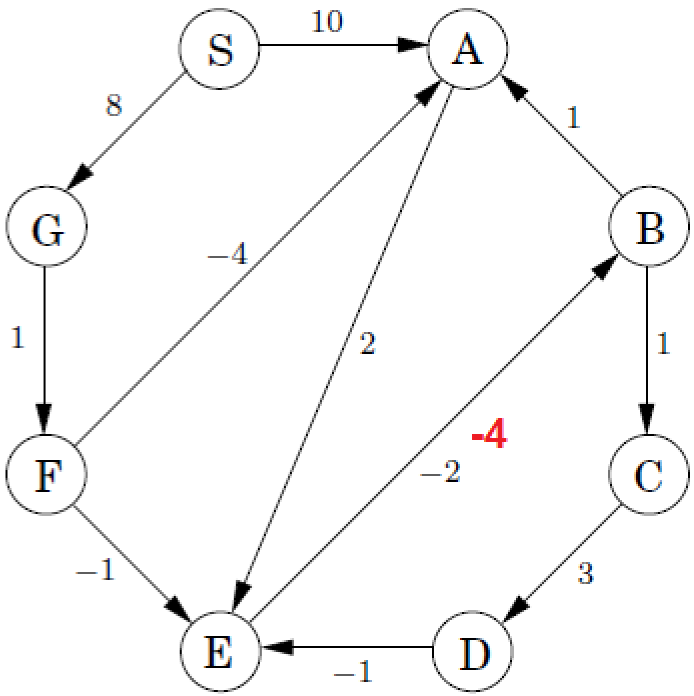
# Algorísmica Avançada
# Algorismes sobre grafs III

Sergio Escalera

# Algorismes sobre grafs

- Cicles negatius




$A \to E \to B \to A$

Els podem trobar amb l'algorisme Bellman-Ford

El camí mínim té com a màxim longitud |V|-1
Podem detectar cicles si fem una iteració extra |V|

→ Hi ha cicle negatiu si a la iteració |V| alguna aresta és actualitzada

# Algorismes sobre grafs

- Cicles negatius
- Hi ha dos tipus de grafs que no tenen cicles negatius: sense pesos negatius i sense cicles
- El primer és directe. Per resoldre el camí mínim en acíclic grafs dirigits negatius:
  - Linealitzar usant DFS
  - Temps lineal!
- Si posem els negatius dels pesos podem trobar els camins de longitud màxima

# Algorismes sobre grafs

**Linealitzar usant DFS**

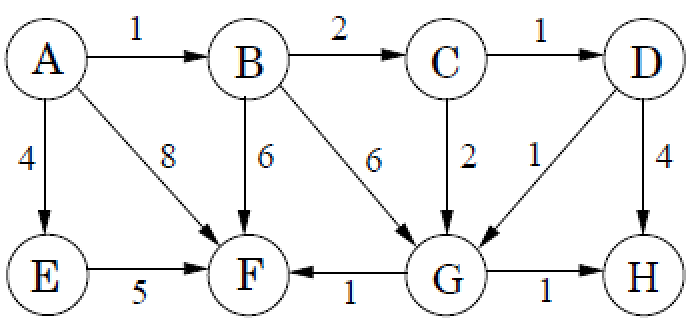
```
procedure dag-shortest-paths(G, l, s)
Input:    Dag G = (V, E);
          edge lengths {l_e : e ∈ E}; vertex s ∈ V
Output:   For all vertices u reachable from s, dist(u) is set
          to the distance from s to u.

for all u ∈ V:
   dist(u) = ∞
   prev(u) = nil

dist(s) = 0
Linearize G
for each u ∈ V, in linearized order:
   for all edges (u, v) ∈ E:
      update(u, v)
```

