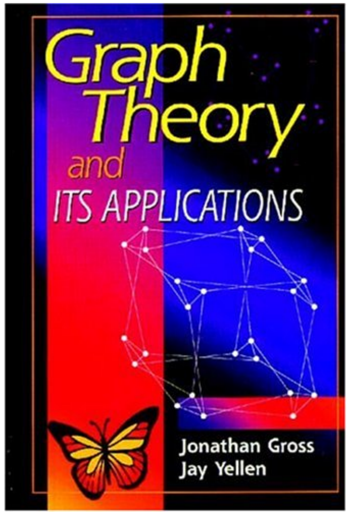
V18 Flows and Cuts in Networks

This lecture follows closely chapter 12.1 in the book on the right on „Flows and Cuts in Networks and Chapter 12.2 on "Solving the Maximum-Flow Problem"



Flow in Networks can mean
- flow of oil or water in pipelines, electricity
- phone calls, emails, traffic networks ...

Equivalences exist between
max-flow min-cut theorem of Ford and Fulkerson
& the **connectivity theorems** of Menger

$\rightarrow$ efficient algorithms were developed to deal with a number of different practical problems that involve solving scheduling and assignment problems.

Single Source – Single Sink Capacitated Networks

Definition: A **single source – single sink network** is a connected digraph that has a distinguished vertex called the **source** with nonzero outdegree and a distinguished vertex called the **sink** with nonzero indegree.

Such a network with source *s* and sink *t* is often referred to as a ***s-t* network**.

Definition: A **capacitated network** is a connected digraph such that each arc *e* is assigned a nonnegative weight ***cap(e),*** called the **capacity** of arc *e.*

Notation: Let *v* be a vertex in a digraph *N*. Then ***Out(v)*** denotes the set of all arcs that are directed **away from** vertex *v*. That is,

$$Out(v) = \{e \in E_N | tail(e) = v\}$$

Correspondingly, ***In(v)*** denotes the set of arcs that are directed **to** vertex *v:*

$$In(v) = \{e \in E_N | head(e) = v\}$$

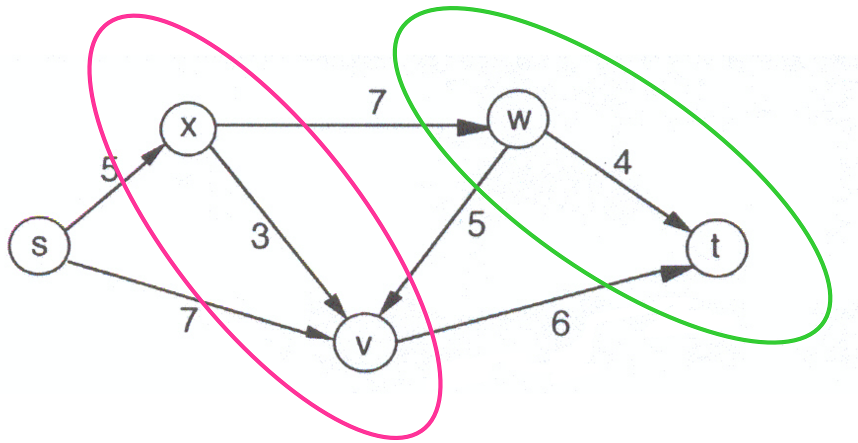
Single Source – Single Sink Capacitated Networks

Notation: For any two vertex subsets X and Y of a digraph N, let $\langle \boldsymbol{X,Y} \rangle$ denote the set of arcs in N that are directed **from** a vertex in X **to** a vertex in Y.

$$\langle X,Y \rangle = \left\{ e \in E_N \middle| tail(e) \in X \text{ and } head(e) \in Y \right\}$$

Example: The figure shows a 5-vertex capacitated s-t-network.

If $\boldsymbol{X} = \{x,v\}$ and $\boldsymbol{Y} = \{w,t\}$, then the elements of arc set $\langle X,Y \rangle$ are the arc directed from vertex x to vertex w and the arc directed from vertex v to sink t.



A 5-vertex capacitated network with source s and sink t.

The only element in arc set $\langle Y,X \rangle$ is the arc directed from vertex w to vertex v.

Feasible Flows

Definition: Let N be a capacitated s-t-network.

A **feasible flow** f in N is a function $f{:}E_N \rightarrow R^+$ that assigns a nonnegative real number to every vertex v in network N, other than source s and sink t, and that fulfills the following two conditions

1. (**capacity constraints**) $f(e) \leq cap(e)$, for every arc e in network N.
2. (**conservation constraints**) $$\sum_{e \in In(v)} f(e) = \sum_{e \in Out(v)} f(e)$$

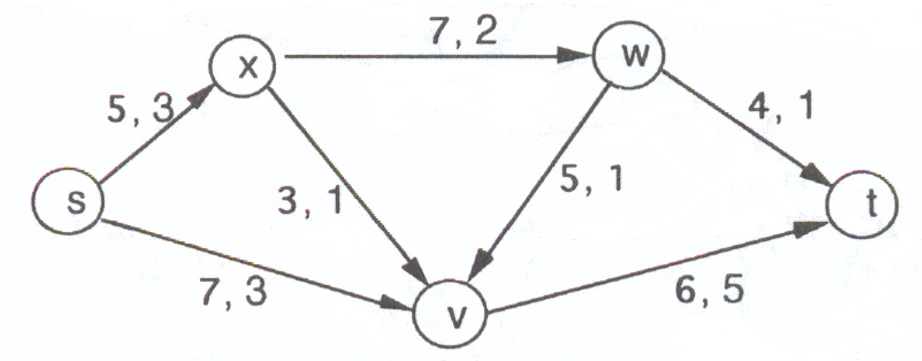
Property 2 above is called the **conservation-of-flow** condition.

E.g. for an oil pipeline, the total flow of oil going into any juncture (vertex) in the pipeline must equal the total flow leaving that juncture.

Notation: to distinguish visually between the flow and the capacity of an arc, we adopt the **convention** in drawings that when both numbers appear, the **capacity** will always be in **bold** and to the left of the flow.

Feasible Flows

Example: The figure shows a feasible flow for the previous network.
Notice that the total amount of flow leaving source *s* equals 6, which is also the net flow entering sink *t.*



Definition: The **value of flow** *f* in a capacitated network *N*, denoted with ***val(f)***, is the net flow leaving the source *s*, that is

$$val(f) = \sum_{e \in Out(s)} f(e) - \sum_{e \in In(s)} f(e)$$

Definition: The **maximum flow *f**** in a capacitated network *N* is a flow in *N* having the maximum value, i.e. *val(f)* $\leq$ *val(f*),* for every flow *f* in *N.*

Cuts in *s-t* Networks

By definition, any nonzero flow must use at least one of the arcs in *Out(s).*
In other words, if all of the arcs in *Out(s)* were deleted from network *N*,
then no flow could get from source *s* to sink *t.*

This is a special case of the following definition, which combines the concepts of **partition-cut** and **s-t separating set.**

From V17
Definition: Let G be a graph, and let X_1 and X_2 form a **partition** of V_G.
The set of all edges of G having one endpoint in X_1 and the other endpoint in X_2 is called a **partition-cut** of G and is denoted $\langle X_1, X_2 \rangle$.

