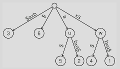


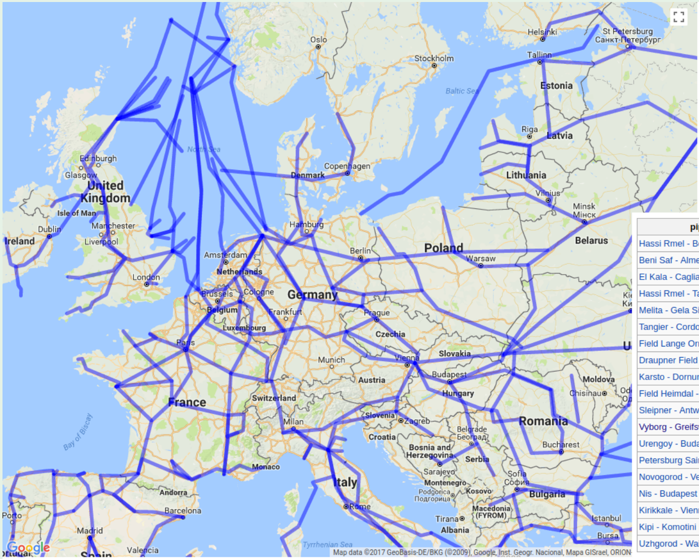
Datenstrukturen und Effiziente Algorithmen

Vorlesung *Datenstrukturen und Effiziente Algorithmen* im WS 18/19

Marc Hellmuth
Institut für Mathematik und Informatik
Universität Greifswald

A Real Flow Network

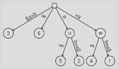
Example 1



pipeline	AnnualCapacityBCM
Hassi Rmel - Bologna Pipeline	27
Beni Saf - Almeria Pipeline	8
El Kala - Cagliari Sardinia Pipeline	8
Hassi Rmel - Tarifa Pipeline	
Melita - Gela Sicily Pipeline	
Tangier - Cordoba Pipeline	
Field Lange Ormen - Teeside Pipeline	25.5
Draupner Field - Dnkirk Pipeline	19.6
Karsto - Dornum Pipeline	24
Field Heimdal - Emden Pipeline	
Sleipner - Antwerp Pipeline	
Vyborg - Greifswald Pipeline	1200
Urengoy - Budapest Pipeline	240
Petersburg Saint - Tallinn Pipeline	
Novogorod - Ventspils Pipeline	
Nis - Budapest Pipeline	
Kirikkale - Vienna Pipeline	
Kipi - Komotini Pipeline	
Uzhgorod - Waidhaus Pipeline	

Major Import Pipelines into the European Union

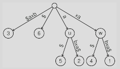
http://enipedia.tudelft.nl/wiki/NaturalGasInfrastructure



Flow Network

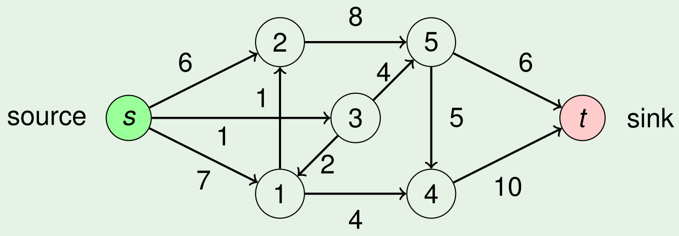
Flow Network

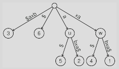
- a material (e.g. a liquid, electrical current, countable items) is produced at a source
- it is moving through a system (e.g. of pipes, wires, roads) that is modeled as a directed graph
- each edge has a given capacity (e.g. diameter of pipe, wire, capacity of road) that limits the amount of material that can pass this edge
- the system is in a steady state, i.e. the amount of material leaving a vertex is equal to the amount of material entering the vertex
- all material ends at a vertex called sink, where it is consumed



Flow Network

Example 2





Flow Network

Definition 3 (Flow network)

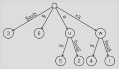
A flow network $G = (V, E)$ is a directed graph such that

- G has no self-loops
- each edge $(u, v) \in E$ has a capacity $c(u, v) \geq 0$
- we define $c(u, v) := 0$ if $(u, v) \notin E$
- V has two distinct vertices, a source s and a sink t

Further assumptions we make

1. $(u, v) \in E \Rightarrow (v, u) \notin E \qquad (\forall u, v \in V)$
2. $\forall v \in V$ there is a path $s \rightsquigarrow v \rightsquigarrow t$

Will later see: both assumptions are not constraining generality.



Flow

Definition 4 (Flow)

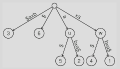
Let $G = (V, E)$ be a flow network as above.
A flow in G is a function $f : V \times V \to \mathbb{R}$ that satisfies

- capacity constraint: $0 \leq f(u, v) \leq c(u, v) \qquad (\forall u, v \in V)$
- flow conservation:

$$\sum_{v \in V} f(u, v) = \sum_{v \in V} f(v, u) \qquad (u \in V \setminus \{s, t\})$$

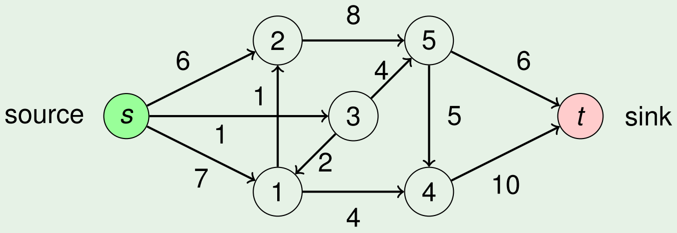
Observation

There is only positive flow along edges:
$(u, v) \notin E \Rightarrow f(u, v) = 0$

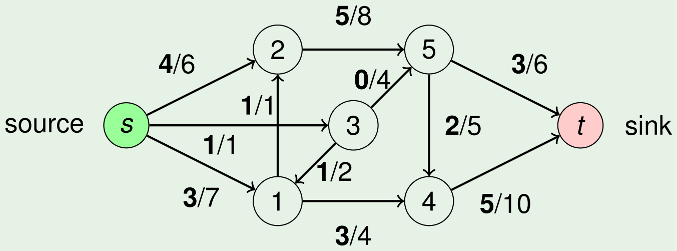


Flow

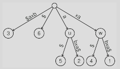
Example 5 (Flow network G)



Example 6 (A flow f in network G)



E.g. $f(s, 2) = 4$, $f(2, 5) = 5$, etc.



