Algorithmics and Complexity
Lecture 4/7 : Flow graphs

CentraleSupélec – Gif

ST2 – Gif

Plan

1. Real-world problem
2. Problem modeling
3. Ford-Fulkerson
4. Flow graphs as a model for other problems
5. Conclusion

The telecommunication operator problem

Context

An operator of a telecommunication network manages its own infrastructure and knows the throughput capacity of each link. He receives remuneration from other operators transiting by its network between the source router s and the target router t , $s \neq t$

The telecommunication operator problem

Context

An operator of a telecommunication network manages its own infrastructure and knows the throughput capacity of each link. He receives remuneration from other operators transiting by its network between the source router s and the target router t , $s \neq t$

Objective ?

Compute the maximum throughput of the network infrastructure between the entry point s and the exit point t .

The telecommunication operator problem

Context

An operator of a telecommunication network manages its own infrastructure and knows the throughput capacity of each link. He receives remuneration from other operators transiting by its network between the source router s and the target router t , $s \neq t$

Objective ?

Compute the maximum throughput of the network infrastructure between the entry point s and the exit point t .

Nature of the problem ?

It is an optimization problem

Maximum flow

There are many real-world applications of the problem :

Maximum flow

There are many real-world applications of the problem :

- What is the **maximum** flow transiting through a hydraulic network of pipes ?

Maximum flow

There are many real-world applications of the problem :

- What is the **maximum** flow transiting through a hydraulic network of pipes ?
- Goods have to be delivered through a network of roads each of which having a maximum capacity of goods that can flow through it. The problem is to find if there is a circulation that satisfies the demand ?

Maximum flow

There are many real-world applications of the problem :

- What is the **maximum** flow transiting through a hydraulic network of pipes ?
- Goods have to be delivered through a network of roads each of which having a maximum capacity of goods that can flow through it. The problem is to find if there is a circulation that satisfies the demand ?
- The enemy transports the steel produced in a location s to a tank manufacture in t with a railway network. What is the **minimum** number of railway links to destroy in order to stop the tanks production ?
- etc.

Plan

1 Real-world problem

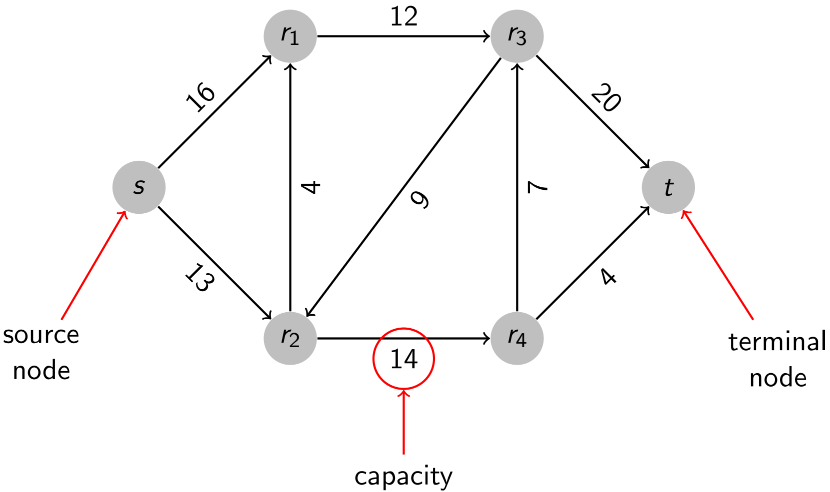
2 Problem modeling
- Example
- Problem

3 Ford-Fulkerson

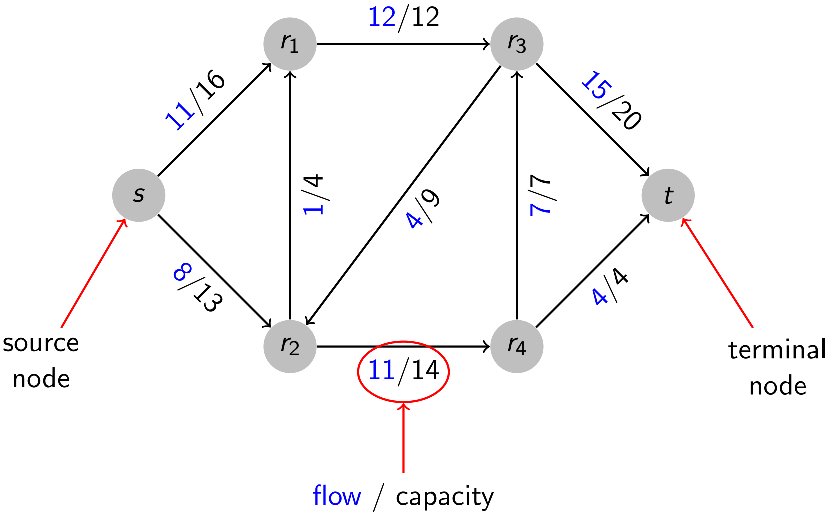
4 Flow graphs as a model for other problems

5 Conclusion

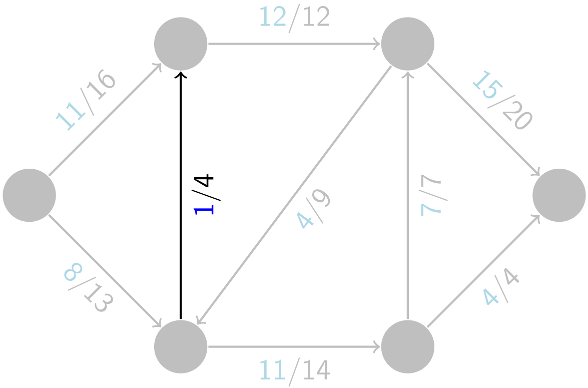
Flow graph example : capacities



Flow graph example : flows

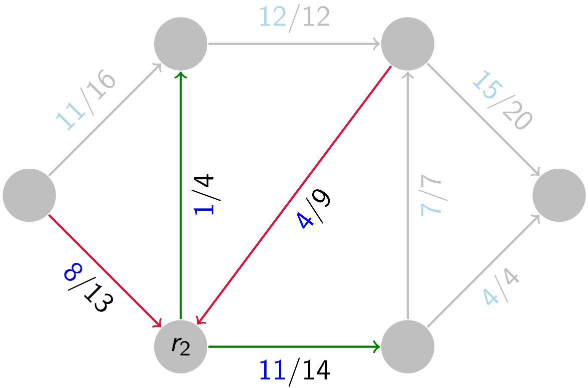


Flow graph example : flow–capacity rule



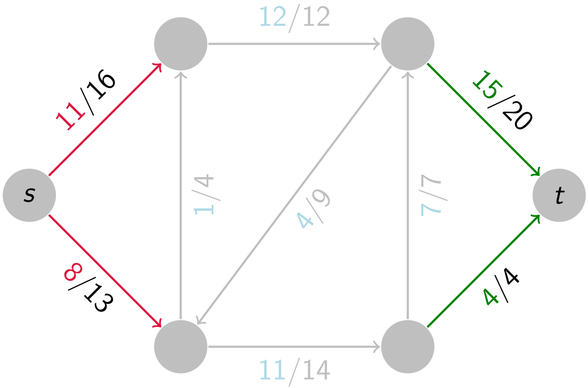
$$\text{Capacity} : \forall u, v \in V \times V \quad 0 \leq f(u, v) \leq c(u, v)$$

Flow graph example : flow conservation



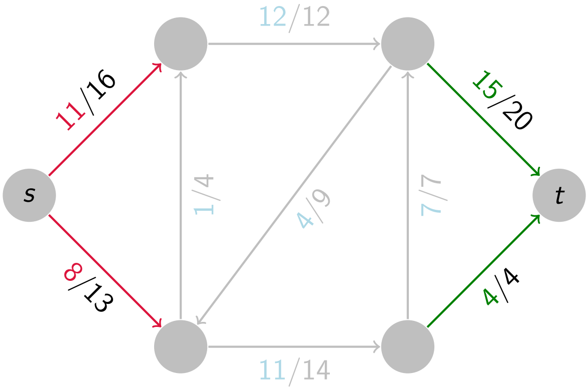
Conservation law : $\forall u \in V \setminus \{s, t\} \quad \sum_{v \in V} f(u, v) = \sum_{v \in V} f(v, u)$

Flow graph example : current flow



Current flow : $|f| = \sum_{u \in V} f(s, u) = \sum_{u \in V} f(u, t)$

Flow graph example : current flow



Current flow : $|f| = \sum_{u \in V} f(s,u) = \sum_{u \in V} f(u,t)$

What is the maximum flow ?