From V17
Definition: Let u and v be distinct vertices in a connected graph G.
A vertex subset (or edge subset) S is u-v **separating** (or **separates** u and v), if the vertices u and v lie in different components of the deletion subgraph $G - S$.

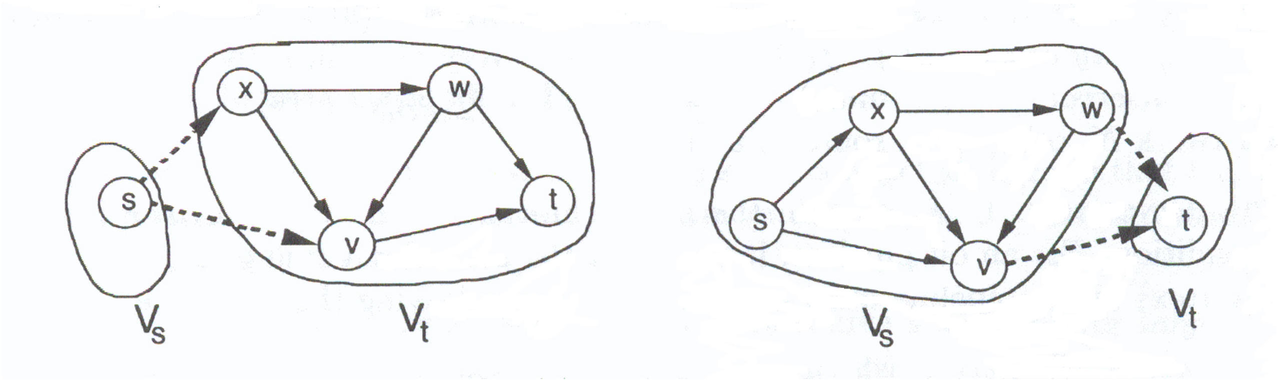
Cuts in *s-t* Networks

Definition: Let N be an *s-t* network, and let V_s and V_t form a **partition** of V_G such that source $s \in V_s$ and sink $t \in V_t$.
Then the set of all arcs that are directed from a vertex in set V_s to a vertex in set V_t is called an ***s-t* cut** of network N and is denoted $\langle V_s, V_t \rangle$.

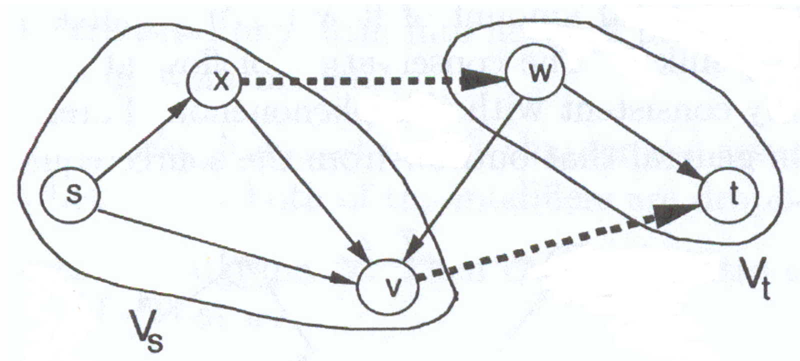
Remark: The arc set ***Out(s)*** for an *s-t* network N is the *s-t* cut $\langle \{s\}, V_N - \{s\} \rangle$, and ***In(t)*** is the *s-t* cut $\langle V_N - \{t\}, \{t\} \rangle$.

Cuts in *s-t* Networks

<u>Example</u>. The figure portrays the arc sets *Out(s)* and *In(t)* as *s-t* cuts, where *Out(s)* = ⟨ {*s*}, {*x,v,w,t*} ⟩ and *In(t)* = ⟨ {*s,x,v,w*},{*t*} ⟩.



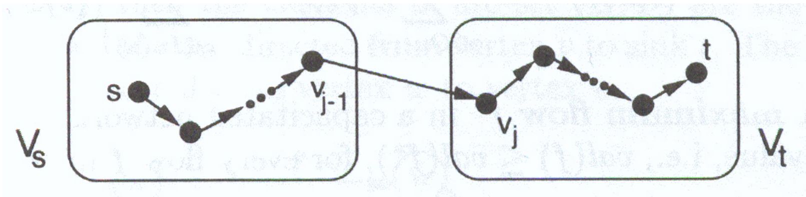
<u>Example</u>: a more general *s-t* cut ⟨ V_s, V_t ⟩ is shown below, where V_s = {*s,x,v*} and V_t = {*w,t*}.



Cuts in *s-t* Networks

Proposition 12.1.1 Let $\langle V_s, V_t \rangle$ be an *s-t* cut of a network *N.*
Then every directed *s-t* path in *N* contains at least one arc in $\langle V_s, V_t \rangle$.

Proof. Let $P = \langle s = v_0, v_1, v_2, \ldots, v_l = t \rangle$ be the vertex sequence of a directed *s-t* path in network *N.*



Since $s \in V_s$ and $t \in V_t$, there must be a first vertex v_j on this path that is in set V_t (see figure below).

Then the arc from vertex v_{j-1} to v_j is in $\langle V_s, V_t \rangle$. □

Relationship between Flows and Cuts

Similar to viewing the set *Out(s)* of arcs directed from source *s* as the *s*-*t* cut $\langle\{s\}, V_N - \{s\}\rangle$, the set *In(s)* may be regarded as the set of „backward" arcs relative to this cut, namely, the arc set $\langle V_N - \{s\}, \{s\}, \rangle$.

From this perspective, the definition of *val(f)* may be rewritten as

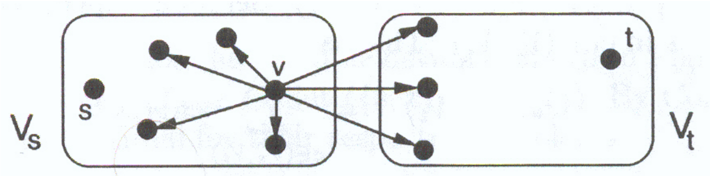
$$val(f) = \sum_{e \in \langle \{s\}, V_N - \{s\} \rangle} f(e) - \sum_{e \in \langle V_N - \{s\}, \{s\} \rangle} f(e)$$

Relationship between Flows and Cuts

Lemma 12.1.2. Let $\langle V_s, V_t \rangle$ be any *s*-*t* cut of an *s*-*t* network *N*. Then

$$\bigcup_{v \in V_s} Out(v) = \langle V_s, V_s \rangle \cup \langle V_s, V_t \rangle \quad \text{and} \quad \bigcup_{v \in V_s} In(v) = \langle V_s, V_s \rangle \cup \langle V_t, V_s \rangle$$

Proof: For any vertex $v \in V_s$, each arc directed from *v* is either in $\langle V_s, V_s \rangle$ or in $\langle V_s, V_t \rangle$. The figure illustrates for a vertex *v* the partition of *Out(v)* into a 4-element subset of $\langle V_s, V_s \rangle$ and a 3-element subset of $\langle V_s, V_t \rangle$.



$$\bigcup_{v \in V_s} Out(v) = \langle V_s, V_s \rangle \cup \langle V_s, V_t \rangle$$