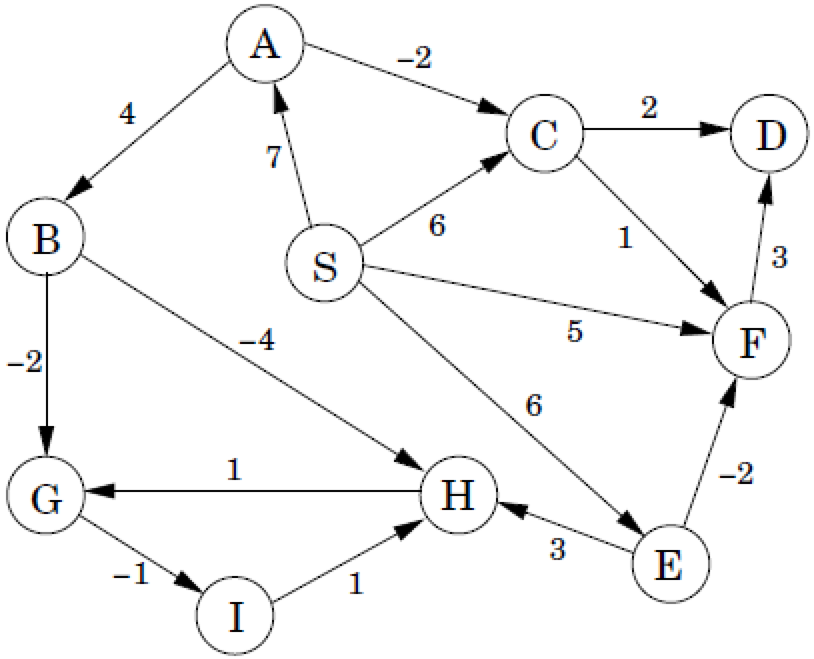
# Algorismes sobre grafs

- Exercicis
- Començant a A: dibuixa la taula de distàncies immediates a tots els nodes a cada iteració.
- Mostra l'arbre de camins mínims



# Algorismes sobre grafs

- Exercicis: el mateix amb Bellman-Ford



# Flujo Máximo

- Imaginemos algún flujo que va desde un sitio ***s***, donde es producido, hasta un sitio ***t*** donde es consumido a la misma tasa de producción.
- Intuitivamente, el flujo en cualquier punto de la red es la tasa a la que se mueve el material.
- Usos: modelado de flujo en tuberías, líneas de ensamblado, corrientes eléctricas, información en redes de comunicación (enroutamientos), optimización sobre datos matriciales, etc.
- Intuitivamente modelable con grafos.

# Flujo Máximo

- Cada arco dirigido puede ser visto como un conducto por donde pasa el material, según las siguientes restricciones:
  - Cada conducto tiene una capacidad máxima finita (>=0).
  - Se cumple la conservación de flujo. $\Sigma f_{input} = \Sigma f_{output}$ (por nodo).

Problema del Flujo Máximo:

¿Cuál es la mayor tasa a la que se puede llevar material sin violar ninguna restricción?

# Redes de Flujo

- Una red de flujo $G=(V,E)$ es un grafo dirigido tal que cada arco $(u,v)\in E$ posee una capacidad $c(u,v)\geq 0$. Si $(u,v)\notin E$, $c(u,v)=0$.
  - Se distinguen 2 vértices, el fuente "***s***" y el destino "***t***". Cada vértice $\boldsymbol{v}\in\boldsymbol{V}$ está en algún $\boldsymbol{s \sim> v \sim> t}$.
  - El grafo es conexo $\Rightarrow |\boldsymbol{E}|\geq|\boldsymbol{V}|-\boldsymbol{1}$

