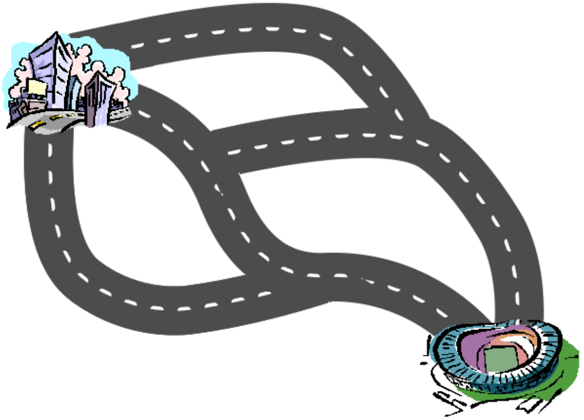
7 Max-Flow Problems



7.1 Max-Flow Problems

- In what follows, we consider a somewhat modified problem constellation
- Instead of costs of transmission, vector c now indicates a maximum capacity that has to be obeyed
- Again, we consider a network with two specifically assigned vertices *s* and *t*
- The objective is to find a maximum flow from source *s* to sink *t*
- E.g., this flow may be a transport of materials from an origin to a destination of consumption

Flow – Inflow and outflow

7.1.1 Definition

Assuming a network $N=(V,E,c)$ is given as above. A mapping $f:E\rightarrow[0,\infty]$ is denoted as an (s,t) flow if and only if the following attributes apply:

1. $0\le f(e)\le c(e),\forall e\in E$
2. $\underbrace{\sum_{j\in V:(i,j)\in E} f((i,j))}_{\text{Outflow from node } i}=\underbrace{\sum_{j\in V:(j,i)\in E} f((j,i))}_{\text{Inflow of node } i},\forall i\in V:i\neq s\wedge i\neq t$

$|f|=\sum_{(s,i)\in E} f((s,i))$ is denoted as the amount of flow. f is denoted as the maximum flow if and only if $|f|$ is maximally chosen.

Observation

- We can transform the equalities (2), which are itemized above, as follows

$$\underbrace{\sum_{j\in V:(i,j)\in E} f((i,j))}_{\text{Outflow from node } i}-\underbrace{\sum_{j\in V:(j,i)\in E} f((j,i))}_{\text{Inflow of node } i}=\begin{cases}|f| & i=s\\ -|f| & i=t\\ 0 & \text{otherwise}\end{cases}$$

Let $\tilde{E}=E\cup\{e_0\},e_0=(t,s)$ and A the vertex - arc adjacency matrix of $(V,\tilde{E})$. Then, $A\cdot f=0^m\wedge f_0=|f|$.

Conclusions

- For what follows, we renumber the arcs, beginning with *1*, i.e., we obtain n arcs with the numbering *1,2,3,…,n*
- Note that this includes the artificial arc 0 (now 1), connecting terminal t with source s
- We know that

$$
\begin{aligned}
&(1,\ldots,1)\cdot A=0\Rightarrow(1,\ldots,1)\cdot A\cdot f=0=(1,\ldots,1)\cdot(A\cdot f)=0\\
&\Rightarrow A\cdot f\le 0\Rightarrow(1,\ldots,1)\cdot(A\cdot f)\le 0\Rightarrow(1,\ldots,1)\cdot A\cdot f\le 0\\
&\Rightarrow(1,\ldots,1)\cdot A\cdot f=0\cdot f=0\Rightarrow A\cdot f=0\\
&\wedge A\cdot f=0\Rightarrow A\cdot f\le 0\Rightarrow A\cdot f\le 0\Leftrightarrow A\cdot f=0
\end{aligned}
$$

Max-Flow Problem

$$\text{Maximize } f_1,\ \text{s.t., } A\cdot f\le 0\wedge f\le c\wedge -f\le 0.$$

$$\text{I.e., }\begin{pmatrix}A\\E_n\\-E_n\end{pmatrix}\cdot f\le\begin{pmatrix}0^m\\c\\0^n\end{pmatrix}, c_1\ge\min\left\{\underbrace{\sum_{j\in V:j>1}c(1,j)}_{\text{Maximum outflow from } s=1},\underbrace{\sum_{i\in V:i<n}c(i,n)}_{\text{Maximum inflow to } n=t}\right\}$$

The dual of Max-Flow

Now, we consider $\tilde{\pi}=(\pi,\gamma,\delta)$, with

$\pi=(\pi_1,\ldots,\pi_m),\gamma=(\gamma_1,\ldots,\gamma_n)$, and $\delta=(\delta_1,\ldots,\delta_n)$

$$\text{Minimize }\sum_{l=1}^{n}c_l\cdot\gamma_l,\ \text{s.t., } A^T\cdot\pi+\gamma-\delta=e^1\wedge(\pi,\gamma,\delta)\ge 0$$

Interpreting the dual

- This time, the dual is given in standard form, i.e., the Simplex Algorithm can be directly applied to it
- Thus, we want to analyze it beforehand
- Let us consider the equalities that have to be fulfilled
- Then, we can transform as follows

$$
\begin{aligned}
&\text{Minimize }\sum_{l=1}^{n}c_l\cdot\gamma_l,\ \text{s.t.,}\\
&\pi_i-\pi_j+\gamma_k-\delta_k=\begin{cases}1 & \text{if } e_k=(t,s)\in E\\ 0 \text{ if } e_k=(i,j)\in E\wedge e_k\ne(t,s)\in E\end{cases}\\
&\wedge\\
&\pi=(\pi_1,\ldots,\pi_m)\ge 0,\gamma=(\gamma_1,\ldots,\gamma_n)\ge 0,\ \text{and } \delta=(\delta_1,\ldots,\delta_n)\ge 0
\end{aligned}
$$

The dual tableau

- Obviously, by conducting the calculation of the Primal Simplex, we obtain a tableau as follows…

$$\begin{array}{c|ccc} 0 & 0 & c^T & 0 \\ \hline e^1 & A^T & E_n & -E_n \end{array}$$

$$\Rightarrow \begin{array}{c|ccc} 0-c_B^T \cdot A_B^{-1} \cdot e^1 & 0-c_B^T \cdot A_B^{-1} \cdot A^T & c^T - c_B^T \cdot A_B^{-1} & 0+c_B^T \cdot A_B^{-1} \cdot E_n \\ \hline A_B^{-1} \cdot e^1 & A_B^{-1} \cdot A^T & A_B^{-1} \cdot E_n & -A_B^{-1} \cdot E_n \end{array}$$

$$\Rightarrow \begin{array}{c|ccc} -f^T \cdot e^1 & -f^T \cdot A^T & c^T - f^T & f^T \\ \hline A_B^{-1} \cdot e^1 & A_B^{-1} \cdot A^T & A_B^{-1} \cdot E_n & -A_B^{-1} \cdot E_n \end{array}$$

Applying the simplex

- The top row of the dual tableau provides comprehensive information about the current state of the calculation
- Specifically, it allows a direct link to the corresponding primal problem which has to be solved originally
- More precisely, we have the following data in the row

$$\Rightarrow \begin{array}{c|ccc} -f^T \cdot e^1 & -f^T \cdot A^T & c^T - f^T & f^T \\ \hline A_B^{-1} \cdot e^1 & A_B^{-1} \cdot A^T & A_B^{-1} \cdot E_n & -A_B^{-1} \cdot E_n \end{array}$$

with:

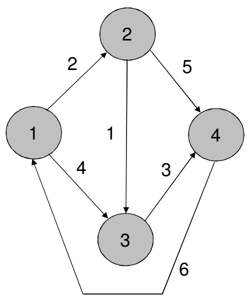
$-f^T \cdot e^1 = -f_1 :$ Objective function value of (P)

