Flujo en redes
(*Maximum flow – Minimum cut*)

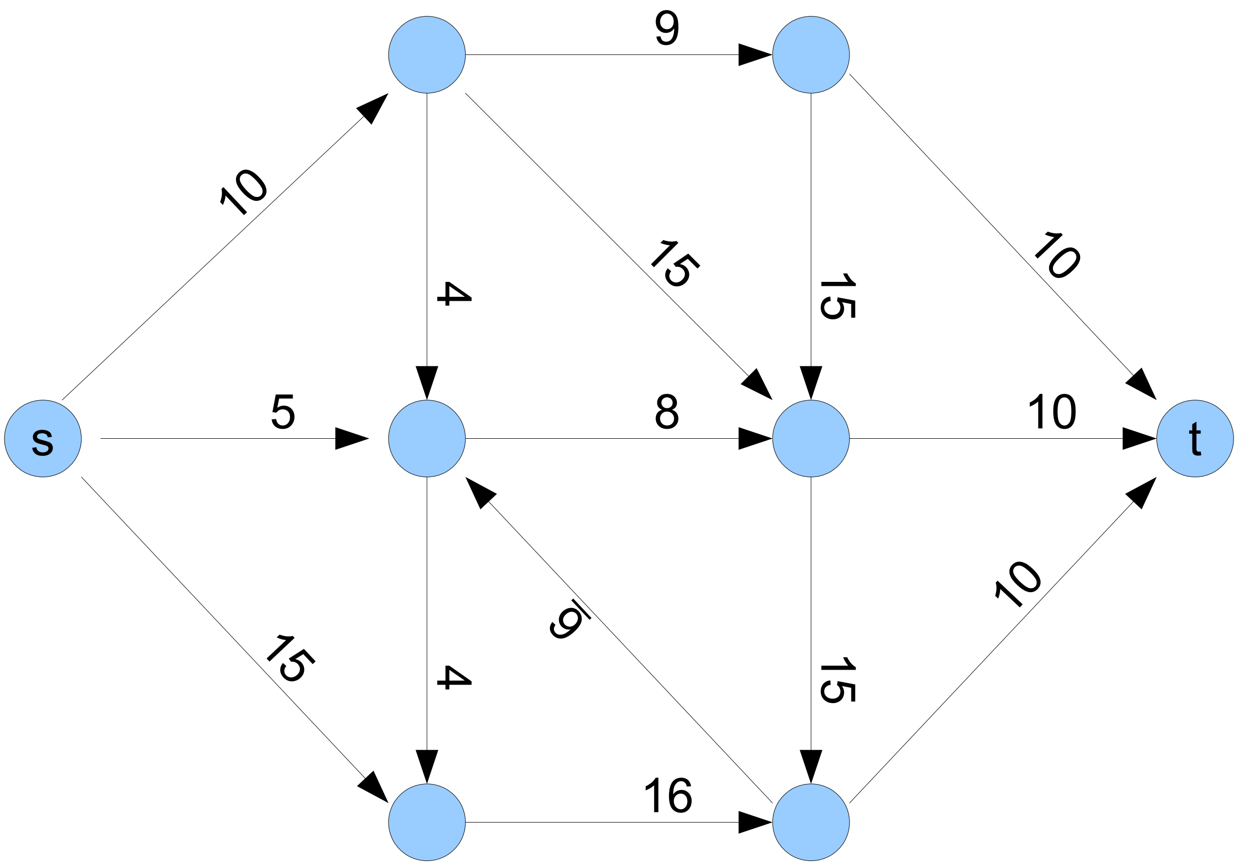
Algoritmo Ford-Fulkerson

Máximo flujo

Sea

- G=<V,E> un grafo conexo, dirigido.
- A cada arista se le asigna una capacidad positiva $c(e)\geq 0$.
- Se distinguen dos vértices:
  - Un origen o fuente del flujo: *s*
  - Un destino o desagüe del flujo: *t*
- Interesa maximizar el flujo total entre *s* y *t*.

