Graph Theory and Optimization
Flow: Ford-Fulkerson Algorithm
Max Flow- Min Cut duality

Nicolas Nisse

Université Côte d'Azur, Inria, CNRS, I3S, France

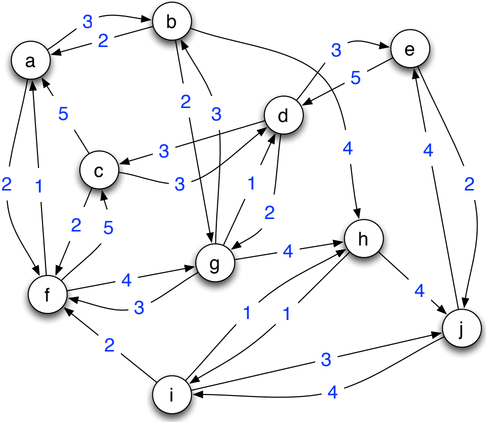
October 2018

Outline

Transportation problem: Modeling

Directed weighted graph: $D = (V, A)$, A : set of arcs, $(x, y) \in A$ ordered pair

arrows

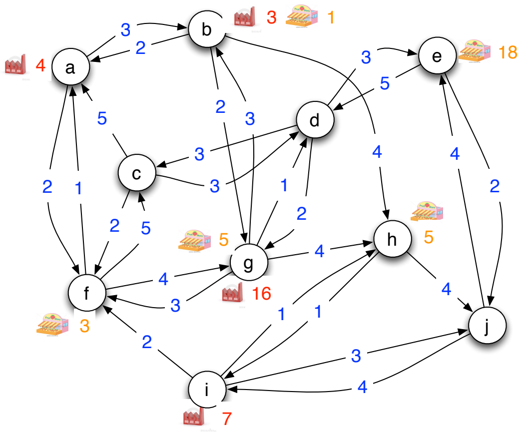


Transportation problem: Modeling

Directed weighted graph: $D = (V, A)$

On vertices: production: $p_{max} : V \to \mathbb{R}^+$; consumption: $cons_{max} : V \to \mathbb{R}^+$

On arcs: capacity: $c : A \to \mathbb{R}^+$

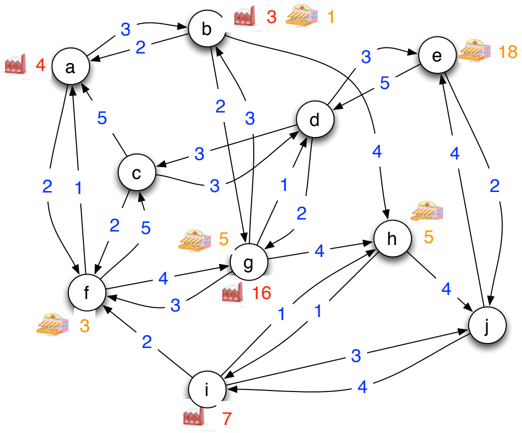


Transportation problem: Modeling

What is the main amount of goods that can be exchanged?

Actual production: $p : V \rightarrow \mathbb{R}^+$ and actual consumption: $cons : V \rightarrow \mathbb{R}^+$

flow: $f : A \rightarrow \mathbb{R}^+$: satisfies capacity and flow conservation

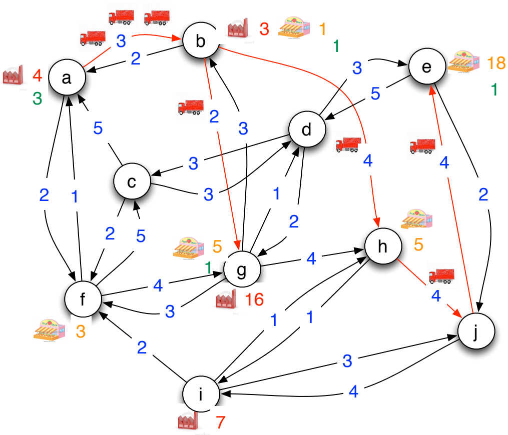


Transportation problem: Modeling

flow: $f : A \rightarrow \mathbb{R}^+$: feasibility: $\forall v \in V$, $p(v) \leq p_{max}(v)$, $cons(v) \leq cons_{max}(v)$

capacity constraint: $\forall a \in A, f(a) \leq c(a)$.

flow conservation: $\forall v \in V$, $p(v) + \sum_{w \in N^-(v)} f(wv) = c(v) + \sum_{w \in N^+(v)} f(vw)$

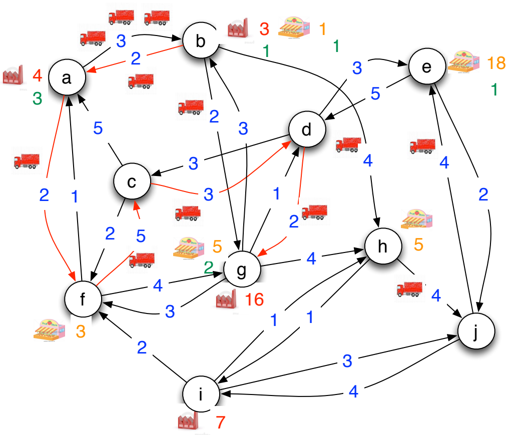


Transportation problem: Modeling

flow: $f : A \to \mathbb{R}^+$: feasibility: $\forall v \in V$, $p(v) \leq p_{max}(v)$, $cons(v) \leq cons_{max}(v)$

capacity constraint: $\forall a \in A, f(a) \leq c(a)$.

flow conservation: $\forall v \in V$, $p(v) + \sum_{w \in N^-(v)} f(wv) = c(v) + \sum_{w \in N^+(v)} f(vw)$

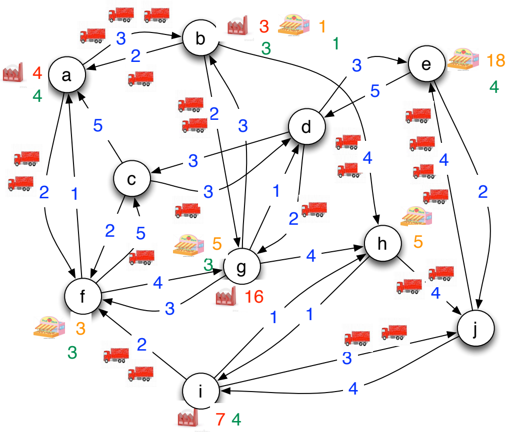


Transportation problem: Modeling

flow: $f : A \rightarrow \mathbb{R}^+$: feasibility: $\forall v \in V$, $p(v) \leq p_{max}(v)$, $cons(v) \leq cons_{max}(v)$

capacity constraint: $\forall a \in A, f(a) \leq c(a)$.

flow conservation: $\forall v \in V$, $p(v) + \sum_{w \in N^-(v)} f(wv) = c(v) + \sum_{w \in N^+(v)} f(vw)$

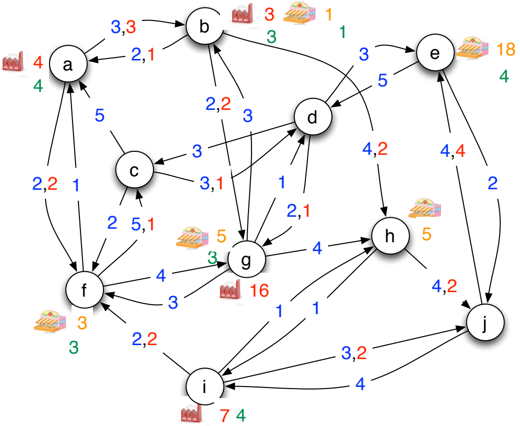


Transportation problem: Modeling

flow: $f : A \rightarrow \mathbb{R}^+$: feasibility: $\forall v \in V$, $p(v) \leq p_{max}(v)$, $cons(v) \leq cons_{max}(v)$

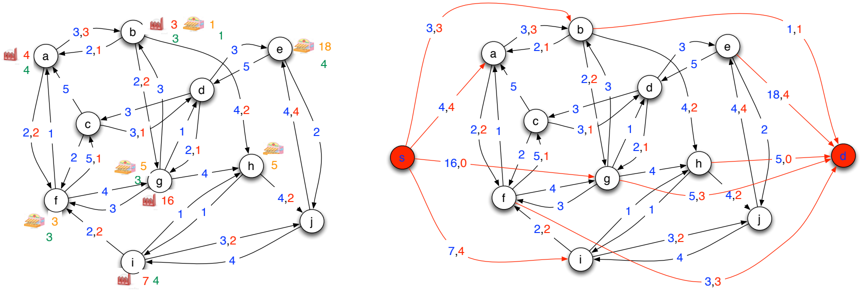
capacity constraint: $\forall a \in A, f(a) \leq c(a)$.

flow conservation: $\forall v \in V$, $p(v) + \sum_{w \in N^-(v)} f(wv) = c(v) + \sum_{w \in N^+(v)} f(vw)$



Transportation problem: Modeling

Simplification: one single source and one single destination



One source s: $\forall v \in V$ add one arc (s, v) with capacity $p_{max}(v)$

One destination d: $\forall v \in V$ add one arc (v, d) with capacity $cons_{max}(v)$

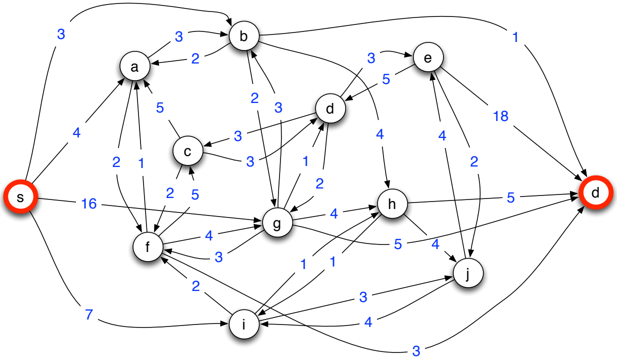
Flows are "equivalent" in both networks

Outline

Elementary Flow Network

Directed graph: $D=(V,A)$, **source** $s \in V$, **destination** $d \in V$, capacity $c : A \rightarrow \mathbb{R}^+$.