Similarly, each arc directed to vertex *v* is either in $\langle V_s, V_s \rangle$ or in $\langle V_t, V_s \rangle$. □

Relationship between Flows and Cuts

Proposition 12.1.3. Let *f* be a flow in an *s-t* network *N*, and let $\langle V_s, V_t \rangle$ be any *s-t* cut of *N*. Then

$$val(f) = \sum_{e \in \langle V_s, V_t \rangle} f(e) - \sum_{e \in \langle V_t, V_s \rangle} f(e)$$

Proof: By definition,

$$val(f) = \sum_{e \in Out(s)} f(e) - \sum_{e \in In(s)} f(e)$$

And by the conservation of flow

$\sum_{e \in Out(v)} f(e) - \sum_{e \in In(v)} f(e) = 0$ for every $v \in V_s$ other than s. Thus one can expand

$$val(f) = \sum_{v \in V_s} \left(\sum_{e \in Out(v)} f(e) - \sum_{e \in In(v)} f(e) \right) = \sum_{v \in V_s} \sum_{e \in Out(v)} f(e) - \sum_{v \in V_s} \sum_{e \in In(v)} f(e) \qquad (1)$$

By Lemma 12.1.2.

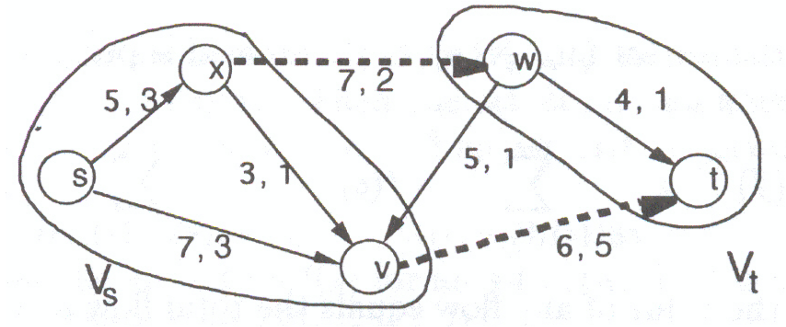
$$\sum_{v \in V_s} \sum_{e \in Out(v)} f(e) = \sum_{e \in \langle V_s, V_s \rangle} f(e) + \sum_{e \in \langle V_s, V_t \rangle} f(e) \quad \text{and}$$

$$\sum_{v \in V_s} \sum_{e \in In(v)} f(e) = \sum_{e \in \langle V_s, V_s \rangle} f(e) + \sum_{e \in \langle V_t, V_s \rangle} f(e) \qquad (2)$$

Now enter the right hand sides of (2) into (1) and obtain the desired equality. □

Example

The flow *f* and cut ⟨{*s,x,v*},{*w,t*}⟩ shown in the figure illustrate Proposition 12.1.3.



$$6 = val(f) = \sum_{e \in \langle\{s,x,v\},\{w,t\}\rangle} f(e) - \sum_{e \in \langle\{w,t\},\{s,x,v\}\rangle} f(e) = 7 - 1$$

The next corollary confirms something that was apparent from intuition: the net flow out of the source *s* equals the net flow into the sink *t.*

Corollary 12.1.4 Let *f* be a flow in an *s-t* network. Then

$$val(f) = \sum_{e \in In(t)} f(e) - \sum_{e \in Out(t)} f(e)$$

Proof: Apply proposition 12.1.3 to the *s-t* cut *In(t)* = ⟨ V_N – {*t*}, {*t*} ⟩. □

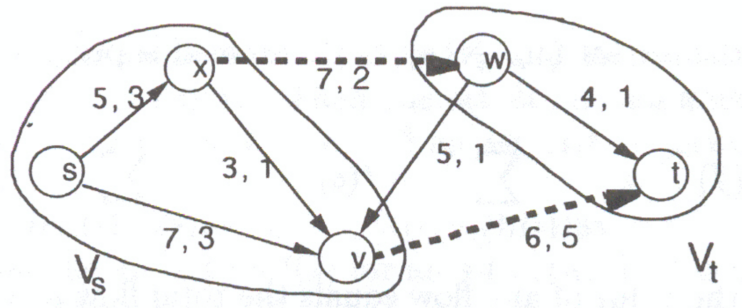
Example

Definition. The **capacity of a cut** $\langle V_s, V_t \rangle$ denoted **cap**$\langle V_s, V_t \rangle$, is the sum of the capacities of the arcs in cut $\langle V_s, V_t \rangle$. That is

$$cap\langle V_s, V_t \rangle = \sum_{e \in \langle V_s, V_t \rangle} cap(e)$$

Definition. The **minimum cut** of a network *N* is a cut with the minimum capacity.

Example. The capacity of the cut shown in the previous figure is 13,
And the cut $\langle$\{s,x,v,w\},\{t\}$\rangle$ with capacity 10, is the only minimum cut.



Maximum-Flow and Minimum-Cut Problems

The problems of finding the maximum flow in a capacitated network N and finding a minimum cut in N are closely related.

These two optimization problems form a *max-min* pair.

The following proposition provides an upper bound for the maximum-flow problem.

Maximum-Flow and Minimum-Cut Problems

Proposition 12.1.5 Let *f* be any flow in an *s*-*t* network, and let $\langle V_s, V_t\rangle$ be any *s*-*t* cut.
Then $$val(f) \le cap\langle V_s, V_t\rangle$$

Proof:

$$\begin{aligned}
val(f) &= \sum_{e\in\langle V_s,V_t\rangle} f(e) - \sum_{e\in\langle V_t,V_s\rangle} f(e) && \text{(by proposition 12.1.3)}\\
&\le \sum_{e\in\langle V_s,V_t\rangle} cap(e) - \sum_{e\in\langle V_t,V_s\rangle} f(e) && \text{(by capacity constraints)}\\
&= cap\langle V_s,V_t\rangle - \sum_{e\in\langle V_t,V_s\rangle} f(e) && \text{(by definition of } cap\langle V_s,\mathrm{V}_t\rangle)\\
&\le cap\langle V_s,V_t\rangle && \text{(since each } f(e) \text{ is nonnegative)} \quad \square
\end{aligned}$$

Maximum-Flow and Minimum-Cut Problems

Corollary 12.1.6 (Weak Duality) Let *f** be a maximum flow in an *s-t* network *N*, and let *K** be a minimum *s-t* cut in *N*. Then

$$val(f^*) \leq cap(K^*)$$

Proof: This follows immediately from proposition 12.1.5.

Corollary 12.1.7 (Certificate of Optimality) Let *f* be a flow in an *s-t* network *N* and *K* an *s-t* cut, and suppose that *val(f) = cap(K).*
Then flow *f* is a **maximum flow** in network *N*, and cut *K* is a **minimum cut**.