Problem definition

Flow graph

- A directed graph $G = (V, E)$
- 2 vertices $s \in V$: source and $t \in V$: terminal
- A capacity function $c : E \longrightarrow \mathbb{R}^+$

Problem definition

Flow graph

- A directed graph $G = (V, E)$
- 2 vertices $s \in V$: source and $t \in V$: terminal
- A capacity function $c : E \longrightarrow \mathbb{R}^+$

More definitions

We call flow a function $f : V \times V \longrightarrow \mathbb{R}$ such that :

Problem definition

Flow graph

- A directed graph $G = (V, E)$
- 2 vertices $s \in V$: source and $t \in V$: terminal
- A capacity function $c : E \longrightarrow \mathbb{R}^+$

More definitions

We call flow a function $f : V \times V \longrightarrow \mathbb{R}$ such that :

- capacity constraint :

$$\forall (u, v) \in E \quad 0 \leq f(u, v) \leq c(u, v)$$

Problem definition

Flow graph

- A directed graph $G = (V, E)$
- 2 vertices $s \in V$: source and $t \in V$: terminal
- A capacity function $c : E \longrightarrow \mathbb{R}^+$

More definitions

We call flow a function $f : V \times V \longrightarrow \mathbb{R}$ such that :

- capacity constraint :

$$\forall (u, v) \in E \quad 0 \leq f(u, v) \leq c(u, v)$$

- flow conservation constraint :

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

We set that $f(u, v) = 0$ when $(u, v) \notin E$

Problem definition II

Flow value

We call flow value of f and we denote

$$|f| = \sum_{v \in V} f(s, v) = \sum_{v \in V} f(v, t)$$

the quantity that flows out of the source. It is also the quantity that flows in the terminal.

Problem definition II

Flow value

We call flow value of f and we denote

$$|f| = \sum_{v \in V} f(s, v) = \sum_{v \in V} f(v, t)$$

the quantity that flows out of the source. It is also the quantity that flows in the terminal.

The maximum flow problem is about finding the maximum possible value $|f|$ for an f flow.

Simplifying hypothesis

For the sake of simplicity, we request the following :

1. each vertex is on a path between s and t

$$\forall v \in V \quad s \rightsquigarrow v \rightsquigarrow t$$

Simplifying hypothesis

For the sake of simplicity, we request the following :

1. each vertex is on a path between s and t

$$\forall v \in V \quad s \rightsquigarrow v \rightsquigarrow t$$

2. all capacities are strictly positive :

$$(u, v) \in E \iff c(u, v) \neq 0$$

Simplifying hypothesis

For the sake of simplicity, we request the following :

1. each vertex is on a path between s and t

$$\forall v \in V \quad s \rightsquigarrow v \rightsquigarrow t$$

2. all capacities are strictly positive :

$$(u, v) \in E \iff c(u, v) \neq 0$$

3. no loop on a vertex :

$$(u, u) \notin E$$

Simplifying hypothesis

For the sake of simplicity, we request the following :

1. each vertex is on a path between s and t

$$\forall v \in V \quad s \rightsquigarrow v \rightsquigarrow t$$

2. all capacities are strictly positive :

$$(u, v) \in E \iff c(u, v) \neq 0$$

3. no loop on a vertex :

$$(u, u) \notin E$$

4. we forbid $(u, v) \in E$ and $(v, u) \in E$ simultaneously

Simplifying hypothesis

For the sake of simplicity, we request the following :

1. each vertex is on a path between s and t

$$\forall v \in V \quad s \rightsquigarrow v \rightsquigarrow t$$

2. all capacities are strictly positive :

$$(u,v) \in E \iff c(u,v) \neq 0$$

3. no loop on a vertex :

$$(u,u) \notin E$$

4. we forbid $(u,v) \in E$ and $(v,u) \in E$ simultaneously

5. to simplify : no edge pointing to s or out of t

$$\forall u \in V \quad (u,s) \notin E \quad \text{et} \quad (t,u) \notin E$$

Plan

1 Real-world problem

2 Problem modeling

3 Ford-Fulkerson
- Idea
- Residual graph
- Example
- Augmentating the flow
- Flow and cut
- Max-Flow-Min-Cut
- Implementation

4 Flow graphs as a model for other problems

5 Conclusion

Ford-Fulkerson method (1962)

General idea

Iterative algorithm

➜ Increase the flow step by step, until the flow is saturating the graph

Ford-Fulkerson method (1962)