Flow

Definition 7 (Value of a flow)

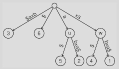
The value $|f|$ of a flow f is defined as

$$|f| := \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s).$$

It is the total flow out of the source minus the total flow into the source.

Remark

Usually, in practially relevant flows, there is no flow into s, i.e. $\sum_{v \in V} f(v,s) = 0$. However, we will need this possibility in intermediate steps in the algorithm that solves the



Flow

Definition 7 (Value of a flow)

The value $|f|$ of a flow f is defined as

$$|f| := \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s).$$

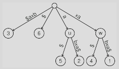
It is the total flow out of the source minus the total flow into the source.

Remark

Usually, in practially relevant flows, there is no flow into s, i.e. $\sum_{v \in V} f(v, s) = 0$. However, we will need this possibility in intermediate steps in the algorithm that solves the

Definition 8 (Maximum flow problem)

Given a flow network G, find a flow in G of maximum value.

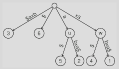


Antiparallel Edges

Antiparallel edges

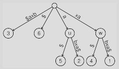
If both (u, v) and (v, u) are edges in a directed graph, then we call these edges antiparallel. A flow network with antiparallel edges (u, v) and (v, u), can be modeled by an equivalent flow network that has no antiparallel edges.

(chalk board)



Multiple sources and sinks

- a flow network may have multiple sources $s_1, \dots, s_m$ and/or multiple sinks $t_1, \dots, t_n$.
- the material is produced at these multiple sources and it is consumed at these multiple sinks



Multiple sources and sinks

- a flow network may have multiple sources $s_1, \dots, s_m$ and/or multiple sinks $t_1, \dots, t_n$.
- the material is produced at these multiple sources and it is consumed at these multiple sinks
- such a flow network can be modeled with an equivalent flow network that has just one source and one sink

(chalk board)

Residual Network

- want to iteratively approach a maximal flow
- an intermediate flow *f* will use some edge capacity and leave some capacity unused
- will define a flow network capturing the *unused* capacity, termed "residual network"

Residual Network

- want to iteratively approach a maximal flow
- an intermediate flow f will use some edge capacity and leave some capacity unused
- will define a flow network capturing the *unused* capacity, termed “residual network”

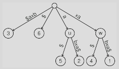
Definition 9 (Residual network)

Let $G = (V, E)$ be a flow network with source s and sink t. Let f be a flow in G. Then for two vertices u, v the residual capacity is defined as

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{, if } (u,v) \in E \\ f(v,u) & \text{, if } (v,u) \in E \\ 0 & \text{, otherwise.} \end{cases}$$

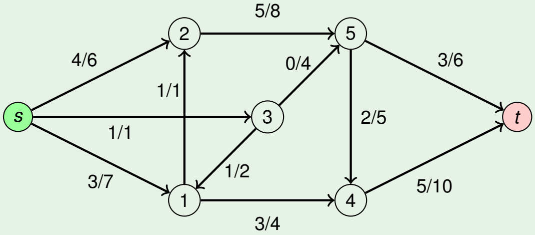
The residual network of G induced by f is $G_f = (V, E_f)$, where

$$E_f = \{(u,v) \in V \times V \mid c_f(u,v) > 0\}.$$

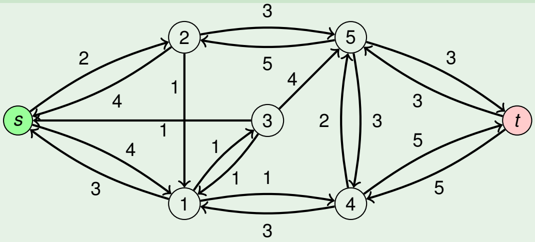


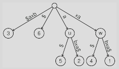
Residual Network

Example 10 (Flow f in network G)



Example 11 (The residual network G_f with residual capacities c_f)

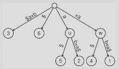




Residual Network

Comments

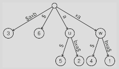
- G_f can have an edge (v, u), where G had an edge (u, v)
- a residual network is very similar to a flow network; however, antiparallel edges are allowed
- will consider flows in the residual network – they are defined just as in flow networks
- a flow f' along such a reversed edge (v, u), models a decrease of the flow in the original direction (u, v)
- f' then partially or completely cancels the flow f
- $|E_f| \leq 2|E|$



Flows in the Residual Network

Augmenting a flow

A flow f' in the residual network can be used to add flow to flow f in the original flow network.



Flows in the Residual Network

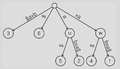
Augmenting a flow

A flow f' in the residual network can be used to add flow to flow f in the original flow network.

Definition 12 (Augmentation of flow)

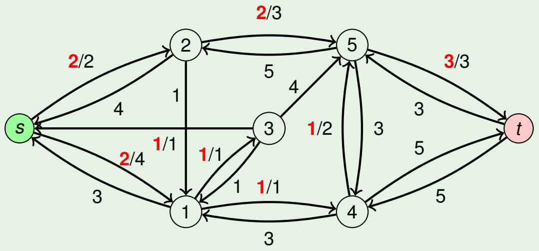
Let f be a flow in G and f' be a flow in the residual network G_f. Define $f \uparrow f' : V \times V \to \mathbb{R}$ – the augmentation of f by f' – as follows

$$(f \uparrow f')(u,v) = \begin{cases} f(u,v) + f'(u,v) - f'(v,u) & \text{, if } (u,v) \in E \\ 0 & \text{, otherwise.} \end{cases}$$

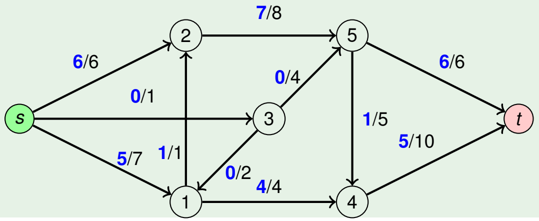


Augmenting a flow

Example 13 (A flow f' in the residual network G_f)



Example 14 (Augmentation $f \uparrow f'$)



Augmentation of a Flow

Lemma 15

Let G, f, f' be as above. Then the function $f \uparrow f'$ is a flow in G with value $|f \uparrow f'| = |f| + |f'|$

Augmentation of a Flow

Lemma 15

Let G, f, f' be as above. Then the function $f \uparrow f'$ is a flow in G with value $|f \uparrow f'| = |f| + |f'|$

Proof...