Proof: Let *f'* be any feasible flow in network *N.*
Proposition 12.1.5 and the premise give

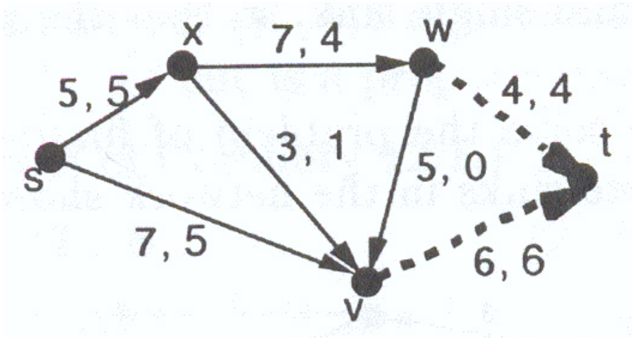
$$val(f') \leq cap(K) = val(f)$$ → *f* is a maximum flow

On the other hand, let $\langle V_s, V_t \rangle$ be any *s-t* cut. Proposition 12.1.5:

$$cap(K) = val(f) \leq cap\langle V_s, V_t \rangle$$ → *K* is a minimum cut. □

Example

Example The flow for the example network shown in the figure has value 10, which is also the capacity of the *s*-*t* cut $\langle\{s,x,v,w\},\{t\}\rangle$.
By corollary 12.1.7, both the flow and the cut are optimal for their respective problem.



A maximum flow and minimum cut.

Corollary 12.1.8 Let $\langle V_s, V_t\rangle$ be an *s*-*t* cut in a network *N*, and suppose that *f* is a flow such that

$$f(e) = \begin{cases} cap(e) & \text{if } e \in \langle V_s, V_t\rangle \\ 0 & \text{if } e \in \langle V_t, V_s\rangle \end{cases}$$

Then *f* is a maximum flow in *N*, and $\langle V_s, V_t\rangle$ is a minimum cut.

Solving the Maximum-Flow Problem

We will present an algorithm that originated by Ford and Fulkerson (1962).
Idea: increase the flow in a network iteratively until it cannot be increased any further $\rightarrow$ **augmenting flow path**.

Suppose that f is a flow in a capacitated s-t network N, and suppose that there exists a directed s-t path

$$P = \langle s, e_1, v_1, e_2, \ldots, e_k, t \rangle$$

in N, such that $f(e_i) < cap(e_i)$ for $i=1, \ldots, k$.

Then considering arc capacities only, the flow on each arc e_i can be increased by as much as $cap(e_i) - f(e_i)$.

But to maintain the **conservation-of-flow** property at each of the vertices v_i, the **increases** on all of the arcs of path P **must be equal**.

Thus, if Δ_P denotes this **increase**,
then the largest possible value for Δ_P is $min\{cap(e_i\} - f(e_i)\}$.

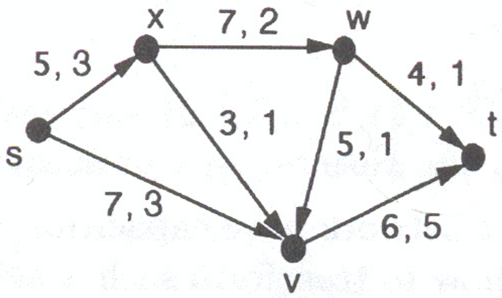
Solving the Maximum-Flow Problem

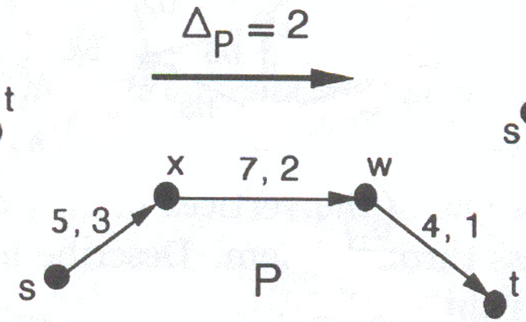
Example: Left: the value of the current flow is 6.

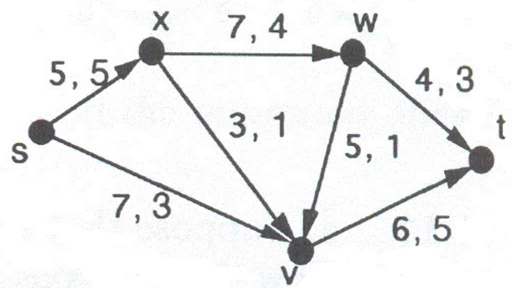
Consider the directed *s-t* path $P = \langle s,x,w,t \rangle$.

The flow on each arc of path P can be increased by $\Delta_P = 2$.

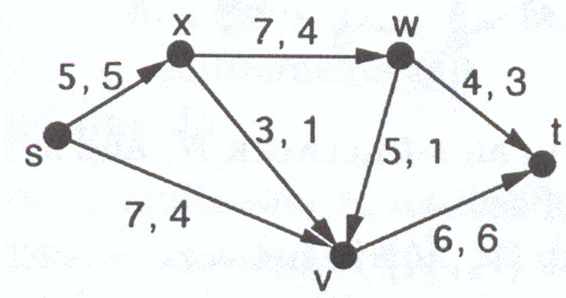
The resulting flow, which has value 8, is shown on the right side.







Using the directed path $\langle s,v,t \rangle$, the flow can be increased to 9. The resulting flow is shown right.



At this point, the flow cannot be increased any further along **directed *s-t* paths**, because each such path must either use the arc directed from *s* to *x* or from *v* to *t*. Both arcs have flow at capacity.

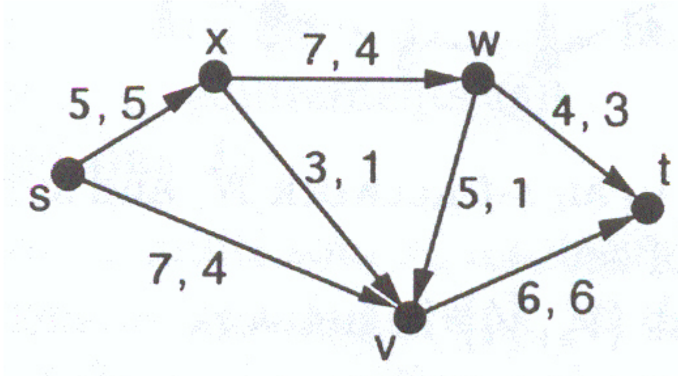
Solving the Maximum-Flow Problem

However, the flow can be increased further.

E.g. increase the flow on the arc from source *s* to vertex *v* by one unit,

<u>decrease</u> the flow on the arc from *w* to *v* by one unit, and

increase the flow on the arc from *w* to *t* by one unit.



f-Augmenting Paths

Definition: An *s-t* **quasi-path** in a network *N* is an alternating sequence

$$\langle s = v_0, e_1, v_1, \ldots, v_{k-1}, e_k, v_k = t \rangle$$

of vertices and arcs that forms an *s-t* path in the underlying undirected graph of *N*.

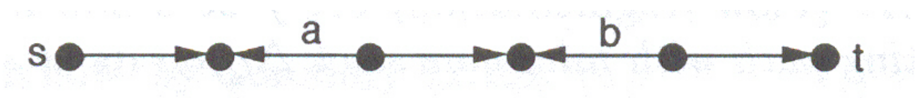
Terminology For a given *s-t* quasi-path

$$Q = \langle s = v_0, e_1, v_1, \ldots, v_{k-1}, e_k, v_k = t \rangle$$

arc e_i is called a **forward arc** if it is directed from vertex v_{i-1} to vertex v_i and
arc e_i is called a **backward arc** if it is directed from v_i to v_{i-1}.