Ejemplo



Flujos

- El flujo por la arista e se denota: f(e)
- El flujo siempre debe ser menor o igual a la capacidad de la arista:

$$f(e) \leqslant c(e)$$

- En todos los vértices *v* distintos a *s,t* la suma de flujos entrantes es igual a la suma de flujos salientes:

$$\sum_{e\ entra\ a\ v} f(e) = \sum_{e\ sale\ de\ v} f(e)$$

"Ley de conservación de flujos"

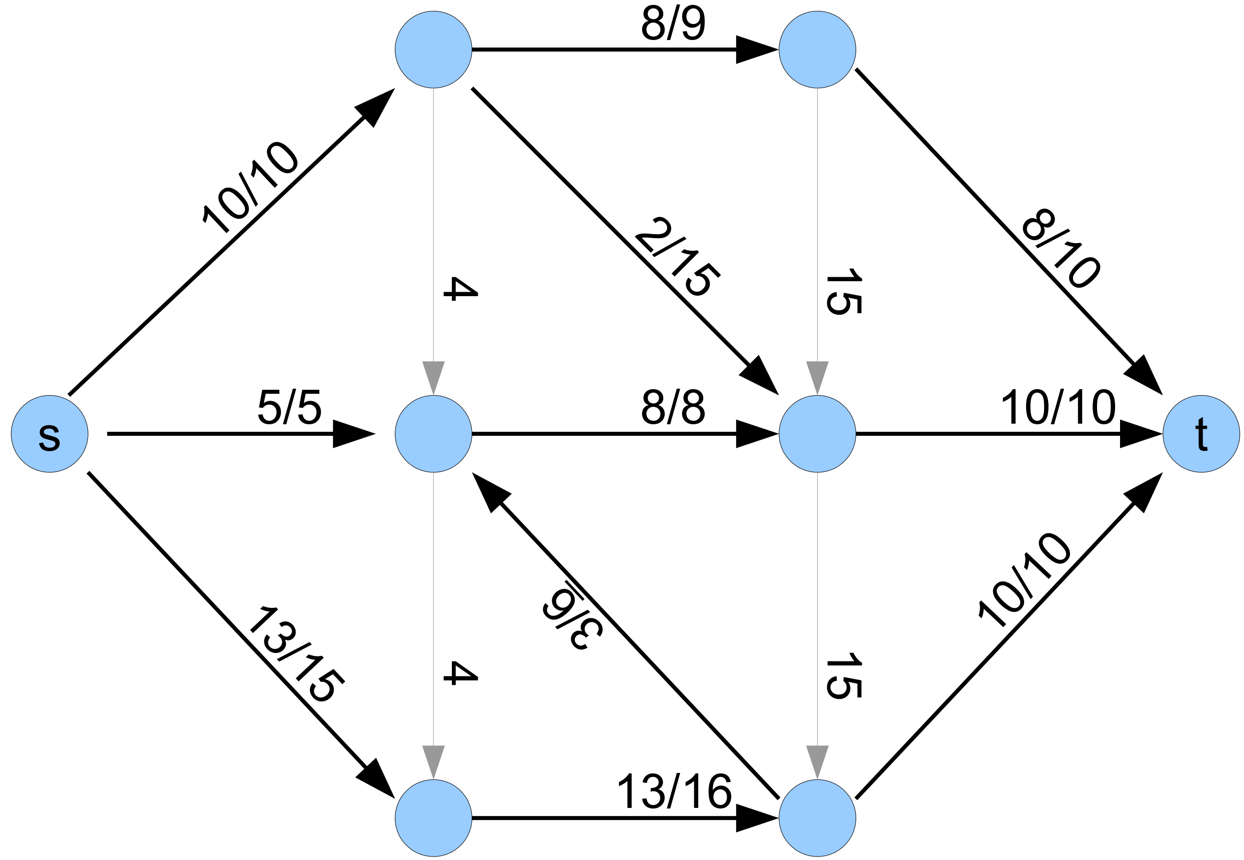
Valor del flujo

- El flujo total se puede medir como el flujo que sale de *s* o el flujo entrante a *t*.