$N^-(s)=\emptyset$ and $N^+(d)=\emptyset$



flow $f : A \rightarrow \mathbb{R}^+$ such that : capacity constraint: $\forall a \in A$, $f(a) \leq c(a)$

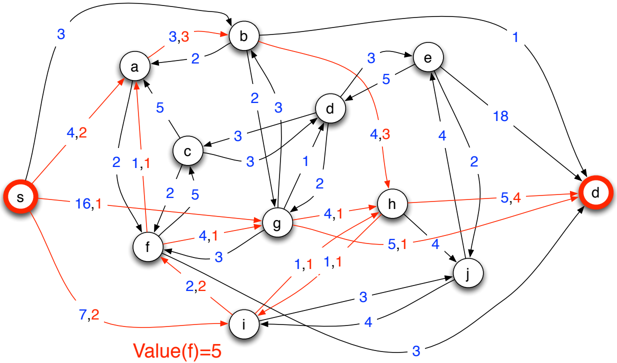
conservation constraint: $\forall v \in V \setminus \{s,d\}$, $\displaystyle\sum_{w \in N^-(v)} f(wv) = \sum_{w \in N^+(v)} f(vw)$

value of flow: $v(f) = \displaystyle\sum_{w \in N^+(s)} f(sw)$.

Elementary Flow Network

Directed graph: $D=(V,A)$, **source** $s \in V$, **destination** $d \in V$, capacity $c : A \to \mathbb{R}^+$.

$N^-(s)=\emptyset$ and $N^+(d)=\emptyset$



Value(f)=5

flow $f : A \to \mathbb{R}^+$ such that : capacity constraint: $\forall a \in A$, $f(a) \leq c(a)$

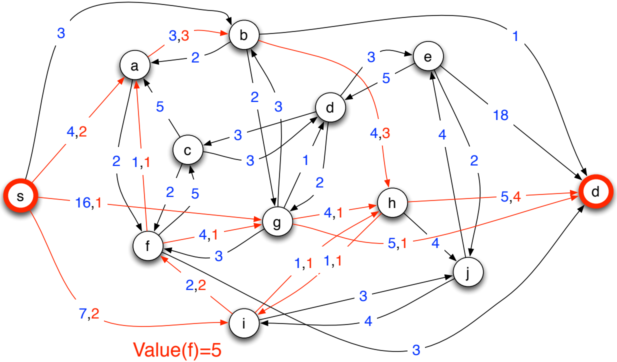
conservation constraint: $\forall v \in V \setminus \{s,d\}$, $\sum_{w \in N^-(v)} f(wv) = \sum_{w \in N^+(v)} f(vw)$

value of flow: $v(f) = \sum_{w \in N^+(s)} f(sw)$.

Elementary Flow Network

Directed graph: $D=(V,A)$, **source** $s \in V$, **destination** $d \in V$, capacity $c : A \rightarrow \mathbb{R}^+$.

$N^-(s)=\emptyset$ and $N^+(d)=\emptyset$



Value(f)=5

flow $f : A \rightarrow \mathbb{R}^+$ such that : capacity constraint: $\forall a \in A$, $f(a) \leq c(a)$

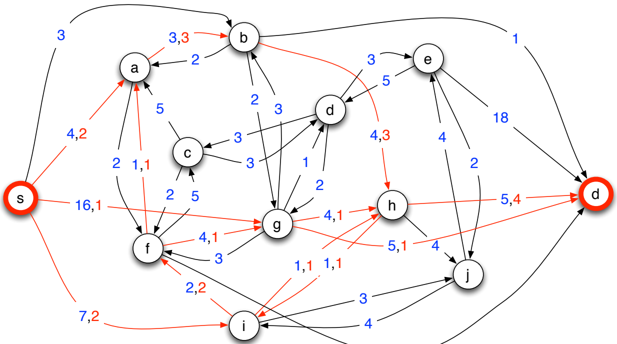
conservation constraint: $\forall v \in V \setminus \{s,d\}$, $\sum_{w \in N^-(v)} f(wv) = \sum_{w \in N^+(v)} f(vw)$

value of flow: $v(f) = \sum_{w \in N^+(s)} f(sw)$.

Exercise: $v(f) = \sum_{w \in N^-(d)} f(wd)$

Elementary Flow Network: Max Flow

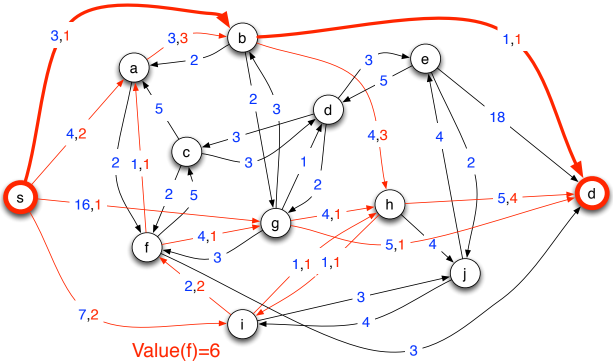
How to compute a flow with maximum value?



Value(f)=5

Elementary Flow Network: Max Flow

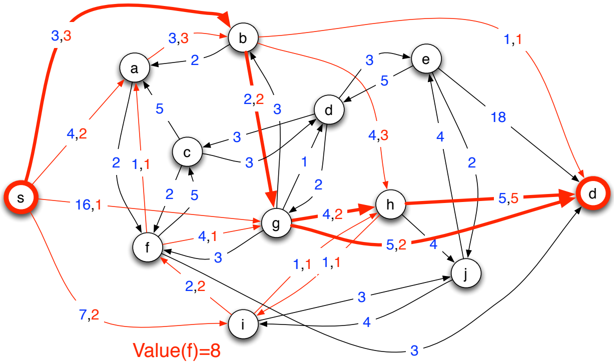
How to compute a flow with maximum value?



Possible to “push" flow along available path

Elementary Flow Network: Max Flow

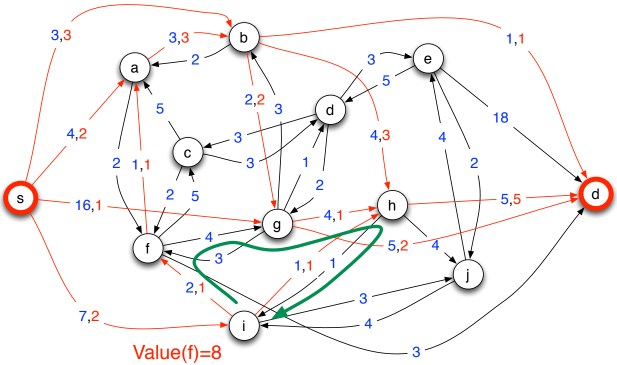
How to compute a flow with maximum value?



Possible to “push" flow along available path

Elementary Flow Network: Max Flow

How to compute a flow with maximum value?

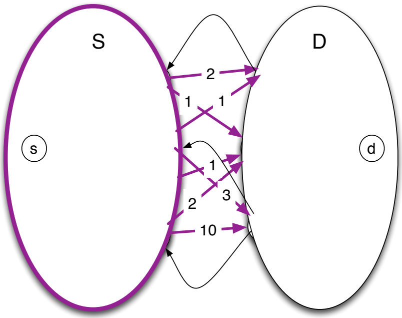


Possible to “push" flow along available path
May be useful to “remove useless flow"

Outline

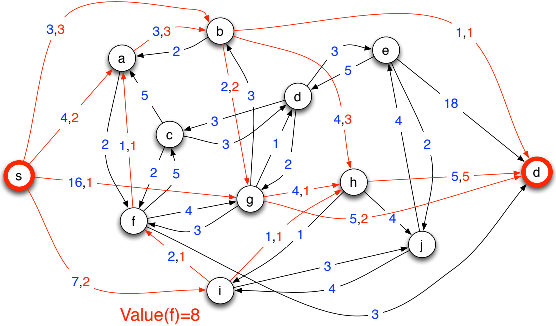
Max Flow, upper bound: Min cut

a cut gives an upper bound on the value of your flow



Max Flow, upper bound: Min cut

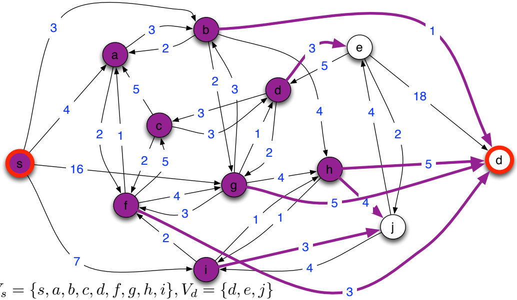
Is this flow maximum?: a cut gives an upper bound on the value of your flow



Value(f)=8

Max Flow, upper bound: Min cut

Is this flow maximum?: a cut gives an upper bound on the value of your flow



$V_s = \{s, a, b, c, d, f, g, h, i\}, V_d = \{d, e, j\}$

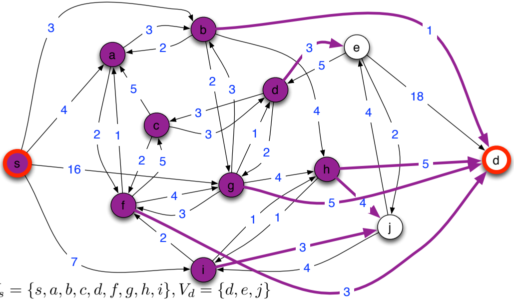
$\delta(V_s, V_d) = 24$

s,*d*-Cut: partition (V_s, V_d) of V with $s \in V_s$ and $d \in V_d$.

Capacity of a *s*,*d*-cut: $\delta(V_s, V_d) = \sum_{u \in V_s, v \in V_d} c(uv)$.

Max Flow, upper bound: Min cut

Is this flow maximum?: a cut gives an upper bound on the value of your flow



$V_s = \{s,a,b,c,d,f,g,h,i\}, V_d = \{d,e,j\}$

$\delta(V_s, V_d) = 24$

s,*d*-Cut: partition (V_s, V_d) of V with $s \in V_s$ and $d \in V_d$.

Capacity of a *s*,*d*-cut: $\delta(V_s, V_d) = \sum_{u \in V_s, v \in V_d} c(uv)$.