Clearly, a directed *s-t* path is a quasi-path whose arcs are all forward.

Example. On the *s-t* quasi-path shown below, arcs *a* and *b* are backward, and the three other arcs are forward.



f-Augmenting Paths

Definition: Let f be a flow in an s-t network N. An ***f*-augmenting path** Q is an s-t quasi path in N such that the flow on each forward arc can be increased, and the flow on each backward arc can be decreased.

Thus, for each arc e on an f-augmenting path Q,

$f(e) < cap(e)$, if e is a forward arc

$f(e) > 0$ if e is a backward arc.

Notation For each arc e on a given f-augmenting path Q, let Δ_e be the quantity given by

$$\Delta_e = \begin{cases} cap(e) - f(e), & \text{if } e \text{ is a forward arc} \\ f(e), & \text{if } e \text{ is a backward arc} \end{cases}$$

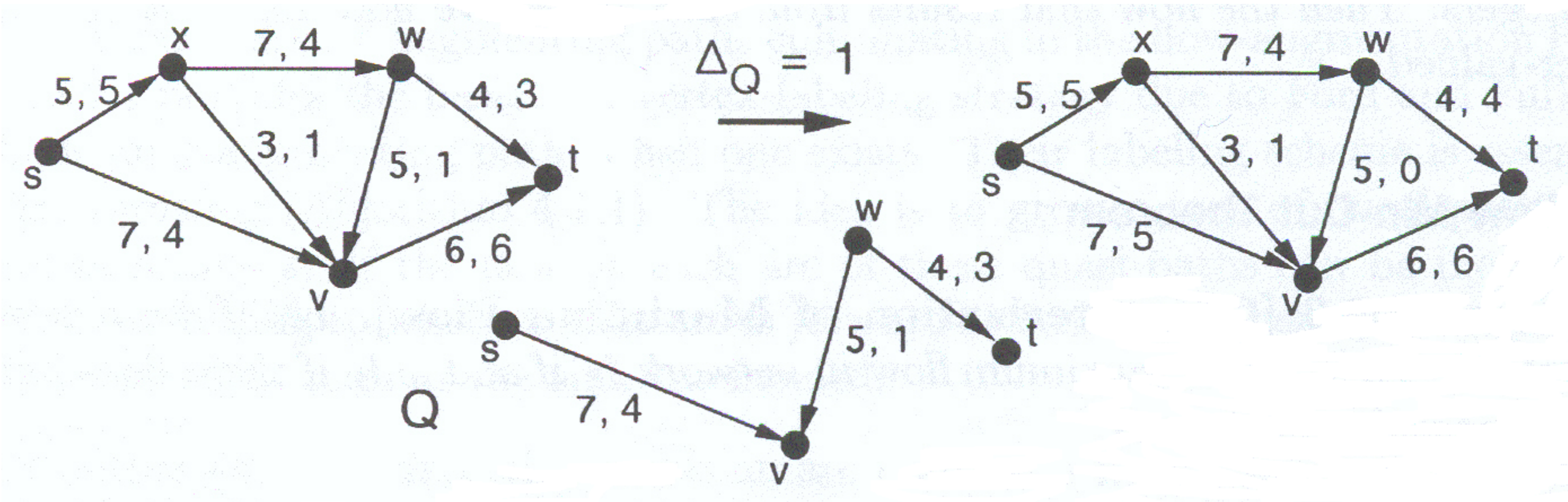
Terminology The quantity Δ_e is called the **slack on arc** e. Its value on a forward arc is the largest possible increase in the flow, and on a backward arc, the largest possible decrease in the flow, disregarding conservation of flow.

f-Augmenting Paths

<u>Remark</u> Conservation of flow requires that the change in the flow on the arcs of an augmenting flow path be of equal magnitude.
Thus, the maximum allowable change in the flow on an arc of quasipath Q is Δ_Q, where

$$\Delta_Q = \min_{e \in Q} \{\Delta_e\}$$

<u>Example</u> For the example network shown below, the current flow *f* has value 9, and the quasi-path Q = $\langle s,v,w,t \rangle$ is an *f*-augmenting path with Δ_Q = 1.



flow augmentation

Proposition 12.2.1 (Flow Augmentation) Let *f* be a feasible flow in a network *N*, and let Q be an *f*-augmenting path with minimum slack Δ_Q on its arcs.
Then the augmented flow *f'* given by

$$f'(e) = \begin{cases} f(e) + \Delta_Q, & \text{if } e \text{ is a forward arc of Q} \\ f(e) - \Delta_Q, & \text{if } e \text{ is a backward arc of Q} \\ f(e) & \text{otherwise} \end{cases}$$

is also a feasible flow in network *N* and $val(f') = val(f) + \Delta_Q$.

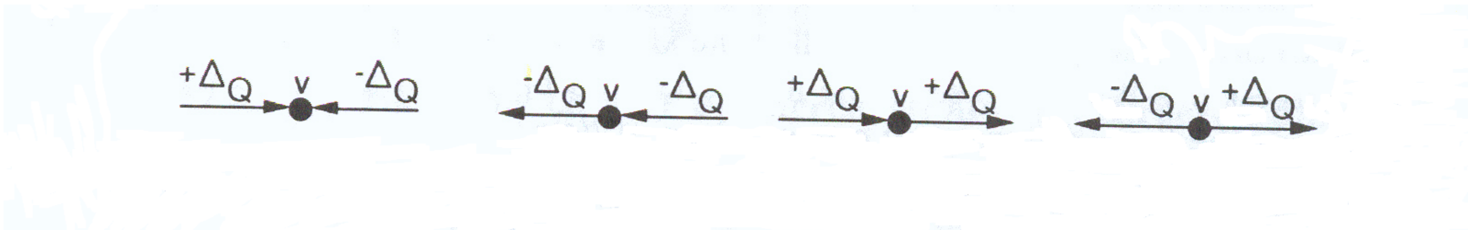
Proof. **Will not be presented in lecture.**

Clearly, $0 \le f'(e) \le cap(e)$, by the definition of Δ_Q.

The only vertices through which the net flow may have changed are those vertices on the augmenting path Q. Thus, to verify that *f'* satisfies conservation of flow, only the internal vertices of Q need to be checked.

f-Augmenting Paths

For a given vertex *v* on augmenting path Q, the two arcs of Q that are incident on *v* are configured in one of four ways, as shown below. In each case, the net flow into or out of vertex *v* does not change, thereby preserving the conservation-of-flow property.



It remains to be shown that the flow has increased by Δ_Q.

The only arc incident on the source *s* whose flow has changed is the first arc e_1 of augmenting path Q.

If e_1 is a forward arc, then $f'(e_1) = f(e_1) + \Delta_Q$, and

if e_1 is a backward arc, then $f'(e_1) = f(e_1) - \Delta_Q$. In either case,

$$val(f') = \sum_{e \in Out(s)} f'(e) - \sum_{e \in In(s)} f'(e) = \Delta_Q + val(f) \quad \square$$

Max-Flow Min-Cut

Theorem 12.2.3 [Characterization of Maximum Flow]

Let *f* be a flow in a network *N.*

Then *f* is a maximum flow in network *N* if and only if there does not exist an *f*-augmenting path in *N.*

Proof: **Will not be presented in lecture.**

Necessity ($\Rightarrow$) Suppose that *f* is a maximum flow in network *N*.