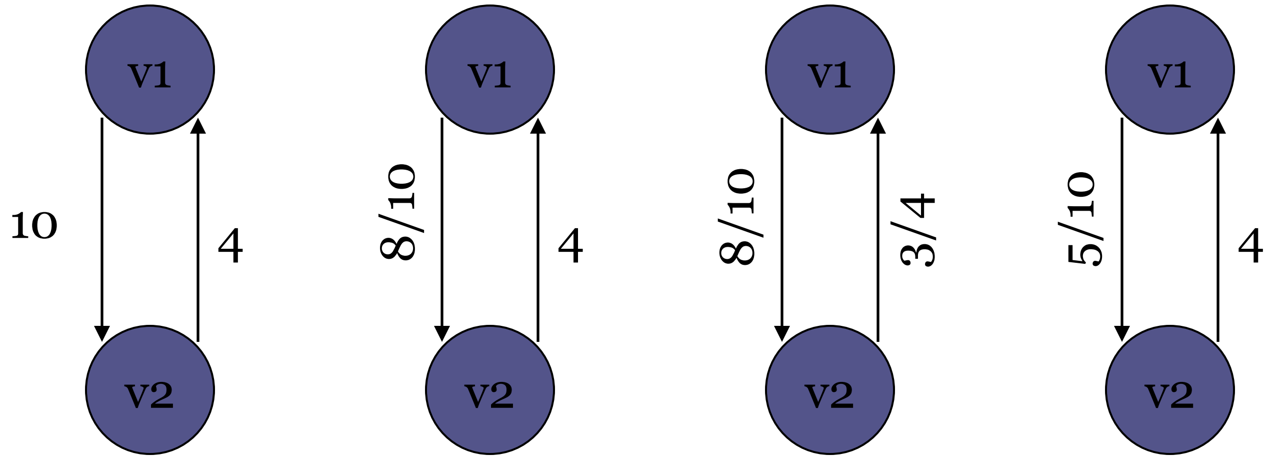
    → *Sino algunos elementos no participarán en el transporte del flujo de **s** a **t***
- El flujo está dado por una función con imagen en los reales. $f:VxV\rightarrow R$ que satisface:
  - Restricción de capacidad: $\forall u,v \in V, f(u,v)\leq c(u,v)$.

    → *El flujo máximo que puede pasar por una conexión está limitado por la capacidad de la conexión*
  - Simetría distorsionada: $\forall u,v \in V, f(u,v)=-f(v,u)$.