$$flujo = \sum_{e \, sale \, de \, s} f(e)$$

- El objetivo es maximizar este flujo

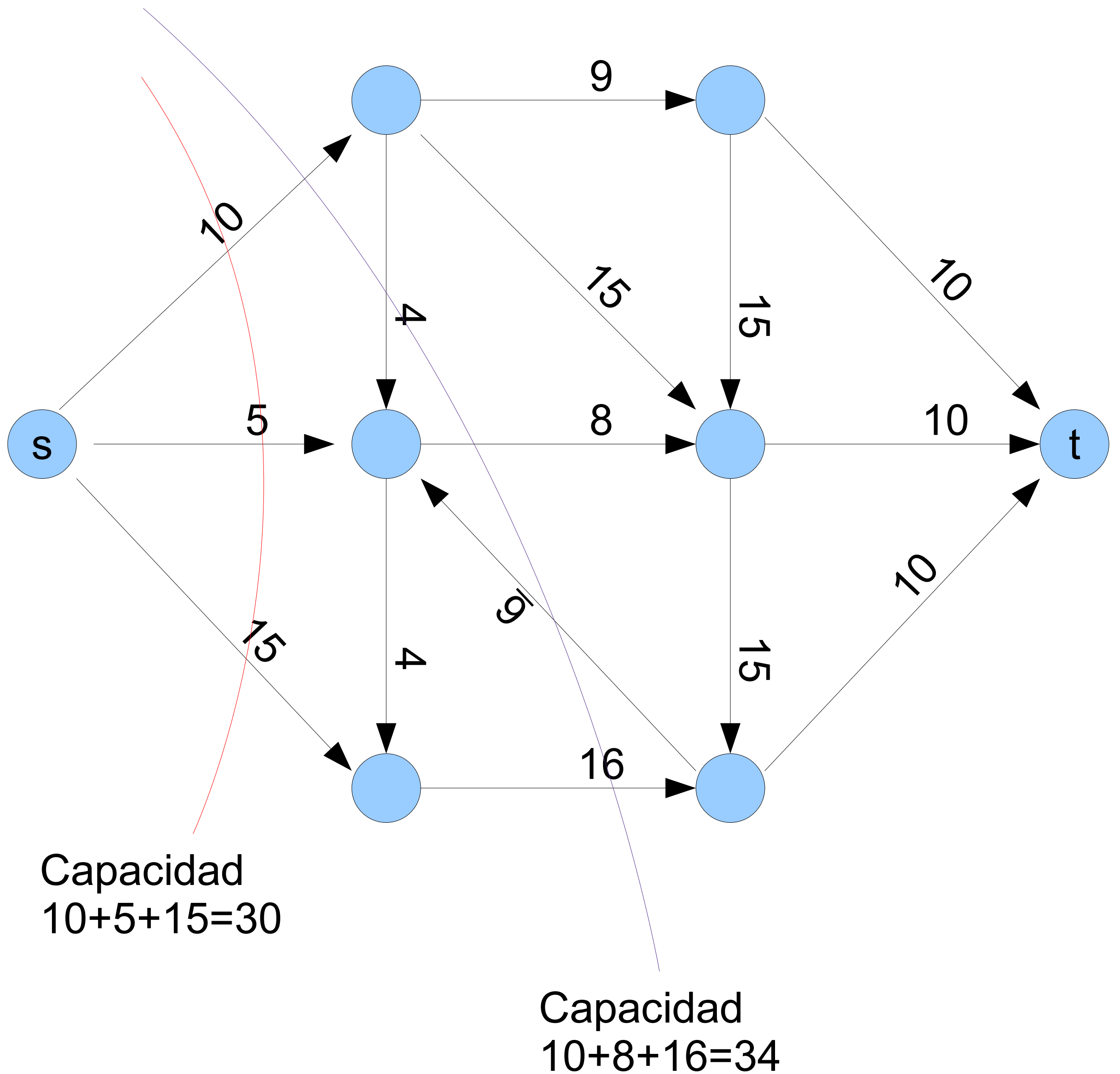
Ejemplo de flujo máximo



Corte mínimo