Then by Proposition 12.2.1, there is no *f*-augmenting path.

Proposition 12.2.1 (Flow Augmentation) Let *f* be a flow in a network *N*, and let Q be an *f*-augmenting path with minimum slack ΔQ on its arcs. Then the augmented flow *f'* given by

$$f'(e) = \begin{cases} f(e) + \Delta_Q, & \text{if } e \text{ is a forward arc of Q} \\ f(e) - \Delta_Q, & \text{if } e \text{ is a backward arc of Q} \\ f(e) & \text{otherwise} \end{cases}$$

is a feasible flow in network *N* and ***val(f') = val(f) + ΔQ**.*

$\rightarrow$ assuming an *f*-augmenting path existed, we could construct a flow *f'* with *val(f') > val(f)* contradicting the maximality of *f.*

Max-Flow Min-Cut

Sufficiency ($\Leftarrow$) Suppose that there does not exist an *f*-augmenting path in network *N.*

Consider the collection of all quasi-paths in network *N* that begin with source *s*, and have the following property: each forward arc on the quasi-path has positive slack, and each backward arc on the quasi-path has positive flow.

Let V_s be the union of the vertex-sets of these quasi-paths.

Since there is no *f*-augmenting path, it follows that sink $t \notin V_s$.

Let $V_t = V_N - V_s$.

Then $\langle V_s, V_t \rangle$ is an *s*-*t* cut of network *N*. Moreover, by definition of the sets V_s and V_t ,

$$f(e) = \begin{cases} cap(e) & \text{if } e \in \langle V_s, V_t \rangle \\ 0 & \text{if } e \in \langle V_t, V_s \rangle \end{cases}$$

(if the flow along these edges *e* were not cap(e) or 0, these edges would belong to V_s!)

Hence, *f* is a maximum flow, by Corollary 12.1.8. □

Max-Flow Min-Cut

Theorem 12.2.4 [Max-Flow Min-Cut] For a given network, the value of a maximum flow is equal to the capacity of a minimum cut.

Proof: The *s*-*t* cut $\langle V_s, V_t \rangle$ that we just constructed in the proof of Theorem 12.2.3 (direction $\Leftarrow$) has capacity equal to the maximum flow. □

The outline of an algorithm for maximizing the flow in a network emerges from Proposition 12.2.1 and Theorem 12.2.3.

Algorithm 12.2.1: Outline for Maximum Flow

Input: an s-t network N.
Output: a maximum flow f^* in network N.

```
[Initialization]
For each arc e in network N
    f*(e) := 0
[Flow Augmentation]
While there exists an f*-augmenting path in network N
    Find an f*-augmenting path Q.
    Let Δ_Q = min_{e∈Q} {Δ_e}.
    For each arc e of augmenting path Q
        If e is a forward arc
            f*(e) := f*(e) + Δ_Q
        Else (e is a backward arc)
            f*(e) := f*(e) − Δ_Q
Return flow f*.
```

Finding an *f*-Augmenting Path

The discussion of *f*-augmenting paths culminating in the flow-augmenting Proposition 12.2.1 provides the basis of a vertex-labeling strategy due to Ford and Fulkerson that finds an *f*-augmenting path, when one exists.

Their labelling scheme is essentially **basic tree-growing**.

The idea is to grow a tree of quasi-paths, each starting at source *s*.

If the flow on each arc of these quasi-paths can be increased or decreased, according to whether that arc is **forward** or **backward**, then an *f*-augmenting path is obtained as soon as the sink *t* is labelled.

Finding an *f*-Augmenting Path

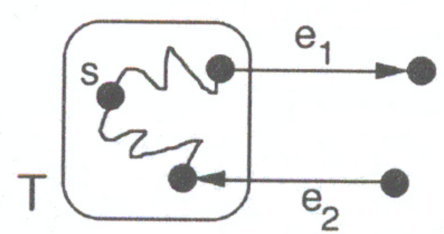
A **frontier arc** is an arc *e* directed from a **labeled** endpoint *v* to an **unlabeled** endpoint *w*.

For constructing an *f*-augmenting path, the frontier path *e* is allowed to be backward (directed from vertex *w* to vertex *v*), and it can be added to the tree as long as it has slack $\Delta_e > 0$.

Finding an *f*-Augmenting Path

<u>Terminology</u>: At any stage during tree-growing for constructing an *f*-augmenting path, let *e* be a frontier arc of tree *T*, with endpoints *v* and *w.*
The arc *e* is said to be **usable** if, for the current flow *f*, either

e is directed from vertex *v* to vertex *w* and $f(e) < cap(e)$, or
e is directed from vertex *w* to vertex *v* and $f(e) > 0$.



Frontier arcs e_1 and e_2 are usable if $f(e_1) < cap(e_1)$ and $f(e_2) > 0$

<u>Remark</u> From this vertex-labeling scheme, any of the existing *f*-augmenting paths could result. But the efficiency of Algorithm 12.2.1 is based on being able to find „good“ augmenting paths.
If the arc capacities are irrational numbers, then an algorithm using the Ford&Fulkerson labeling scheme might not terminate (strictly speaking, it would not be an algorithm).

Finding an *f*-Augmenting Path

Even when flows and capacities are restricted to be integers,
problems concerning efficiency still exist.

E.g., if each flow augmentation were to increase the flow by only one unit,
then the number of augmentations required for maximization would equal
the capacity of a minimum cut.

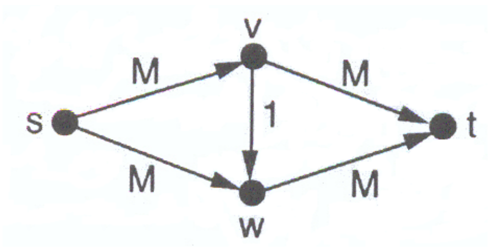
Such an algorithm would depend on the size of the
arc capacities instead of on the size of the network.

Finding an *f*-Augmenting Path

Example: For the network shown below, the arc from vertex v to vertex w has flow capacity 1, while the other arcs have capacity M, which could be made arbitrarily large.

If the choice of the augmenting flow path at each iteration were to alternate between the directed path $\langle s,v,w,t \rangle$ and the quasi path $\langle s,w,v,t \rangle$, then the flow would increase by only one unit at each iteration.

Thus, it could take as many as $2M$ iterations to obtain the maximum flow.



Finding an *f*-Augmenting Path

Edmonds and Karp avoid these problems with this algorithm.

It uses **breadth-first search** to find an *f*-augmenting path with the smallest number of arcs.

Algorithm 12.2.2: Finding an Augmenting Path

Input: a flow f in an s-t network N.
Output: an f-augmenting path Q or a minimum s-t cut with capacity $val(f)$.