$-f^T \cdot A^T :$ Flow balance in the vertices, i.e., is $=0$ for feasible f

$c^T - f^T :$ Remaining capacity of the arcs

$f^T :$ Current corresponding solution to (P)

A simple example

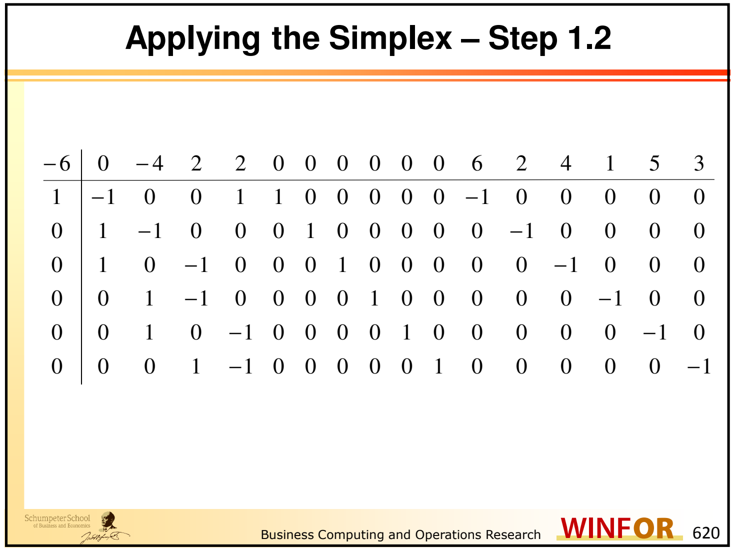


$$A = \begin{pmatrix} -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & -1 & -1 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & -1 \end{pmatrix}$$

$$\wedge\, c = \begin{pmatrix} 6 \\ 2 \\ 4 \\ 1 \\ 5 \\ 3 \end{pmatrix}$$

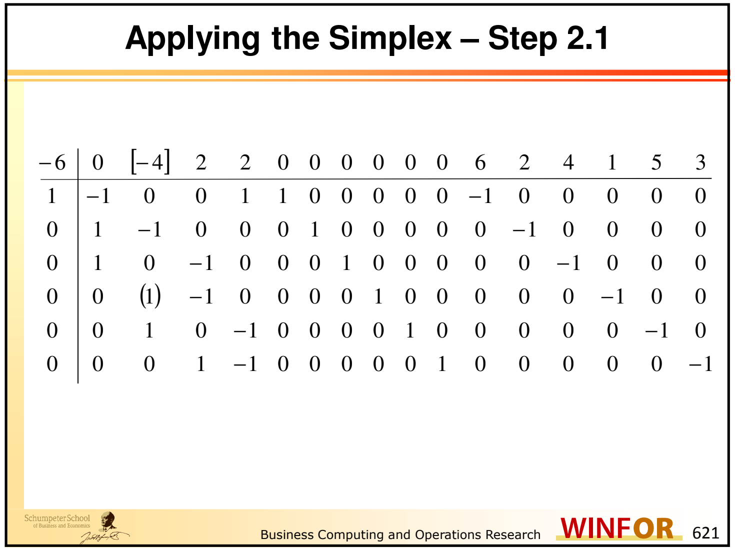
Applying the Simplex – Step 1.1

$$\begin{array}{c|cccccccccccccccc} 0 & 0 & 0 & 0 & 0 & 6 & 2 & 4 & 1 & 5 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \end{array}$$



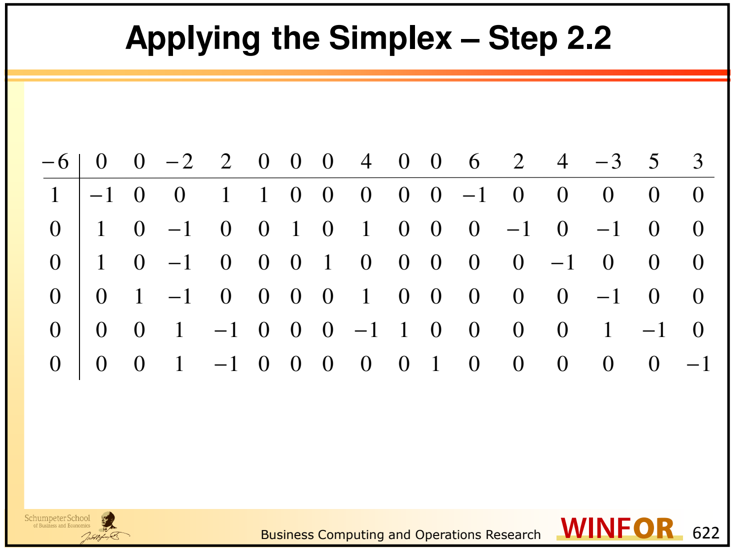
Applying the Simplex – Step 1.2

$$\begin{array}{c|cccccccccccccccc}
-6 & 0 & -4 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 2 & 4 & 1 & 5 & 3 \\ \hline
1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1
\end{array}$$



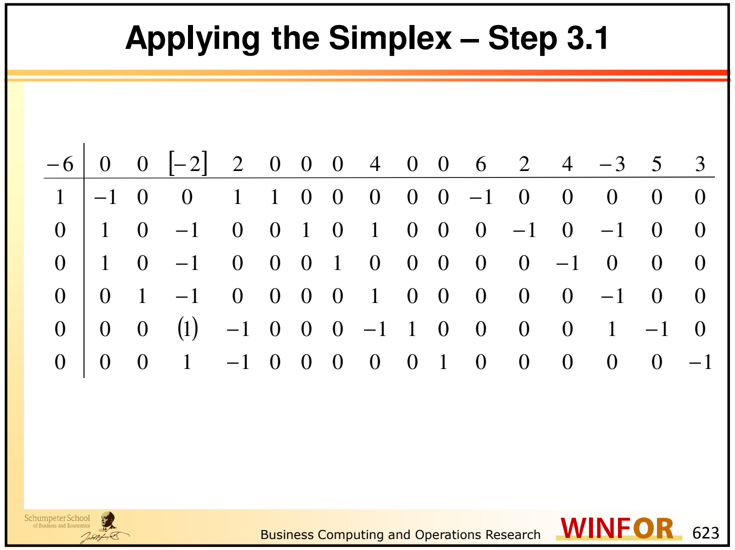
Applying the Simplex – Step 2.1

$$\begin{array}{c|cccccccccccccccc}
-6 & 0 & [-4] & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 2 & 4 & 1 & 5 & 3 \\ \hline
1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & (1) & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1
\end{array}$$



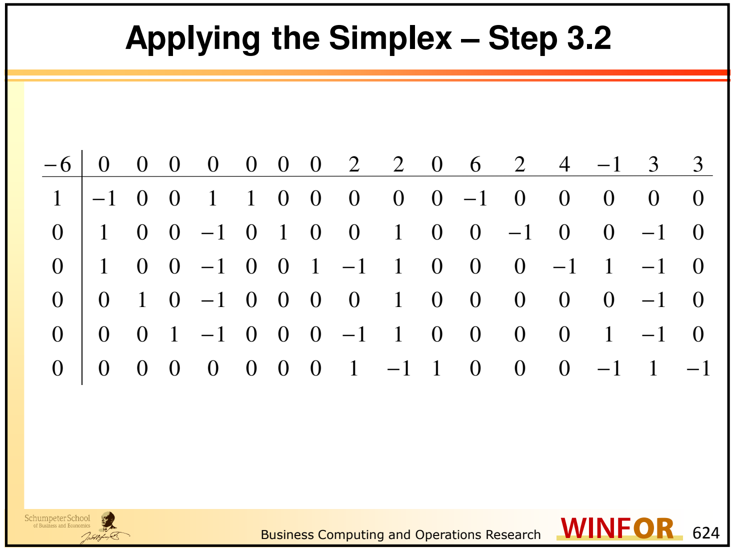
Applying the Simplex – Step 2.2

$$\begin{array}{c|cccccccccccccccc}
-6 & 0 & 0 & -2 & 2 & 0 & 0 & 0 & 4 & 0 & 0 & 6 & 2 & 4 & -3 & 5 & 3 \\ \hline
1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & -1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 \\
0 & 1 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1
\end{array}$$