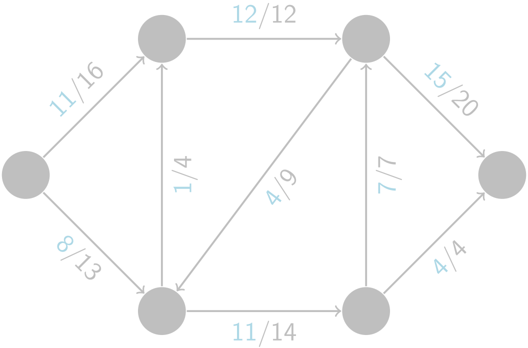
General idea

Iterative algorithm

- ➜ Increase the flow step by step, until the flow is saturating the graph

Relies on residual capacities and augmenting paths

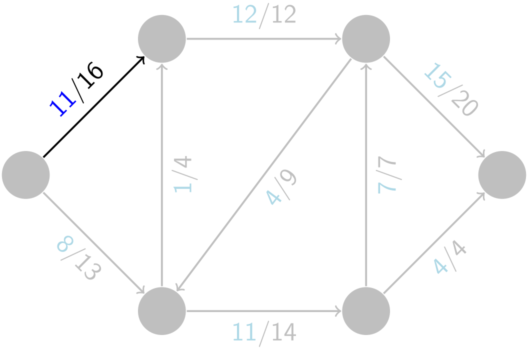
Residual capacity



Residual capacity

- What we can still push along an edge

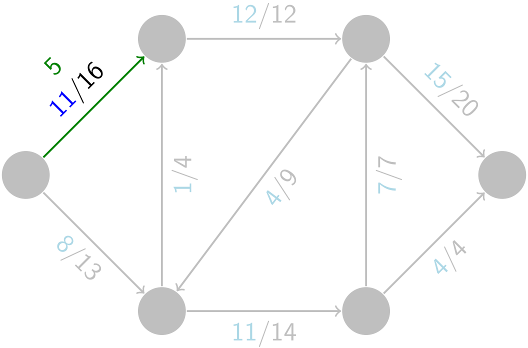
Residual capacity



Residual capacity

- What we can still push along an edge

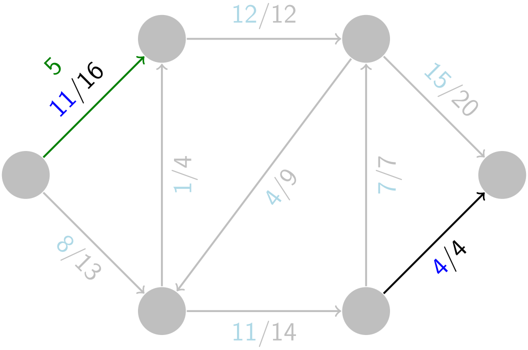
Residual capacity



Residual capacity

- What we can still push along an edge

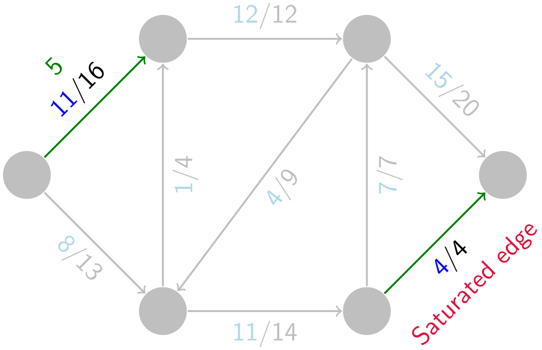
Residual capacity



Residual capacity

- What we can still push along an edge

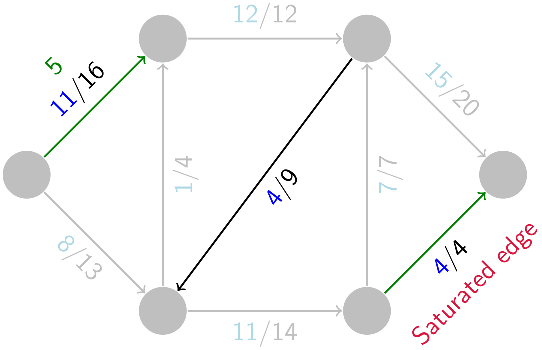
Residual capacity



Residual capacity

- What we can still push along an edge

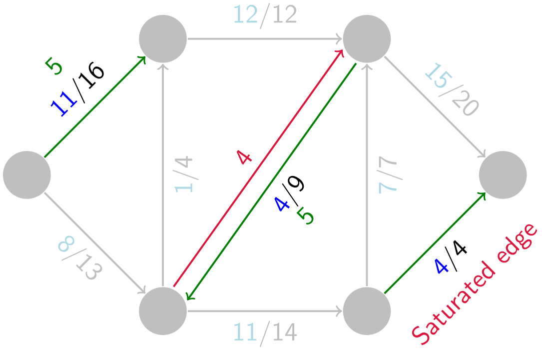
Residual capacity



Residual capacity

- What we can still push along an edge

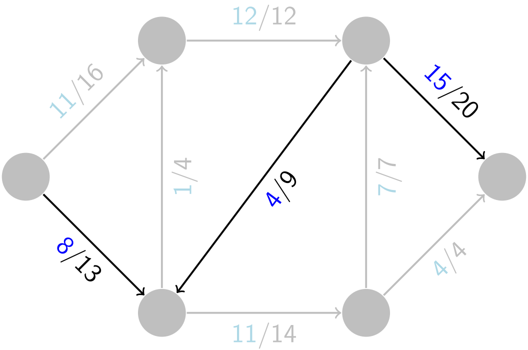
Residual capacity



Residual capacity

- What we can still push along an edge
- but also what we can cancel in the reverse direction
  - to keep it simple : no cancelling towards s or from t (justification later on).

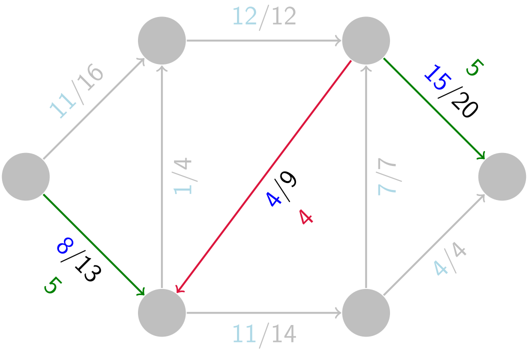
Augmenting path (main idea)



Augmentanting path

➜ « simple path » from s to t with residual capacities
edges taken forward or backward...

Augmenting path (main idea)



Augmentanting path

➜ ≪ simple path ≫ from s to t with residual capacities

edges taken forward or backward. . .

Ford-Fulkerson method (1962)

Main idea

Iterative algorithm :

➜ Augment the flow gradually until it is saturated

Sketch of the algorithm

1. Find an augmenting path with residual capacities between s and t
   by some not yet specified method. . .
2. Augment as much as possible the flow along this path
3. Repeat until you cannot augment the flow anymore

Ford-Fulkerson method (1962)

Main idea

Iterative algorithm :

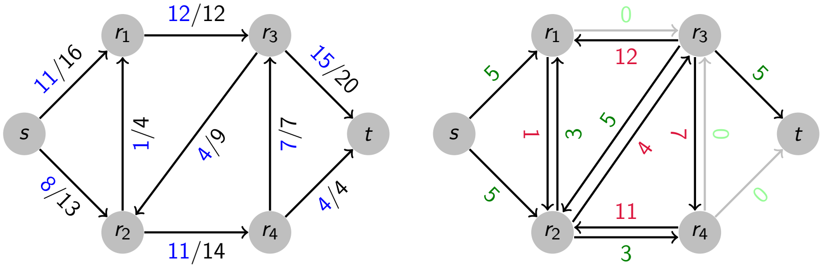
➜ Augment the flow gradually until it is saturated

Sketch of the algorithm

1. Find an augmenting path with residual capacities between s and t
   by some not yet specified method. . .
2. Augment as much as possible the flow along this path
3. Repeat until you cannot augment the flow anymore

➜ How to implement this algorithm ?

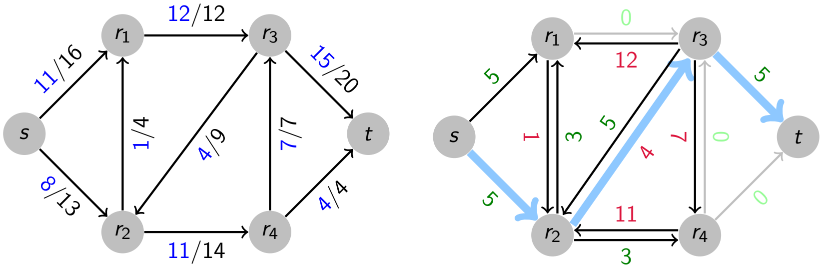
Definition of residual graph



Residual graph

- Graph induced by residual capacities

Augmenting path : example



Augmenting path

An augmenting path is a path between s and t in the residual graph

➜ The flow can be augmented by the value of the minimum residual capacity on an augmenting path

Definitions

Residual capacity

The residual capacity along (u, v) is the value of supplementary flow that can be sent from u to v, either directly or by cancelling flow in the reverse direction :

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

Residual graph

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

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

Definitions

Augmentanting path

An augmentanting path in G is a path between s and t in the residual graph of G induced by f.

Residual capacity of a path

The residual capacity of an augmenting path p is the maximum value by which we can augment the flow along this p path :

$$c_f(p) = \min\{c_f(u, v) \mid (u, v) \in p\}$$

Definitions

Augmentanting path

An augmentanting path in G is a path between s and t in the residual graph of G induced by f.

Residual capacity of a path

The residual capacity of an augmenting path p is the maximum value by which we can augment the flow along this p path :

$$c_f(p) = \min\{c_f(u,v) \mid (u,v) \in p\}$$

While augmenting :

- The flow is modified only along the path under consideration,
- the conservation law is kept valid.

  The sum of in-flows is equal to the sum of out-flows.

