Problemas de flujo en redes

Tecnología Digital V: Diseño de Algoritmos
Universidad Torcuato Di Tella

Datos de entrada

1. Un grafo dirigido $G = (N, A)$.
2. Nodos $s, t \in N$ de origen y destino.
3. Una función de capacidad $u : A \longrightarrow \mathbb{Z}_+$ asociada con los arcos.

Flujo en redes

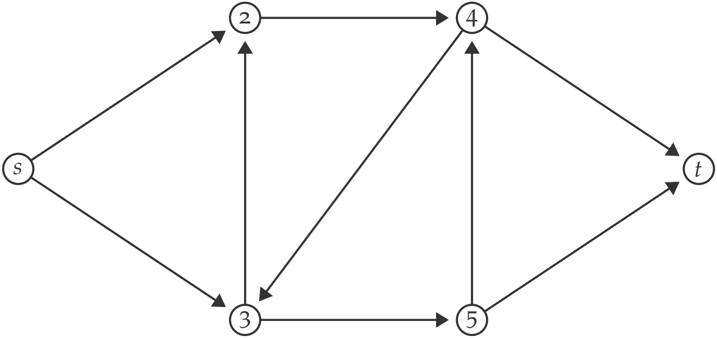
Datos de entrada

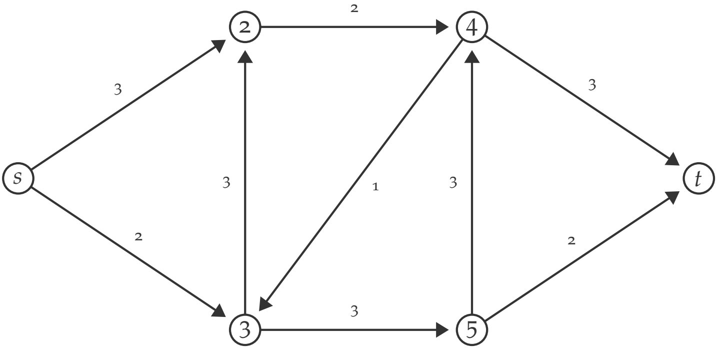
1. Un grafo dirigido $G = (N, A)$.
2. Nodos $s, t \in N$ de origen y destino.
3. Una función de capacidad $u : A \longrightarrow \mathbb{Z}_+$ asociada con los arcos.

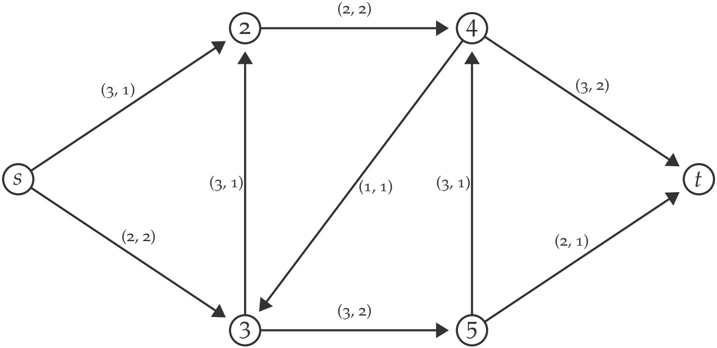
Problema

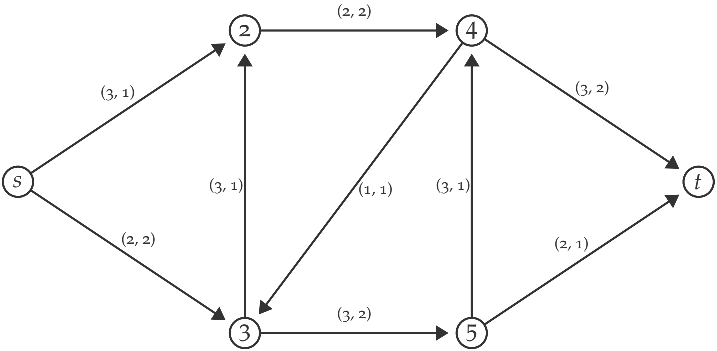
Encontrar un flujo (cantidad a enviar por cada arco) entre s y t de mayor valor posible.

1. Salvo s y t, en cada nodo la cantidad de flujo que entra al nodo debe ser igual a la cantidad de flujo que sale del nodo.
2. La cantidad x_{ij} enviada por el arco $ij \in A$ debe cumplir $0 \leq x_{ij} \leq u_{ij}$.
3. El valor de un flujo es la cantidad de flujo neto que sale de s.