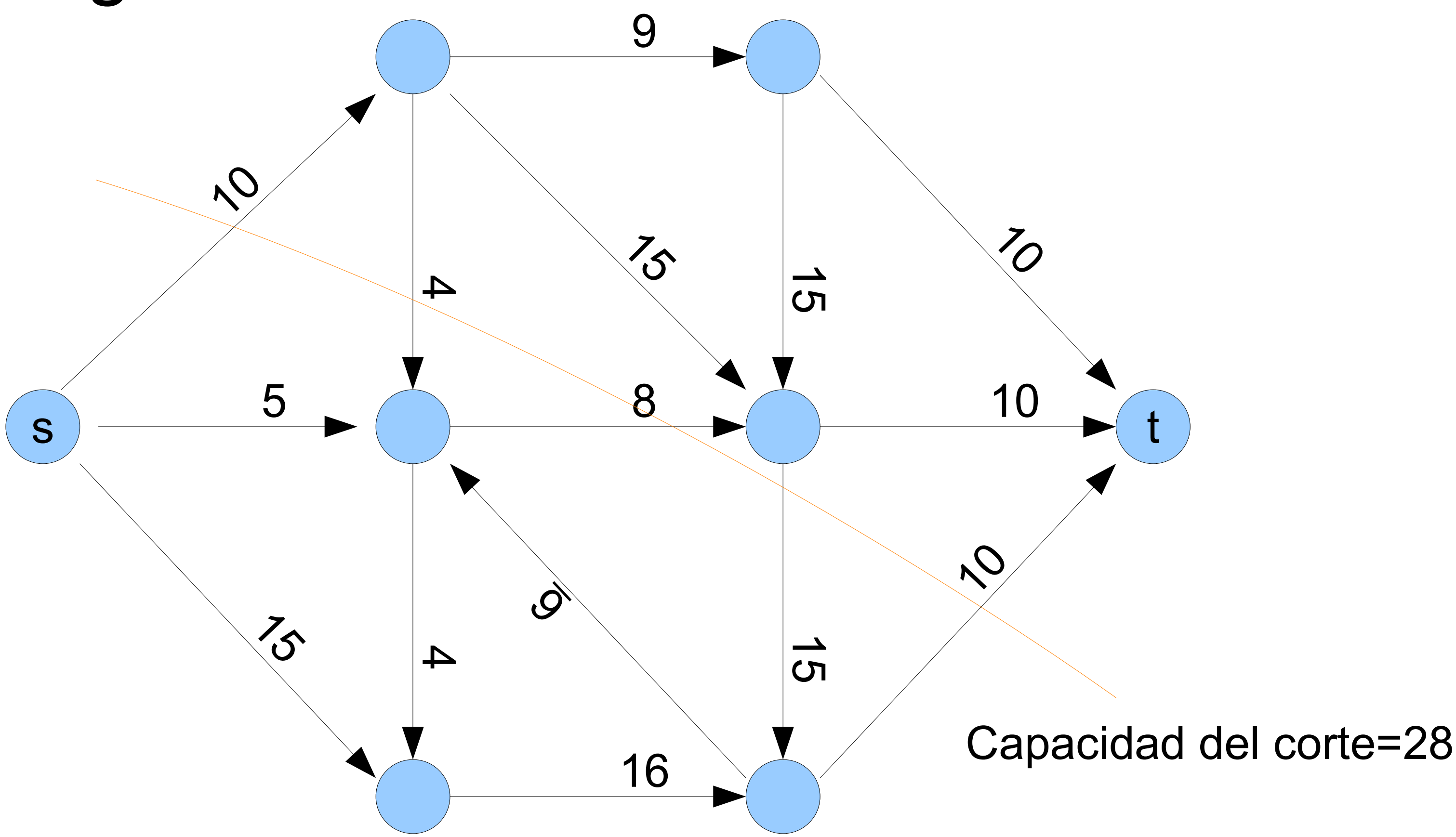
- Un corte es una partición (A,B) de los vértices tal que $s \in A$ y $t \in B$.
- La capacidad de la partición es la suma de las **aristas salientes de A.**