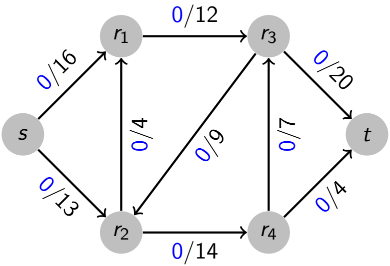
Ford-Fulkerson method (1962)

General idea

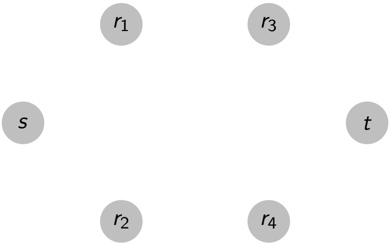
```
def FordFulkerson(G, s, t):
    # initialize Gr with G and f with 0
    Gr, f = ...

    while True:
        # find an augmenting path
        aug_path = search_aug_path(Gr, s, t)
        if not aug_path :
            break

        # compute the residual capacity for this path
        aug_flow = cf_path(Gr, aug_path)

        # update flow and residual graph
        Gr, f = update_flow_graph(Gr, f, aug_path, aug_flow)

    return f
```

Ford-Fulkerson : demonstration

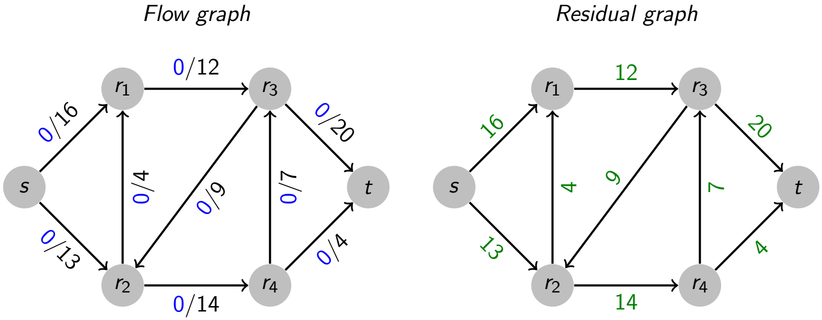
Flow graph



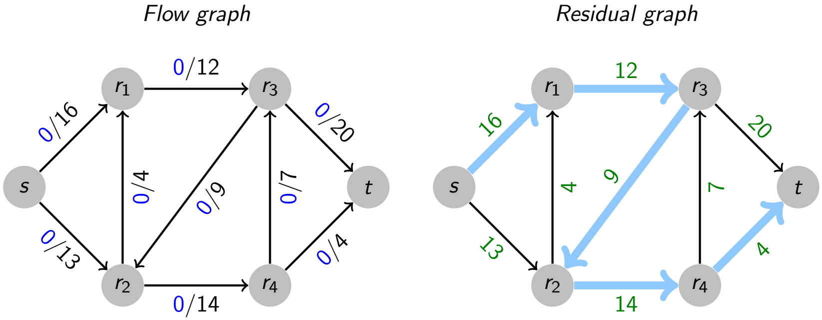
Residual graph



Ford-Fulkerson : demonstration

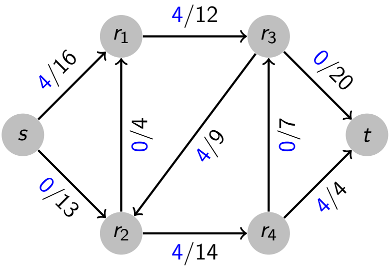


Ford-Fulkerson : demonstration

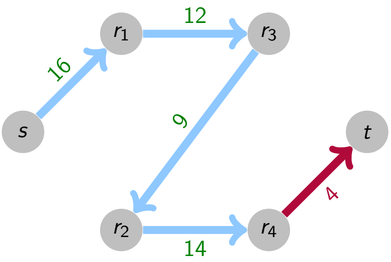


Ford-Fulkerson : demonstration

Flow graph

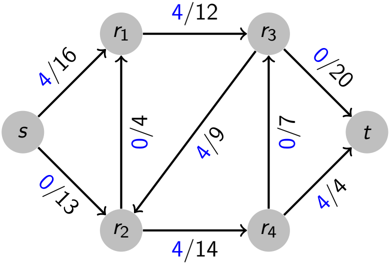


Residual graph

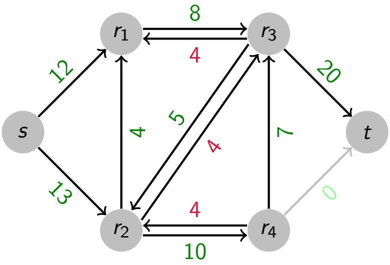


Ford-Fulkerson : demonstration

Flow graph

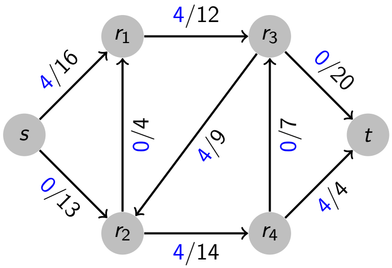


Residual graph

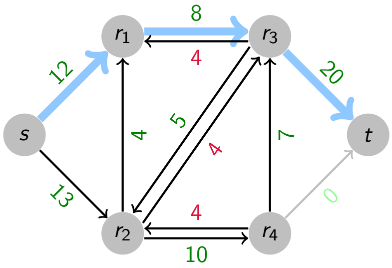


Ford-Fulkerson : demonstration

Flow graph

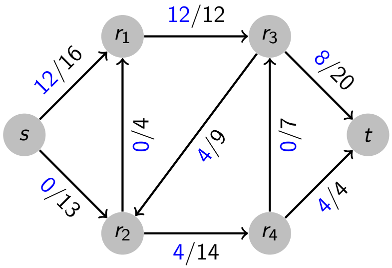


Residual graph

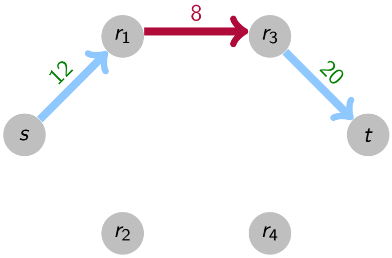


Ford-Fulkerson : demonstration

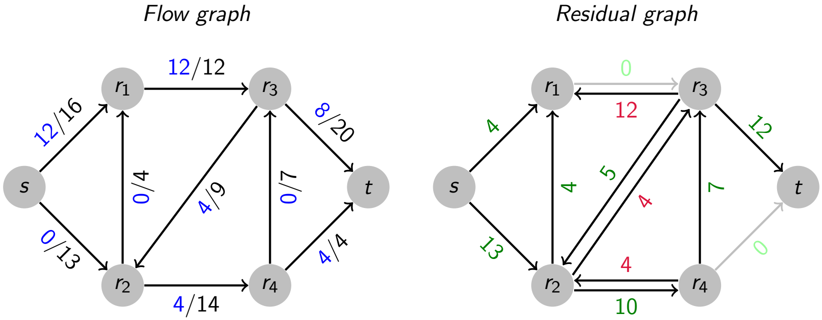
Flow graph



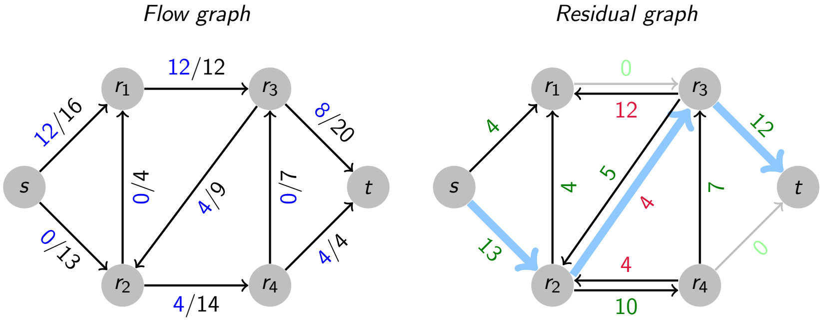
Residual graph



Ford-Fulkerson : demonstration

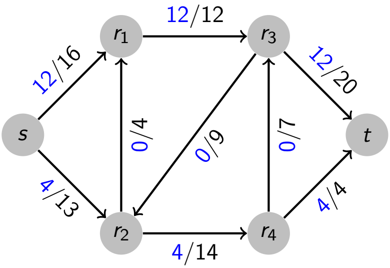


Ford-Fulkerson : demonstration