Capacity constraint:
If $(u, v) \notin E$ then $(f \uparrow f')(u, v) = 0$ satisfies the capacity constraint.
Let $(u, v) \in E$. Then

$$\begin{aligned}
(f \uparrow f')(u, v) &= f(u, v) + f'(u, v) - f'(v, u) \\
&\geq f(u, v) - f'(v, u) \\
&\geq 0 && \text{(as } f'(v, u) \leq c_f(v, u) = f(u, v)\text{)}
\end{aligned}$$

We also have

$$\begin{aligned}
(f \uparrow f')(u, v) &= f(u, v) + f'(u, v) - f'(v, u) \\
&\leq f(u, v) + f'(u, v) \\
&\leq f(u, v) + c_f(u, v) \\
&= f(u, v) + c(u, v) - f(u, v) \\
&= c(u, v)
\end{aligned}$$

$f \uparrow f'$ therefore satisfies the capacity constraint.

Augmentation of a Flow

Proof.

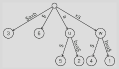
Flow conservation:
Both f and f' obey flow conservation. Therefore, we have for any $u \in V \setminus \{s, t\}$

$$\begin{aligned}
\sum_{v \in V} (f \uparrow f')(u, v) &= \sum_{v \in V} (f(u, v) + f'(u, v) - f'(v, u)) \\
&= \sum_{v \in V} f(v, u) + \sum_{v \in V} f'(v, u) - \sum_{v \in V} f'(u, v) \\
&= \sum_{v \in V} (f \uparrow f')(v, u).
\end{aligned}$$

Therefore, $f \uparrow f'$ obeys flow conservation.
For computing the **value of** $f \uparrow f'$, let $F = \{v \in V \mid (s, v) \in E\}$ be the set of vertices with edges From s and let $T = \{v \in V \mid (v, s) \in E\}$ be the set of vertices with edges To s. $F \cap T = \emptyset$ because G has no antiparallel edges.

$$\begin{aligned}
|f \uparrow f'| &= \sum_{v \in F} (f \uparrow f')(s, v) - \sum_{v \in T} (f \uparrow f')(v, s) \\
&= \sum_{v \in F} (f(s, v) + f'(s, v) - f'(v, s)) - \sum_{v \in T} (f(v, s) + f'(v, s) - f'(s, v)) \\
&= |f| + \sum_{v \in F \cup T} (f'(s, v) - f'(v, s)) \\
&= |f| + |f'|
\end{aligned}$$

□



Augmenting Path

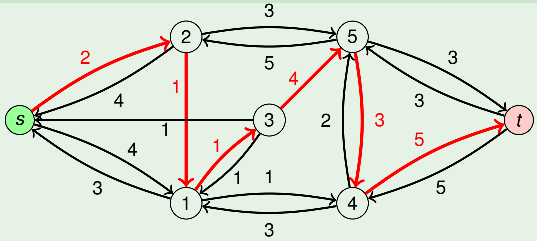
Choice of f'

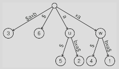
- above lemma allows augmenting a flow f with an *arbitrary* flow f' in the residual network
- will only consider a simple kind of flow for f'

Definition 16 (Augmenting path)

Given a flow network $G = (V, E)$ and a flow f in G, an augmenting path is a simple path from s to t in the residual network G_f.

Example 17 (An augmenting path in the residual network G_f)





Augmenting Path

Augmenting path

- let p be an augmenting path
- p naturally defines a flow in the residual network:
  there is a positive flow on all the edges of p
  there is no flow elsewhere

Definition 18 (residual capacity of a path)

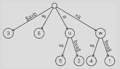
Let

$$c_f(p) = \min\{c_f(u,v) \mid (u,v) \text{ is on } p\}.$$

Definition 19 (flow defined by augmenting path)

The flow $f_p : V \times V \to \mathbb{R}$ defined by augmenting path p is

$$f_p(u,v) = \begin{cases} c_f(p) & \text{, if } (u,v) \text{ is on } p \\ 0 & \text{, otherwise.} \end{cases}$$

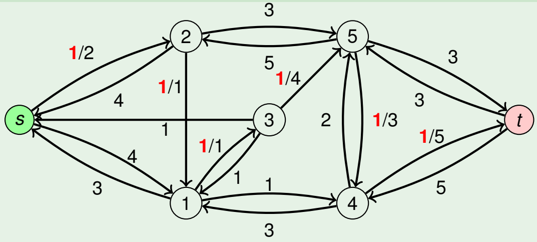


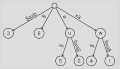
Augmenting Path

Augmenting path

- let p be an augmenting path
- f_p really is a flow in G_f
- $|f_p| = c_f(p) > 0$
- $f \uparrow f_p$ is a flow in G with value $|f| + |f_p| > |f|$

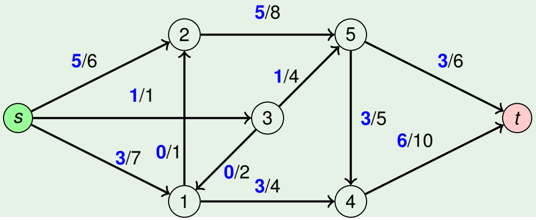
Example 20 (The flow f_p in the residual network G_f)

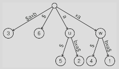




Augmenting Path

Example 21 (Augmentation $f \uparrow f_p$ of value 9)

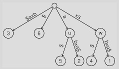




Ford-Fulkerson Method

Ford-Fulkerson Method

```
1: initialize flow $f$ to 0 everywhere
2: **while** there is an augmenting path $p$ in the residual network $G_f$ **do**
3:     augment flow $f$ along $p$ by setting $f \leftarrow f \uparrow f_p$
4: return $f$
```



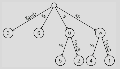
Ford-Fulkerson Method

Ford-Fulkerson Method

```
1: initialize flow $f$ to 0 everywhere
2: **while** there is an augmenting path $p$ in the residual network $G_f$ **do**
3:     augment flow $f$ along $p$ by setting $f \leftarrow f \uparrow f_p$
4: return $f$
```

Optimality

- above method increases the flow until no augmenting path exists
- is the resulting network f globally optimal or does the result value depend on the choices of the augmenting paths?