2
4
s
t
3
5

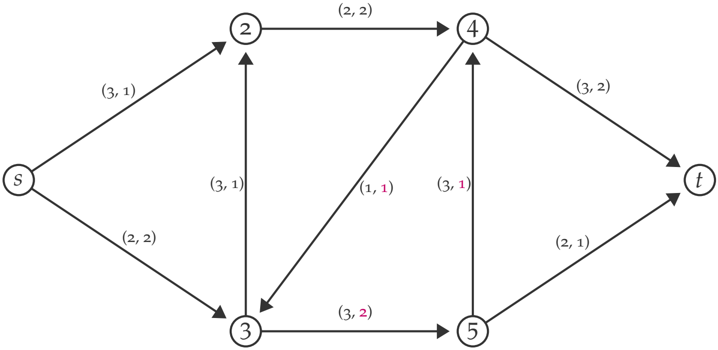

2
4
3
3
s
3
1
3
t
2
2
3
3
5

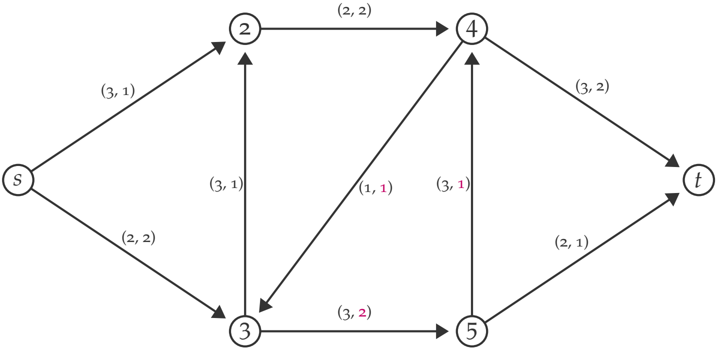

s
2
3
4
5
t
(3, 1)
(2, 2)
(2, 2)
(3, 1)
(1, 1)
(3, 2)
(3, 1)
(3, 2)
(2, 1)



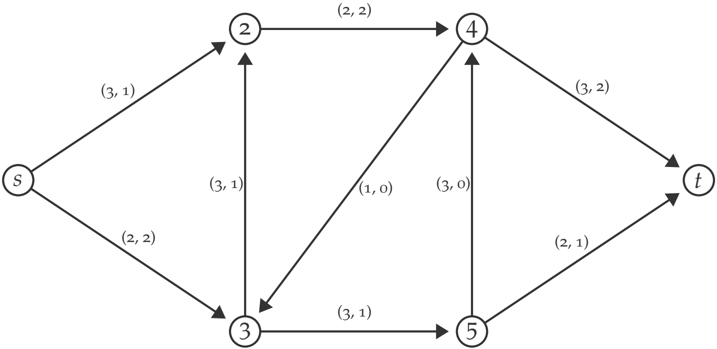
- $F = 3$
- flujo se conserva en cada nodo que no es s o t