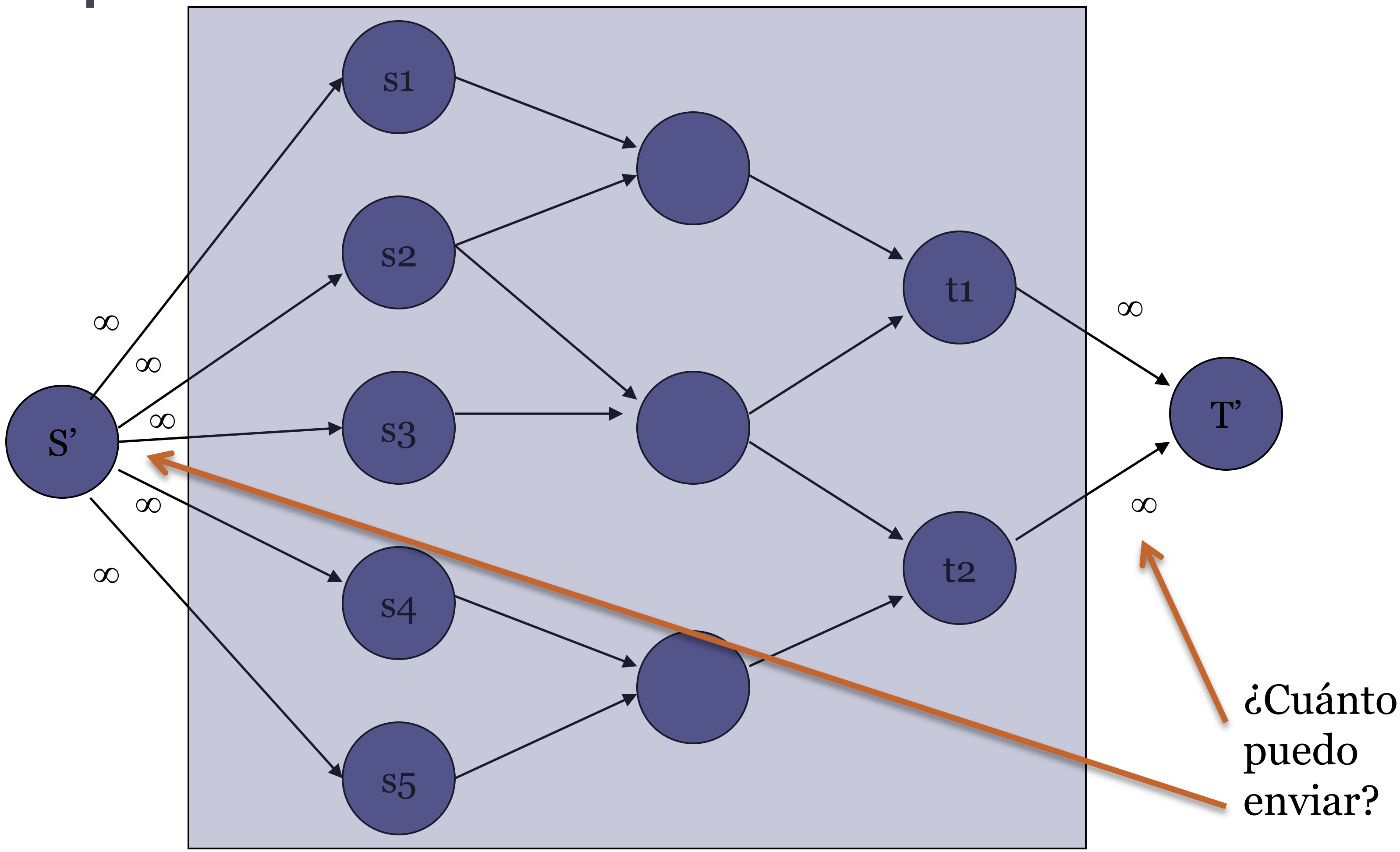
    → *El flujo es direccional*

# Relación entre 2 nodos

8 de v1 a v2 | 3 de v2 a v1 | cancelación



# Redes de múltiples entradas y múltiples salidas



# Método Ford-Fulkerson

- El método iterativo depende de tres ideas importantes:
  - **Red residual**
  - **Aumento de camino**
  - **Cortes**

Para ello usaremos el teorema:

**max-flow/min-cut** que caracteriza el flujo máximo en términos de cortes de la red de flujo.

# Teorema max-flow min-cut

**Teorema**: El máximo valor de entre todos los flujos que llegan a ***t*** en una red es igual a la *capacidad mínima* de entre todos los cortes que dividen la red.

**Prueba**: Es suficiente con mostrar un flujo y un corte tal que sean iguales en valor. Luego, el flujo ha de ser máximo pues no puede rebasar la capacidad del corte y el corte ha de ser mínimo porque ninguna otra capacidad puede ser menor que el valor actual del flujo.

**Objetivo**: Saturar la red para satisfacer el teorema!!!

# Iteración

- **En cada iteración se va consiguiendo un valor de flujo que aumenta el camino, es decir, podemos aumentar el flujo en un camino de *s* a *t*. Este proceso se repite hasta que no haya más posibilidad de aumentar.**

FORD-FULKERSON-METHOD(*G*,*s*,*t*)
1 Inicializar el flujo de toda conexión a 0
2 Mientras exista un camino de acceso por flujo ***p*** de ***s*** a ***t*** (*path*)
3 Aumentar el flujo f encontrado a través de todo el camino ***p***
4 *return f*.

# Red Residual

La red residual consiste en arcos que admiten más flujo. Dada una red de flujo $G=(V,E)$ con fuente $\boldsymbol{s}$ y destino $\boldsymbol{t}$. Sea $f$ el flujo en $G$, y considere un par de vértices $u,v \in V$, la cantidad de flujo adicional que se puede verter sobre $u,v$ es la **capacidad residual**.

$c_f(u,v) = c(u,v) - f(u,v)$

Ejemplo:

$c(u,v)=16,\ f(u,v)=10 \Rightarrow c_f(u,v)=6$

Capacidad residual conexión (u,v) en paso 1

Capacidad residual conexión (u,v) en paso 2

# Aumento de Caminos

- Dada una red de flujo $G=(V,E)$, un camino aumentado $p$ es un camino simple de $s$ a $t$ en la red residual $G_f$. Este camino sólo admite flujo positivo.
- La cantidad máxima de flujo que puede llevarse por los arcos en un aumento de $p$ se denomina **capacidad residual** de $p$, y está dado por:

