

The Max Flow Problem

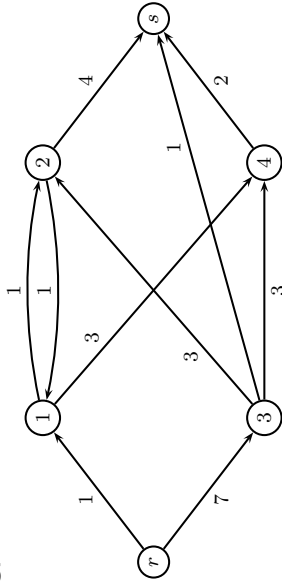
Augmenting path method

Jesper Larsen & Jens Clausen & David Pisinger
jesla, jenc1, pisinger@man.dtu.dk

DTU Management
Technical University of Denmark

The Max Flow Problem

- We consider a **directed graph** $G = (V, E)$ which for each e has a **capacity** $u_e \in \mathbb{R}_+$.
- Furthermore two “special” vertices r and s are given; these are called resp. the **source** and the **sink**.
- Aim is to find the maximum flow from the source to the sink in the graph while respecting the capacities of the edges.

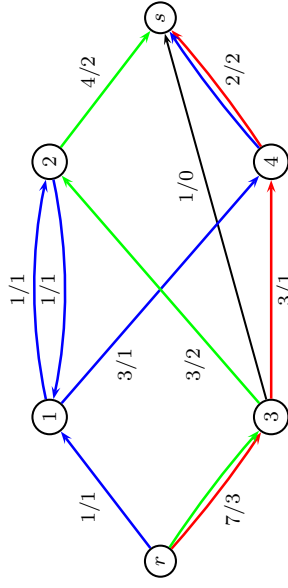


Flow

- A **flow** in G is a function $x : E \rightarrow \mathbb{R}_+$ satisfying:
 - (1) $\forall i \in V \setminus \{r, s\} : \sum_{(j,i) \in E} x_{ji} - \sum_{(i,j) \in E} x_{ij} = 0$
 - (2) $\forall (i, j) \in E : 0 \leq x_{ij} \leq u_{ij}$
- For a flow we define: $f_x(i) = \sum_{(j,i) \in E} x_{ji} - \sum_{(i,j) \in E} x_{ij}$
- $f_x(i)$ is the **net flow into i** or the **excess for x in i** . $f_x(s)$ is called the **value of x** .

Example with a flow

Here we have added a flow of value 4 to our graph.



Notation: u_{ij} / x_{ij}

Math. Programming Model of Max Flow

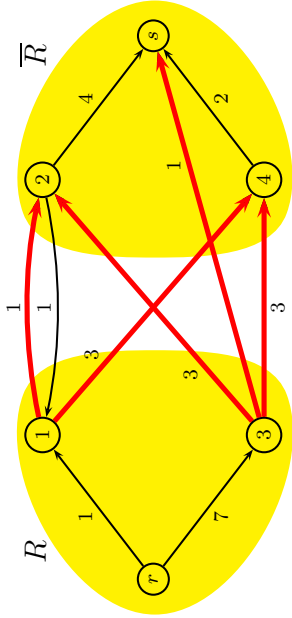
$$\begin{array}{ll} \max & f_x(s) \\ \text{s.t.} & f_x(i) = 0 \quad i \in V \setminus \{r, s\} \\ & 0 \leq x_e \leq u_e \quad e \in E \\ & x_e \in \mathbb{Z}_+ \quad e \in E \end{array}$$

$$\text{where } f_x(i) = \sum_{(j,i) \in E} x_{ji} - \sum_{(i,j) \in E} x_{ij}$$

Max Flow and Cuts II

- In an undirected graph we have $\delta(R) = \delta(\overline{R})$, this does not hold here.
- R is an r,s -cut if $r \in R$ and $s \notin R$.
- The capacity of the cut R is defined as:

$$u(\delta(R)) = \sum_{(i,j) \in E, i \in R, j \in \overline{R}} u_{ij}$$

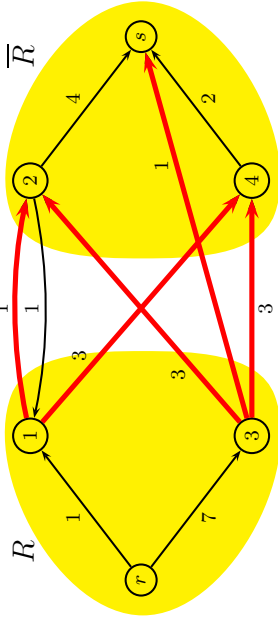


Max Flow and Cuts I

- We now consider $R \subset V$. $\delta(R)$ is the set of edges incident from a vertex in R to a vertex in $\overline{R}$. $\delta(R)$ is also called the cut **generated** by R :