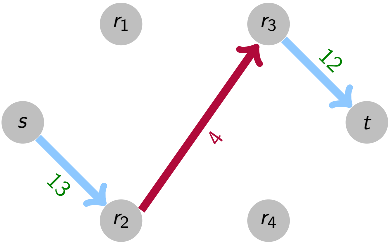


Ford-Fulkerson : demonstration

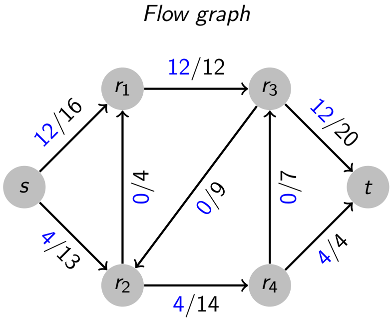
Flow graph

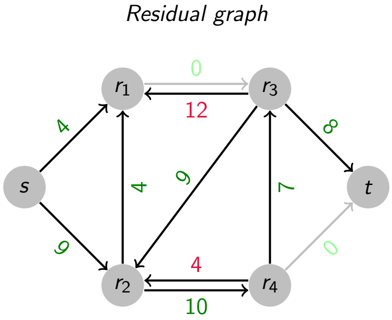


Residual graph

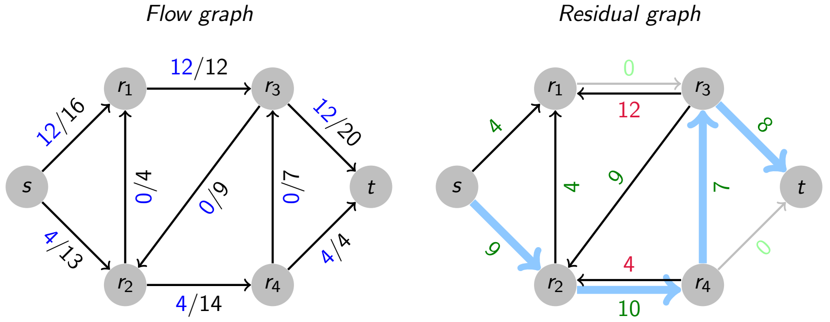


Ford-Fulkerson : demonstration

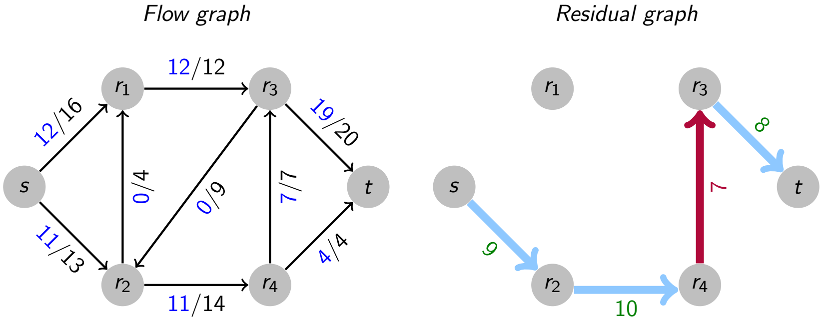




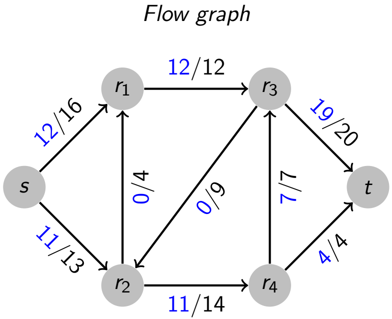
Ford-Fulkerson : demonstration

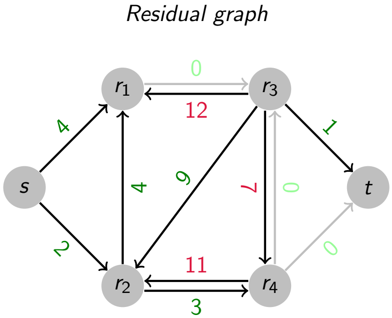


Ford-Fulkerson : demonstration

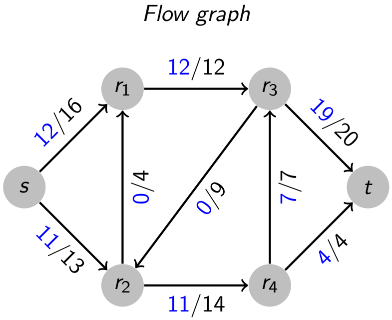


Ford-Fulkerson : demonstration





Ford-Fulkerson : demonstration



Termination

The algorithm stops when there is no path between s and t in the residual graph.

Ford-Fulkerson and flow augmentation

Theorem

- Let f be a flow in a flow graph G.
- Let $c_f(p)$ be the residual capacity of an augmenting path p in the residual graph G_f induced by f on G.

➜ The new flow $f' = f + c_f(p)$ computed by adding $c_f(p)$ along p in f is also a flow on G and $|f'| > |f|$

- Ford-Fulkerson does actually augment repeatedly the flow,

Ford-Fulkerson and flow augmentation

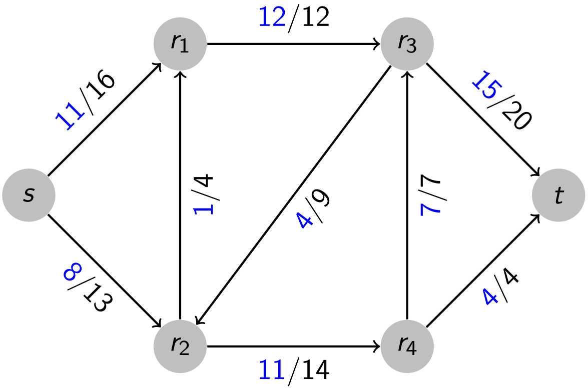
Theorem

- Let f be a flow in a flow graph G.
- Let $c_f(p)$ be the residual capacity of an augmenting path p in the residual graph G_f induced by f on G.

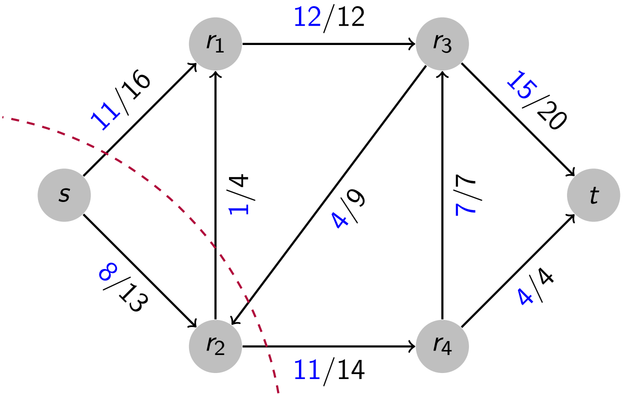
➜ The new flow $f' = f + c_f(p)$ computed by adding $c_f(p)$ along p in f is also a flow on G and $|f'| > |f|$

- Ford-Fulkerson does actually augment repeatedly the flow,
- why does Ford-Fulkerson converge towards the maximum flow ?