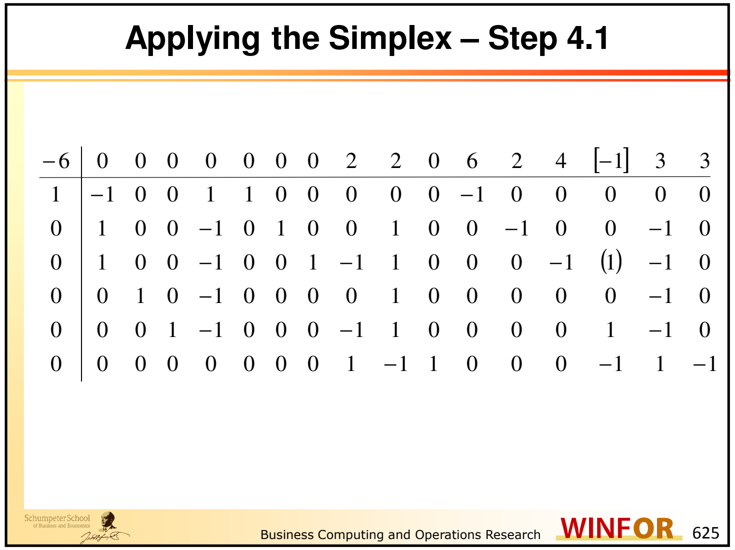
Applying the Simplex – Step 3.1

$$\begin{array}{c|cccccccccccccccc}
-6 & 0 & 0 & [-2] & 2 & 0 & 0 & 0 & 4 & 0 & 0 & 6 & 2 & 4 & -3 & 5 & 3 \\ \hline
1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & -1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 \\
0 & 1 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & (1) & -1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1
\end{array}$$



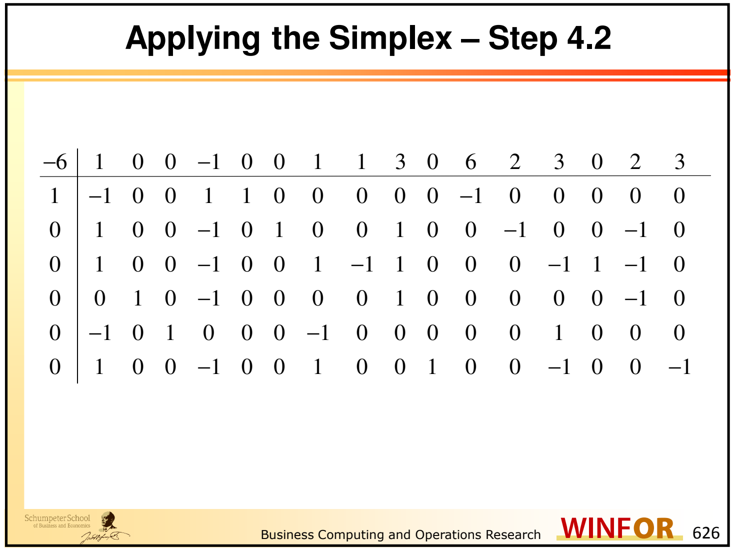
Applying the Simplex – Step 3.2

$$\begin{array}{c|cccccccccccccccc}
-6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 0 & 6 & 2 & 4 & -1 & 3 & 3 \\
\hline
1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 & -1 & 1 & 0 & 0 & 0 & -1 & 1 & -1 & 0 \\
0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 0 & 0 & 0 & -1 & 1 & -1
\end{array}$$



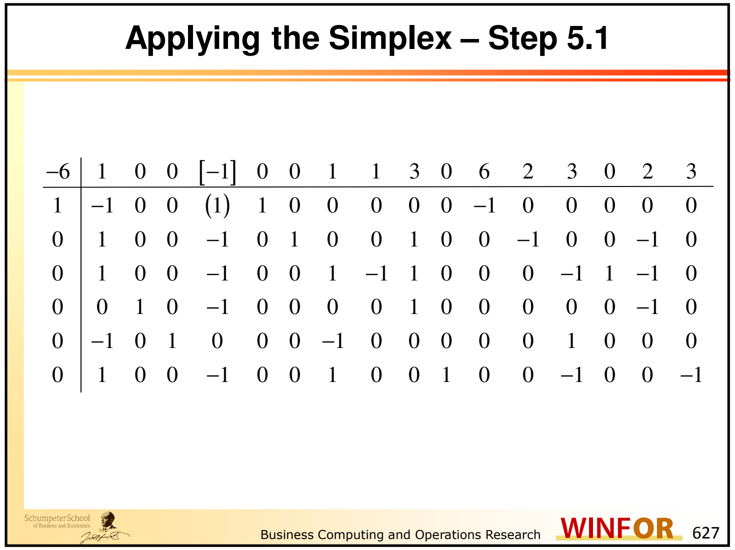
Applying the Simplex – Step 4.1

$$\begin{array}{c|cccccccccccccccc}
-6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 & 0 & 6 & 2 & 4 & [-1] & 3 & 3 \\
\hline
1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 & -1 & 1 & 0 & 0 & 0 & -1 & (1) & -1 & 0 \\
0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 0 & 0 & 0 & -1 & 1 & -1
\end{array}$$



Applying the Simplex – Step 4.2

$$\begin{array}{c|cccccccccccccccc}
-6 & 1 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 3 & 0 & 6 & 2 & 3 & 0 & 2 & 3 \\
\hline
1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 & -1 & 1 & 0 & 0 & 0 & -1 & 1 & -1 & 0 \\
0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & -1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & -1
\end{array}$$



Applying the Simplex – Step 5.1

$$\begin{array}{c|cccccccccccccccc}
-6 & 1 & 0 & 0 & [-1] & 0 & 0 & 1 & 1 & 3 & 0 & 6 & 2 & 3 & 0 & 2 & 3 \\
\hline
1 & -1 & 0 & 0 & (1) & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 & -1 & 1 & 0 & 0 & 0 & -1 & 1 & -1 & 0 \\
0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & -1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & -1
\end{array}$$

Applying the Simplex – Step 5.2

$$
\begin{array}{r|rrrrrrrrrrrrrrrr}
-5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 3 & 0 & 5 & 2 & 3 & 0 & 2 & 3 \\
\hline
1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & -1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 0 & -1 & 0 & -1 & 1 & -1 & 0 \\
1 & -1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & -1 & 0 \\
0 & -1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & -1 & 0 & -1 & 0 & 0 & -1
\end{array}
$$

$f = (5,2,3,0,2,3) \wedge$

$\tilde{\pi} = (\pi,\gamma,\delta) = (0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0)$, i.e.,

$\pi = (0 \quad 1 \quad 0 \quad 1) \wedge \gamma = (0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1) \wedge \delta = (0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0)$

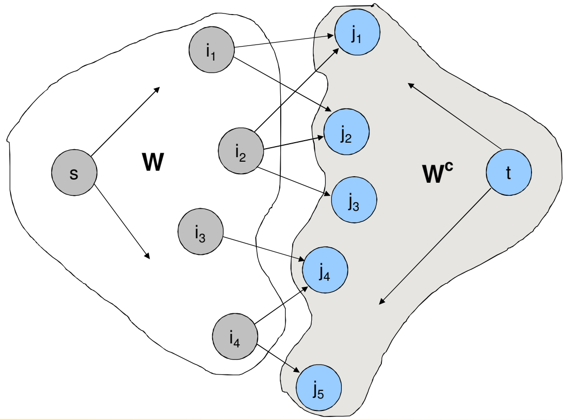
7.2 Min-Cut Problems

7.2.1 Definition:

Assuming $N = (V, E, c, s, t)$ is a network with two labeled nodes s and t. A partition $V = W \cup W^c$ is denoted as an s - t cut if and only if $s \in W$ and $t \in W^c$. $\sum_{(i,j)\in E \text{ with } i\in W \wedge j\in W^c} c(i,j)$ is denoted as the capacity of the cut.