Ejemplos


9
10
4
15
15
10
s
5
8
10
t
15
4
9
15
10
16
Capacidad
10+5+15=30
Capacidad
10+8+16=34

Problema del corte mínimo

- Encontrar el corte de menor capacidad en el grafo, *e.g.*

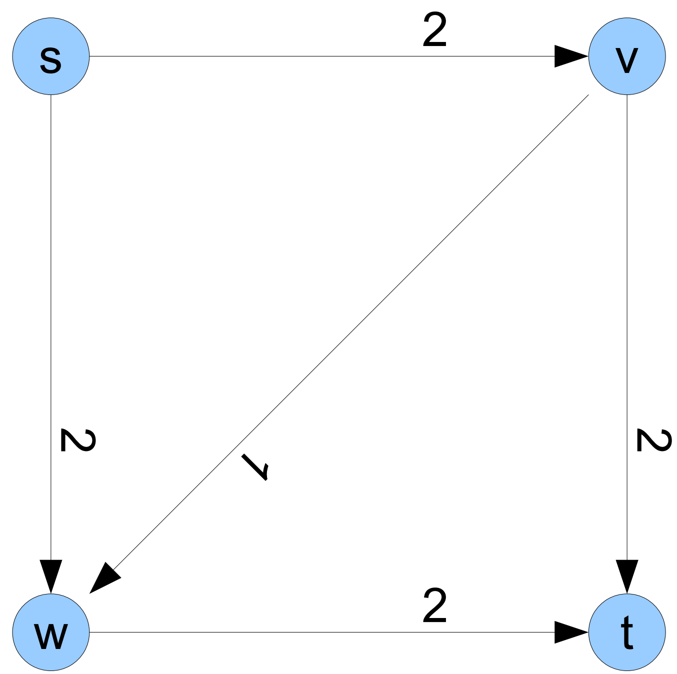


Hacia una solución al problema del máximo flujo

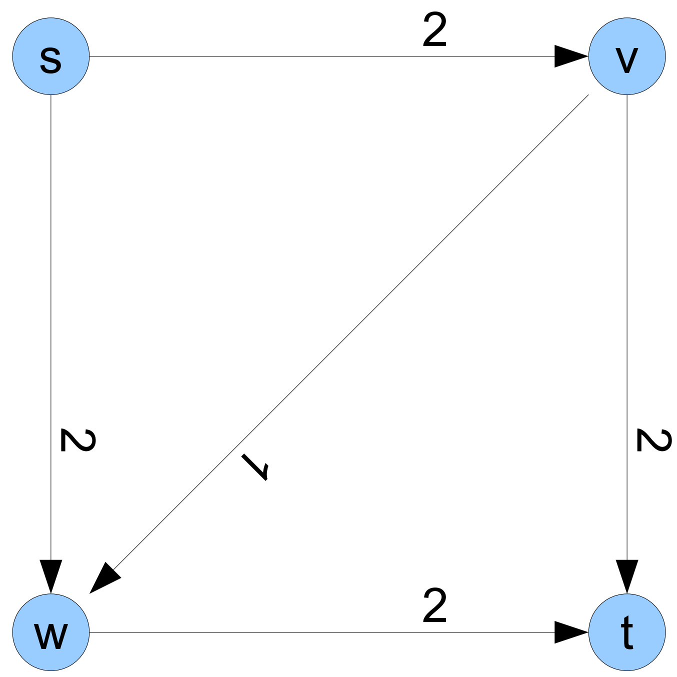
Un algoritmo voraz:

1) Inicializar $f(e)=0$ para todas las aristas.
2) Encontrar un camino P entre s-t donde todas las aristas tengan $f(e)<c(e)$.
3) Aumentar el flujo a lo largo del camino.
4) Repetir 2 hasta no encontrar caminos factibles.