Cuts of Flow Networks

Definition 22 (Cut)

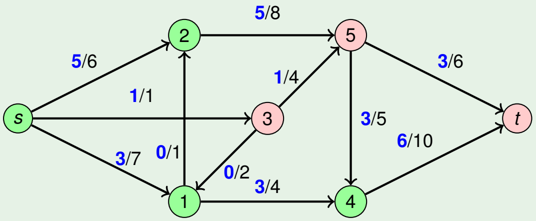
A cut (S, T) of flow network $G = (V, E)$ is a partition of V into S and $T = V \setminus S$ such that $s \in S$ and $t \in T$.

Definition 23 (Net flow across cut)

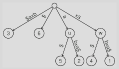
Let f be a flow in G. The net flow $f(S, T)$ across the cut (S, T) is

$$f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u).$$

Example 24 (A cut (S, T) with net flow $f(S, T) = 12 - 3 = 9$)



$S = \{s, 1, 2, 4\}$, $T = \{t, 3, 5\}$



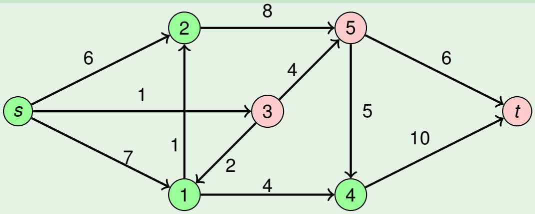
Cuts of Flow Networks

Definition 25 (Capacity of a cut)

The capacity $c(S,T)$ of the cut (S,T) is

$$c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v).$$

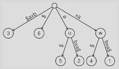
Example 26 (A cut (S,T) with capacity $c(S,T) = 19$)



$S = \{s,1,2,4\}$, $T = \{t,3,5\}$

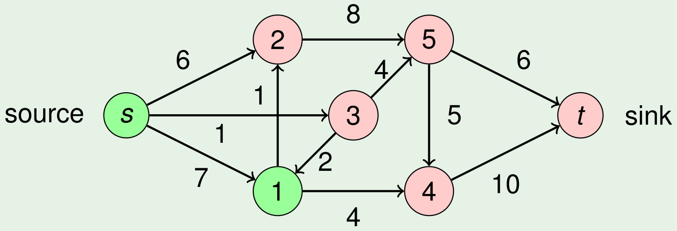
Definition 27 (Minimum cut)

A minimum cut of a flow network is a cut whose capacity is minimal over all cuts of the network.



Minimum cut

Example 28 (A minimum cut)



mininum cut $(\{s, 1\}, \{t, 2, 3, 4, 5\})$ with capacity 12

Net Flow Independent of Cut

Lemma 29

Let f be a flow in G and let (S, T) be any cut of G. Then the net flow across (S, T) is

$$f(S, T) = |f|.$$

Net Flow Independent of Cut

Lemma 29

Let f be a flow in G and let (S,T) be any cut of G. Then the net flow across (S,T) is

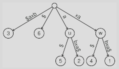
$$f(S,T) = |f|.$$

Proof.

Adding a “trick-0” to the right-hand side of the definition of $f(S,T)$ we get

$$\begin{aligned}
f(S,T) &= \sum_{u\in S}\sum_{v\in T} f(u,v) - \sum_{u\in S}\sum_{v\in T} f(v,u) + \underbrace{\sum_{u\in S}\sum_{v\in S} f(u,v) - \sum_{u\in S}\sum_{v\in S} f(v,u)}_{=0} \\
&= \sum_{u\in S}\sum_{v\in V} f(u,v) - \sum_{u\in S}\sum_{v\in V} f(v,u) \qquad (\text{as } S \,\dot\cup\, T = V) \\
&= \sum_{v\in V} f(s,v) - \sum_{v\in V} f(v,s) \qquad (1) \\
&= |f| \qquad (\text{by definition of } |f|)
\end{aligned}$$

Here, (1) follows from the definition of flow conservation for all $u \in S \setminus \{s\}$. □

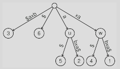


Capacity of Cut Upper Boundary for Maximal Flow

Corollary 30

Let f be any flow in a flow network and (S, T) be any cut. Then

$$|f| \leq c(S, T).$$



Capacity of Cut Upper Boundary for Maximal Flow

Corollary 30

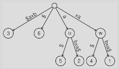
Let f be any flow in a flow network and (S, T) be any cut. Then

$$|f| \leq c(S, T).$$

Proof.

$$\begin{aligned}
|f| &= f(S, T) && \text{(by lemma 29)}\\
&= \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u) && \text{(by definition of net flow)}\\
&\leq \sum_{u \in S} \sum_{v \in T} f(u, v)\\
&\leq \sum_{u \in S} \sum_{v \in T} c(u, v)\\
&= c(S, T)
\end{aligned}$$

$\square$

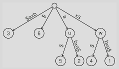


Max-Flow Min-Cut Theorem

Theorem 31 (Max-flow min-cut)

Let $G = (V, E)$ be a flow network and f be a flow in G. Then the following three conditions are equivalent:

1) *f is a maximum flow in G.*
2) *The residual network G_f contains no augmenting paths.*
3) *$|f| = c(S, T)$ for some cut (S, T) of G.*



Max-Flow Min-Cut Theorem

Theorem 31 (Max-flow min-cut)

Let $G = (V, E)$ be a flow network and f be a flow in G. Then the following three conditions are equivalent:

1) *f is a maximum flow in G.*
2) *The residual network G_f contains no augmenting paths.*
3) *$|f| = c(S, T)$ for some cut (S, T) of G.*

Proof...

1) $\Rightarrow$ 2):
Let f be a maximum flow in G and let the residual network G_f contain an augmenting path p. Then, by lemma 15, $|f \uparrow f_p|$ is a flow with value $|f| + |f_p| > |f|$. ↯

Max-Flow Min-Cut Theorem

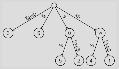
...Proof...