A cut (W, W^c) is denoted as a minimum cut if $\sum_{(i,j)\in E \text{ with } i\in W \wedge j\in W^c} c(i,j)$ is minimal.

Illustration



Problem definition

We introduce $\pi^T = (\pi_1,\ldots,\pi_m)$, with $\pi_i = \begin{cases} 1 & \text{if } i \in W^c \\ 0 & \text{if } i \in W \end{cases}$

and $\gamma^T = (\gamma_1,\ldots,\gamma_n)$, with $\gamma_k = \begin{cases} 1 & \text{if } e_k = (i,j) \wedge i \in W \wedge j \in W^c \\ 0 & \text{otherwise} \end{cases}$

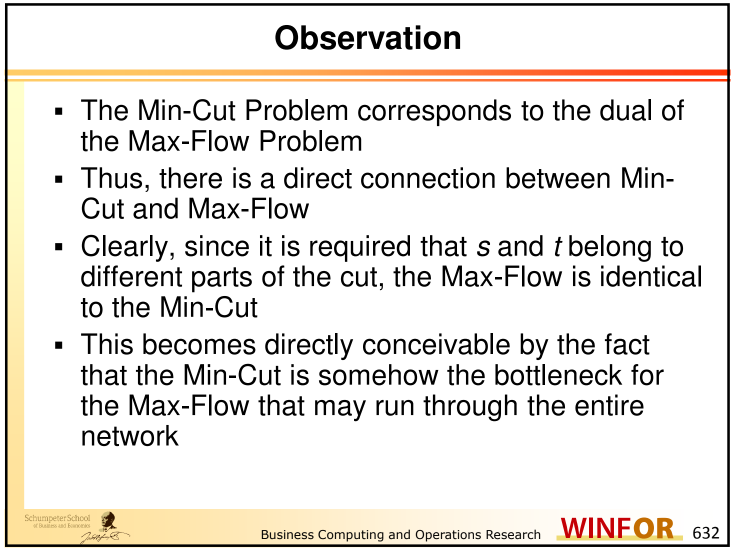
Since $i \in W \wedge j \in W^c \Leftrightarrow \pi_i = 0 \wedge \pi_j = 1 \Leftrightarrow \pi_i - \pi_j = -1$ and $i \in W^c \wedge j \in W \Leftrightarrow \pi_i = 1 \wedge \pi_j = 0 \Leftrightarrow \pi_i - \pi_j = 1$, we obtain the following problem:

Minimize $\sum_{k=1}^{n} c_k \cdot \gamma_k$, s.t.,

$\forall e_k = (i,j)(\neq (t,s)) \in E : \pi_i - \pi_j + \gamma_k \geq 0 \wedge \pi_t - \pi_s + \gamma_1 \geq 1$

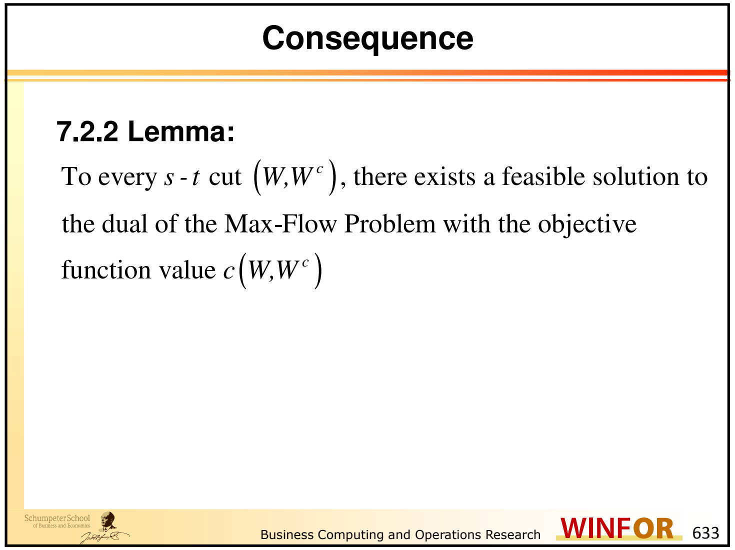
$\Leftrightarrow$

Minimize $\sum_{l=1}^{n} c_l \cdot \gamma_l$, s.t., $A^T \cdot \pi + \gamma - \delta = e^1 \wedge (\pi,\gamma,\delta) \geq 0$



Observation

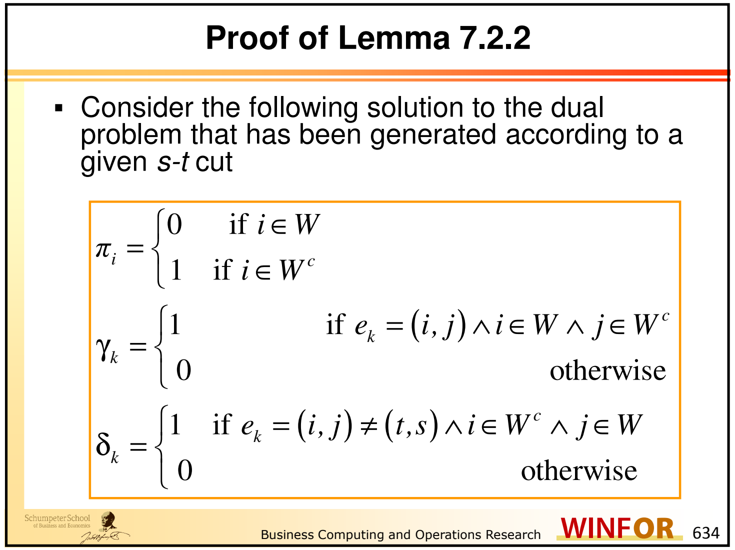
- The Min-Cut Problem corresponds to the dual of the Max-Flow Problem
- Thus, there is a direct connection between Min-Cut and Max-Flow
- Clearly, since it is required that *s* and *t* belong to different parts of the cut, the Max-Flow is identical to the Min-Cut
- This becomes directly conceivable by the fact that the Min-Cut is somehow the bottleneck for the Max-Flow that may run through the entire network



Consequence

7.2.2 Lemma:

To every s - t cut $\left(W,W^c\right)$, there exists a feasible solution to the dual of the Max-Flow Problem with the objective function value $c\left(W,W^c\right)$