$$\delta(R) = \{(i, j) \in E : i \in R, j \in \overline{R}\}$$

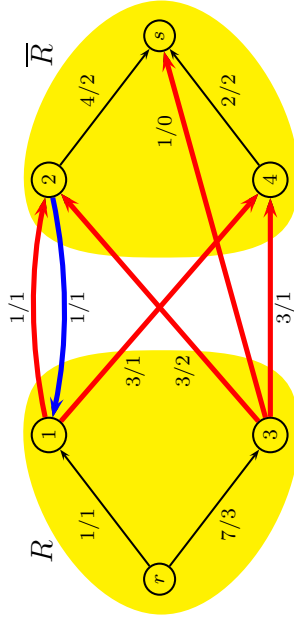
- Note the difference to the cut in an undirected graph. In a directed graph the cut only consists of arcs going from R to $\overline{R}$ and not the opposite direction.



Relationship between flows and cuts

- **Proposition:** For any r, s -cut $\delta(R)$ and any flow x we have:
 $x(\delta(R)) - x(\delta(\overline{R})) = f_x(s)$
- **Corollary:** For any feasible flow x and any r, s -cut $\delta(R)$, we have:

$$f_x(s) \leq u(\delta(R)).$$



The Residual graph

- Suppose that x is a flow in G . The **residual graph** shows **how flow excess can be moved** in G given that the flow x is already present.
- The residual graph G_x for G wrt. x is defined by:

$$\begin{aligned} V(G_x) &= V \\ E(G_x) &= \{(i, j) : (i, j) \in E \wedge x_{ij} < u_{ij}\} \cup \\ &\quad \{(j, i) : (i, j) \in E \wedge x_{ij} > 0\} \end{aligned}$$

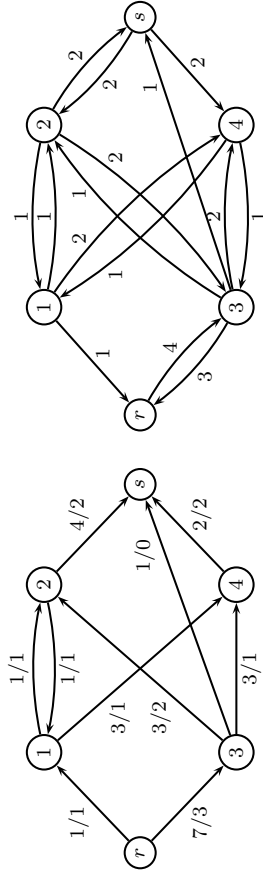
Short we write $V_x = V(G_x)$ and $E_x = E(G_x)$

- Some textbooks use the word “auxiliary graph” instead of residual graph.

The Max Flow Problem – p.9/77

The Residual graph

Graph with flow of 4, and corresponding residual graph



The Max Flow Problem – p.10/77

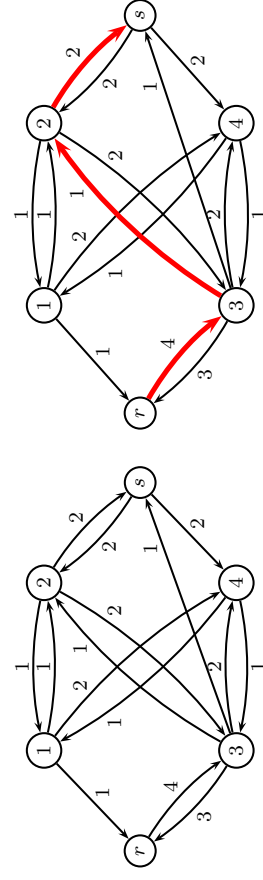
Incrementing and augmenting path

- A path is called **x -incrementing** if for every forward arc e $x_e < u_e$ and for every backward arc $x_e > 0$.
- A **x -incrementing path** from r to s is called **x -augmenting**.
- Note: this relates to the original graph. In the Residual graph all arcs in a path are forward arcs.

The Max Flow Problem – p.11/77

Augmenting path

Residual graph and augmenting path of capacity 1



The Max Flow Problem – p.12/77

Basic Approach

1. Build residual graph
2. Build augmenting path
3. If not possible **stop**
4. Increase flow along the augmenting path.
5. Go to step 1

The Max Flow Problem – p.13/77

Finding an augmenting path I

- 🔴 **Input:** The network $G = (V, E, u)$ and the current feasible flow x .
- 🔴 **Output:** An augmenting path from r to s and the capacity for the flow augmentation.