Un ejemplo



Un contra-ejemplo



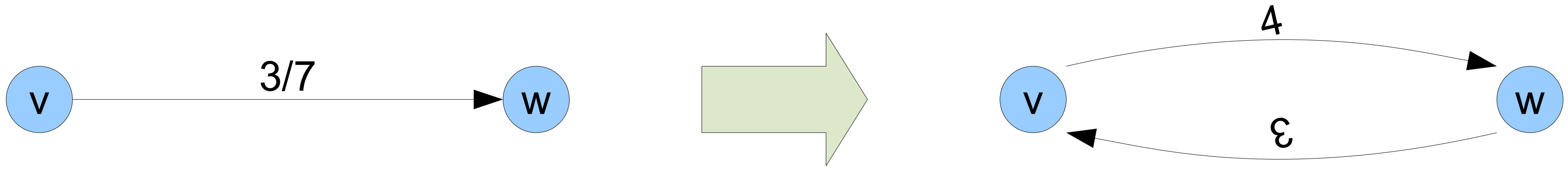
La estrategia voraz no encuentra la solución óptima

- Una característica de todo algoritmo voraz es que una decisión una vez tomada, no se cambia.
- Una mala decisión nos lleva a una situación donde no es posible encontrar el óptimo.

Algoritmo de Ford-Fulkerson

- Se apoya en el concepto de la “red residual”:
- Si por una arista de capacidad c circula un flujo f, se reemplaza por dos aristas:
  - Una en sentido directo de capacidad c-f
  - Una en sentido inverso de capacidad f.

Red residual



Caminos aumentantes (*Augmenting paths*)

- Son caminos P entre s-t sobre la red residual.
- La capacidad “cuello de botella” δ es la menor de las capacidades residuales en el camino P.
- Se define el proceso de “aumentación” como el incremento del flujo en el valor de la capacidad cuello de botella.
- El flujo por las aristas originales se incrementa en δ y por las arista reversas en $-\delta$.

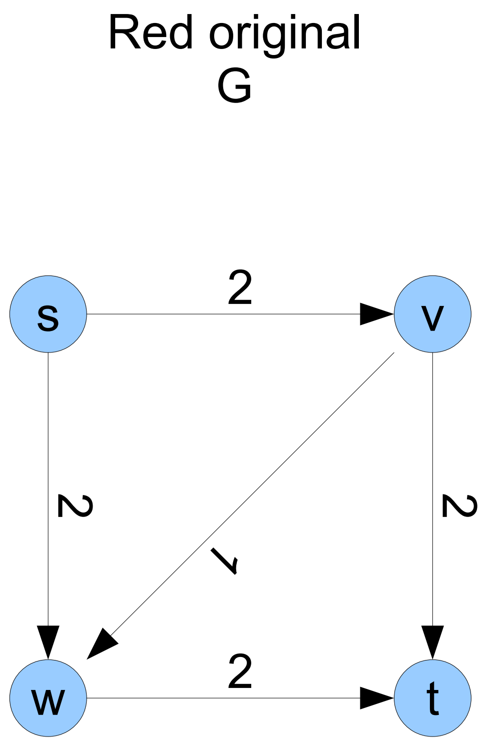
Proceso de aumentación

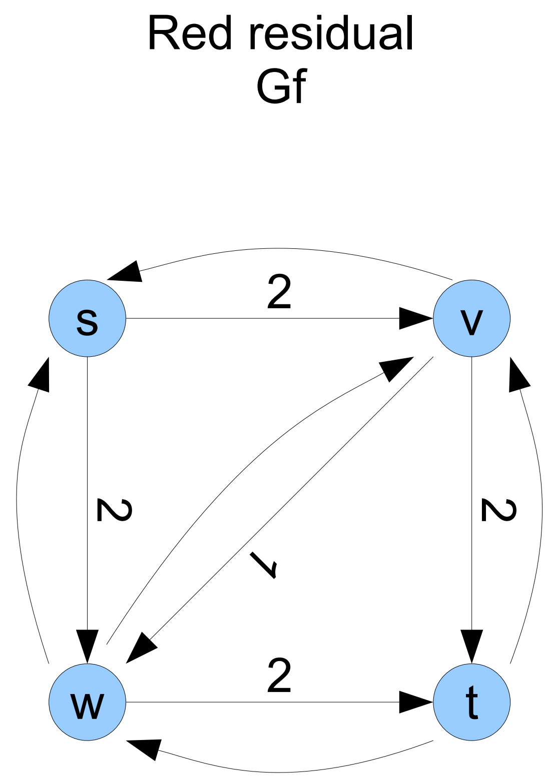
```
Aumentacion(f,c,P):
   δ ← Capacidad “cuello de botella” del camino P
   Para cada e en P:
      if e ∈ E:
         f(e) ← f(e) + δ
      else:
         f(e’) ← f(e’) - δ
   Return f
```