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

- **¿Cómo creéis que lo podemos hacer?**

# Corte de la red de flujo

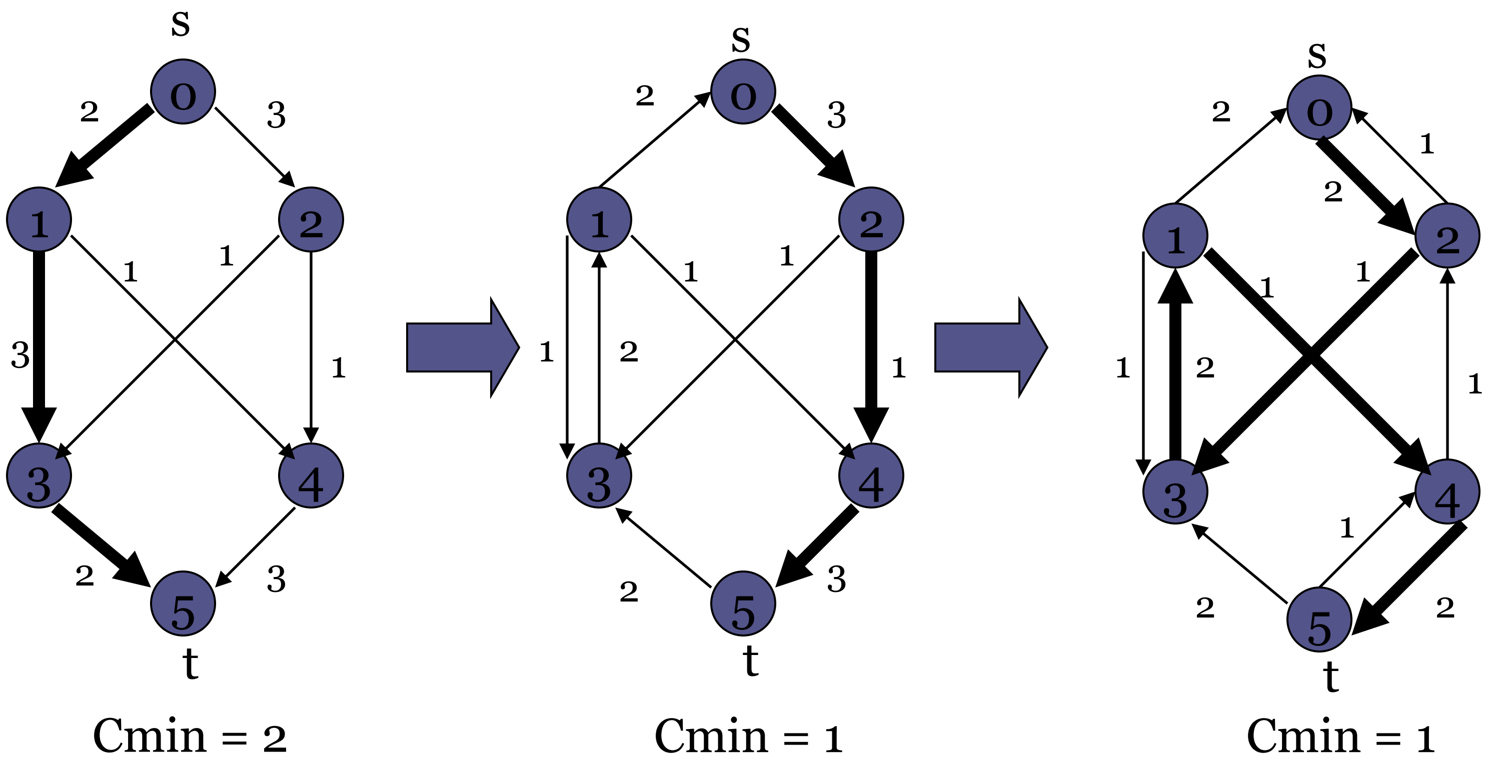
- El método aumenta repetidamente el flujo a través de los caminos de aumento $p$ hasta alcanzar el máximo.