2) $\Rightarrow$ **3)**:
Let G_f contain no augmenting paths. Define

$$S = \{v \in V \mid \text{there is a path in } G_f \text{ from } s \text{ to } v\}$$

and $T := V \setminus S$.
Clearly, $s \in S$. Also $t \in T$ as otherwise G_f would contain an augmenting path. Therefore, (S, T) is a cut. Consider a pair $u \in S, v \in T$ of vertices. Observe, that $c_f(u, v) = 0$, as otherwise v would be reachabe by an edge in G_f from u and therefore also be in S. By definition of c_f this means that $f(u, v) = c(u, v)$ for $(u, v) \in E$ and $f(v, u) = 0$ for $(v, u) \in E$. We get

$$\begin{aligned} |f| &= f(S, T) && \text{(by lemma 29)} \\ &= \sum_{\substack{u \in S, v \in T \\ (u,v) \in E}} f(u, v) - \sum_{\substack{u \in S, v \in T \\ (v,u) \in E}} f(v, u) \\ &= \sum_{u \in S} \sum_{v \in T} c(u, v) \\ &= c(S, T) \end{aligned}$$



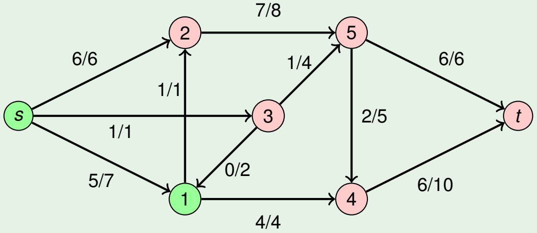
Max-Flow Min-Cut Theorem

...Proof.

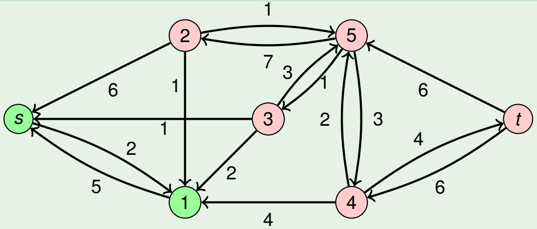
3) $\Rightarrow$ 1):
By corollary 30 we know that $|f'| \leq c(S, T)$ for any flow f'.
$|f| = c(S, T)$ therefore implies that f is maximal. $\square$

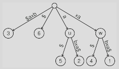
Max-Flow Min-Cut Theorem

Example 32 (max-flow f of value 12 and min-cut $(\{s, 1\}, \{t, 2, 3, 4, 5\})$ of capacity 12)



Example 33 (The residual network G_f has no augmenting paths)

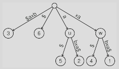




Ford-Fulkerson Algorithm

Ford-Fulkerson Algorithm

```
1: for each edge $(u,v) \in E$ do
2:     $f(u,v) \leftarrow 0$
3: while there is a path $p$ from $s$ to $t$ in the residual network $G_f$ do
4:     $c_f(p) \leftarrow \min\{c_f(u,v) \mid (u,v)$ is on $p\}$
5:     for each edge $(u,v)$ in $p$ do
6:         if $(u,v) \in E$ then
7:             $f(u,v) \leftarrow f(u,v) + c_f(p)$
8:         else
9:             $f(v,u) \leftarrow f(v,u) - c_f(p)$
10: return $f$ as maximum flow
```



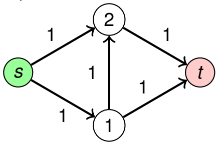
Ford-Fulkerson Algorithm

Ford-Fulkerson Algorithm

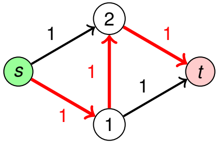
- correctness follows from Max-Flow Min-Cut Theorem
- running time depends on the choice of the augmenting path

Ford-Fulkerson Algorithm, Example

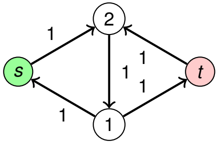
1) a flow network G



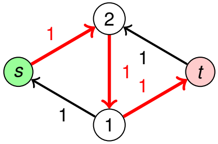
2) an augmenting path p in $G = G_f$ when $f \equiv 0$



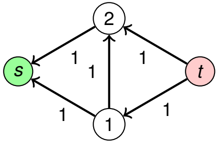
3) residual network G_f



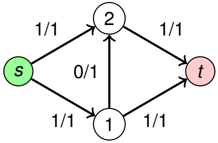
4) an augmenting path p in G_f

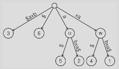


5) residual network G_f



6) maximal flow f in G of value 2





Ford-Fulkerson Algorithm

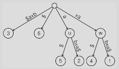
Running time

- Let $m = |E|$ be the number of edges.
- Let $E' = \{(u, v) \mid (u, v) \in E \text{ or } (v, u) \in E\}$. Then at any given time, we only need to consider edges in E'.
- $|E'| \leq 2m$
- Finding an augmenting path can be done with search algorithms that run in time $O(m)$ (e.g. depth-first search, breadth-first search).
- When capacities are integers, then $|f|$ increases each step by at least 1, therefore the running time is then in $O(m \cdot |f^*|)$, where f^* is a maximum flow.
- $\Theta(m \cdot |f^*|)$ can really be achieved in the worst case.

Ford-Fulkerson Algorithm, Example

Example 34 (flow network, where FF can take $\Theta(m \cdot |f^*|)$ time)

$$\begin{tikzcd}
 & 2 \arrow[dr, "1000"] & \\
s \arrow[ur, "1000"] \arrow[dr, "1000"'] & & t \\
 & 1 \arrow[u, "1"] \arrow[ur, "1000"'] &
\end{tikzcd}$$



Edmonds-Karp Algorithm

Choice of augmenting path

- worst-case running time of FF is unsatisfying
- want to pick augmenting paths in a way that guarantees a better worst-case running time
- shortest augmenting path: shortest path from s to t in the residual network (minimal number of edges)

Edmonds-Karp algorithm

Follow the general Ford-Fulkerson method, and choose for each update step a shortest augmenting path.