Theorem: for any network flow $D = (V, A)$, $s, d \in V$ and $c : A \rightarrow \mathbb{R}^+$

For any flow $f : A \rightarrow \mathbb{R}^+$ and *s*,*d*-cut (V_s, V_d), $v(f) \leq \delta(V_s, V_d)$

Max Flow, upper bound: Min cut

Theorem: for any network flow $D=(V,A)$, $s,d \in V$ and $c : A \rightarrow \mathbb{R}^+$

For any flow $f : A \rightarrow \mathbb{R}^+$ and s,d-cut (V_s, V_d), $v(f) \leq \delta(V_s, V_d)$

Proof:

conservation constraint: $\forall v \in V \setminus \{s,d\}$, $\sum_{w \in N^-(v)} f(wv) = \sum_{w \in N^+(v)} f(vw)$.

sum over all vertices in $V_s \setminus \{s\}$:

$$0 = \sum_{v \in V_s \setminus \{s\}} \Big(\sum_{w \in N^-(v)} f(wv) - \sum_{w \in N^+(v)} f(vw) \Big) =$$

$$\sum_{w \in N^+(s)} f(sw) + \sum_{v \in V_s \setminus \{s\}; w \in V_d} f(wv) - \sum_{v \in V_s; w \in V_d} f(vw)$$

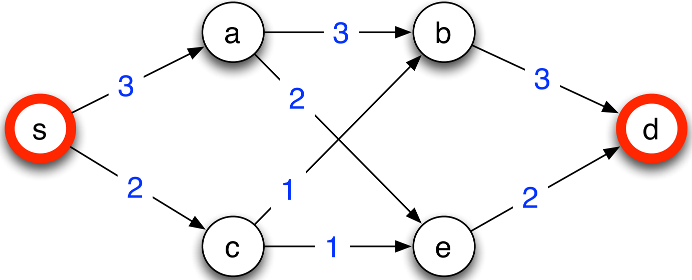
So

$$v(f) = \sum_{v \in V_s; w \in V_d} f(vw) - \sum_{v \in V_s \setminus \{s\}; w \in V_d} f(wv) \leq \sum_{v \in V_s; w \in V_d} f(vw) \leq \sum_{v \in V_s; w \in V_d} c(vw) = \delta(V_s, V_d)$$

Corollary: Max flow $\leq$ Min Cut

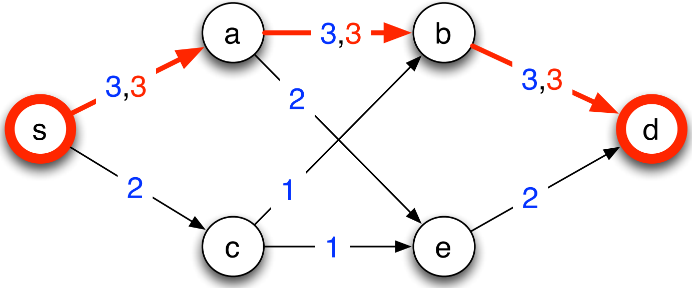
Outline

Algorithm for Max Flow: Intuition



Let us compute a max flow from s to d.

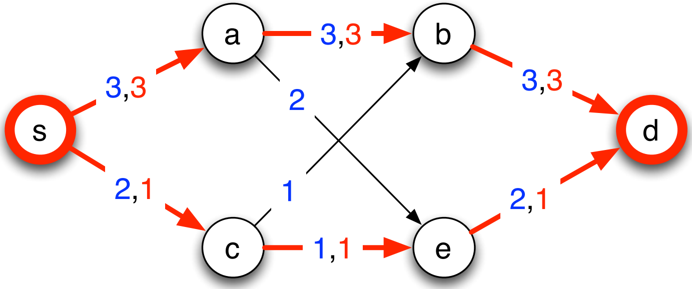
Algorithm for Max Flow: Intuition



We can “push" 3 units of flow along an available path (s, a, b, d). $v(f) = 3$.

The only remaining available path is (s, c, e, d)

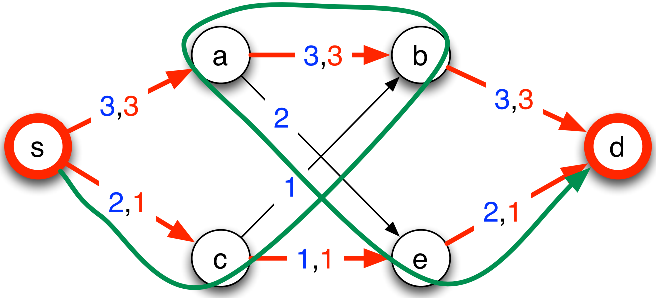
Algorithm for Max Flow: Intuition



We can “push" 1 units of flow along (s, c, e, d). $v(f) = 4$.

No path from s to d remains available, but...

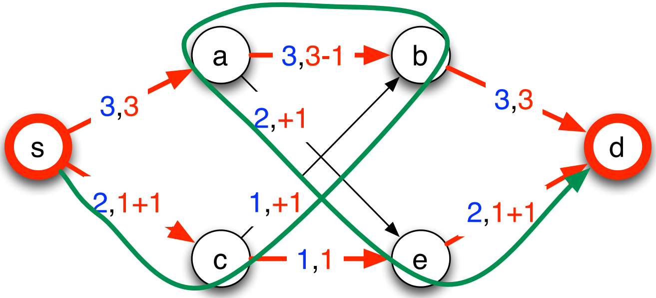
Algorithm for Max Flow: Intuition



We want to “push" some flow along the "path" (s, c, b, a, e, d)

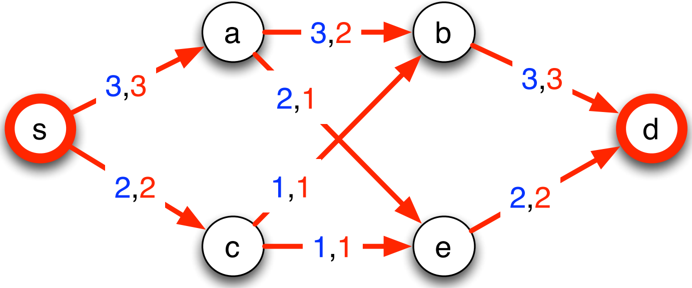
It is NOT a directed path (because $(b, a) \notin A$)

Algorithm for Max Flow: Intuition



Somehow, we "reverse" some flow along the arc (a, b)

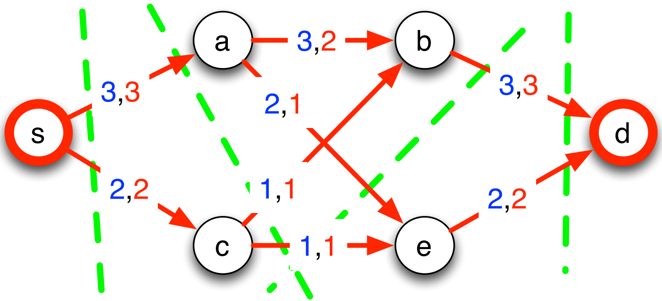
Algorithm for Max Flow: Intuition



So we got a flow with value $v(f) = 5$.

Exercise: Why is it optimal ?

Algorithm for Max Flow: Intuition

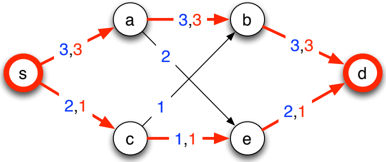


Recall that Max flow $\leq$ Min Cut

If there is a flow f and a cut (V_s, V_d) with $v(f) = \delta(V_s, V_d)$, then f is maximum and (V_s, V_d) is a minimum cut.

Ford-Fulkerson Algorithm 1^{st} example

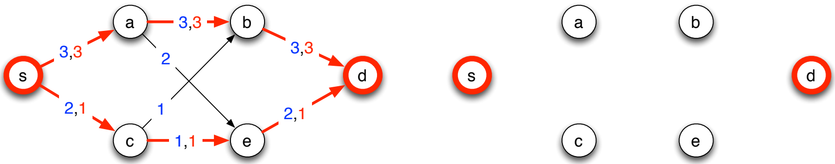
Problem here: there is no path where to push flow



1^{st} **Phase of FF-algorithm:** Compute an auxiliary graph where to find a path

Ford-Fulkerson Algorithm 1^{st} example

Problem here: there is no path where to push flow



1^{st} **Phase of FF-algorithm:** Compute an auxiliary graph where to find a path

For all $u, v \in V(G)$, create an arc with capacity $c_{aux}(uv) = c(uv) - f(uv) + f(vu)$

$f(uv)$ current flow from u to v, $f(vu)$ current flow from v to u

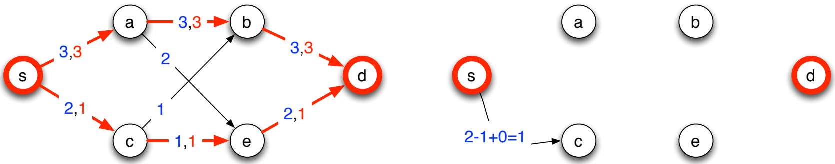
$c(uv) - f(uv)$ is the residual capacity

Remarks:

- $c_{aux}(uv)$ may be positive even if $(u, v) \notin A(G)$
- If $(u, v) \notin A(G)$ and $(v, u) \notin A(G)$, then $c_{aux}(uv) = c_{aux}(vu) = 0$

Ford-Fulkerson Algorithm 1^{st} example

Problem here: there is no path where to push flow



1^{st} **Phase of FF-algorithm:** Compute an auxiliary graph where to find a path

For all $u,v \in V(G)$, create an arc with capacity $c_{aux}(uv) = c(uv) - f(uv) + f(vu)$

$f(uv)$ current flow from u to v, $f(vu)$ current flow from v to u

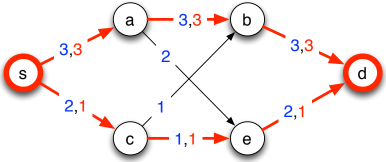
$c(uv) - f(uv)$ is the residual capacity

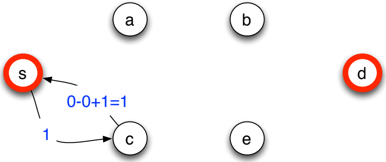
Remarks:

- $c_{aux}(uv)$ may be positive even if $(u,v) \notin A(G)$
- If $(u,v) \notin A(G)$ and $(v,u) \notin A(G)$, then $c_{aux}(uv) = c_{aux}(vu) = 0$

Ford-Fulkerson Algorithm 1^{st} example

Problem here: there is no path where to push flow





1^{st} **Phase of FF-algorithm:** Compute an auxiliary graph where to find a path