- Un corte $(S,T)$ de la red de flujo $G=(V,E)$ es una partición de $V$ en $S$ y $T=V-S$ tal que $s \in S$ y $t \in T$. Si $f$ es el flujo, entonces **el flujo de red a través del corte $(S,T)$ se define $f(S,T)$. La capacidad del corte $(S,T)$ es $C(S,T)$.**

# Algoritmo de Ford-Fulkerson

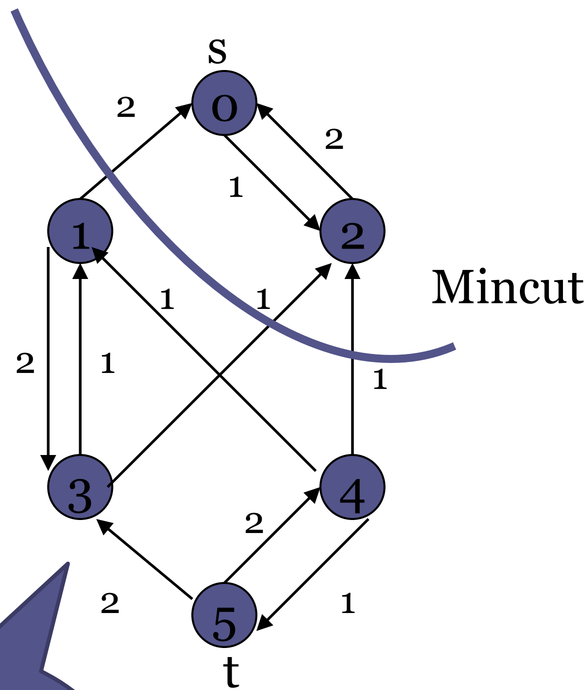
FORD-FULKERSON(G,s,t)<br>
**for** cada arista $(u,v) \in E[G]$<br>
**do** $f[u,v] \leftarrow 0$ , $c_f(u,v)=c(u,v)$<br>
$f[v,u] \leftarrow 0$ , $c_f(v,u)=c(v,u)$<br>
**while** existe camino *p* de *s* a *t* en la red residual $G_f$<br>
**do** $c_f(p)=\min\{c_f(u,v) : (u,v) \in p\}$<br>
**for** cada arista (u,v) en *p*<br>
**do** $f[u,v] \leftarrow f[u,v] + c_f(p)$<br>
$f[v,u] \leftarrow -f[u,v]$<br>
$c_f(u,v)=c_f(u,v)-c_f(p)$<br>
$c_f(v,u)=c_f(v,u)+c_f(p)$