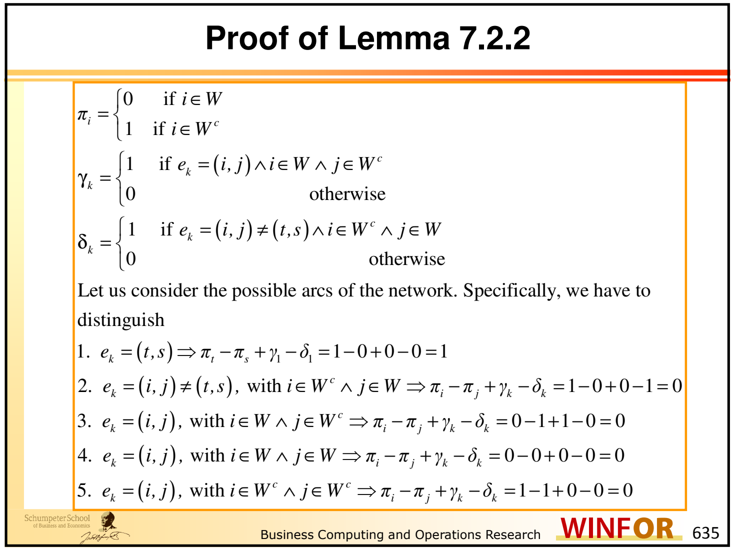
Proof of Lemma 7.2.2

- Consider the following solution to the dual problem that has been generated according to a given *s*-*t* cut

$$\pi_i=\begin{cases}0 & \text{if } i\in W\\ 1 & \text{if } i\in W^c\end{cases}$$

$$\gamma_k=\begin{cases}1 & \text{if } e_k=(i,j)\wedge i\in W\wedge j\in W^c\\ 0 & \text{otherwise}\end{cases}$$

$$\delta_k=\begin{cases}1 & \text{if } e_k=(i,j)\neq(t,s)\wedge i\in W^c\wedge j\in W\\ 0 & \text{otherwise}\end{cases}$$



Proof of Lemma 7.2.2

$$\pi_i=\begin{cases}0 & \text{if } i\in W\\ 1 & \text{if } i\in W^c\end{cases}$$

$$\gamma_k=\begin{cases}1 & \text{if } e_k=(i,j)\wedge i\in W\wedge j\in W^c\\ 0 & \text{otherwise}\end{cases}$$

$$\delta_k=\begin{cases}1 & \text{if } e_k=(i,j)\neq(t,s)\wedge i\in W^c\wedge j\in W\\ 0 & \text{otherwise}\end{cases}$$

Let us consider the possible arcs of the network. Specifically, we have to distinguish

1. $e_k=(t,s)\Rightarrow\pi_t-\pi_s+\gamma_1-\delta_1=1-0+0-0=1$
2. $e_k=(i,j)\neq(t,s)$, with $i\in W^c\wedge j\in W\Rightarrow\pi_i-\pi_j+\gamma_k-\delta_k=1-0+0-1=0$
3. $e_k=(i,j)$, with $i\in W\wedge j\in W^c\Rightarrow\pi_i-\pi_j+\gamma_k-\delta_k=0-1+1-0=0$
4. $e_k=(i,j)$, with $i\in W\wedge j\in W\Rightarrow\pi_i-\pi_j+\gamma_k-\delta_k=0-0+0-0=0$
5. $e_k=(i,j)$, with $i\in W^c\wedge j\in W^c\Rightarrow\pi_i-\pi_j+\gamma_k-\delta_k=1-1+0-0=0$

The objective function value

- We calculate the total weight of arcs crossing the cut from W to W^c
- Thus, we may conclude

$$c\left(W,W^c\right)=\sum_{e_k=(i,j),i\in W\wedge j\in W^c} c\left(e_k\right)=\sum_{e_k=(i,j),\gamma_k=1} c\left(e_k\right)=\sum_{e_k\in E}\gamma_k\cdot c\left(e_k\right)$$

Direct consequences

- In what follows, our primal problem is…

$$\text{Minimize } \sum_{l=1}^{n} c_l\cdot\gamma_l,\ \text{s.t., } A^T\cdot\pi+\gamma-\delta=e^1\wedge(\pi,\gamma,\delta)\geq 0$$

- …and the corresponding dual…

$$\text{Maximize } f_1,\ \text{s.t., } A\cdot f\leq 0\wedge f\leq c\wedge -f\leq 0$$

Max-Flow-Min-Cut Theorem

7.2.3 Theorem:

1. For each feasible s-t-flow f and each feasible s-t cut $\left(W,W^c\right)$ it holds: $|f|\leq c\left(W,W^c\right)$
2. A feasible s-t-flow f is maximal and the s-t cut $\left(W,W^c\right)$ that is constructed as defined in the Proof of Lemma 7.2.2 is minimal if it holds: $f_k=\begin{cases}0 & \text{if } e_k=(i,j)\wedge i\in W^c\wedge j\in W\\ c_k & \text{if } e_k=(i,j)\wedge i\in W\wedge j\in W^c\end{cases}$
3. To a feasible Max-Flow f, there exists a Min-Cut $\left(W,W^c\right)$ with $|f|=c\left(W,W^c\right)$

Proof of Theorem 7.2.3 – Part 1

Since the objective function value of each dual solution (Max-Flow) is a lower bound to each feasible solution to the primal problem (Min-Cut), the proposition 1 follows immediately.

Proof of Theorem 7.2.3 – Part 2

In order to prove the proposition 2, we make use of the Theorem of the complementary slackness, i.e., Theorem 5.1. Specifically, we have to analyze the rows where the dual program leaves no slack at all.

For this purpose, let us consider the following calculations Since f is assumed to be feasible, we know by the results obtained in Section 7.1 that $A \cdot f = 0$.

Consequently, the corresponding primal variables, i.e., π, may be defined arbitrarily.

Proof of Theorem 7.2.3 – Part 2

Let us now consider

$$E_n \cdot f \le c \Leftrightarrow f_k \le c_k, \forall e_k \in E \Rightarrow f_k = \begin{cases} c_k & \text{if } e_k = (i,j) \wedge i \in W \wedge j \in W^c \\ 0 & \text{if } e_k = (i,j) \wedge i \in W^c \wedge j \in W \end{cases}$$

Corresponding variables are γ. These variables are defined accordingly,

$$\text{i.e., } \gamma_k = \begin{cases} 1 & \text{if } e_k = (i,j) \wedge i \in W \wedge j \in W^c \\ 0 & \text{otherwise} \end{cases}$$

Thus, whenever there is no gap in the dual (this is the case if $f_k = c_k$), the one-value of the primal does not disturb. Other way round, if there is a gap in the dual (this is the case if $f_k = 0$), the primal fixes it by zero-values.

Proof of Theorem 7.2.3 – Part 2

Finally, we consider

$$-E_n \cdot f \le 0 \Leftrightarrow -f_k \le 0, \forall e_k \in E \Rightarrow f_k = \begin{cases} c_k & \text{if } e_k = (i,j) \wedge i \in W \wedge j \in W^c \\ 0 & \text{if } e_k = (i,j) \wedge i \in W^c \wedge j \in W \end{cases}$$

Corresponding variables are δ. These variables are defined just reversely,

$$\text{i.e., } \delta_k = \begin{cases} 1 & \text{if } e_k = (i,j) \wedge i \in W^c \wedge j \in W \\ 0 & \text{otherwise} \end{cases}$$

Thus, whenever there is no gap in the dual (this is now the case $f_k = 0$(!)), the one-value of the primal does not disturb.

Other way round, if there is a gap in the dual (this is now the case $f_k = c_k$(!)), the primal fixes it by zero-values.