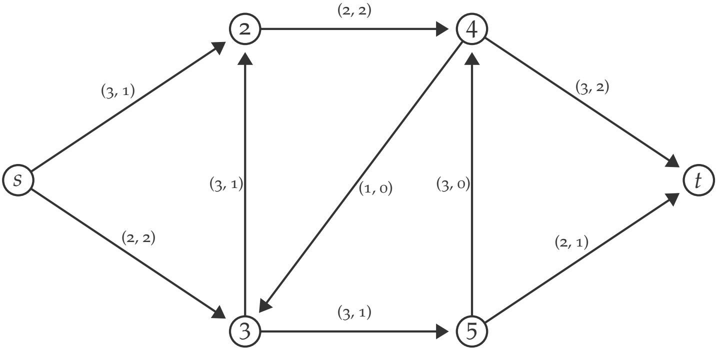
Flujo en redes



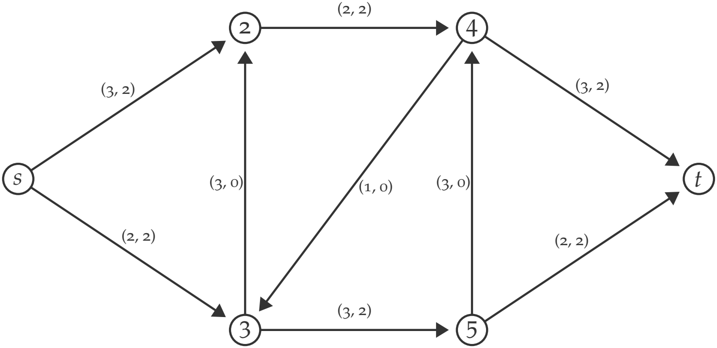


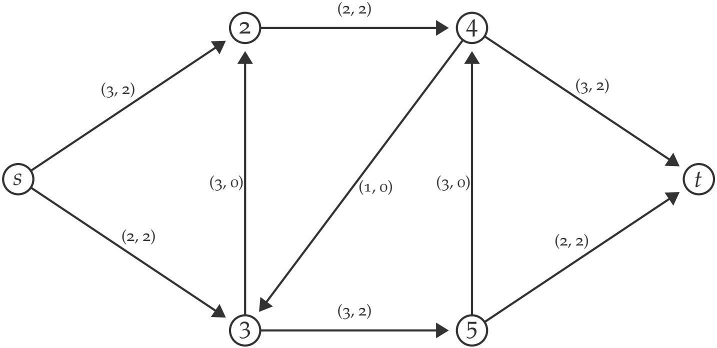
- Podemos eliminar flujo en ciclo


(2, 2)
2
4
(3, 1)
(3, 2)
s
(3, 1)
(1, 0)
(3, 0)
t
(2, 2)
(2, 1)
(3, 1)
3
5

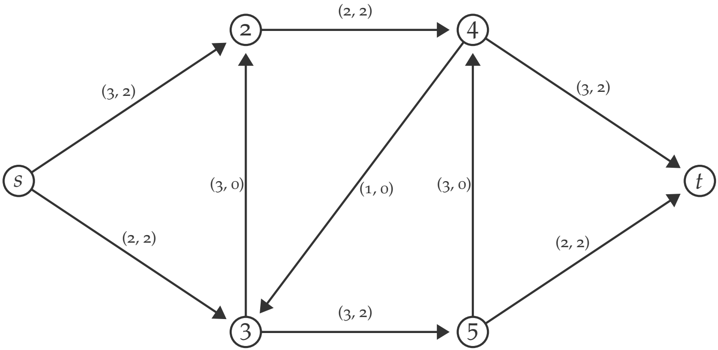


- Podemos aumentar el flujo? Cómo?

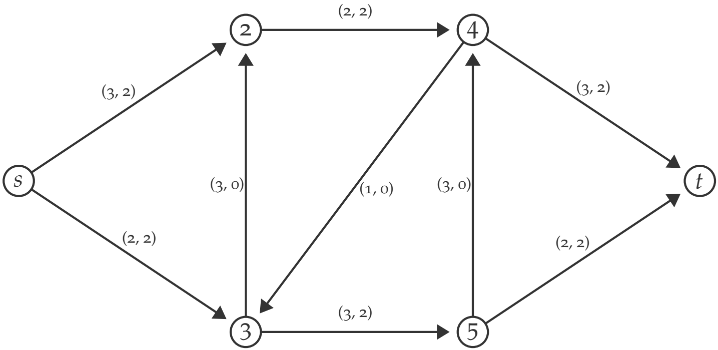

(2, 2)
2
4
(3, 2)
(3, 2)
s
(3, 0)
(1, 0)
(3, 0)
t
(2, 2)
(2, 2)
(3, 2)
3
5



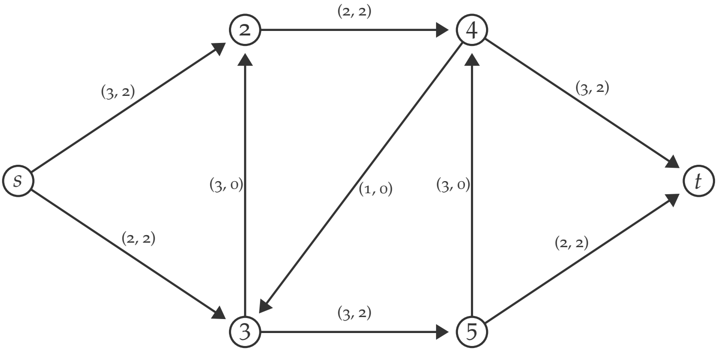
- $F = 4$ es el flujo máximo



- $F = 4$ es el flujo máximo
- Necesitamos:



- $F = 4$ es el flujo máximo
- Necesitamos:
  - un algoritmo sistemático para encontrar el flujo máximo



- $F = 4$ es el flujo máximo
- Necesitamos:
  - un algoritmo sistemático para encontrar el flujo máximo
  - una forma de comprobar que encontramos el flujo máximo

- Un corte en la red $G = (N, A)$ es un subconjunto $S \subseteq N \setminus \{t\}$ tal que $s \in S$.

