x_B^T = [10, 5, 5, 5, 10, 15] \]
(second crossbar order)
Reduce cost computation (I)

- **Reduced cost:**

\[ r_{ij} = c_{ij} - \pi^T \mathcal{N}_{ij} = c_{ij} - \pi_i + \pi_j , \forall (i,j) \in A \]

- Any constant \( k \) it can be summed to the Lagrange multipliers \( \pi_i \) without change the value of \( r_{ij} \):

\[ r_{ij} = c_{ij} - \pi_i + \pi_j = c_{ij} - (\pi_i+k) + (\pi_j+k) \]

- This implies that we can fix arbitrarily the value of the Lagrange multiplier in a node:

\[ \pi_i = 0 \]

- The rest of the Lagrange multipliers can be calculated using the reduced cost expression of the basic arcs:

\[ r_{ij} = c_{ij} - \pi_i + \pi_j = 0 , \forall (i,j) \in T \]
Langrange multipliers computation (II)

- Procedure:
  1. Assign the value \( \pi_i = 0 \) to the source node.
  2. Visit the following node according to the cross bar order.
  3. Compute the Lagrange multiplier based on:
     \[
     c_{ij} - \pi_i + \pi_j = 0
     \]
  4. Go to step 2.

\[
\pi^T = [ -5, -3, -7, -2, -2, -1 ]
\]
(Cross bar order)
Leaving variable selection

- No basic arc \((k,l)\) introduction to base with flow \(\theta\):
  \[
x_B(\theta) = B^{-1}b - B^{-1} N_{kl} \theta = x_B - y_{kl} \theta = x_B + \Delta x_B(\theta)
\]

- The vector \(y_{kl}\) is defined by the fundamental cycle associated to \((k,l)\):
  - If \((k,l) = (7,4)\) and \(\theta = +1\):
    \[
    \Delta x_B(+1) = -y_{kl} = \begin{bmatrix} 0, -1, 0, -1, +1, +1 \end{bmatrix}^T
    \]
  - \(\theta\) is increase until the flow of a basic arc is cancelled.
    - In the example, the arc\((5,2)\) would leave and \((7,4)\) would enter to the base with a value of 2.
Base change

- Pivotation between the outgoing arc (5,2) and the entering arc (7,4):

- The values of $x_B$ are updated, $\pi$ and $r$ from the new spanning tree.
Simplex algorithm for network flows

Begin

A initial feasible tree T is defined;

While there are no basic arcs \((i,j) \notin T\) with \(r_{ij} < 0\) Do

The Lagrange multipliers \(\pi\) are defined and the flow \(x\) associated to \(T\);

An entering arc \((k,l)\) is selected with \(r_{kl} < 0\);

The arc \((k,l)\) is added to the tree and the outgoing arc \((p,q)\);

The tree \(T\) is updated;

End while

End
Example

Solve the problem of network flows associated to the following network.

(Solution)
Exercise

1. Formulate a network flow feasible problem of minimum cost with 6 nodes and 9 arcs.
2. Find a initial spanning tree feasible.
3. Solve the problem from the spanning tree found in the last section with the simplex algorithm for network flows.
4. Find the optimal solution with the Giden support.
Example (I)

- Solve the network flows problem associated to the network.

- Begin:
  - Initial feasible tree:
    \[ T = \{(1,2),(2,3),(2,5),(2,4),(4,6)\} \]
  - \( x_B = [9,0,2,7,5]^T \)
  - \( \pi = [0,-3,-5,-5,-8,-11]^T \)

- 1st iteration:
  - \( A \setminus T = \{(1,3),(3,5),(5,4),(5,6)\} \)
  - \( r = [-3, 4, 2, -2]^T \Rightarrow (1,3) \) entering arc
  - (2,3) outgoing arc
Example (II)

- **2\textsuperscript{nd} iteration:**
  - Feasible tree: \( T = \{(1,3),(1,2),(2,5),(2,4),(4,6)\} \)
  - \( x_B = [0,9,2,7,5]^T \) (invariant for all arcs \( \notin \) fundamental cycles)
  - \( \pi = [0,-2,-3,-5,-8,-11]^T \) (invariant from node 2)
  - \( A \setminus T = \{(2,3),(3,5),(5,4),(5,6)\} \)
  - \( r = [3,1,2,-2]^T \Rightarrow (5,6) \) entering arc
  - \( (4,6) \) outgoing arc.

- **3\textsuperscript{rd} iteration:**
  - Feasible tree: \( T = \{(1,3),(1,2),(2,5),(2,4),(4,6)\} \)
  - \( x_B = [0,9,7,5,2]^T \)
  - \( \pi = [0,-2,-3,-5,-9,-8]^T \)
  - \( A \setminus T = \{(2,3),(3,5),(5,4),(4,6)\} \)
  - \( r = [3,1,2,2]^T > 0 \Rightarrow \text{optimum} \)