Initialize vertex set $V_s := \{s\}$.
Write label 0 on vertex s.
Initialize label counter $i := 1$
While vertex set V_s does not contain sink t
  If there are usable arcs
    Let e be a usable arc whose labeled endpoint v has the smallest possible label.
    Let w be the unlabeled endpoint of arc e.
    Set $backpoint(w) := v$.
    Write label i on vertex w.
    $V_s := V_s \cup \{w\}$
    $i := i + 1$
  Else
    Return s-t cut $\langle V_s, V_N - V_s \rangle$.
Reconstruct the f-augmenting path Q by following backpointers, starting from sink t.
Return f-augmenting path Q.

FFEK algorithm: Ford, Fulkerson, Edmonds, and Karp

Algorithm 12.2.3 combines Algorithms 12.2.1 and 12.2.2

Algorithm 12.2.3: FFEK - Maximum Flow

Input: an s-t network N.
Output: a maximum flow f^* in network N.

```
[Initialization]
For each arc e in network N
    f*(e) := 0
[Flow Augmentation]
Repeat
    Apply Algorithm 12.2.2 to find an f*-augmenting path Q.
    Let Δ_Q = min_{e∈Q} {Δ_e}.
    For each arc e of augmenting path Q
        If e is a forward arc
            f*(e) := f*(e) + Δ_Q
        Else (e is a backward arc)
            f*(e) := f*(e) − Δ_Q
Until an f*-augmenting path cannot be found in network N.
Return flow f*.
```

FFEK algorithm: Ford, Fulkerson, Edmonds, and Karp

<u>Example</u>: the figures illustrate algorithm 12.2.3.

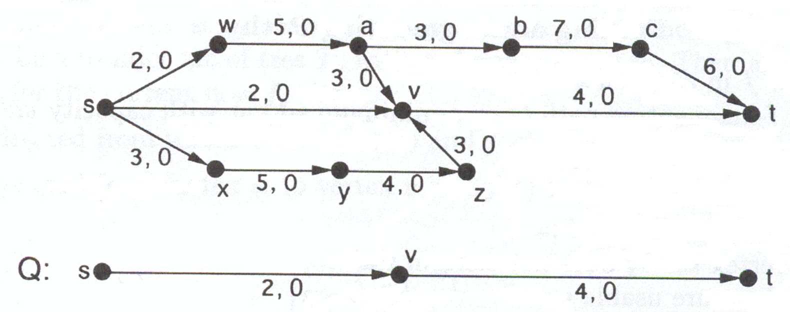


Figure 12.2.8 Iteration 0: $val(f) = 0$; augmenting path Q has $\Delta_Q = 2$.

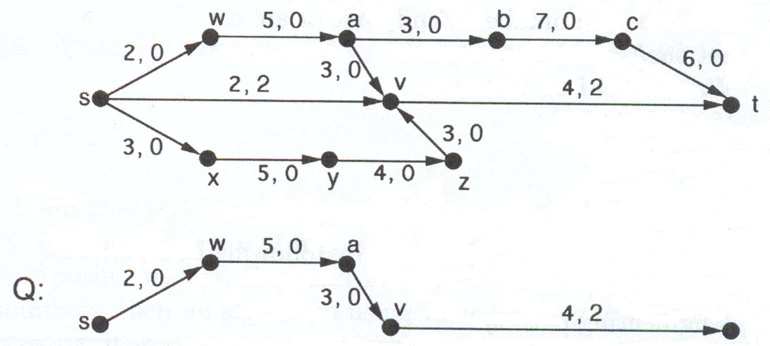


Figure 12.2.9 Iteration 1: $val(f) = 2$; augmenting path Q has $\Delta_Q = 2$.

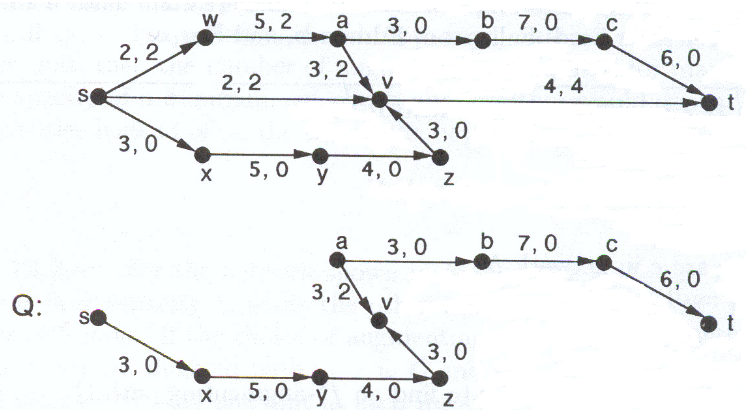


Figure 12.2.10 Iteration 2: $val(f) = 4$; augmenting path Q has $\Delta_Q = 2$.

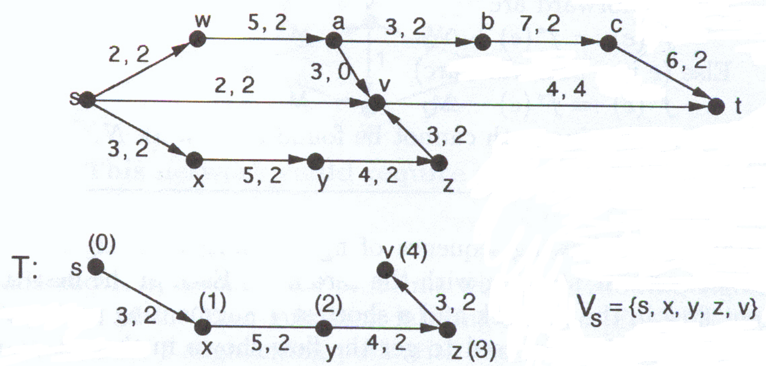


Figure 12.2.11 Final iteration: $val(f) = 6 = cap\langle\{s, x, y, z, v\}, \{w, a, b, c, t\}\rangle$.

<{s, x, y, z, v}, {w, a, b, c, t}> is the *s-t* cut with capacity equal to the current flow, establishing optimality.

FFEK algorithm: Ford, Fulkerson, Edmonds, and Karp

At the end of the final iteration, the two arcs from source *s* to vertex *w* and the arc directed from vertex *v* to sink *t* form the minimum cut $\langle \{s,x,y,z,v\}, \{w,a,b,c,t\} \rangle$. Neither of them is usable, i.e. the flow(e) = cap(e).

This illustrates the *s-t* cut that was constructed in the proof of theorem 12.2.3.

From Graph connectivity to Metabolic networks

We will now use the theory of network flows to give constructive proofs of Menger's theorem.

These proofs lead directly to algorithms for determining the edge-connectivity and vertex-connectivity of a graph.

The strategy to prove Menger's theorems is based on properties of certain **networks** whose arcs all have **unit capacity**.

These **0-1 networks** are constructed from the original graph.

Determining the connectivity of a graph

Lemma 12.3.1. Let *N* be an *s-t* network such that

outdegree(s) > indegree(s),

indegree(t) > outdegree (t), and

outdegree(v) = indegree(v) for all other vertices *v.*

Then, there exists a directed *s-t* path in network *N.*

Proof. Let *W* be a longest directed trail (trail = walk without repeated edges; path = trail without repeated vertices) in network *N* that starts at source *s*, and let *z* be its terminal vertex.