For all $u,v \in V(G)$, create an arc with capacity $c_{aux}(uv) = c(uv) - f(uv) + f(vu)$

$f(uv)$ current flow from u to v, $f(vu)$ current flow from v to u

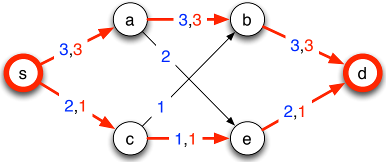
$c(uv) - f(uv)$ is the residual capacity

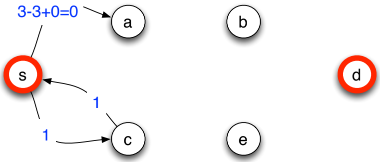
Remarks:

- $c_{aux}(uv)$ may be positive even if $(u,v) \notin A(G)$
- If $(u,v) \notin A(G)$ and $(v,u) \notin A(G)$, then $c_{aux}(uv) = c_{aux}(vu) = 0$

Ford-Fulkerson Algorithm 1^{st} example

Problem here: there is no path where to push flow





1^{st} **Phase of FF-algorithm:** Compute an auxiliary graph where to find a path

For all $u, v \in V(G)$, create an arc with capacity $c_{aux}(uv) = c(uv) - f(uv) + f(vu)$

$f(uv)$ current flow from u to v, $f(vu)$ current flow from v to u

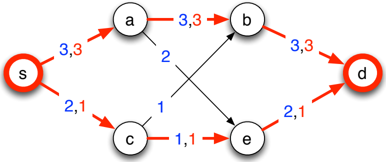
$c(uv) - f(uv)$ is the residual capacity

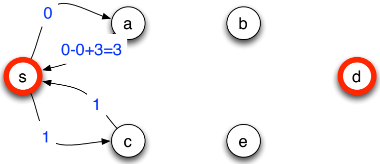
Remarks:

- $c_{aux}(uv)$ may be positive even if $(u, v) \notin A(G)$
- If $(u, v) \notin A(G)$ and $(v, u) \notin A(G)$, then $c_{aux}(uv) = c_{aux}(vu) = 0$

Ford-Fulkerson Algorithm 1^{st} example

Problem here: there is no path where to push flow





1^{st} **Phase of FF-algorithm:** Compute an auxiliary graph where to find a path

For all $u, v \in V(G)$, create an arc with capacity $c_{aux}(uv) = c(uv) - f(uv) + f(vu)$

$f(uv)$ current flow from u to v, $f(vu)$ current flow from v to u

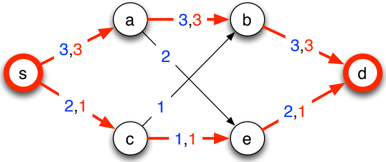
$c(uv) - f(uv)$ is the residual capacity

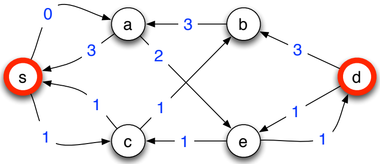
Remarks:

- $c_{aux}(uv)$ may be positive even if $(u, v) \notin A(G)$
- If $(u, v) \notin A(G)$ and $(v, u) \notin A(G)$, then $c_{aux}(uv) = c_{aux}(vu) = 0$

Ford-Fulkerson Algorithm 1^{st} example

Problem here: there is no path where to push flow





1^{st} **Phase of FF-algorithm:** Compute an auxiliary graph where to find a path

For all $u, v \in V(G)$, create an arc with capacity $c_{aux}(uv) = c(uv) - f(uv) + f(vu)$

$f(uv)$ current flow from u to v, $f(vu)$ current flow from v to u

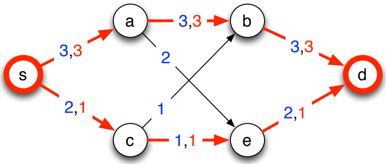
$c(uv) - f(uv)$ is the residual capacity

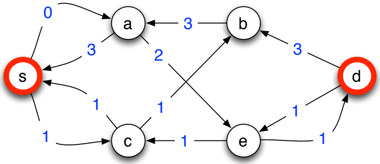
Remarks:

- $c_{aux}(uv)$ may be positive even if $(u, v) \notin A(G)$
- If $(u, v) \notin A(G)$ and $(v, u) \notin A(G)$, then $c_{aux}(uv) = c_{aux}(vu) = 0$

Ford-Fulkerson Algorithm 1^{st} example

Problem here: there is no path where to push flow

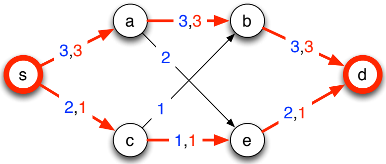


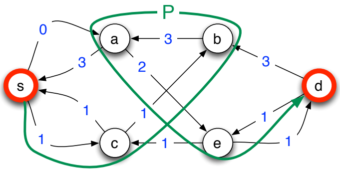


2^{nd} **Phase of FF-algorithm:** Look for a path P from s to d in auxiliary graph

Ford-Fulkerson Algorithm 1^{st} example

Problem here: there is no path where to push flow



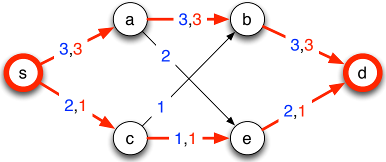


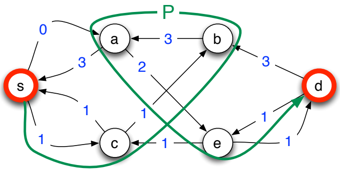
2^{nd} **Phase of FF-algorithm:** Look for a path P from s to d in auxiliary graph

Here $P = (s,c,b,a,e,d)$ and its minimum capacity is $\varepsilon = 1 > 0$

Ford-Fulkerson Algorithm 1^{st} example

Problem here: there is no path where to push flow





2^{nd} **Phase of FF-algorithm:** Look for a path P from s to d in auxiliary graph

Here $P = (s,c,b,a,e,d)$ and its minimum capacity is $\varepsilon = 1 > 0$

We will "push" ε units of flow along P in G

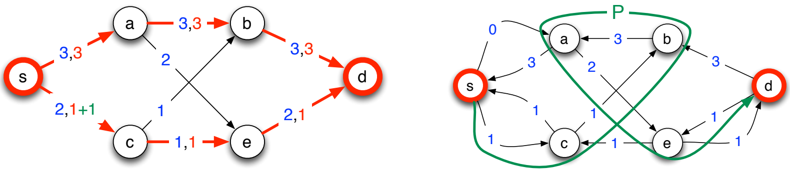
For all arcs (u,v) of P

- Add ε to the current flow of (u,v) if $f(uv) + \varepsilon \leq c(uv)$
- Otherwise add $c(uv) - f(uv)$ to the current flow of (u,v)

  note that $\varepsilon - (c(uv) - f(uv))$ are "lacking to be pushed"
- In the latter case, remove $\varepsilon - (c(uv) - f(uv))$ from the current flow from v to u.

Ford-Fulkerson Algorithm 1^{st} example

Problem here: there is no path where to push flow



2^{nd} **Phase of FF-algorithm:** Look for a path P from s to d in auxiliary graph

Here $P = (s,c,b,a,e,d)$ and its minimum capacity is $\varepsilon = 1 > 0$

We will “push" ε units of flow along P in G

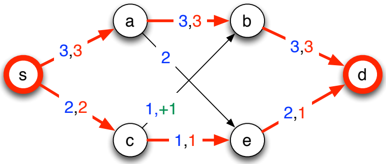
For all arcs (u,v) of P

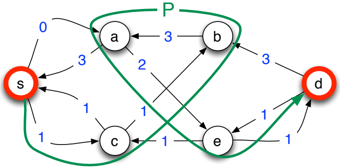
- Add ε to the current flow of (u,v) if $f(uv)+\varepsilon \leq c(uv)$
- Otherwise add $c(uv)-f(uv)$ to the current flow of (u,v)

  note that $\varepsilon-(c(uv)-f(uv))$ are “lacking to be pushed"
- In the latter case, remove $\varepsilon-(c(uv)-f(uv))$ from the current flow from v to u.

Ford-Fulkerson Algorithm 1^{st} example

Problem here: there is no path where to push flow





2^{nd} **Phase of FF-algorithm:** Look for a path P from s to d in auxiliary graph

Here $P = (s, c, b, a, e, d)$ and its minimum capacity is $\varepsilon = 1 > 0$

We will “push" ε units of flow along P in G

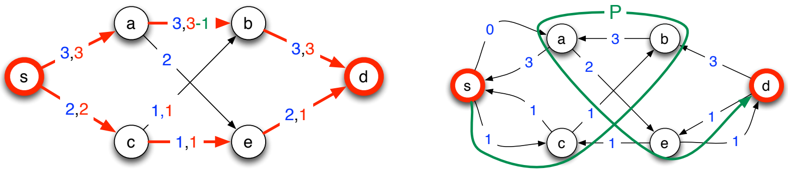
For all arcs (u, v) of P

- Add ε to the current flow of (u, v) if $f(uv) + \varepsilon \leq c(uv)$
- Otherwise add $c(uv) - f(uv)$ to the current flow of (u, v)

  note that $\varepsilon - (c(uv) - f(uv))$ are “lacking to be pushed"
- In the latter case, remove $\varepsilon - (c(uv) - f(uv))$ from the current flow from v to u.

Ford-Fulkerson Algorithm 1^{st} example

Problem here: there is no path where to push flow



2^{nd} **Phase of FF-algorithm:** Look for a path P from s to d in auxiliary graph

Here $P = (s, c, b, a, e, d)$ and its minimum capacity is $\varepsilon = 1 > 0$

We will “push" ε units of flow along P in G

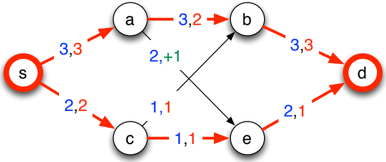
For all arcs (u, v) of P

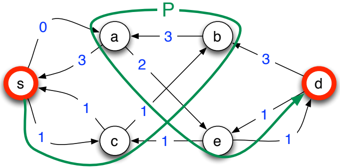
- Add ε to the current flow of (u, v) if $f(uv) + \varepsilon \leq c(uv)$
- Otherwise add $c(uv) - f(uv)$ to the current flow of (u, v)

  note that $\varepsilon - (c(uv) - f(uv))$ are “lacking to be pushed"
- In the latter case, remove $\varepsilon - (c(uv) - f(uv))$ from the current flow from v to u.