Proof of Theorem 7.2.3 – Part 3

- This proof is temporarily postponed until we have introduced the algorithm of Ford and Fulkerson that generates a Min-Cut according to a given Max-Flow
- This is provided in Section 7.4

7.3 A Primal-Dual Algorithm

- We commence with the dual problem

Maximize f_1, s.t., $A \cdot f \le 0 \wedge f \le c \wedge -f \le 0$, i.e., $\begin{pmatrix} A \\ E \\ \text{-}E \end{pmatrix} \cdot f \le \begin{pmatrix} 0 \\ c \\ 0 \end{pmatrix}$

- Obviously, an initial feasible solution is *f=0*

By using a feasible dual solution, we get the set J that comprises three groups of indices. Specifically, we have: $J = J_\pi \cup J_\gamma \cup J_\delta$,
$J_\pi = \left\{ i \mid \left(A \cdot f\right)_i = 0 \right\}, J_\gamma = \left\{ k \mid f_k = c_k \right\}, J_\delta = \left\{ k \mid f_k = 0 \right\}$
Since $A \cdot f = 0$ for all feasible f, we obtain $J_\pi = \{1, 2, 3, ..., m\}$

The reduced primal (RP)

Minimize $\left(1^n\right)^T \cdot \alpha$, s.t.,

$$\alpha \ge 0, \pi \ge 0, \gamma_{\left(J_\gamma\right)} \ge 0, \delta_{\left(J_\gamma\right)} \ge 0 \wedge \left(E, A^T, E^{\left(J_\gamma\right)}, -E^{\left(J_\delta\right)}\right) \cdot \begin{pmatrix} \alpha \\ \pi \\ \gamma_{\left(J_\gamma\right)} \\ \delta_{\left(J_\delta\right)} \end{pmatrix} = e^1.$$

Note that

$E^{\left(J_\gamma\right)}$ is generated out of matrix E_n by erasing all columns that do not belong to set J_γ

$E^{\left(J_\delta\right)}$ is generated out of matrix E_n by erasing all columns that do not belong to set J_δ

The dual of the reduced primal (DRP)

Maximize g_1, s.t., $\begin{pmatrix} E \\ A \\ \left(E^{\left(J_\gamma\right)}\right)^T \\ \left(-E^{\left(J_\delta\right)}\right)^T \end{pmatrix} \cdot g \le \begin{pmatrix} 1^n \\ 0^m \\ 0^{\left|J_\gamma\right|} \\ 0^{\left|J_\delta\right|} \end{pmatrix}$,

i.e., $g \le 1 \wedge A \cdot g \le 0 \wedge g_i \le 0, i \in J_\gamma \wedge g_i \ge 0, i \in J_\delta$

Updating *f*

- As provided by the design of primal-dual algorithm, an optimal solution of DRP may either indicate that *f* is already optimal or allow an improvement of *f*
- Thus, we have to find an appropriate λ_0 which ensures an improved but still feasible dual solution
- Specifically, …

... assuming $\tilde{g}$ as the optimal solution of (DRP), we update f by $f_{new} := f_{old} + \lambda_0 \cdot \tilde{g}$

Ensuring feasibility I

In order to ensure feasibility, we have to guarantee the following:

1. $A \cdot \left(f_{old} + \lambda_0 \cdot \tilde{g} \right) \leq 0$. We already know

$$A \cdot \left(f_{old} + \lambda_0 \cdot \tilde{g} \right) = A \cdot f_{old} + A \cdot \lambda_0 \cdot \tilde{g} = 0 + A \cdot \lambda_0 \cdot \tilde{g}$$

$$= \lambda_0 \cdot \underbrace{A \cdot \tilde{g}}_{\text{Since } \tilde{g} \text{ is feasible, } A \cdot \tilde{g} \leq 0} \leq 0, \text{ for all } \lambda_0 \geq 0$$

2. $\left(f_{old} + \lambda_0 \cdot \tilde{g} \right) \leq c \Rightarrow \left(f_k + \lambda_0 \cdot \tilde{g}_k \right) \leq c_k, \forall k$

$$\Leftrightarrow \lambda_0 \leq \frac{c_k - f_k}{\tilde{g}_k}, \tilde{g}_k > 0 \wedge \underbrace{\lambda_0 \geq \frac{c_k - f_k}{\tilde{g}_k}}_{\text{Since } \lambda_0 \geq 0 \text{ and } f \text{ feasible, this is always fulfilled}}, \tilde{g}_k < 0$$

$$\Leftrightarrow \lambda_0 \leq \frac{c_k - f_k}{\tilde{g}_k}, \tilde{g}_k > 0$$

Ensuring feasibility II

And finally, we have to guarantee the following:

3. $(-1) \cdot \left(f_{old} + \lambda_0 \cdot \tilde{g} \right) \leq 0 \Rightarrow \left(-f_k - \lambda_0 \cdot \tilde{g}_k \right) \leq 0, \forall k$

$$\Leftrightarrow \underbrace{\lambda_0 \geq \frac{f_k}{-\tilde{g}_k}, \tilde{g}_k > 0}_{\text{Since } \lambda_0 \geq 0 \text{ and } f \text{ feasible, this is always fulfilled}} \wedge \lambda_0 \leq \frac{f_k}{-\tilde{g}_k}, \tilde{g}_k < 0$$

$$\Leftrightarrow \lambda_0 \leq \frac{f_k}{-\tilde{g}_k}, \tilde{g}_k < 0$$

Interpreting DRP

- Obviously, DRP can be interpreted as a specifically defined accessibility problem, i.e., a path is searched in a reduced graph
- This reduced graph restricts the searching process as follows
  - Arcs that are already used up to capacity may only be used in backward direction, i.e., the flow is reduced
  - Arcs that are unused, i.e., $f_k=0$, may only be used in forward direction
  - All other arcs can be used in any direction
  - All induced flows are restricted by 1, i.e., a flow of maximum capacity 1 is sought

Augmenting the flow

- Obviously, by solving DRP, we are aspiring an augmenting path
- Hence, it is not feasible to augment an already saturated flow or to decrease a zero flow along some edge
- Consequently, if there is an augmentation possible, we are able to generate a flow f that induces only 1, -1, or 0 values at the respective edges
- This considerably simplifies the updating of the dual solution in the Primal-Dual Algorithm

Ensuring feasibility with g=1,0,-1

In order to ensure feasibility, we have to guarantee the following:

1. $A\cdot\left(f_{old}+\lambda_0\cdot\tilde{g}\right)\le 0$ is fulfilled for all $\lambda_0\ge 0$

2. $\left(f_{old}+\lambda_0\cdot\tilde{g}\right)\le c \Leftrightarrow \lambda_0\le\frac{c_k-f_k}{\tilde{g}_k}, \tilde{g}_k=1 \Leftrightarrow \lambda_0\le c_k-f_k$

3. $\left(-1\right)\cdot\left(f_{old}+\lambda_0\cdot\tilde{g}\right)\le 0 \Leftrightarrow \lambda_0\le\frac{f_k}{-\tilde{g}_k}, \tilde{g}_k=-1 \Leftrightarrow \lambda_0\le f_k$

$\Rightarrow \lambda_0\le\min\left\{\min\left\{c_k-f_k \mid \tilde{g}_k=1\right\},\min\left\{f_k \mid \tilde{g}_k=-1\right\}\right\}$

7.4 Ford-Fulkerson Algorithm

- This algorithm is a modified primal-dual solution procedure
- The DRP is directly solved, however, that is why no Simplex procedure is necessary for this step
- On the other side, this has considerable consequences according to the termination of the solution procedure
- This will be discussed thoroughly later

A reduced network

7.4.1 Definition:

Assuming $N=\left(V,E,c,s,t\right)$ is an s - t-network and f a feasible s - t - flow. Then, we introduce

$E_f=E_f^f\cup E_f^b$, with

$E_f^f=\left\{e_k=\left(i,j\right)\mid\exists e_k=\left(i,j\right)\in E\wedge f_k<c_k\right\}$ and

$E_f^b=\left\{e_k=\left(i,j\right)\mid\exists e_{\tilde{k}}=\left(j,i\right)\in E\wedge f_{\tilde{k}}>0\right\}$.

E_f^f denotes the set of forward arcs while E_f^b defines the backward arcs. Then, we denote $\left(V,E_f,c,s,t\right)$ as the corresponding reduced network.