If vertex *z* were not the sink *t,* then there would be an arc not in trail *W* that is directed from *z* (since *indegree(z) = outdegree(z)).*

But this would contradict the maximality of trail *W.*

Thus, *W* is a directed trail from source *s* to sink *t.*

If *W* has a repeated vertex, then a part of *W* determines a directed cycle, which can be deleted from *W* to obtain a shorter directed *s-t* trail.

This deletion step can be repeated until no repeated vertices remain, at which point, the resulting directed trail is an *s-t* path. □

Determining the connectivity of a graph

<u>Proposition 12.3.2</u>. Let N be an s-t network such that

$$\mathit{outdegree}(s) - \mathit{indegree}(s) = m = \mathit{indegree}(t) - \mathit{outdegree}\ (t),$$

and $\quad \mathit{outdegree}(v) = \mathit{indegree}(v)$ for all vertices $v \neq s,t$.

Then, there exist m disjoint directed s-t path in network N.

<u>Proof</u>. **Will not be presented in lecture.**

If $m = 1$, then there exists an open eulerian directed trail T from source s to sink t by Theorem 6.1.3.

Review: An eulerian trail in a graph is a trail that visits every edge of that graph exactly once**.**

Theorem 6.1.3. A connected digraph D has an open eulerian trail from vertex x to vertex y if and only if $\mathit{indegree}(x) + 1 = \mathit{outdegree}(x)$, $\mathit{indegree}(y) = \mathit{outdegree}(y) + 1$, and all vertices except x and y have equal indegree and outdegree.

Euler proved that a necessary condition for the existence of Eulerian circuits is that all vertices in the graph have an even degree.

Theorem 1.5.2. Every open x-y walk W is either an x-y path or can be reduced to an x-y path.

Therefore, trail T is either an s-t directed path or can be reduced to an s-t path.

Determining the connectivity of a graph

By way of induction, assume that the assertion is true for $m = k$, for some $k \geq 1$, and consider a network N for which the condition holds for $m = k + 1$.
There does exist at least one directed s-t path P by Lemma 12.3.1.

If the arcs of path P are deleted from network N, then the resulting network $N - P$ satisfies the condition of the proposition for $m = k$.

By the induction hypothesis, there exist k arc-disjoint directed s-t paths in network $N - P$. These k paths together with path P form a collection of $k + 1$ arc-disjoint directed s-t paths in network N. □

Basic properties of 0-1 networks

Definition A **0-1 network** is a capacitated network whose arc capacities are either 0 or 1.

Proposition 12.3.3. Let *N* be an *s-t* network such that *cap(e)* = 1 for every arc *e.* Then the value of a maximum flow in network *N* equals the maximum number of arc-disjoint directed *s-t* paths in *N.*

Proof: Let *f** be a maximum flow in network *N*, and let *r* be the maximum number of arc-disjoint directed *s-t* paths in *N.*
Consider the network *N** obtained by deleting from *N* all arcs *e* for which *f*(e)* = 0.
Then *f*(e)* = 1 for all arcs *e* in network *N*.*
It follows from the definition that for every vertex *v* in network *N*,*

$$\sum_{e \in Out(v)} f^*(e) = |Out(v)| = outdegree(v)$$

and

$$\sum_{e \in In(v)} f^*(e) = |In(v)| = indegree(v)$$

Basic properties of 0-1 networks

Thus by the definition of *val(f*)* and by the conservation-of-flow property,

$$outdegree(s) - indegree\ (s) = val(f^*) = indegree(t) - outdegree(t)$$

and $outdegree(v) = indegree(v)$, for all vertices $v \neq s,t$.

By Proposition 12.3.2., there are *val(f*)* arc-disjoint *s*-*t* paths in network *N**, and hence, also in *N*, which implies that $val(f^*) \leq r$.

To obtain the reverse inequality, let $\{P_1, P_2, \ldots, P_r\}$ be the largest collection of arc-disjoint directed *s*-*t* paths in *N*, and consider the function $f: E_N \to R^+$ defined by

$$f(e) = \begin{cases} 1, & \text{if some path } P_i \text{ uses arc } e \\ 0, & \text{otherwise} \end{cases}.$$

Then *f* is a feasible flow in network *N*, with $val(f) = r$.

It follows that $val(f^*) \geq r$. □

Separating Sets and Cuts

Review from § 5.3

Let s and t be distinct vertices in a graph G. An s-t **separating edge set** in G is a set of edges whose removal destroys all s-t paths in G.

Thus, an s-t separating edge set in G is an edge subset of E_G that contains at least one edge of every s-t path in G.

Definition: Let s and t be distinct vertices in a digraph D.
An s-t **separating arc set** in D is a set of arcs whose removal destroys all directed s-t paths in D.

Thus, an s-t separating arc set in D is an arc subset of E_D that contains at least one arc of every directed s-t path in digraph D.

Remark: For the degenerate case in which the original graph or digraph has no s-t paths, the empty set is regarded as an s-t separating set.

Separating Sets and Cuts

Proposition 12.3.4 Let N be an s-t network such that $cap(e)$ = 1 for every arc e. Then the capacity of a minimum s-t cut in network N equals the minimum number of arcs in an s-t separating arc set in N.

Proof: Let $K^* = \langle V_s, V_t \rangle$ be a minimum s-t cut in network N, and let q be the minimum number of arcs in an s-t separating arc set in N.
Since K^* is an s-t cut, it is also an s-t separating arc set. Thus $cap(K^*) \geq q$.

To obtain the reverse inequality, let S be an s-t separating arc set in network N containing q arcs, and let R be the set of all vertices in N that are reachable from source s by a directed path that contains no arc from set S.

Then, by the definitions of arc set S and vertex set R, $t \notin R$, which means that $\langle R, V_N - R \rangle$ is an s-t cut.

Moreover, $\langle R, V_N - R \rangle \subseteq S$. Therefore

Separating Sets and Cuts

$$
\begin{aligned}
cap(K^*) &\leq cap\langle R, V_N - R\rangle && \text{since } K^* \text{ is a minimum } s-t \text{ cut} \\
&= |\langle R, V_N - R\rangle| && \text{since all capacities are 1} \\
&\leq |S| && \text{since } \langle R, V_N - R\rangle \subseteq S \\
&= q
\end{aligned}
$$

which completes the proof. □

Arc and Edge Versions of Menger's Theorem Revisited

Theorem 12.3.5 [Arc form of Menger's theorem]

Let s and t be distinct vertices in a digraph D. Then the maximum number of arc-disjoint directed s-t paths in D is equal to the minimum number of arcs in an s-t separating set of D.

Proof: Let N be the s-t network obtained by assigning a unit capacity to each arc of digraph D. Then the result follows from Propositions 12.3.3. and 12.3.4., together with the max-flow min-cut theorem. □

Theorem 12.2.4 [Max-Flow Min-Cut] For a given network, the value of a maximum flow is equal to the capacity of a minimum cut.

Proposition 12.3.3. Let N be an s-t network such that $cap(e) = 1$ for every arc e. Then the value of a maximum flow in network N equals the maximum number of arc-disjoint directed s-t paths in N.

Proposition 12.3.4 Let N be an s-t network such that $cap(e) = 1$ for every arc e. Then the capacity of a minimum s-t cut in network N equals the minimum number of arcs in an s-t separating arc set in N.