Ford-Fulkerson Algorithm 1^{st} example

Problem here: there is no path where to push flow





2^{nd} **Phase of FF-algorithm:** Look for a path P from s to d in auxiliary graph

Here $P = (s, c, b, a, e, d)$ and its minimum capacity is $\varepsilon = 1 > 0$

We will “push" ε units of flow along P in G

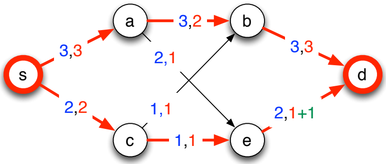
For all arcs (u, v) of P

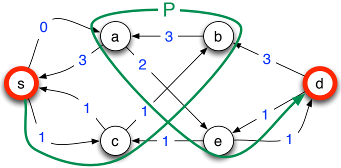
- Add ε to the current flow of (u, v) if $f(uv) + \varepsilon \leq c(uv)$
- Otherwise add $c(uv) - f(uv)$ to the current flow of (u, v)

  note that $\varepsilon - (c(uv) - f(uv))$ are “lacking to be pushed"
- In the latter case, remove $\varepsilon - (c(uv) - f(uv))$ from the current flow from v to u.

Ford-Fulkerson Algorithm 1^{st} example

Problem here: there is no path where to push flow





2^{nd} **Phase of FF-algorithm:** Look for a path P from s to d in auxiliary graph

Here $P = (s, c, b, a, e, d)$ and its minimum capacity is $\varepsilon = 1 > 0$

We will "push" ε units of flow along P in G

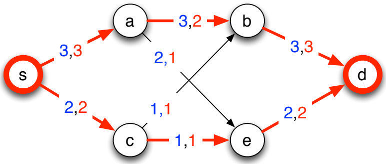
For all arcs (u, v) of P

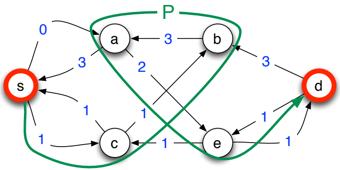
- Add ε to the current flow of (u, v) if $f(uv) + \varepsilon \leq c(uv)$
- Otherwise add $c(uv) - f(uv)$ to the current flow of (u, v)

  note that $\varepsilon - (c(uv) - f(uv))$ are "lacking to be pushed"
- In the latter case, remove $\varepsilon - (c(uv) - f(uv))$ from the current flow from v to u.

Ford-Fulkerson Algorithm 1^{st} example

Problem here: there is no path where to push flow





2^{nd} **Phase of FF-algorithm:** Look for a path P from s to d in auxiliary graph

Here $P = (s, c, b, a, e, d)$ and its minimum capacity is $\varepsilon = 1 > 0$

We will "push" ε units of flow along P in G

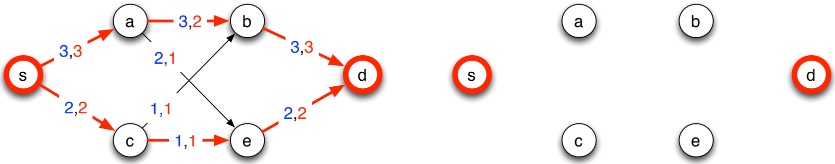
For all arcs (u, v) of P

- Add ε to the current flow of (u, v) if $f(uv) + \varepsilon \leq c(uv)$
- Otherwise add $c(uv) - f(uv)$ to the current flow of (u, v)

  note that $\varepsilon - (c(uv) - f(uv))$ are "lacking to be pushed"
- In the latter case, remove $\varepsilon - (c(uv) - f(uv))$ from the current flow from v to u.

Ford-Fulkerson Algorithm 1^{st} example

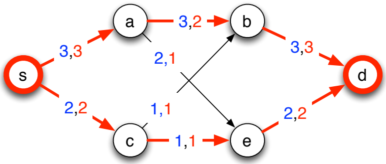
Problem here: there is no path where to push flow

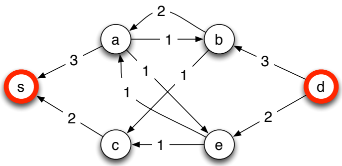


back to the 1^{st} Phase of FF-algorithm: Compute in auxiliary graph

Ford-Fulkerson Algorithm 1^{st} example

Problem here: there is no path where to push flow

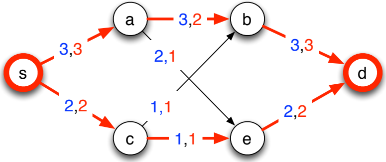


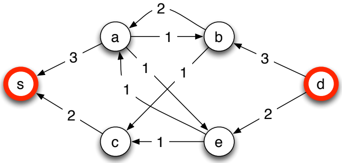


back to the 1^{st} **Phase of FF-algorithm:** Compute in auxiliary graph

Ford-Fulkerson Algorithm 1^{st} example

Problem here: there is no path where to push flow





back to the 1^{st} Phase of FF-algorithm: Compute in auxiliary graph

Here no path with > 0 capacity from s to d.
The set of nodes reachable from s (here, only s) defines a cut

Exercise: What is the capacity of this cut? Why the flow is maximum?

Ford-Fulkerson Algorithm

$D=(V,A)$, $c:A\rightarrow\mathbb{R}^+$ and $s,d\in V$

Let $f:A\rightarrow\mathbb{R}^+$ be a valid flow in D *Initially, f may be null*

1. Compute an auxiliary graph $D_{aux}=(V,A_{aux})$ and $c_{aux}:A_{aux}\rightarrow\mathbb{R}^+$

2. If there is a directed path P from s to d in D_{aux}
   (auxiliary capacity of arcs of P must be >0)
   - Let $\varepsilon>0$ be the minimum capacity c_{aux} of the arcs of P
   - "Push" ε units of flow along P in D
   - Go to 1 *(Note that the value of the flow has increased)*

   Else Return f *f is a maximum flow.*

Ford-Fulkerson Algorithm

$D = (V, A)$, $c : A \to \mathbb{R}^+$ and $s, d \in V$

Let $f : A \to \mathbb{R}^+$ be a valid flow in D *Initially, f may be null*

1. Compute an auxiliary graph $D_{aux} = (V, A_{aux})$ and $c_{aux} : A_{aux} \to \mathbb{R}^+$

   $\forall u, v \in V$, add an arc $(u, v) \in A_{aux}$ with capacity

   $c_{aux}(uv) = c(uv) - f(uv) + f(vu)$.

2. If there is a directed path P from s to d in D_{aux}

   (auxiliary capacity of arcs of P must be > 0)

   - Let $\varepsilon > 0$ be the minimum capacity c_{aux} of the arcs of P
   - "Push" ε units of flow along P in D
   - Go to 1 *(Note that the value of the flow has increased)*

   Else Return f *f is a maximum flow.*

Ford-Fulkerson Algorithm

$D = (V, A)$, $c : A \to \mathbb{R}^+$ and $s, d \in V$

Let $f : A \to \mathbb{R}^+$ be a valid flow in D *Initially, f may be null*

1. Compute an auxiliary graph $D_{aux} = (V, A_{aux})$ and $c_{aux} : A_{aux} \to \mathbb{R}^+$

   $\forall u, v \in V$, add an arc $(u, v) \in A_{aux}$ with capacity

   $c_{aux}(uv) = c(uv) - f(uv) + f(vu)$.

2. If there is a directed path P from s to d in D_{aux}

   (auxiliary capacity of arcs of P must be > 0)

   - Let $\varepsilon > 0$ be the minimum capacity c_{aux} of the arcs of P
   - "Push" ε units of flow along P in D

     For each arc $(uv) \in A(P)$ of P:

     $f(uv) \leftarrow \min\{c(uv); f(uv) + \varepsilon\}$

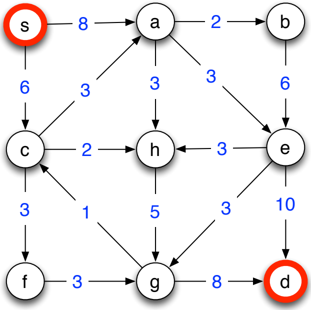
     push ε but if it exceeds the capacity of the link

     $f(vu) \leftarrow f(vu) - \max\{0; f(uv) + \varepsilon - c(uv)\}$

     "reverse" some flow
   - Go to 1 *(Note that the value of the flow has increased)*

   Else Return f *f is a maximum flow.*

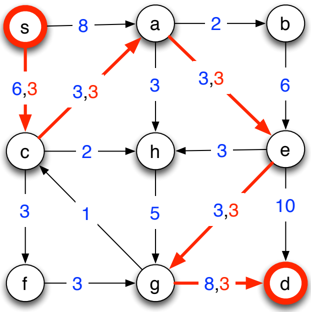
Ford-Fulkerson Algorithm Example



Digraph $D = (V, A)$, capacity $c : A \to \mathbb{R}^+$

Let us compute a max flow from $s \in V$ to $d \in V$.

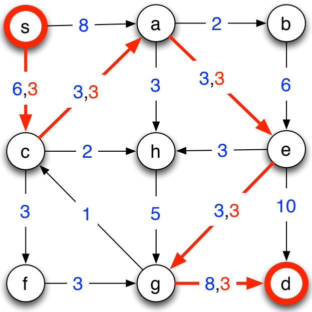
Ford-Fulkerson Algorithm Example

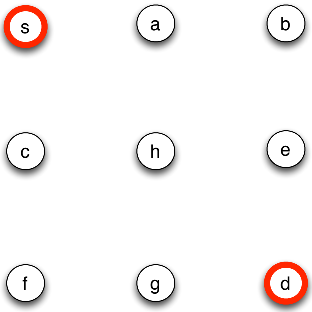


Let us compute a max flow from $s \in V$ to $d \in V$.

start from a given initial flow: in the example $v(f) = 3$.