# Ejemplo



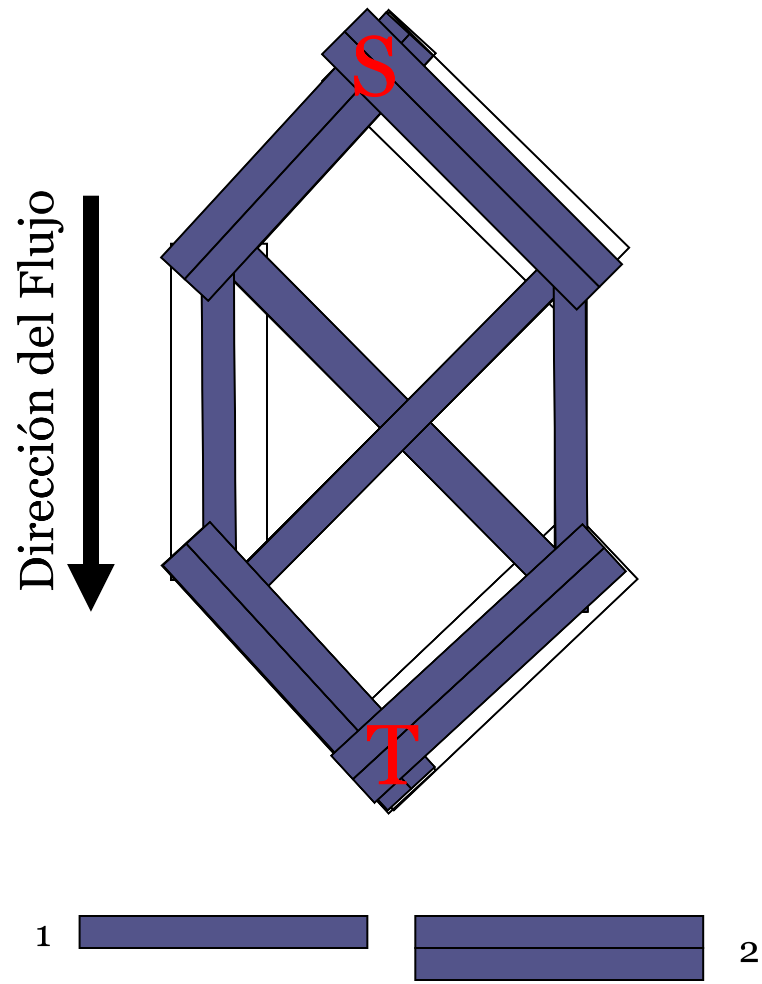

**Path *p*: p.e. usando BFS (sin criterio de ordenación de vértices)**

# Ejemplo



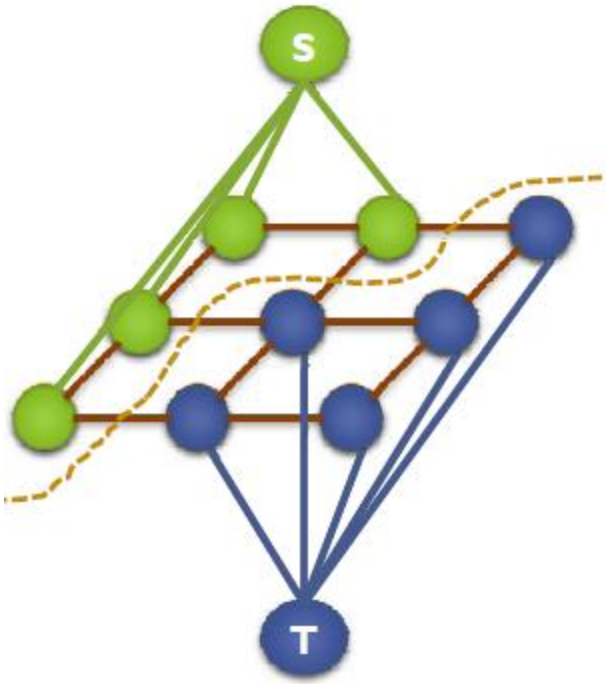

Maxflow
es 2+2=4

Flujo Máximo

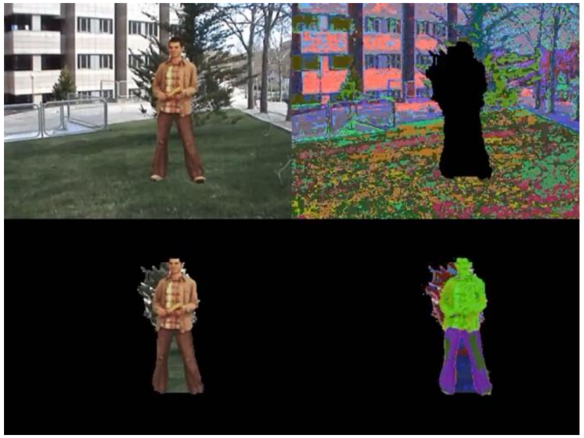


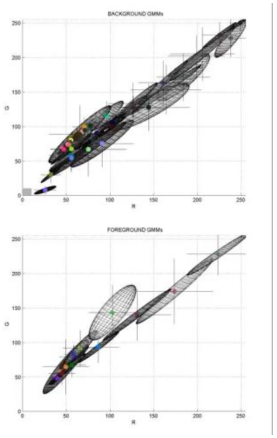
# Aplicaciones

- Segmentación de imágenes – Ford-Fullkerson




“Graph Cut”
-Imagen como un grafo
-Los costes son probabilidades de pertenecer a uno de dos posibles grupos
-La saturación define la segmentación aplicando Ford-Fulkerson





# Algorismes sobre grafs

- Exercici
- Aplica Ford-Fullkerson: identifica **min-cut** y **max-flow**