- Un corte en la red $G = (N, A)$ es un subconjunto $S \subseteq N \setminus \{t\}$ tal que $s \in S$.
- Dados $S, T \subseteq N$, definimos $ST = \{ij : i \in S \text{ y } j \in T\}$

- Un corte en la red $G = (N, A)$ es un subconjunto $S \subseteq N \setminus \{t\}$ tal que $s \in S$.
- Dados $S, T \subseteq N$, definimos $ST = \{ij : i \in S \text{ y } j \in T\}$

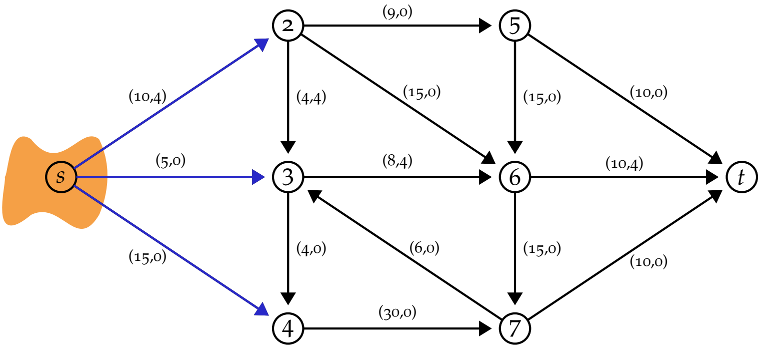
Proposición

Sea x un flujo definido en una red $G = (N, A)$ y sea S un corte. Entonces

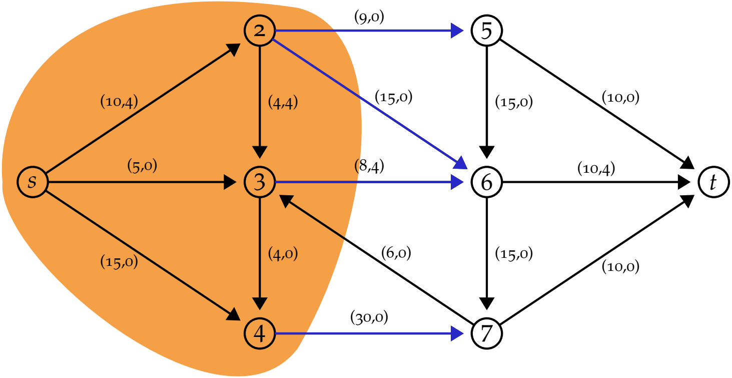
$$F = \sum_{ij \in S\bar{S}} x_{ij} - \sum_{ij \in \bar{S}S} x_{ij}$$

donde $\bar{S} = N \setminus S$.

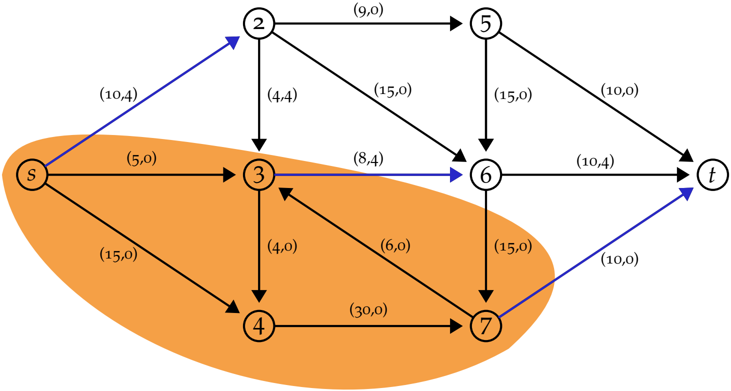
$$F = 4$$



$$F = 4$$



$$F = 4$$



- La capacidad de un corte S se define como

$$u(S) = \sum_{ij \in S\bar{S}} u_{ij}.$$

- La capacidad de un corte S se define como

$$u(S) = \sum_{ij \in S\bar{S}} u_{ij}.$$

Proposición

Si x es un flujo con valor F y S es un corte en N, entonces $F \leq u(S)$.