Ford-Fulkerson Algorithm Example

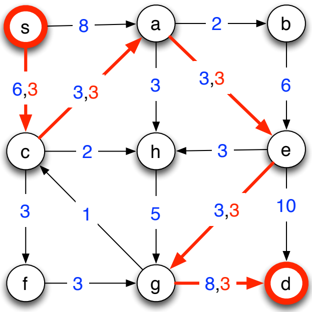


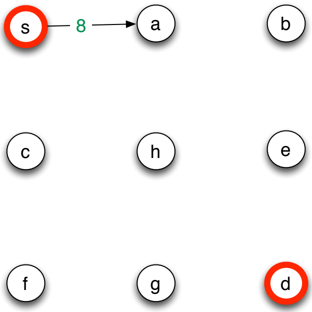


1^{st} **step:** Compute the first auxiliary digraph D_{aux}

start with same vertices as D

Ford-Fulkerson Algorithm Example



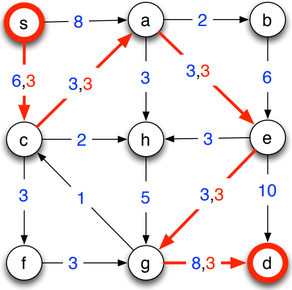


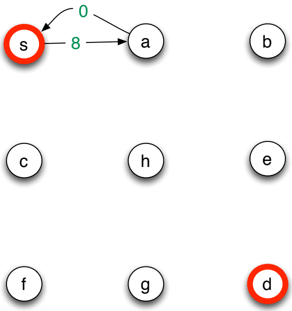
1^{st} **step:** Compute the first auxiliary digraph D_{aux}

for all $u, v \in V \times V$, $c_{aux}(uv) = c(uv) - f(uv) + f(vu)$:

in the example $c_{aux}(sa) = 8 - 0 + 0 = 8$

Ford-Fulkerson Algorithm Example



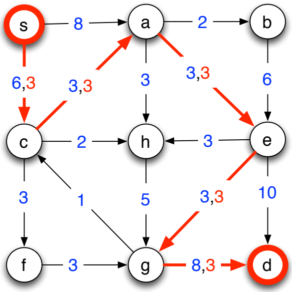


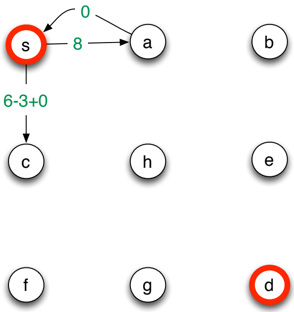
1^{st} **step:** Compute the first auxiliary digraph D_{aux}

for all $u, v \in V \times V$, $c_{aux}(uv) = c(uv) - f(uv) + f(vu)$:

in the example $c_{aux}(as) = 0 - 0 + 0 = 0$

Ford-Fulkerson Algorithm Example



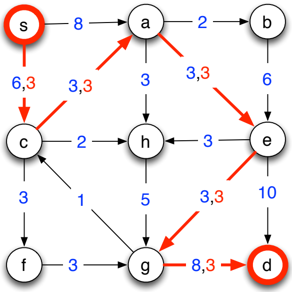


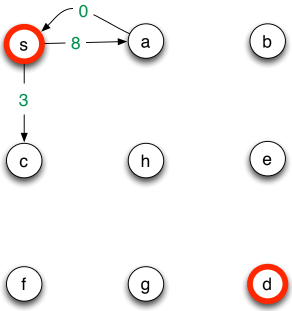
1^{st} **step:** Compute the first auxiliary digraph D_{aux}

for all $u, v \in V \times V$, $c_{aux}(uv) = c(uv) - f(uv) + f(vu)$:

in the example $c_{aux}(sc) = 6 - 3 + 0 = 3$

Ford-Fulkerson Algorithm Example



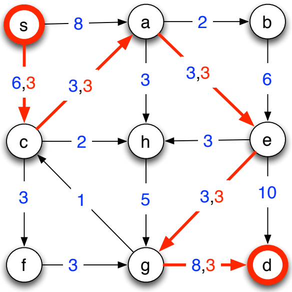


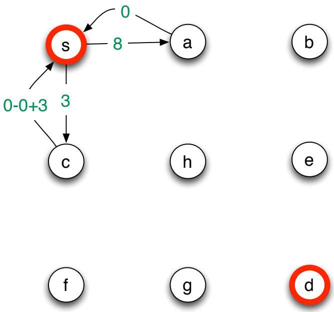
1st **step:** Compute the first auxiliary digraph D_{aux}

for all $u, v \in V \times V$, $c_{aux}(uv) = c(uv) - f(uv) + f(vu)$:

in the example $c_{aux}(sc) = 6 - 3 + 0 = 3$

Ford-Fulkerson Algorithm Example



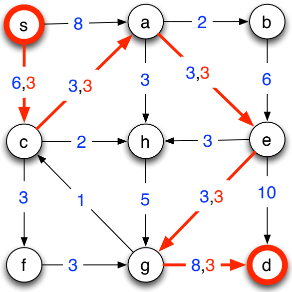


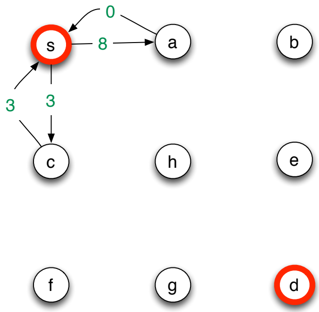
1^{st} **step:** Compute the first auxiliary digraph D_{aux}

for all $u, v \in V \times V$, $c_{aux}(uv) = c(uv) - f(uv) + f(vu)$:

in the example $c_{aux}(cs) = 0 - 0 + 3 = 3$

Ford-Fulkerson Algorithm Example



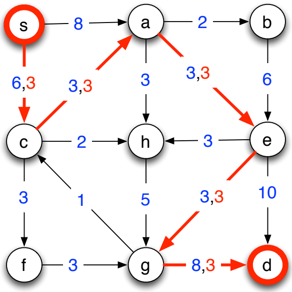


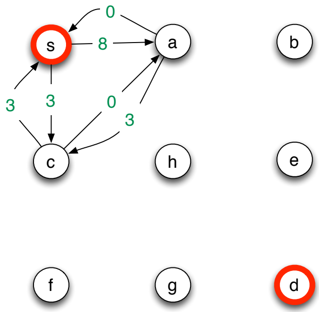
1^{st} **step:** Compute the first auxiliary digraph D_{aux}

for all $u, v \in V \times V$, $c_{aux}(uv) = c(uv) - f(uv) + f(vu)$:

in the example $c_{aux}(cs) = 0 - 0 + 3 = 3$

Ford-Fulkerson Algorithm Example



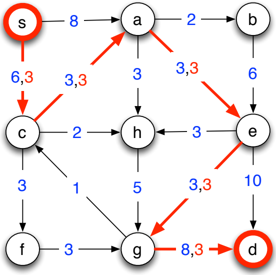


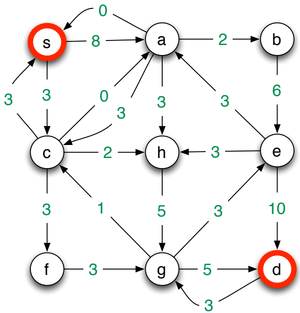
1^{st} **step:** Compute the first auxiliary digraph D_{aux}

for all $u, v \in V \times V$, $c_{aux}(uv) = c(uv) - f(uv) + f(vu)$:

in the example $c_{aux}(ca) = 3 - 3 + 0 = 0$ and $c_{aux}(ac) = 0 - 0 + 3 = 3$

Ford-Fulkerson Algorithm Example



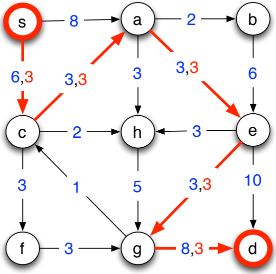


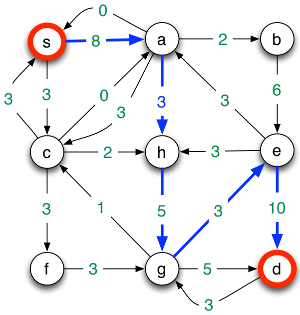
1^{st} **step:** Compute the first auxiliary digraph D_{aux}

for all $u, v \in V \times V$, $c_{aux}(uv) = c(uv) - f(uv) + f(vu)$:

and so on

Ford-Fulkerson Algorithm Example

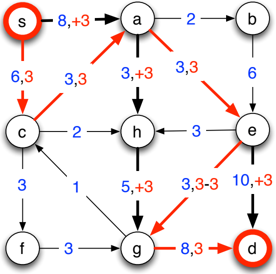


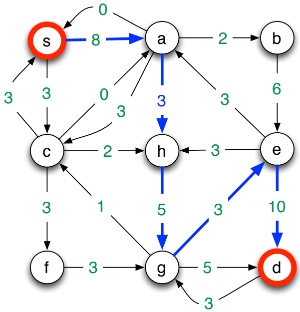


2^{nd} **step:** Find in D_{aux} a directed s,d-path with positive capacity

in the example $P = (s,a,h,g,e,d)$ and $\varepsilon = \min_{arc \in P} c_{aux}(arc) = 3$

Ford-Fulkerson Algorithm Example





2^{nd} **step:** Push ε units of flow along P in D. For each arc (uv) of P:

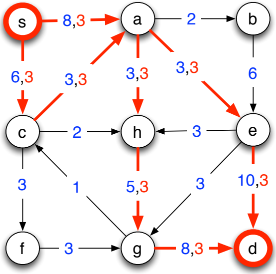
$$f(uv) \leftarrow \min\{c(uv); f(uv)+\varepsilon\}$$