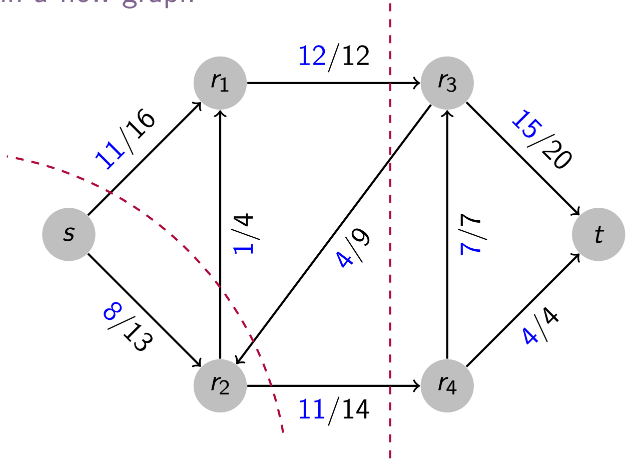
Cut in a flow graph



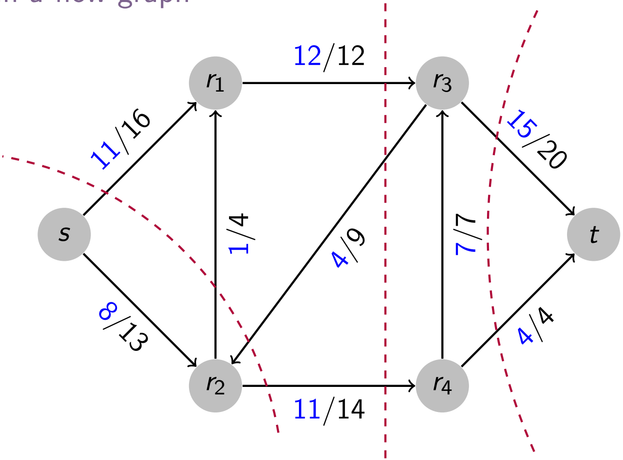
Cut in a flow graph



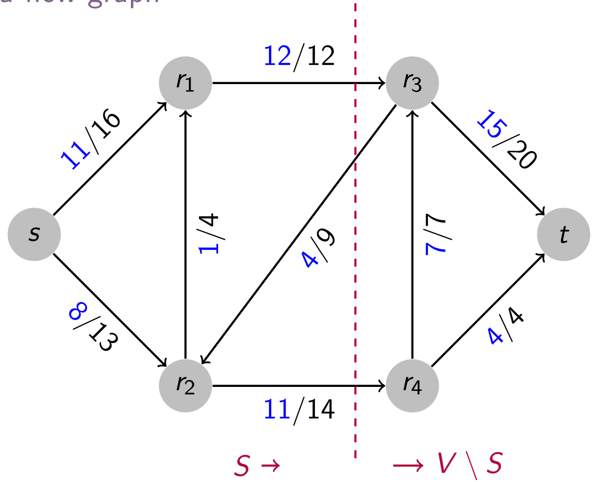
Cut in a flow graph



Cut in a flow graph



Cut in a flow graph



Cut in a flow graph

Definition

An s–t cut is a partition of V in S and $T = V \setminus S$ such that $s \in S$ and $t \in T = V \setminus S$.

Cut in a flow graph

Definition

An s–t cut is a partition of V in S and $T = V \setminus S$ such that $s \in S$ and $t \in T = V \setminus S$.
Its capacity is

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

Cut in a flow graph

Definition

An s–t cut is a partition of V in S and $T = V \setminus S$ such that $s \in S$ and $t \in T = V \setminus S$.
Its capacity is

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

The net flow across this cut is

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

How to compute the value of the flow on G ?

Definition and theorem

The value $|f|$ of the flow in G is equal to $f(S, T)$ for any s–t cut.

How to compute the value of the flow on G ?

Definition and theorem

The value $|f|$ of the flow in G is equal to $f(S, T)$ for any s–t cut.

$\rightarrow$ It can be proved that it is always the same !

Hint : if you move one vertex from S to T, this won't change the flow...

How to compute the value of the flow on G ?

Definition and theorem

The value $|f|$ of the flow in G is equal to $f(S, T)$ for any s–t cut.

$\rightarrow$ It can be proved that it is always the same !

Hint : if you move one vertex from S to T, this won't change the flow. . .

Two special s–t cuts

- $S = \{s\}$ (value at the source)
- $S = V \setminus \{t\}$ (value at the terminal)

Some observations

The flow value is bounded by any cut capacity :

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

Some observations

The flow value is bounded by any cut capacity :

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

Definition

An arc $(u, v) \in E$ is said to be saturated if $f(u, v) = c(u, v)$.

By extension, an s–t cut is saturated if $|f| = c(S, T)$.

(The flow is equal to the cut capacity)

Some observations

The flow value is bounded by any cut capacity :

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

Definition

An arc $(u, v) \in E$ is said to be saturated if $f(u, v) = c(u, v)$.

By extension, an s–t cut is saturated if $|f| = c(S, T)$.

(The flow is equal to the cut capacity)

Definition

A minimum cut is a cut of minimum capacity among all the cuts.

Some observations

The flow value is bounded by any cut capacity :

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

Definition

An arc $(u, v) \in E$ is said to be saturated if $f(u, v) = c(u, v)$.

By extension, an s–t cut is saturated if $|f| = c(S, T)$.

(The flow is equal to the cut capacity)

Definition

A minimum cut is a cut of minimum capacity among all the cuts.

Corollary : A saturated cut is a minimum cut.

Main theorem : max flow $\Leftrightarrow$ min cut $\Leftrightarrow$ no augmenting path

Theorem

The three propositions below are equivalent :

1. The flow $|f|$ between s and t is maximum.
2. There is no augmenting path.
3. There exists an s–t cut whose capacity is equal to $|f|$.

Main theorem : max flow $\Leftrightarrow$ min cut $\Leftrightarrow$ no augmenting path

Theorem

The three propositions below are equivalent :

1. The flow $|f|$ between s and t is maximum.
2. There is no augmenting path.
3. There exists an s–t cut whose capacity is equal to $|f|$.

Demonstration

1. 3 $\Rightarrow$ 1 : cut=flow $\Rightarrow$ max-flow
2. 1 $\Rightarrow$ 2 : max-flow $\Rightarrow$ no augmenting path
3. 2 $\Rightarrow$ 3 : no augmenting path $\Rightarrow$ cut=flow

skip proof

$3 \Rightarrow 1$: min-cut and flow-max

- By flow definition ($\forall e, f(e) \leq c(e)$) and by cut capacity definition (capacities sum), the flow value is bounded by the cut capacity :

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

$\rightarrow$ This holds for all cuts...

$3 \Rightarrow 1$: min-cut and flow-max

- By flow definition ($\forall e, f(e) \leq c(e)$) and by cut capacity definition (capacities sum), the flow value is bounded by the cut capacity :

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

$\rightarrow$ This holds for all cuts...

- If $c(S, T) = |f| \qquad (3)$ then necessarily :
  - $|f|$ is maximum $\qquad (1)$
  - $c(S, T)$ is minimum

$3 \Rightarrow 1$: min-cut and flow-max

- By flow definition ($\forall e, f(e) \leq c(e)$) and by cut capacity definition (capacities sum), the flow value is bounded by the cut capacity :

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

$\rightarrow$ This holds for all cuts...

- If $c(S,T) = |f| \qquad (3)$ then necessarily :
  - $|f|$ is maximum $\qquad (1)$
  - $c(S,T)$ is minimum

We showed :

$$3 \Rightarrow 1$$

but also : $\quad 3 \Rightarrow$ the cut is minimum

$1 \Rightarrow 2$: max-flow and no augmenting path

Proof by contraposition

If there exists an augmenting path...

... then we can augment the flow (hence it was not maximum)

$1 \Rightarrow 2$: max-flow and no augmenting path

Proof by contraposition

If there exists an augmenting path. . .

. . . then we can augment the flow (hence it was not maximum)

By contraposition, max-flow (1) $\Rightarrow$ no augmenting path (2)

$2 \Rightarrow 3$: no augmenting path and min-cut

Proof (1956)

Proof given at the same time by Ford and Fulkerson and by Elias, Feinstein and Shannon

Let $|f|$ be the value of a flow with no augmenting path.