Reminder: Breadth-First Search

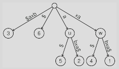
Breadth-first search

Let $G = (V, E)$ be a (directed) graph and $s \in V$ be the source.

```
 1: $Q \leftarrow$ empty queue
 2: **for** each vertext $v \in V \setminus \{s\}$ **do**
 3:     $p(v) \leftarrow NULL$ // predecessor on the shortest path to $s$
 4:     $d(v) \leftarrow \infty$ // distance to $s$
 5:     color$(v) \leftarrow$ white // white: not queued yet
 6: $d(s) \leftarrow 0$, color$(s) \leftarrow$ gray, $p(s) \leftarrow NULL$
 7: insert $s$ into $Q$
 8: **while** $Q$ not empty **do**
 9:     $u \leftarrow$ remove first element of $Q$
10:     **for** each $v$ such that $(u, v) \in E$ **do**
11:         **if** color$(v)$ = white **then**
12:             $d(v) \leftarrow d(u) + 1$
13:             $p(v) \leftarrow u$
14:             color$(v) \leftarrow$ gray
15:             insert $v$ into $Q$
```

Running time: $O(|V| + |E|)$

For any node v (e.g. the sink in a flow network) we can find the shortest path to s by iteratively following the links $p(v)$ until reaching s.



Edmonds-Karp Algorithm

Lemma 35

Let $G = (V, E)$ be a flow network with source s and sink t, let $v \in V \setminus \{s, t\}$. Let f be a flow. Let $d_f(v)$ be the shortest-path distance from s to v in the residual network G_f.
Then, during the Edmonds-Karp algorithm, $d_f(v)$ increases monotonically with each flow augmentation.

Edmonds-Karp Algorithm

Proof.

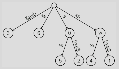
Assume for the sake of contradiction that the lemma was not true and the distance from s decreases for some vertex for some updating step. Let f and f' be the flows just before and after this updating step, respectively. Let $v \in V \setminus \{s, t\}$ be such that $d_f(v) > d_{f'}(v)$ and such that $d_{f'}(v)$ is minimal among such vertices.
Let u be the last vertex visited on a shortest path from s to t in $G_{f'}$: $s \rightsquigarrow u \to v$. Then $d_{f'}(u) = d_{f'}(v) - 1$ by definition of a shortest path.
Suppose we had $(u, v) \in E_f$ as well. Then

$$\begin{aligned} d_f(v) &\leq d_f(u) + 1 && \text{(triangle inequality)} \\ &\leq d_{f'}(u) + 1 && \text{(as } v \text{ was the closest vertext to } s \text{ in } G_f' \text{ violating monotonicity)} \\ &= d_{f'}(v) \end{aligned}$$

would contradict our assumption $d_f(v) > d_{f'}(v)$. Therefore, $(u, v) \notin E_f$.
Since we have $(u, v) \notin E_f$ and $(u, v) \in E_{f'}$ the augmenting path in G_f must contain the edge (v, u). Therefore

$$\begin{aligned} d_f(v) &= d_f(u) - 1 \\ &\leq d_{f'}(u) - 1 \\ &= d_{f'}(v) - 2 \\ &< d_{f'}(v). \end{aligned}$$

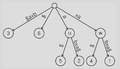
↯This contradicts the assumption that v violates the monotonicity from f to f'. Therefore, the claim of the lemma must be true. □



Edmonds-Karp Algorithm

Theorem 36

Let $G = (E, V)$ be a flow network with $n = |V|$ vertices and $m = |E|$ edges. The total number of flow augmentations performed by the Edmonds-Karp algorithm is $O(nm)$ and the total running time to find the maximum flow is $O(nm^2)$.



Edmonds-Karp Algorithm

Theorem 36

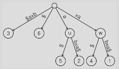
Let $G = (E, V)$ be a flow network with $n = |V|$ vertices and $m = |E|$ edges. The total number of flow augmentations performed by the Edmonds-Karp algorithm is $O(nm)$ and the total running time to find the maximum flow is $O(nm^2)$.

Proof...

We say that an edge (u, v) is critical in a residual network G_f on an augmenting path p if $c_f(p) = c_f(u, v)$. Therefore, after augmenting the flow along the augmenting path, a critical edge is removed from the residual network. Consider some vertex pair (u, v) such that either $(u, v) \in E$ or $(v, u) \in E$. If that vertex pair ever becomes a critical edge in some residual network, then let f be the flow the first time it becomes critical. Then

$$d_f(v) = d_f(u) + 1$$

as the augmenting path is a shortest path in G_f. If (u, v) should ever become critial again, then in the meantime an augmenting path must have included the reverse edge (v, u). Let f' be the flow of the residual network at that time.



Edmonds-Karp Algorithm

...Proof.

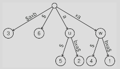
Then

$$d_{f'}(u) = d_{f'}(v) + 1.$$

We get

$$\begin{aligned} d_{f'}(u) &= d_{f'}(v) + 1 \\ &\geq d_f(v) + 1 && \text{(by lemma 35)} \\ &= d_f(u) + 2 \end{aligned}$$

The distance to the source must have increased by at least 2 each time (u, v) becomes critical again. As $u \neq t$ the distance to the source is limited by $n - 2$. By lemma 35 the distance can only increase, which limits the total number of times that (u, v) becomes critical to $1 + (n-2)/2 = n/2$. The total number of vertex pairs that can become critical edges is bounded by $2m$. As each augmenting path has at least one critical edge on it, there can be at most nm flow augmentation steps.
Each search for an augmenting path by breadth-first search takes $O(n + m)$ time, which is also $O(m)$ as we have assumed that there is a path $s \rightsquigarrow v \rightsquigarrow t$ for all $v \in V$. We therefore get a total worst-case time bound of $O(nm^2)$. $\square$



Example: Assigning Seminar Topics

Example 37 (Topics in a Bioinformatics seminar)

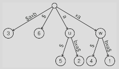
t_1	RNA-Sekundärstrukturvorhersage mit dem Nussinov-Algorithmus
t_2	Ein Wettspiel und die Häufigkeitsverteilung eines Musters in einer Sequenz
t_3	Spaced Seeds **(2)**
t_4	Flüsse in Netzwerken und Multiples Sequenzalignment
t_5	Hirschbergs Algorithmus mit linearem Speicherverbrauch
t_6	Ant Colony Optimization
t_7	Genetische Algorithmen
t_8	Ein Kompressionsalgorithmus für DNA
t_9	Einführung in Knotentheorie
t_{10}	Polynomialzeitreduktion auf Multiples Alignment **(2)**
t_{11}	Spliced Alignment mittels Netzwerk Alignment
t_{12}	HMMs und das Logarithmische Zahlensystem
t_{13}	Frameshift-korrigierendes HMM von ESTScan
t_{14}	Schätzen einer Multinomialverteilung mit einem Bayes-Ansatz **(2)**
t_{15}	Einführung in Neuronale Netze **(2)**
t_{16}	Genvorhersage in der Metagenomik mit dem Programm Orphelia
t_{17}	Erkennung von chimärischen 16S rRNA Sequenzen
t_{18}	Topologie von Transkriptionsnetzwerken von Säugern
t_{19}	Bewerten der Konserviertheit eines Residuums