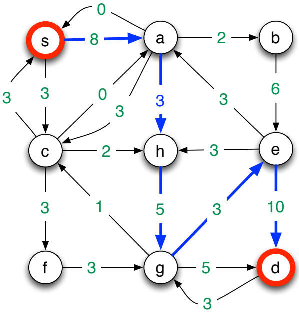
ex: $f(sa) = \min\{8, 0+3\} = 3$
$f(ge) = \min\{0, 0+3\} = 0$

$$f(vu) \leftarrow f(vu) - \max\{0; f(uv)+\varepsilon - c(uv)\}$$

ex: $f(eg) = 3 - \max\{0, 3+3-3\} = 0$

Ford-Fulkerson Algorithm Example

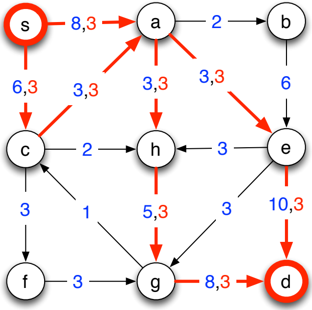


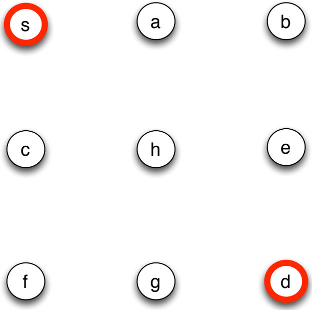


The flow has been increased:

$$v(f) = 6$$

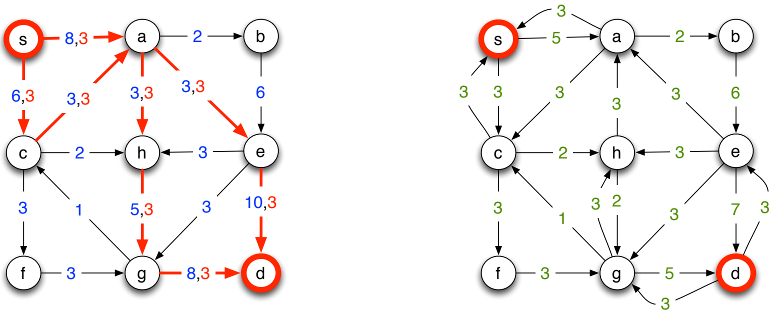
Ford-Fulkerson Algorithm Example





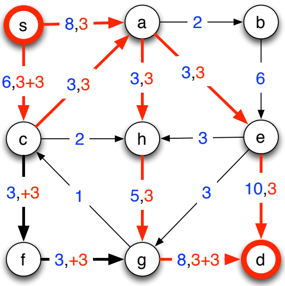
1^{st} **step:** We have to compute the auxiliary digraph starting from the new current flow

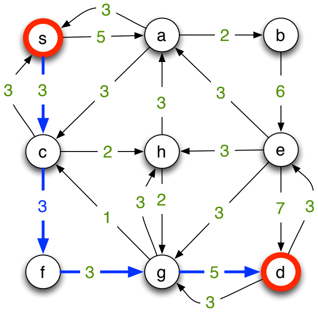
Ford-Fulkerson Algorithm Example



1^{st} **step:** We have to compute the auxiliary digraph starting from the new current flow

Ford-Fulkerson Algorithm Example

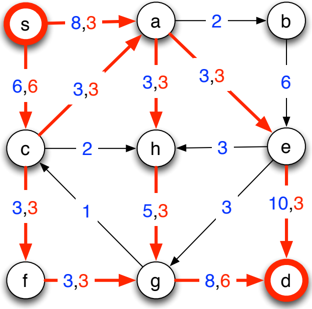


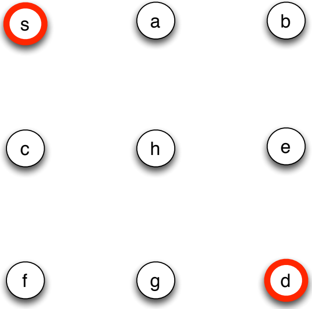


2^{nd} **step:** Find in D_{aux} a directed s,d-path with positive capacity

in the example $P=(s,c,f,g,d)$ and $\varepsilon = \min_{arc \in P} c_{aux}(arc) = 3$ and Push ε units of flow along P in D

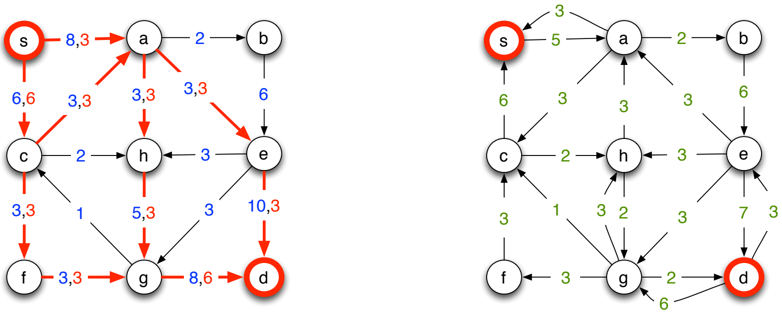
Ford-Fulkerson Algorithm Example





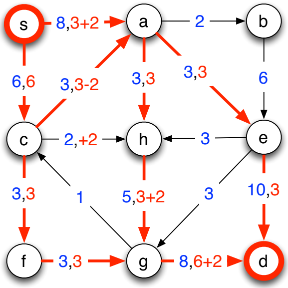
1^{st} **step:** We have to compute the auxiliary digraph starting from the new current flow $v(f) = 9$

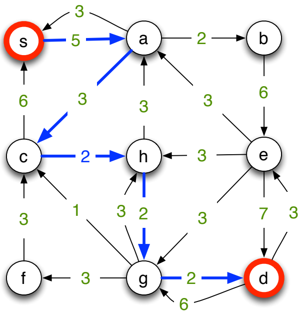
Ford-Fulkerson Algorithm Example



1[st] **step:** We have to compute the auxiliary digraph starting from the new current flow

Ford-Fulkerson Algorithm Example

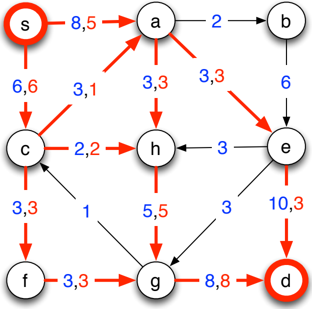


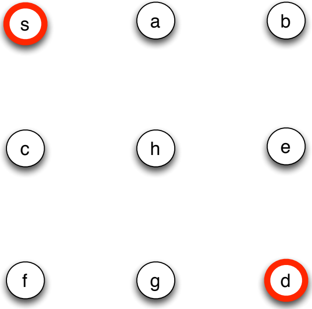


2nd **step:** Find in D_{aux} a directed s,d-path with positive capacity

in the example $P = (s, a, c, h, g, d)$ and $\varepsilon = \min_{arc \in P} c_{aux}(arc) = 2$ and Push ε units of flow along P in D

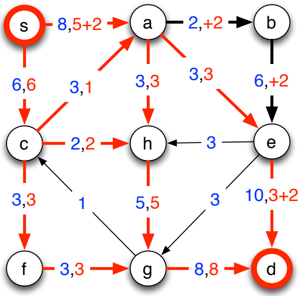
Ford-Fulkerson Algorithm Example

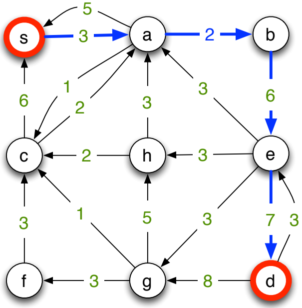




1^{st} **step:** We have to compute the auxiliary digraph starting from the new current flow $v(f) = 11$

Ford-Fulkerson Algorithm Example





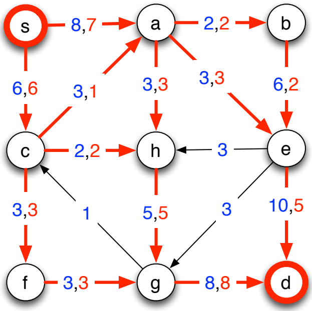
1^{st} **step:** We have to compute the auxiliary digraph

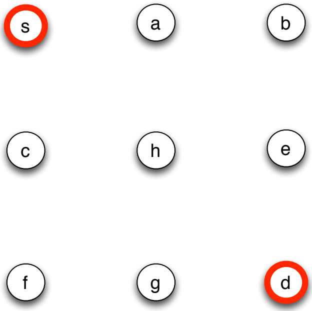
2^{nd} **step:** Find in $\underline{D_{aux}}$ a directed s,d-path with positive capacity

in the example $P = (s,a,b,e,d)$ and $\varepsilon = \min_{arc \in P} c_{aux}(arc) = 2$

and Push ε units of flow along P in $\underline{D}$

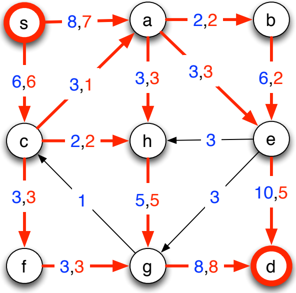
Ford-Fulkerson Algorithm Example

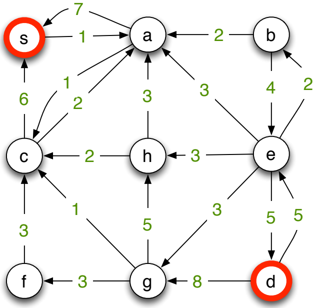




1^{st} **step:** We have to compute the auxiliary digraph
starting from the new current flow $v(f) = 13$

Ford-Fulkerson Algorithm Example



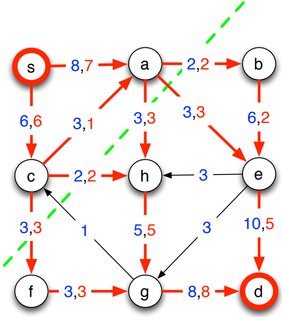


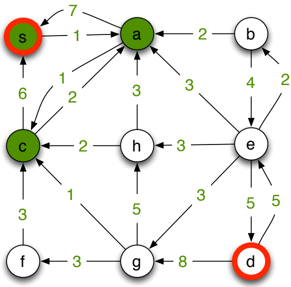
2^{nd} **step:** Find in $\underline{D_{aux}}$ a directed s,d-path with positive capacity

in the example, there is no such path !!

Exercise: Prove that the current flow is maximum

Ford-Fulkerson Algorithm Example





Aux. digraph defines a cut: $V_s = \{$vertices reachable from $s\}$; $V_d = V \setminus V_s$.

$\delta(V_s, V_d) = 13 = v(f) \Rightarrow$ *f* is maximum.

Outline

Ford-Fulkerson Algorithm: Min Cut=Max flow

Exercise: Let $f : A \rightarrow \mathbb{R}^+$ be computed by FF-algorithm

1. Prove that f is a valid flow
2. Prove that f is a maximum flow

idea of proof of 2: FF-algorithm returns a flow $f : A \rightarrow \mathbb{R}^+$
it also returns a cut with capacity $v(f)$.