Ford-Fulkerson

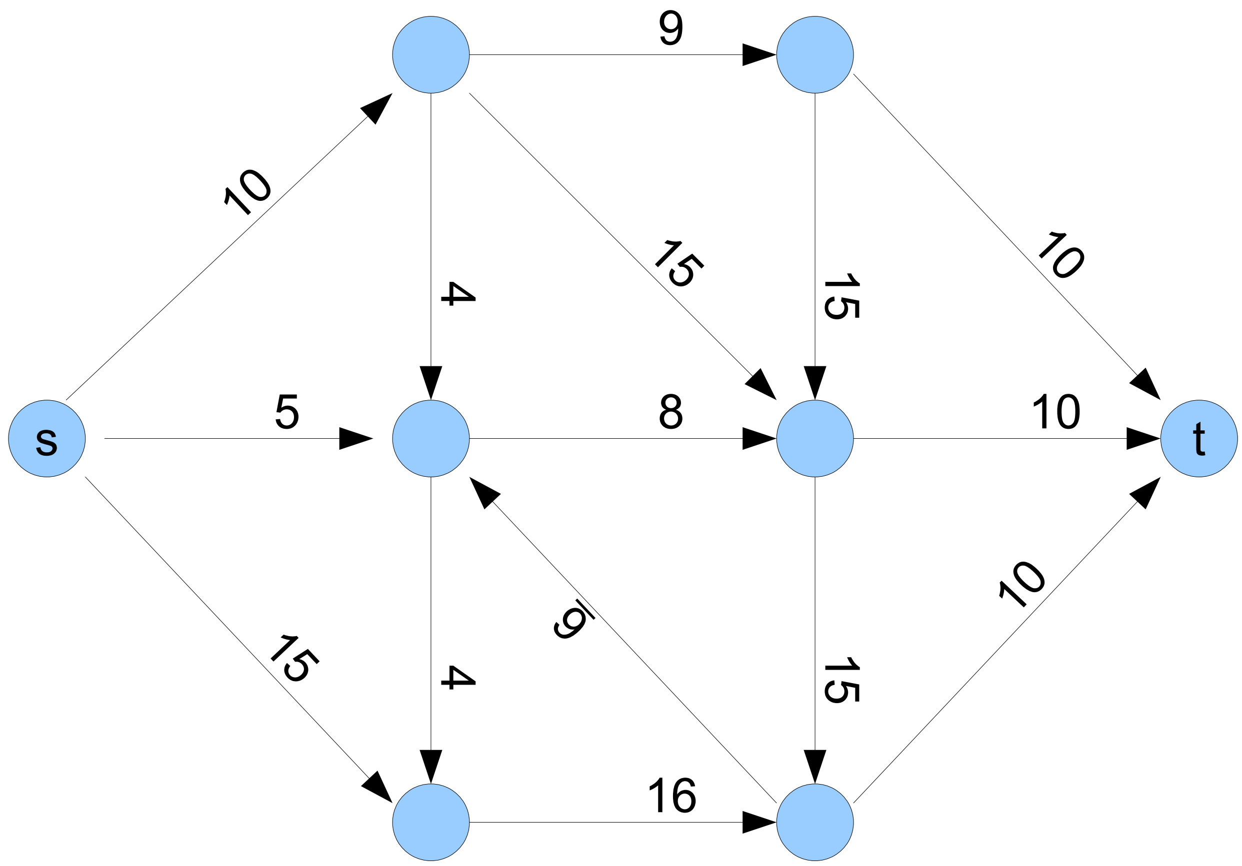
```
Ford-Fulkerson(G):
   Para cada e∈E:
      f(e) ← 0

   Gf ← Red residual (aristas inversas f(e’)←0)

   Mientras existe P entre s-t en Gf:
      f ← Aumentación(f,c,P)
      Actualizar Gf

   Return f
```