Interpretation

- Forward arcs
  - are used by the current flow *f*, but they are not used up to capacity
  - I.e., they are not saturated by now
- Backward arcs
  - are not used by the current flow *f*, but the inverted arc is used by flow *f*
  - Consequently, these arcs are used in opposite direction by the current flow *f*
- Consequently,
  - forward arcs are candidates for augmenting the flow in the current direction (since they offer remaining capacities)
  - backward arcs are candidates for reducing the flow (since the opposite direction transfers something)

Observation

7.4.2 Lemma:

A path $(i_0,\ldots,i_k)$ with $i_0 = 1,\ i_k = n,$ and $(i_{l-1},i_l)\in E_f$ indicates an optimal solution to (DRP).

Proof of Lemma 7.4.2

Based on the path $p = (i_0 = s,\ldots,i_k = t)$, we define as follows:

$$g_k = \begin{cases} 1 \text{ if } e_k = (i,j) = (i_{l-1},i_l)\in E, \text{ for } l\in\{1,\ldots,k\} \text{ or if } e_k = (n,1) \\ -1 \qquad\qquad \text{if } e_k = (i,j) = (i_l,i_{l-1})\in E, \text{ for } l\in\{1,\ldots,k\} \\ 0 \qquad \text{otherwise} \end{cases}$$

Since p is a path, each visited node is reached and left by arcs once. If this is done according to arc directions, we use $g_k = 1$, otherwise we have $g_k = -1$. Since the 1 and -1 values in A are changed accordingly, we obtain in both cases for the respective row i:$(A\cdot g)_i = 0$.

In addition, it holds:

$g \le 1 \wedge g_i \le 0, i\in J_\gamma = \{k \mid f_k = c_k\} \wedge g_i \ge 0, i\in J_\delta = \{k \mid f_k = 0\}$. Thus, g is feasible. Since $g_1 = 1$, it is also an optimal solution to (DRP).

Conclusions

Let us assume that such a path between s and t cannot be established in the reduced network.

We define for this constellation:

$W = \{i\in V \mid \exists p = (s = i_1,\ldots,i_k = i) : (i_{l-1},i_l)\in E_f\} \wedge W^c = V \setminus W$

and additionally...

$$\pi_i = \begin{cases} 0 & \text{if } i\in W \\ 1 & \text{if } i\in W^c \end{cases},\ \gamma_k = \begin{cases} 1 & \text{if } e_k = (i,j)\wedge i\in W \wedge j\in W^c \\ 0 & \text{otherwise} \end{cases},$$

and finally $\delta_k = \begin{cases} 1 & \text{if } e_k = (i,j) \ne (t,s) \wedge i\in W^c \wedge j\in W \\ 0 & \text{otherwise} \end{cases}$

The *s*-*t*-cut

We obtain :

$$c(W,W^c) = \sum_{e_k=(i,j), i\in W \wedge j\in W^c} c(e_k) = \sum_{e_k=(i,j), \gamma_k=1} c(e_k) = \sum_{e_k\in E} \gamma_k \cdot c(e_k)$$

Since all nodes of W^c were not reachable, all arcs bridging the cut (W,W^c) are used up to capacity by flow f. Consequently, we know

$f_1 = |f| = \sum_{e_k\in E} \gamma_k \cdot c(e_k) = c(W,W^c)$. In addition, f cannot be augmented and is therefore maximal.

Maximum augmentation

- The maximum augmentation δ that is possible for the current flow, is determined by

$$\delta = \min \left\{ \begin{array}{l} \min_{\text{arcs of path } p} \left\{ c_k - f_k \mid e_k \text{ is forward arc} \right\}, \\ \min_{\text{arcs of path } p} \left\{ f_k \mid e_k \text{ is backward arc} \right\} \end{array} \right\}$$

Ford-Fulkerson Algorithm

- In what follows, we introduce the description provided by Papadimitriou and Steiglitz (1982) p.123

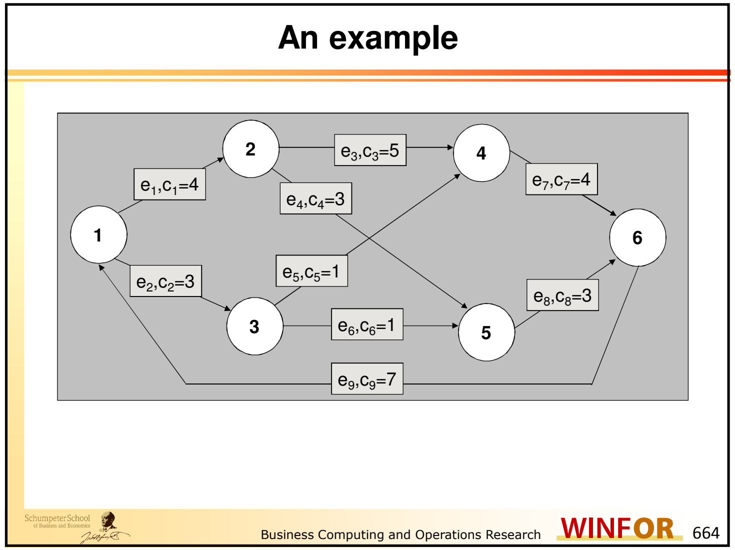
Ford-Fulkerson Algorithm

- Input: Network $N=(s,t,V,E,c)$
- Output: Max-Flow f
- Set $f=0$, $E_f=E$;
- While an augmenting *s*-*t*-path with min capacity value δ > 0 can be found in the reduced network E_f:
  - Set f = f + δ;
  - Update reduced network E_f (decrease capacities in path direction by value δ and increase capacities in opposite direction by value δ for all edges on the augmenting path)
- End while

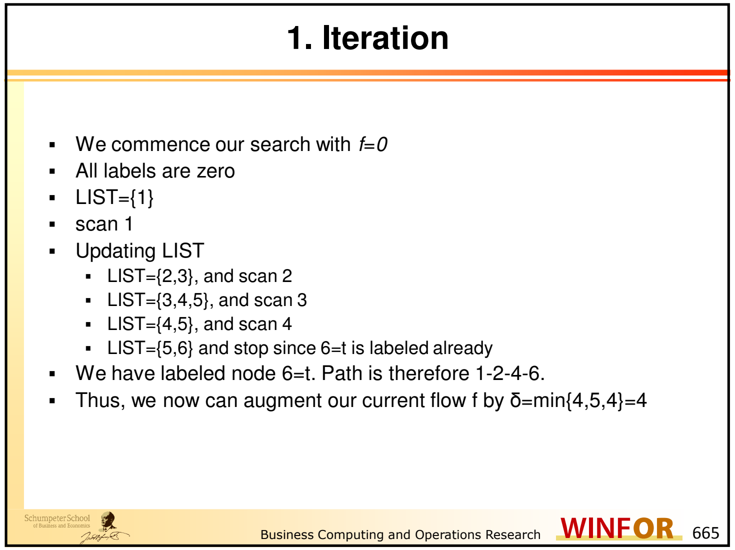
An augmenting path can be found with the labeling algorithm on the next slide.

Labeling Algorithm

- We try to label every node with one possible predecessor on a path from *s* until we reach *t*:
- LIST={s};
- While LIST not empty and *t* not in LIST:
  - **Scan *x*:** Remove *x* from LIST. Label not all labeled yet adjacent nodes to *x* in E_f with *x* as predecessor and put them on LIST.
- End while
- If *t* is labeled, we can create the augmenting path by considering the predecessors in the labels.