- La capacidad de un corte S se define como

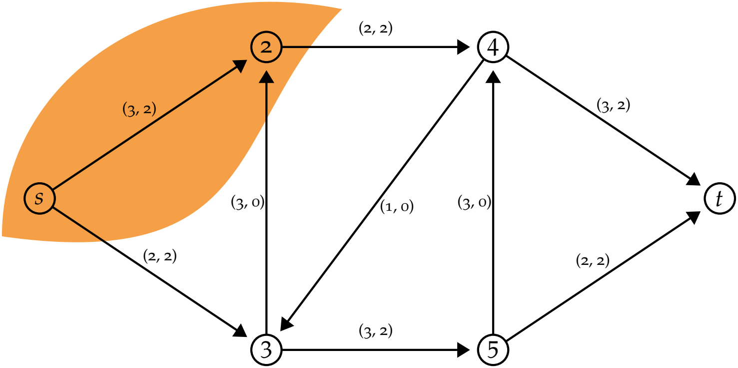
$$u(S) = \sum_{ij \in S\bar{S}} u_{ij}.$$

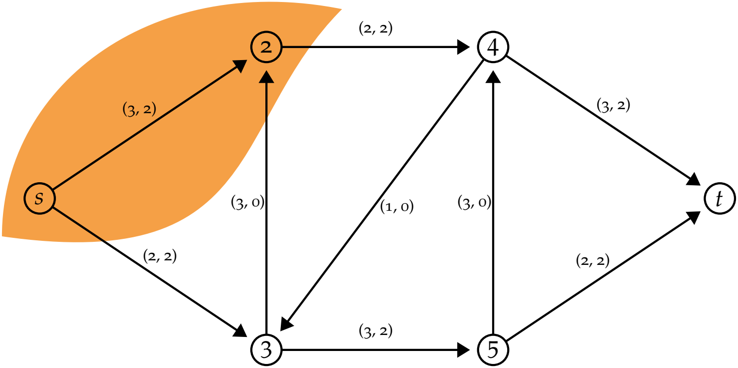
Proposición

Si x es un flujo con valor F y S es un corte en N, entonces $F \leq u(S)$.

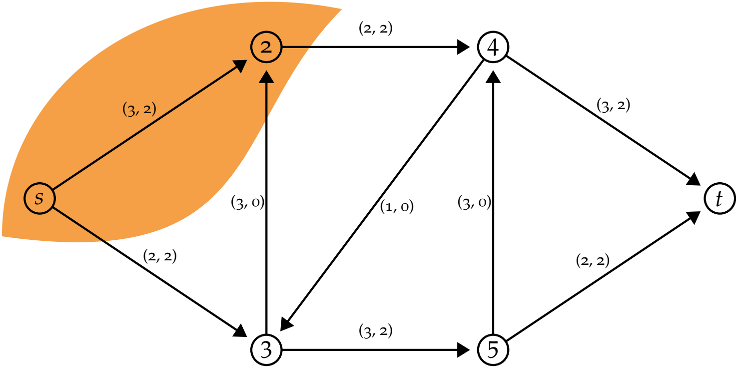
Corolario (certificado de optimalidad)

Si F es el valor de un flujo x y S un corte en G tal que $F = u(S)$ entonces x define un flujo máximo y S un corte de capacidad mínima.