Volvamos al caso problema:





Retomando el ejemplo



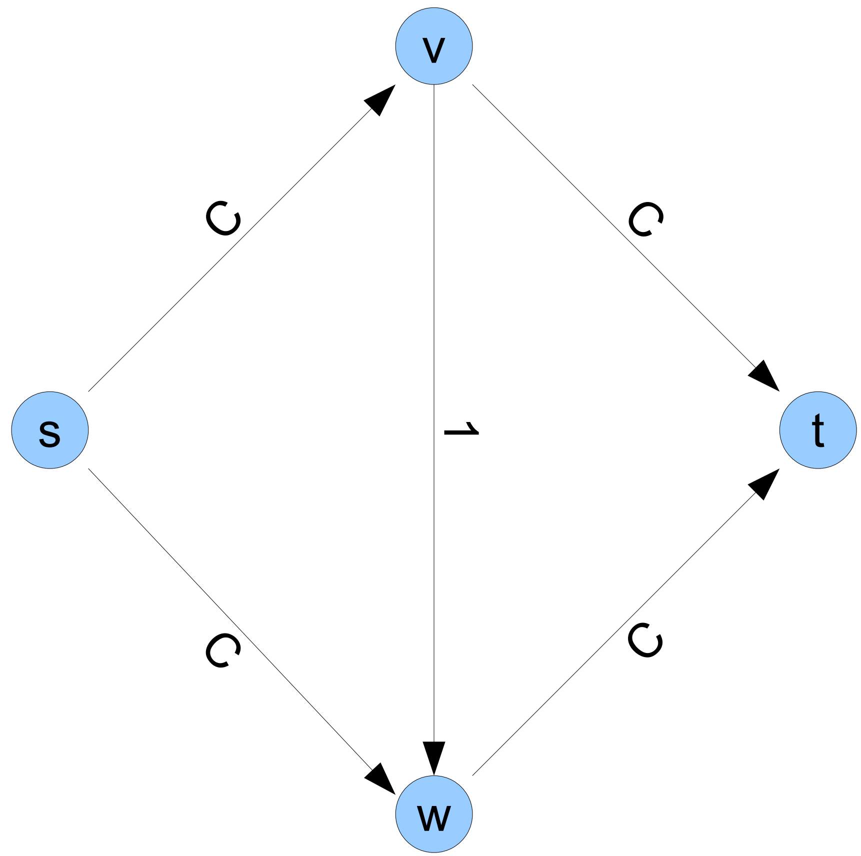
Teorema del flujo máximo y corte mínimo

- Observamos que el flujo total siempre coincide con el flujo de cualquier corte.
- Cuando en un corte la capacidad residual es cero, significa que no existe un camino que permita aumentar el flujo.
- Por tanto, el menor de los cortes limita el máximo flujo posible en la red.

Análisis de Ford-Fulkerson

- Se asume: Capacidades $\in \mathbb{Z}$.
- Observación: Cada que se encuentra un camino, el flujo aumenta en un valor entero.
- Como mínimo el flujo aumenta en 1 cada aumentación. Cada aumentación es de tiempo constante.
- Para encontrar el camino aumentante se puede utilizar DFS: ~V
- Sin embargo el número de aumentaciones puede ser ~ kC, siendo C la capacidad del corte mínimo.

Peor caso para Ford-Fulkerson



Mejoras a Ford-Fulkerson

Hay muchas estrategias que buscan mejorar el desempeño de peor caso:

- Capacity-scaling algorithm,
- Shortest aumenting paths,
- Dinitz' algorithm,
- y más...

Bibliografía

Kleinberg & Tardos.

Algorithm Design, Cap. 7