Example: Assigning Seminar Topics

Seminar assignment problem

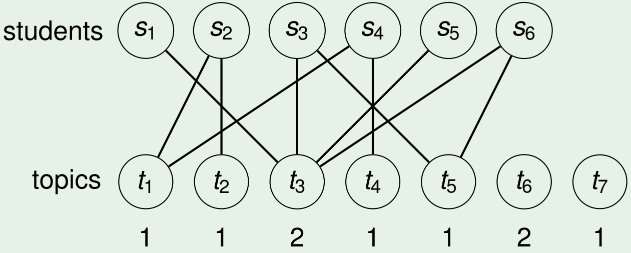
- 16 students $s_1, \ldots, s_{16}$, 19 topics $t_1, \ldots, t_{19}$
- some topics can be presented by two students
- each student is asked to give a nonempty list of preferred topics
- consider feasible assignments of topics to students, such that
  - every student gets exactly one topic
  - every topic is taken by at most 1 or 2 (if appropriate) students

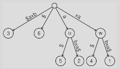
Find a feasible assignment of topics to students, such that all students get a topic they prefer or, if not possible, such that a maximal number of students get a topic of their preference.



Example: Assigning Seminar Topics

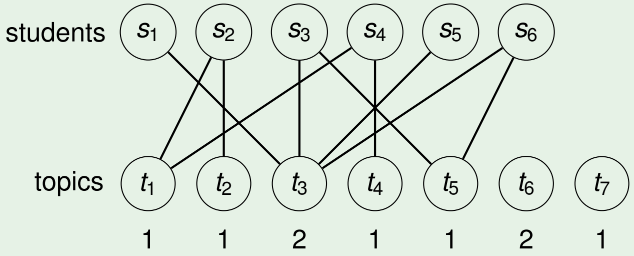
Example 38 (Seminar topic preferences)





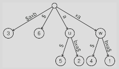
Example: Assigning Seminar Topics

Example 38 (Seminar topic preferences)



Disallowing topics that can be taken more than once

For convenience, we will allow each topic to be taken at most once. When there are topics that can be taken by two or more students, then we can formulate an equivalent seminar assignment problem in which each topic can only be taken once. How?

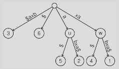


(Bipartite) Matching

Definition 39 (Matching)

Let $G = (V, E)$ be an undirected graph.

- A matching is a subset $M \subset E$ of edges such that for all vertices $v \in V$, *at most one* edge of M is incident on v.
- If v is incident to an edge in M, then v is matched, otherwise it is unmatched.
- A maximum matching is a matching of maximal size.



(Bipartite) Matching

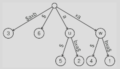
Definition 39 (Matching)

Let $G = (V, E)$ be an undirected graph.

- A matching is a subset $M \subset E$ of edges such that for all vertices $v \in V$, *at most one* edge of M is incident on v.
- If v is incident to an edge in M, then v is matched, otherwise it is unmatched.
- A maximum matching is a matching of maximal size.

Definition 40 (Bipartite graph (German: “bipartit” oder “paar”))

A bipartite graph is an undirected graph $G = (V, E)$, in which V can be partitioned into two sets L, R, such that for all $\{u, v\} \in E$ either $u \in L$ and $v \in R$ or $u \in R$ and $v \in L$.



(Bipartite) Matching

Definition 39 (Matching)

Let $G = (V, E)$ be an undirected graph.

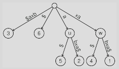
- A matching is a subset $M \subset E$ of edges such that for all vertices $v \in V$, *at most one* edge of M is incident on v.
- If v is incident to an edge in M, then v is matched, otherwise it is unmatched.
- A maximum matching is a matching of maximal size.

Definition 40 (Bipartite graph (German: “bipartit” oder “paar”))

A bipartite graph is an undirected graph $G = (V, E)$, in which V can be partitioned into two sets L, R, such that for all $\{u, v\} \in E$ either $u \in L$ and $v \in R$ or $u \in R$ and $v \in L$.

Maximum bipartite matching problem

Find a maximum matching in a given bipartite graph.

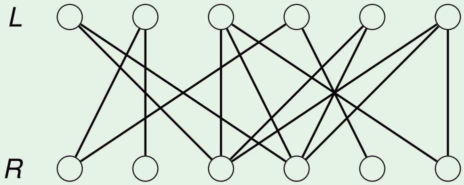


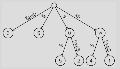
Corresponding Flow Network

Flow network corresponding to bipartite graph

- let $G = (V, E)$ be a bipartite graph with partition $V = L \dot\cup R$
- assumption: No vertex has degree 0. (Otherwise, remove them first.)
- design a corresponding flow network $G' = (V', E')$

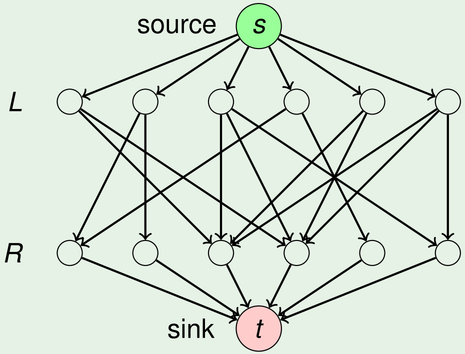
Example 41 (Bipartite graph G from seminar assignment example)

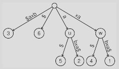




Corresponding Flow Network

Example 42 (Corresponding flow network G')



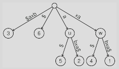


Corresponding Flow Network

Definition 43 (Flow network corresponding to bipartite graph)