Theorem: *Max Flow = Min Cut* *(duality)*

For any digraph D, $c : A(D) \rightarrow \mathbb{R}^+$ and $s, d \in V(D)$:
maximum value of a flow from s to d = minimum capacity of a s,d-cut

Ford-Fulkerson Algorithm: Complexity

Remark: the FF-algorithm may not terminate

Exercise: Assume capacities are integral $c : A \rightarrow \mathbb{N}$

- Prove that FF-algorithm always terminates
- Prove that the maximum value of a flow is integral

Complexity with integral capacities

FF-algorithm terminates in time $O(v_{max}|A(D)|)$

Ford-Fulkerson Algorithm: Complexity

Remark: the FF-algorithm may not terminate

Exercise: Assume capacities are integral $c : A \rightarrow \mathbb{N}$

- Prove that FF-algorithm always terminates
- Prove that the maximum value of a flow is integral

Complexity with integral capacities

FF-algorithm terminates in time $O(v_{max}|A(D)|)$

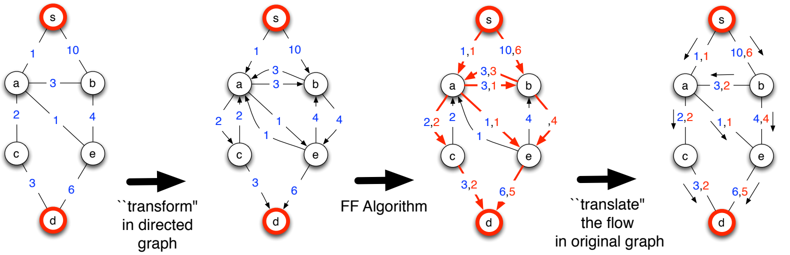
Ford-Fulkerson Algorithm: undirected graphs

In undirected graph $G = (V, E)$

$f : V \times V \to \mathbb{R}^+$ is defined on ordered pairs of vertices

Flow conservation: for any $v \in V \setminus \{s, d\}$, $\sum_{u \in N(v)} f(uv) = \sum_{u \in N(v)} f(vu)$

Capacity: for any $e = \{u, v\} \in E$, $f(uv) + f(vu) \leq c(\{u, v\})$.



Outline

Ford-Fulkerson Algorithm: Application to Connectivity

Edge-disjoint Paths: $G = (V, E)$, $s, d \in V$, $k \in \mathbb{N}$

Exercise: $\exists$ k edge-disjoint paths from s to d in G

$\Leftrightarrow$ $\exists$ a s,d-flow with value k in G (with capacity 1 on edges)

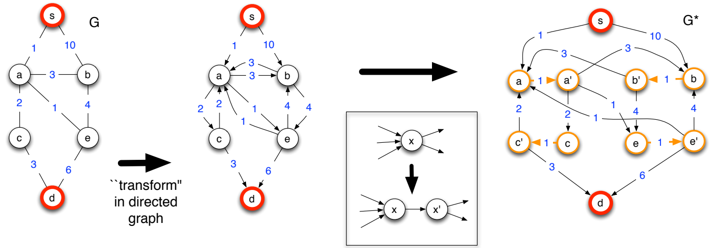
Idea: use the fact that there exists an integral maximum flow

Ford-Fulkerson Algorithm: Application to Connectivity

Vertex-disjoint Paths: $G=(V,E)$, $s,d \in V$, $k \in \mathbb{N}$

Exercise: $\exists$ k (internally) vertex-disjoint paths from s to d in G

$\Leftrightarrow$ $\exists$ a s,d-flow with value k in G^* (see Picture)

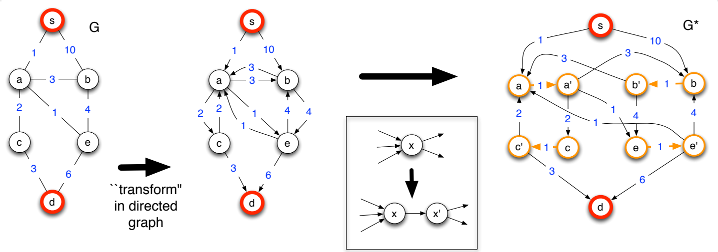


Ford-Fulkerson Algorithm: Application to Connectivity

Vertex-disjoint Paths: $G = (V, E)$, $s, d \in V$, $k \in \mathbb{N}$

Exercise: $\exists$ k (internally) vertex-disjoint paths from s to d in G

$\Leftrightarrow$ $\exists$ a s,d-flow with value k in G^* (see Picture)



$S \subseteq V$ is a s,d-separator if s and d in different components of $G \setminus S$

Menger's Theorem (1927): *(duality)*

For any graph $G = (V, E)$, $s, d \in V$ non-adjacent, $k \in \mathbb{N}$:

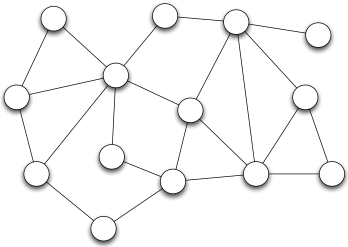
$\exists$ k internally-vertex-disjoint paths from s to d $\Leftrightarrow$ $\nexists$ s,d-separator of size $< k$.

Outline

Ford-Fulkerson Algorithm: Application to Matching

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

Matching: A set M of pairwise disjoint edges in a graph $(M \subseteq E)$

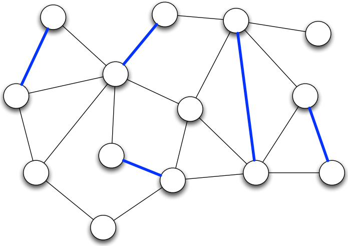


Problem: Compute a matching of maximum cardinality

Ford-Fulkerson Algorithm: Application to Matching

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

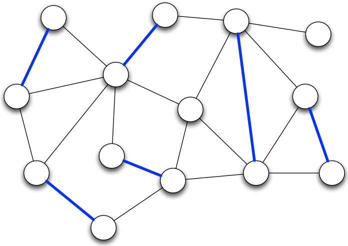
Matching: A set M of pairwise disjoint edges in a graph ($M \subseteq E$)



Question: is this matching maximum?

Ford-Fulkerson Algorithm: Application to Matching

Let $G = (V, E)$ be a graph.
Matching: A set M of pairwise disjoint edges in a graph ($M \subseteq E$)

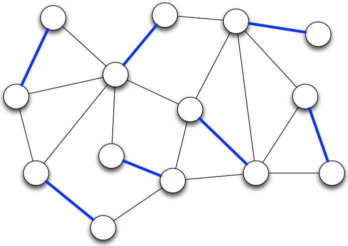


Question: and this one?

Ford-Fulkerson Algorithm: Application to Matching

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

Matching: A set M of pairwise disjoint edges in a graph ($M \subseteq E$)

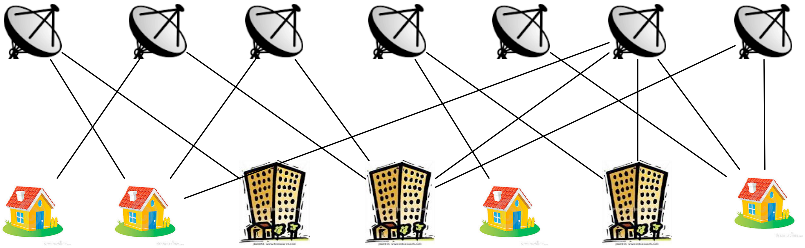


Question: and this one?

Exercise: Prove that any matching M is such that $|M| \leq \lfloor |V|/2 \rfloor$

Ford-Fulkerson Algorithm: Application to Matching

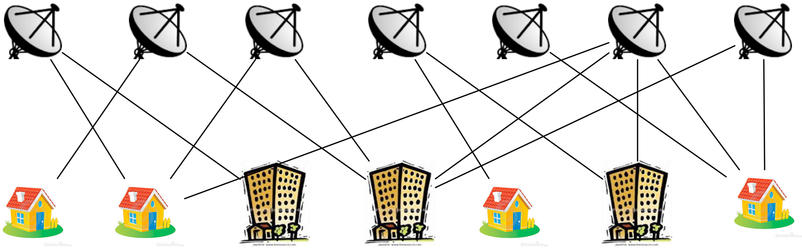
Example of application of matching:



stable set: set of vertices pariwise non-adjacent
Bipartite graph: $G = (V, E)$ and $V = A \cup B$ can be partitioned into 2 stable sets A and B.

Ford-Fulkerson Algorithm: Application to Matching

Example of application of matching:



stable set: set of vertices pariwise non-adjacent

Bipartite graph: $G=(V,E)$ and $V=A\cup B$ can be partitioned into 2 stable sets A and B.

Matching in bipartite graphs $G=(A\cup B,E)$

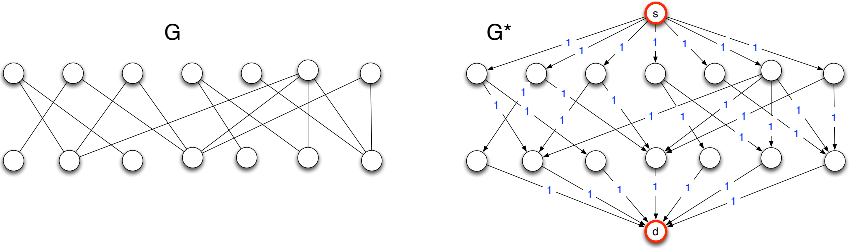
- Hall's Theorem (1935): $\exists$ matching saturating $A \Leftrightarrow \forall S\subseteq A, |N(S)|\geq |S|$.
- Hungarian Method [Kuhn, 1955]: compute a maximum matching in time $O(n^3)$

Ford-Fulkerson Algorithm: Application to Matching

$G=(A\cup B,E)$ a bipartite graph, $k\in\mathbb{N}$

Exercise: Prove that there exists a matching of size $\geq k$
$\Leftrightarrow$ exists a s,d-flow of value $\geq k$ in G^* (*see Picture*)

Idea: use the fact that there exists an integral maximum flow



Summary: To be remembered

- flow, cut
- Ford-Fulkerson algorithm $O(|E| flow_{max})$ for rational capacities
- Min Cut = Max Flow
- Menger Theorem (max $\#$ disjoint paths = min size of separator)
- Matching (Hungarian method, Edmonds algorithm, Hall Theorem)