Let S be the set of vertices reachable from s by following augmenting paths

- $s \in S$ (by definition) and $t \notin S$
  (because there is no augmenting path reaching t)

$2 \Rightarrow 3$: no augmenting path and min-cut

Proof (1956)

Proof given at the same time by Ford and Fulkerson and by Elias, Feinstein and Shannon

Let $|f|$ be the value of a flow with no augmenting path.

Let S be the set of vertices reachable from s by following augmenting paths

- $s \in S$ (by definition) and $t \notin S$
  (because there is no augmenting path reaching t)
- So we have an s–t cut with net flow :

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

(def. of the net flow for a cut)

$2 \Rightarrow 3$: no augmenting path and min-cut

Proof (1956)

Proof given at the same time by Ford and Fulkerson and by Elias, Feinstein and Shannon

Let $|f|$ be the value of a flow with no augmenting path.

Let S be the set of vertices reachable from s by following augmenting paths

- $s \in S$ (by definition) and $t \notin S$
  (because there is no augmenting path reaching t)
- So we have an s–t cut with net flow :

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

(the outgoing arcs are saturated, else we could reach T)

$2 \Rightarrow 3$: no augmenting path and min-cut

Proof (1956)

Proof given at the same time by Ford and Fulkerson and by Elias, Feinstein and Shannon

Let $|f|$ be the value of a flow with no augmenting path.

Let S be the set of vertices reachable from s by following augmenting paths

- $s \in S$ (by definition) and $t \notin S$
  (because there is no augmenting path reaching t)
- So we have an s–t cut with net flow :

$$f(S,T) = \sum_{u \in S, v \in T} c(u,v) - \sum_{u \in T, v \in S} 0$$

(The flow coming from T is null, else we could cancel it)

$2 \Rightarrow 3$: no augmenting path and min-cut

Proof (1956)

Proof given at the same time by Ford and Fulkerson and by Elias, Feinstein and Shannon

Let $|f|$ be the value of a flow with no augmenting path.

Let S be the set of vertices reachable from s by following augmenting paths

- $s \in S$ (by definition) and $t \notin S$
  (because there is no augmenting path reaching t)
- So we have an s–t cut with net flow :

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

$\Rightarrow$ flow value $=$ cut capacity ■

Ford-Fulkerson algorithm in Python

```
def FordFulkerson(G, s, t):
    # initialisze Gr and f
    ...

    while True:
        # find an augmenting path
        aug_path = search_aug_path(Gr, s, t)
        if aug_path == None:
            break
        # compute the residual capacity of the path
        aug_flow = cf_path(Gr, aug_path)

        for k in range(len(aug_path)-1):
            u, v = aug_path[k], aug_path[k+1]
            # update flow and residual graph
            # along the augmenting path
            if v in neighbours(G, u) :
                f[u][v] += aug_flow
                cf[u][v], cf[v][u] = c[u][v] - f[u][v], f[u][v]
            else :
                f[v][u] -= aug_flow
                cf[v][u], cf[u][v] = c[v][u] - f[v][u], f[v][u]
```

Searching an augmenting path in Python

```
def search_aug_path(Gr,s,t):

    lnext = [s]
    parent = {s:None}

    while len(lnext)>0:
      n = pop_end(lnext) # DFS or pop_begin for BFS

      if n==t:
          return path(parent, t) # returns the augmenting path

      for v in neighbours(Gr, n):
        if not v in parent:
          add_end(v,lnext)
          parent[v] = n

    return None # no augmenting path
```

N.B. Classical F-F is using DFS, but you are free to choose another method to find an augmenting path. For example, Edmonds and Karp suggest to use BFS.

Complexity and convergence of the Ford-Fulkerson algorithm

Assume that $c : E \longrightarrow \mathbb{N}$

- DFS is in $\mathcal{O}(|V| + |E|)$.

Complexity and convergence of the Ford-Fulkerson algorithm

Assume that $c : E \longrightarrow \mathbb{N}$

- DFS is in $\mathcal{O}(|V| + |E|)$.
- F-F algorithm complexity ?

Complexity and convergence of the Ford-Fulkerson algorithm

Assume that $c : E \longrightarrow \mathbb{N}$

- DFS is in $\mathcal{O}(|V| + |E|)$.
- F-F algorithm complexity : $\mathcal{O}((|V| + |E|) \times |f|_{max})$
  - depends on the value of the answer $|f|_{max} \in \mathbb{N}$
  - and when $|f|_{max} >> |V|$, for example : $|f|_{max} \approx 2^{|V|}$! !

Complexity and convergence of the Ford-Fulkerson algorithm

Assume that $c : E \longrightarrow \mathbb{N}$

- DFS is in $\mathcal{O}(|V| + |E|)$.
- F-F algorithm complexity : $\mathcal{O}((|V| + |E|) \times |f|_{max})$
  - depends on the value of the answer $|f|_{max} \in \mathbb{N}$
  - and when $|f|_{max} >> |V|$, for example : $|f|_{max} \approx 2^{|V|}$!!

Assume that $c : E \longrightarrow \mathbb{Q}$?

Complexity and convergence of the Ford-Fulkerson algorithm

Assume that $c : E \longrightarrow \mathbb{N}$

- DFS is in $\mathcal{O}(|V| + |E|)$.
- F-F algorithm complexity : $\mathcal{O}((|V| + |E|) \times |f|_{max})$
  - depends on the value of the answer $|f|_{max} \in \mathbb{N}$
  - and when $|f|_{max} >> |V|$, for example : $|f|_{max} \approx 2^{|V|}$! !

Assume that $c : E \longrightarrow \mathbb{Q}$

- F-F algorithm complexity is in $\mathcal{O}((|V| + |E|) \times |f|_{max} \times d)$! where d is the common denominator

Complexity and convergence of the Ford-Fulkerson algorithm

Assume that $c : E \longrightarrow \mathbb{N}$

- DFS is in $\mathcal{O}(|V| + |E|)$.
- F-F algorithm complexity : $\mathcal{O}((|V| + |E|) \times |f|_{max})$
  - depends on the value of the answer $|f|_{max} \in \mathbb{N}$
  - and when $|f|_{max} >> |V|$, for example : $|f|_{max} \approx 2^{|V|}$! !

Assume that $c : E \longrightarrow \mathbb{Q}$

- F-F algorithm complexity is in $\mathcal{O}((|V| + |E|) \times |f|_{max} \times d)$! where d is the common denominator

Assume that $c : E \longrightarrow \mathbb{R}$?

Complexity and convergence of the Ford-Fulkerson algorithm

Assume that $c : E \longrightarrow \mathbb{N}$

- DFS is in $\mathcal{O}(|V| + |E|)$.
- F-F algorithm complexity : $\mathcal{O}((|V| + |E|) \times |f|_{max})$
  - depends on the value of the answer $|f|_{max} \in \mathbb{N}$
  - and when $|f|_{max} >> |V|$, for example : $|f|_{max} \approx 2^{|V|}$! !

Assume that $c : E \longrightarrow \mathbb{Q}$

- F-F algorithm complexity is in $\mathcal{O}((|V| + |E|) \times |f|_{max} \times d)$! where d is the common denominator

Assume that $c : E \longrightarrow \mathbb{R}$

- it may happen that F-F never terminates !
- augmenting paths with smaller and smaller residual capacities
- safe for a computer ! (see implementation exercise in TD3)

Alternatives

Algorithms whose complexity does not depend on $|f|_{max}$

- Edmonds-Karp (F-F based on BFS), 1970 :
  - in $\mathcal{O}(|E|^2 \times |V|)$

Alternatives

Algorithms whose complexity does not depend on $|f|_{max}$

- Edmonds-Karp (F-F based on BFS), 1970 :
  - in $\mathcal{O}(|E|^2 \times |V|)$
  - converges for capacities in $\mathbb{N}$, $\mathbb{Q}$ or $\mathbb{R}$
  - in less than $|V| \times |E|$ augmenting paths (iterations)

Alternatives

Algorithms whose complexity does not depend on $|f|_{max}$

- Edmonds-Karp (F-F based on BFS), 1970 :
  - in $\mathcal{O}(|E|^2 \times |V|)$
  - converges for capacities in $\mathbb{N}$, $\mathbb{Q}$ or $\mathbb{R}$
  - in less than $|V| \times |E|$ augmenting paths (iterations)
- Dinic (Dinitz), 1970, en $\mathcal{O}(|E| \times |V|^2)$
- Orlin, 2013, in $\mathcal{O}(|E| \times |V|)$ and even in $\mathcal{O}(\frac{|V|^2}{\log(|V|)})$ when $|E|$ is in $\mathcal{O}(|V|)$

Plan

1. Real-world problem
2. Problem modeling
3. Ford-Fulkerson
4. Flow graphs as a model for other problems
   - Student residence
   - Week of BDE (Student Office)
5. Conclusion

Managing university residences

Context

A university has M residences, the number of students that each residence can host is $m_i, i = 1, \ldots, M$.
The university welcomes N students. A student communicates a list of the residences in which he wishes to be accommodated.