Let $G = (V, E)$ be a bipartite graph with partition $V = L \dot\cup R$. The corresponding flow network $G' = (V', E')$ is defined as follows:

- $V' := V \cup \{s, t\}$, where source s and sink t are two new vertices
- $$\begin{aligned} E' \quad := \quad & \{(u, v) \mid u \in L, v \in R, \{u, v\} \in E\} \\ & \cup \{(s, v) \mid v \in L\} \\ & \cup \{(v, t) \mid v \in R\} \end{aligned}$$
- $c(u, v) = 1$ if $(u, v) \in E'$, $c(u, v) = 0$, otherwise. (“unit capacity”)



Corresponding Flow Network

Definition 43 (Flow network corresponding to bipartite graph)

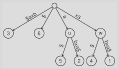
Let $G = (V, E)$ be a bipartite graph with partition $V = L \dot\cup R$. The corresponding flow network $G' = (V', E')$ is defined as follows:

- $V' := V \cup \{s, t\}$, where source s and sink t are two new vertices
- $$\begin{aligned} E' \quad := \quad & \{(u,v) \mid u \in L, v \in R, \{u,v\} \in E\} \\ & \cup \{(s,v) \mid v \in L\} \\ & \cup \{(v,t) \mid v \in R\} \end{aligned}$$
- $c(u, v) = 1$ if $(u, v) \in E'$, $c(u, v) = 0$, otherwise. ("unit capacity")

Observation

$$|E| \leq |E'| = |E| + |V| \leq 3|E|$$

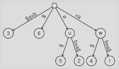
as G has no isolated nodes and therefore $|E'| = \Theta(E)$.



Correspondence Between Matchings and Flows

Definition 44 (integer-valued)

A flow f is integer-valued if $f(u, v)$ is an integer for all $(u, v) \in V \times V$



Correspondence Between Matchings and Flows

Definition 44 (integer-valued)

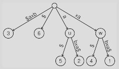
A flow f is integer-valued if $f(u, v)$ is an integer for all $(u, v) \in V \times V$

Lemma 45 (Correspondence between matchings and flows)

Let G, G' be as above.

a) *If M is a matching in G, then there is an integer-valued flow f in G' with value $|f| = |M|$.*

b) *If f is an integer-valued flow in G', then there is a matching M in G of size $|M| = |f|$.*



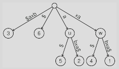
Correspondence Between Matchings and Flows

Proof.

a)
Let M be a matching in G. Define f as follows. For any $\{u, v\} \in M$ with $u \in L, v \in R$, let $f(s, u) = f(u, v) = f(v, t) = 1$. Let the flow f be 0 on all other edges in E'. This f satisfies the capacity constraint and flow conservation as each vertex in E is incident to at most one edge in M. Therefore, f is an integer-valued flow. Consider the cut $(S, T) = (L \cup \{s\}, R \cup \{t\})$. The net flow across this cut is $f(S, T) = |M|$ by definition of f and according to lemma 29 $f(S, T) = |f|$.
b)
Let f be an integer-valued flow in G'. As capacities in G' are either 0 or 1, also $f(u, v) \in \{0, 1\}$. Define M as follows.

$$M := \{\{u, v\} \mid u \in L, v \in R, f(u, v) = 1\}$$

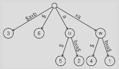
Again, $|f| = f(L \cup \{s\}, R \cup \{t\})$ by lemma 29. The net flow across this cut is equal to the number of edges between L and R with a flow of 1, which is $|M|$. $\square$



Integrality of Flow

Integrality

In general, in a flow network corresponding to a bipartite graph, a maximum flow must have a value that is an integer, but need itself not be integer-valued. However, ...



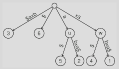
Integrality of Flow

Integrality

In general, in a flow network corresponding to a bipartite graph, a maximum flow must have a value that is an integer, but need itself not be integer-valued. However, ...

Lemma 46

If the capacity function c takes only integer values, then $|f|$ is an integer and the maximum flow f produced by the Ford Fulkerson method is integer-valued.



Integrality of Flow

Integrality

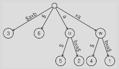
In general, in a flow network corresponding to a bipartite graph, a maximum flow must have a value that is an integer, but need itself not be integer-valued. However, ...

Lemma 46

If the capacity function c takes only integer values, then $|f|$ is an integer and the maximum flow f produced by the Ford Fulkerson method is integer-valued.

Proof.

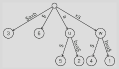
The algorithm starts with $f \equiv 0$, an integer-valued flow. Let f be an integer-valued flow (induction hypothesis) and p be an augmenting path in G_f. $c_f(p)$ is an integer as the capacities are integers. Therefore, the flow $f \uparrow f_p$ is again integer-valued. The claim that the maximum flow produced FF is integer-valued follows by induction. That $|f|$ is integer-valued is a direct consequence. $\square$



Maximum Matching and Maximum Flow

Corollary 47

The size $|M|$ of a maximum matching M in a bipartite graph G is equal to the value $|f|$ of a maximum flow f in its corresponding flow network G'.



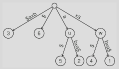
Maximum Matching and Maximum Flow

Corollary 47

The size $|M|$ of a maximum matching M in a bipartite graph G is equal to the value $|f|$ of a maximum flow f in its corresponding flow network G'.

Proof.

Let M be a maximum matching. By lemma 45 a), there is a flow f with value $|f| = |M|$. If $|f|$ was not the value of a maximum flow, then there was another flow f' of higher value, $|f'| > |f|$, which can by lemma 46 be assumed to be integer-valued. This would imply, by lemma 45 b), that there is another match M' of size $|M'| = |f'| > |f|$, which contradicts that M is maximal. Therefore, $|f|$ is the value of a maximum flow. $\square$



Finding a Maximum Matching

Finding a maximum matching

Let G be a bipartite graph.

- create the flow network G'
- find a maximum flow f in G' with a Ford-Fulkerson method
- construct a maximum matching M from f as in lemma 45 b)

Running time

- the worst-case running time of FF is $O(|E'| \cdot |f^*|)$, where $|f^*|$ is the value of a maximum flow
- $|E'| = \Theta(|E|)$ and $|f^*| = O(|V|)$
- $\Rightarrow$ the running time of above algorithm is $O(nm)$, where $n = |V|$ and $m = |E|$.