(2, 2)
2
4
(3, 2)
(3, 2)
s
(3, 0)
(1, 0)
(3, 0)
t
(2, 2)
(2, 2)
(3, 2)
3
5

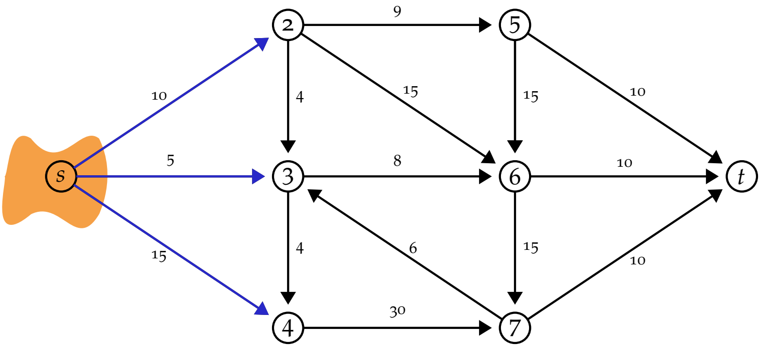


- $F = 4$
- $U = 4$

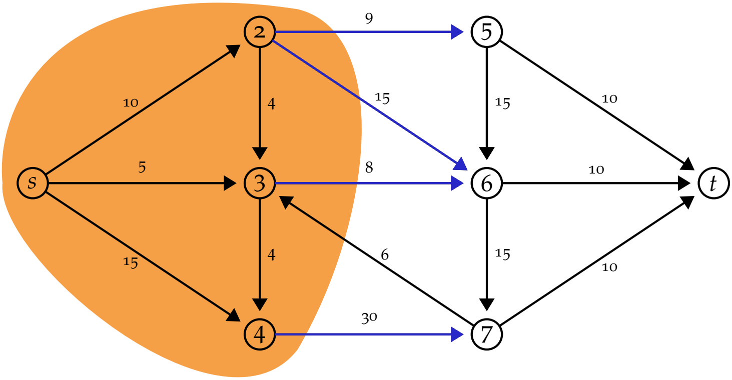


- $F = 4$
- $U = 4$
- $\Rightarrow F = 4$ es el flujo máximo.

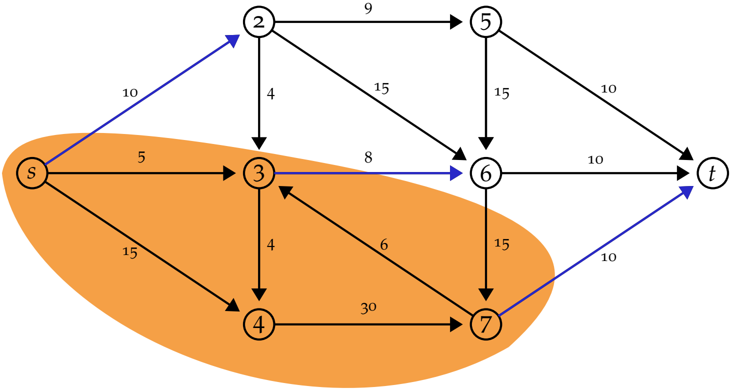
$$U = 30$$



$$U = 62$$

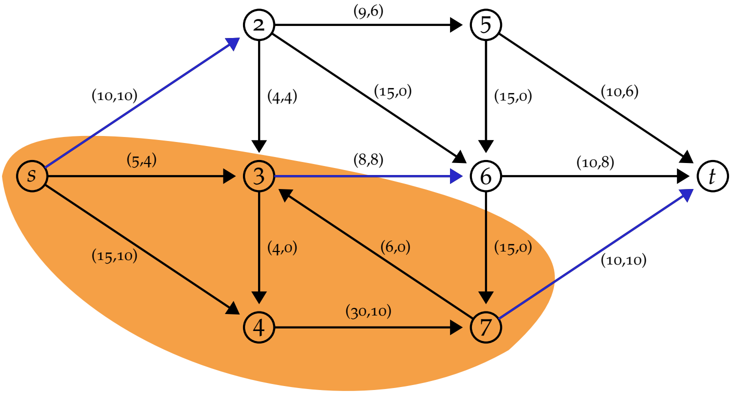


$$U = 28$$



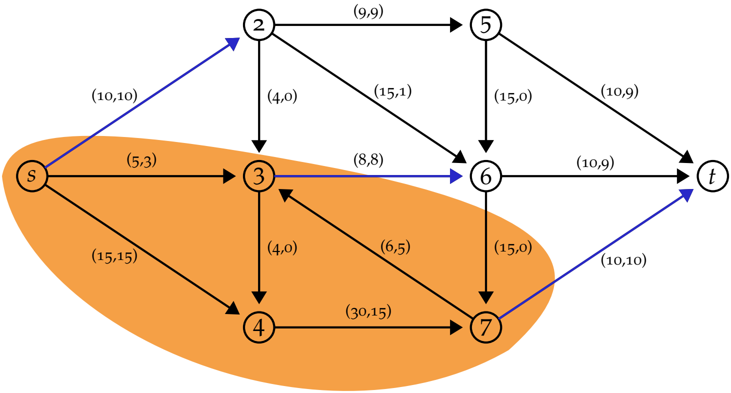
$$U = 28$$

$$F = 24$$


2
5
s
3
6
t
4
7
(9,6)
(10,10)
(4,4)
(15,0)
(15,0)
(10,6)
(5,4)
(8,8)
(10,8)
(15,10)
(4,0)
(6,0)
(15,0)
(10,10)
(30,10)