Managing university residences

Context

A university has M residences, the number of students that each residence can host is $m_i, i = 1, \ldots, M$.
The university welcomes N students. A student communicates a list of the residences in which he wishes to be accommodated.

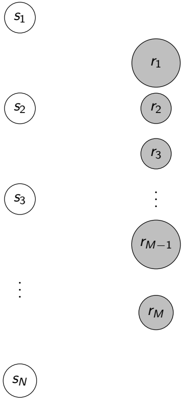
Goal

Suggest an allocation of residences by maximizing the number of accepted students.

Model construction : flow graph

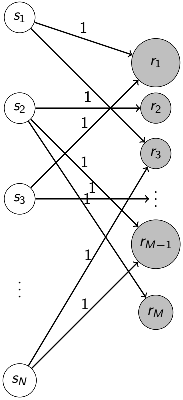
Model construction : flow graph

1. a bipartite graph $(V_N \cup V_M, E = V_N \times V_M)$, where V_N are the students vertices and V_M the residences vertices



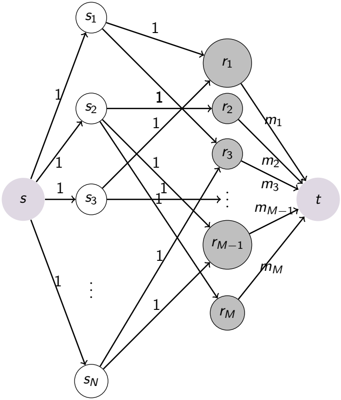
Model construction : flow graph

1. a bipartite graph $(V_N \cup V_M, E = V_N \times V_M)$, where V_N are the students vertices and V_M the residences vertices
2. an arc $(s_i, r_j) \in V_N \times V_M$ for each wish of the student s_i to get the residence r_j with $c((s_i, r_j)) = 1$



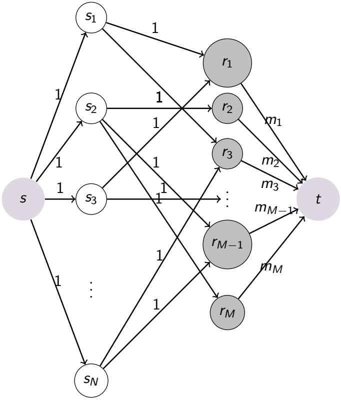
Model construction : flow graph

1. a bipartite graph $(V_N \cup V_M, E = V_N \times V_M)$, where V_N are the students vertices and V_M the residences vertices
2. an arc $(s_i, r_j) \in V_N \times V_M$ for each wish of the student s_i to get the residence r_j with $c((s_i, r_j)) = 1$
3. add two artificial nodes, s and t and connect them by the following arcs :
   - $e = (s, s_i))$ for each $s_i \in V_N$ with $c(e) = 1$
   - $e = (r_i, t)$ for each $r_i \in V_M$ with $c(e) = m_i$



Model construction : flow graph

1. a bipartite graph $(V_N \cup V_M, E = V_N \times V_M)$, where V_N are the students vertices and V_M the residences vertices
2. an arc $(s_i, r_j) \in V_N \times V_M$ for each wish of the student s_i to get the residence r_j with $c((s_i, r_j)) = 1$
3. add two artificial nodes, s and t and connect them by the following arcs :
   - $e = (s, s_i))$ for each $s_i \in V_N$ with $c(e) = 1$
   - $e = (r_i, t)$ for each $r_i \in V_M$ with $c(e) = m_i$



Conclusion

Solving the students housing problem is reduced to finding the max-flow of the above graph.

Week of BDE

Context

During the day, the students of CentraleSupélec go from the student residence (in the morning) to the canteen (at noon), passing through various classrooms.

Problem

The BDE team wants to distribute volunteers on the students' route so that no one can avoid the distribution of leaflets. Depending on the size of the passage areas, it may be necessary to put several volunteers to cover a wide passage.

Week of BDE

Context

During the day, the students of CentraleSupélec go from the student residence (in the morning) to the canteen (at noon), passing through various classrooms.

Problem

The BDE team wants to distribute volunteers on the students' route so that no one can avoid the distribution of leaflets. Depending on the size of the passage areas, it may be necessary to put several volunteers to cover a wide passage.

Goal

Propose an optimal assignment with the minimum number of volunteers so as not to miss any student.

Model construction :

1. A graph whose vertices represent the residence, the classrooms and the canteen. The edges represent the different paths allowing to go directly from one point to another of the campus.

Model construction :

1. A graph whose vertices represent the residence, the classrooms and the canteen. The edges represent the different paths allowing to go directly from one point to another of the campus.
2. We define the capacity of an edge as the number of volunteers that must be positioned to intercept all the students who pass by this path.

Model construction :

1. A graph whose vertices represent the residence, the classrooms and the canteen. The edges represent the different paths allowing to go directly from one point to another of the campus.
2. We define the capacity of an edge as the number of volunteers that must be positioned to intercept all the students who pass by this path.
3. Two special vertices : the residence (s) and the canteen (t).

minimum cut problem

What is the minimum cut in this graph separating s and t ?

Model construction :

1. A graph whose vertices represent the residence, the classrooms and the canteen. The edges represent the different paths allowing to go directly from one point to another of the campus.
2. We define the capacity of an edge as the number of volunteers that must be positioned to intercept all the students who pass by this path.
3. Two special vertices : the residence (s) and the canteen (t).

minimum cut problem

What is the minimum cut in this graph separating s and t ?

➜ The capacity of this cup gives the number of volunteers required.

Model construction :

1. A graph whose vertices represent the residence, the classrooms and the canteen. The edges represent the different paths allowing to go directly from one point to another of the campus.
2. We define the capacity of an edge as the number of volunteers that must be positioned to intercept all the students who pass by this path.
3. Two special vertices : the residence (s) and the canteen (t).

minimum cut problem

What is the minimum cut in this graph separating s and t ?

→ The capacity of this cup gives the number of volunteers required.

Exercise : it is about a non-oriented graph, how to transform it into a flow graph ?

Plan

1. Real-world problem
2. Problem modeling
3. Ford-Fulkerson
4. Flow graphs as a model for other problems
5. Conclusion

To keep in mind

- Directed graph : $G = (V, E)$ with capacities $c : E \to \mathbb{R}^+$
- Main theorem :

max flow $\Leftrightarrow$ min cut $\Leftrightarrow$ no augmenting path

- Ford-Fulkerson algorithm :
  - Finding augmenting paths (free traversal) ;
  - Complexity within $\mathcal{O}((|V| + |E|) \times |f|_{max})$;
  - When F-F terminates, we obtain the maximum flow (theorem) ;
  - ➜ Edmonds-Karp's variant based on BFS within $\mathcal{O}(|E|^2 \times |V|)$
- Many practical applications :
  - networks of any kind. . .