Initialize: all vertices are unlabeled;

$Q \leftarrow \{r\}, S \leftarrow \emptyset, p[\cdot] \leftarrow 0;$

$u_{\max}[i] \leftarrow +\infty$ **for** $i \in G$

The Max Flow Problem – p.15/77

The Augmenting Path Algorithm

- 1 $x_{ij} \leftarrow 0$ for all $(i, j) \in E;$
- 2 **repeat**
- 3 construct $G_x;$
- 4 find an augmenting path $P;$
- 5 **if** s is reached **then**
- 6 determine max amount K of augmentation;
- 7 augment x along the path P by $K;$
- 8 **until** s is not reachable from r in $G_x;$

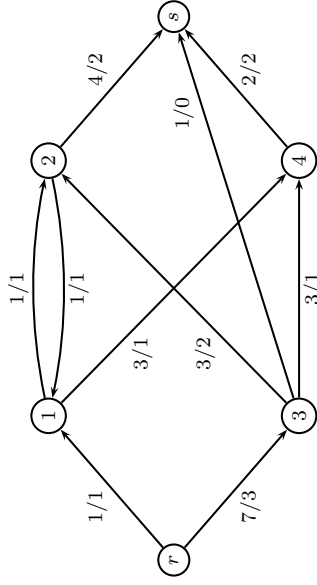
The Max Flow Problem – p.14/77

Finding an augmenting path II

- 1 **while** $Q \neq \emptyset$ and s is not labeled **yet do**
- 2 select a $i \in Q$ and scan $(i, j) \in E_x;$
- 3 **if** $(i, j) \in E$ and j unlabeled **then**
- 4 label $j; Q \leftarrow Q \cup \{j\}; p[j] \leftarrow i;$
- 5 $u_{\max}[j] \leftarrow \min\{u_{\max}[i], u_{ij} - x_{ij}\};$
- 6 **if** $(j, i) \in E$ and j unlabeled **then**
- 7 label $j; Q \leftarrow Q \cup \{j\}; p[j] \leftarrow i;$
- 8 $u_{\max}[j] \leftarrow \min\{u_{\max}[i], x_{ji}\};$
- 9 $Q \leftarrow Q \setminus \{i\}; S \leftarrow S \cup \{i\};$

The Max Flow Problem – p.16/77

Example



The Max Flow Problem – p.17/77

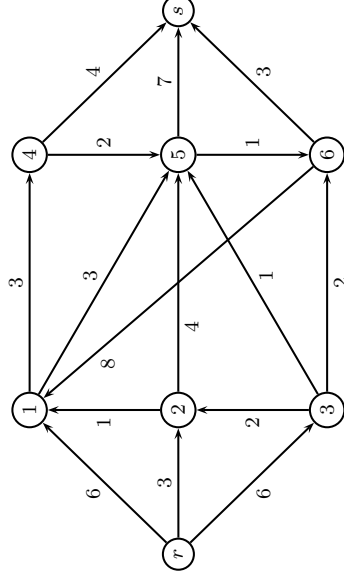
Augmenting a feasible flow x

- Input: The network $G = (V, E, u)$, the current feasible flow x , an augmenting path P from r to s in G_x and the x -width of P .
- Output: An “updated” feasible flow x' .
- An updated flow is achieved by:

for $(i, j) \in P$ $(i, j) \in E : x'_{ij} \leftarrow x_{ij} + u_{\max}[s]$
 for $(i, j) \in P$ $(j, i) \in E : x'_{ji} \leftarrow x_{ji} - u_{\max}[s]$
 for all other $(i, j) \in E : x'_{ij} \leftarrow x_{ij}$

The Max Flow Problem – p.18/77

Example

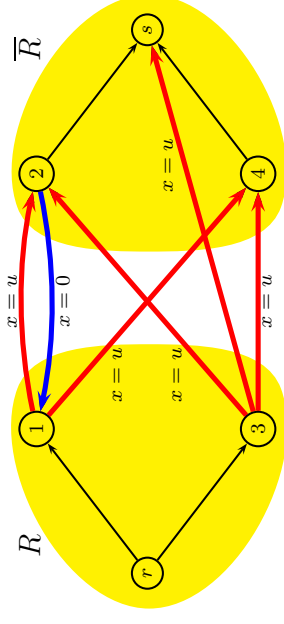


The Max Flow Problem – p.19/77

Max Flow - Min Cut Theorem

Given a network $G = (V, E, u)$ and a current feasible flow x . The following 3 statements are equivalent:

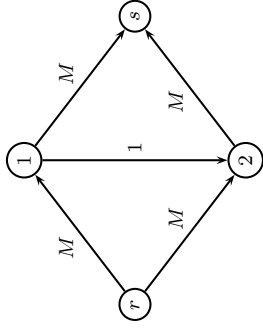
- x is a maximum flow in G .
- A flow augmenting path does not exist (r - s -path in G_x).
- An (r, s) -cut R exists with capacity equal to the value of x , i.e. $u(R) = f_x(s)$.



The Max Flow Problem – p.20/77

Time complexity

If we choose edge with capacity 1 in each iteration, we use $O(M)$ iterations.



The Max Flow Problem – p.21/77

Concluding Remarks

- We call an augmenting path from r to s **shortest** if it has the minimum possible number of arcs.
- The augmenting path algorithm with a breadth-first search solves the maximum flow problem in $O(nm^2)$.
- Breadth-first can easily be established using a queue.

The Max Flow Problem – p.22/77