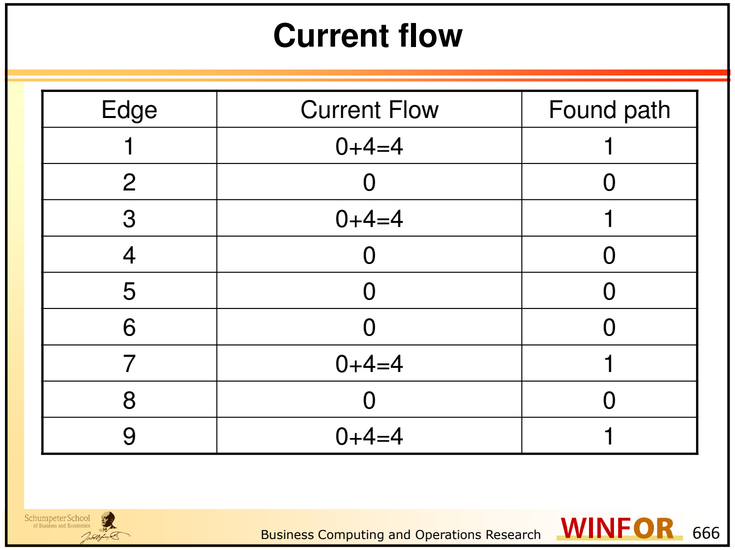


An example



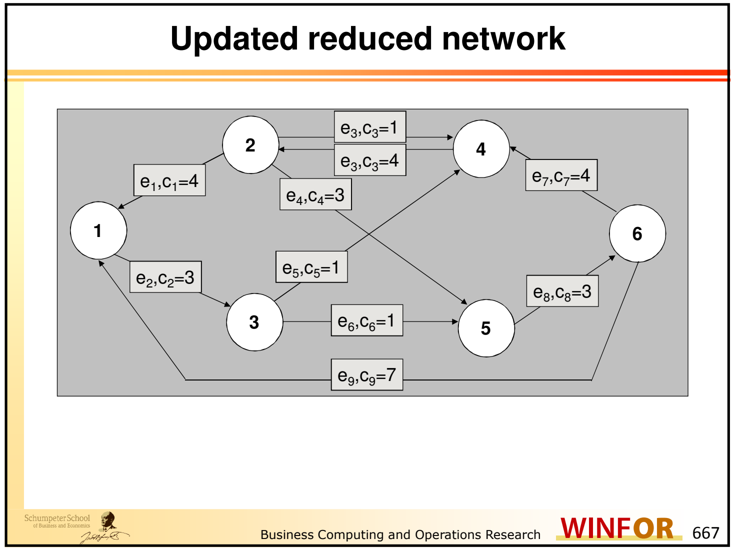
1. Iteration

- We commence our search with *f=0*
- All labels are zero
- LIST={1}
- scan 1
- Updating LIST
  - LIST={2,3}, and scan 2
  - LIST={3,4,5}, and scan 3
  - LIST={4,5}, and scan 4
  - LIST={5,6} and stop since 6=t is labeled already
- We have labeled node 6=t. Path is therefore 1-2-4-6.
- Thus, we now can augment our current flow f by δ=min{4,5,4}=4



Current flow

Edge	Current Flow	Found path
1	0+4=4	1
2	0	0
3	0+4=4	1
4	0	0
5	0	0
6	0	0
7	0+4=4	1
8	0	0
9	0+4=4	1



Updated reduced network

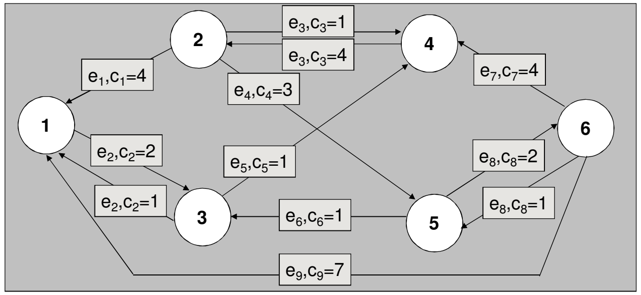
2. Iteration

- We commence our search with *f*
- All labels are zero
- LIST={1}
- scan 1
- Updating LIST
  - LIST={3}, and scan 3
  - LIST={4,5}, and scan 4
  - LIST={5,2}, and scan 5
  - LIST={6} and stop since 6=t is labeled already
- We have labeled node 6=t. Path is therefore 1-3-5-6.
- Thus, we now can augment our current flow f by δ=min{3,1,3}=1

Current flow

Edge	Current Flow	Found path
1	4	0
2	0+1=1	1
3	4	0
4	0	0
5	0	0
6	0+1=1	1
7	4	0
8	0+1=1	1
9	5	1

Updated reduced network



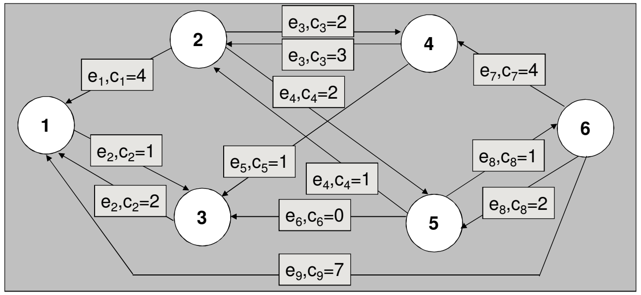
3. Iteration

- We commence our search with *f*
- All labels are zero
- LIST={1}
- scan 1
- Updating LIST
  - LIST={3}, and scan 3
  - LIST={1,4}. Since 1 is labeled, LIST={4}, and scan 4
  - LIST={2}, and scan 2
  - LIST={1,4,5} Since 1,4 are labeled, LIST={5}, and scan 5
  - LIST={6} and stop since 6=t is labeled already
  - We have labeled node 6=t. Path is therefore 1-3-4-2-5-6.
- Thus, we now can augment our current flow f by δ=min{2,1,4,3,2}=1

Current flow

Edge	Current Flow	Found path
1	4	0
2	1+1=2	1
3	4-1=3	-1
4	0+1=1	1
5	0+1=1	1
6	1	0
7	4	0
8	1+1=2	1
9	5+1=6	1

Updated reduced network



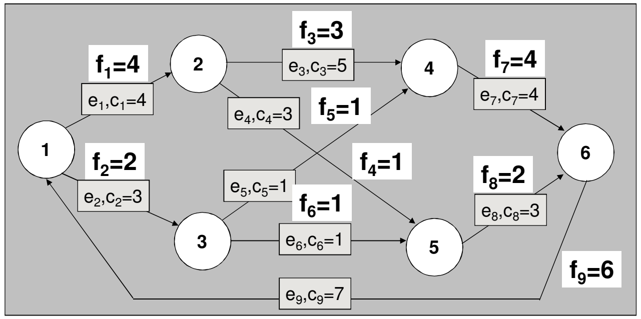
4. Iteration

- We commence our search with *f*
- All labels are zero
- LIST={1}
- scan 1
- Updating LIST
  - LIST={3}, and scan 3
  - LIST={1}. Since 1 is labeled, LIST={}, and terminate
  - Thus, we obtain the s-t cut
    - W={1,3} and W^c={2,4,5,6}
    - The cut has total costs $c_1+c_5+c_6=4+1+1=6$

Maximal flow

Edge	Flow
1	4
2	2
3	3
4	1
5	1
6	1
7	4
8	2
9	6

Updated reduced network



Optimality

- Clearly, the optimality of the procedure depicted above may be directly derived from the Primal-Dual Algorithm design
- There are, however, some specific interesting attributes coming along with the procedure of Ford and Fulkerson that are worth mentioning
- In what follows, we briefly discuss or just mention them

Correctness of the procedure

7.4.3 Lemma:

When the Ford and Fulkerson labeling algorithm terminates, it does so at optimal flow.

Proof of Lemma 7.4.3

- When the algorithm of Ford and Fulkerson terminates, there are some nodes that are already labeled while others are still unlabeled. We define W and W^c as above
- Consequently, all arcs that are running from W to W^c are saturated now
- Additionally, arcs running in the opposite direction have flow zero
- Therefore, by Theorem 7.2.3, the *s*-*t*-cut (W,W^c) and flow f are optimal