$$U = 28$$

$$F = 28$$

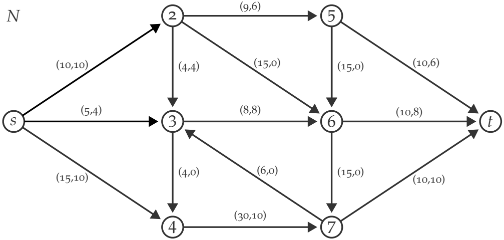

(9,9)
(10,10)
(4,0)
(15,1)
(15,0)
(10,9)
(5,3)
(8,8)
(10,9)
(15,15)
(4,0)
(6,5)
(15,0)
(10,10)
(30,15)
2
5
s
3
6
t
4
7

- Dada una red $G = (N, A)$ con función de capacidad u y un flujo factible x, definimos la red residual $R(G, x) = (N, A_R)$, donde:

  1. $ij \in A_R$ si $x_{ij} < u_{ij}$,
  2. $ji \in A_R$ si $x_{ij} > 0$.

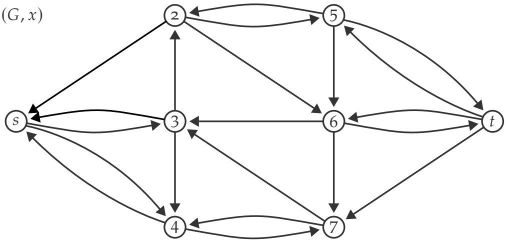
- Dada una red $G = (N, A)$ con función de capacidad u y un flujo factible x, definimos la red residual $R(G, x) = (N, A_R)$, donde:
  1. $ij \in A_R$ si $x_{ij} < u_{ij}$,
  2. $ji \in A_R$ si $x_{ij} > 0$.
- Un camino de aumento es un camino orientado de s a t en $R(G, x)$.

N

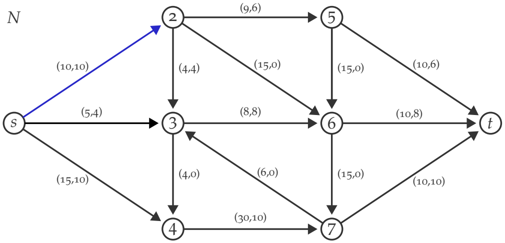


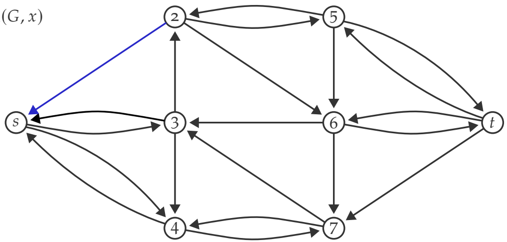
(9,6)
(10,10)
(4,4)
(15,0)
(15,0)
(10,6)
(5,4)
(8,8)
(10,8)
(15,10)
(4,0)
(6,0)
(15,0)
(10,10)
(30,10)
s
2
3
4
5
6
7
t

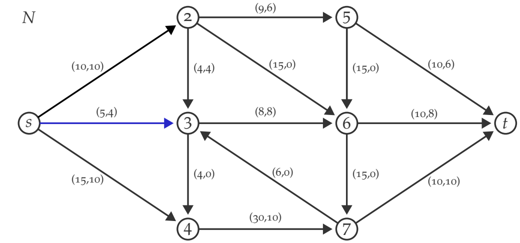
$R(G, x)$

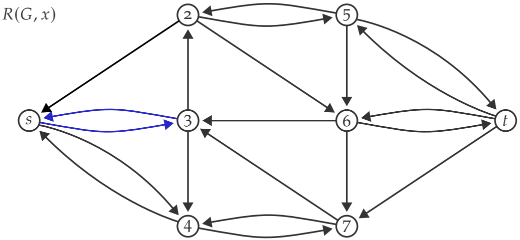


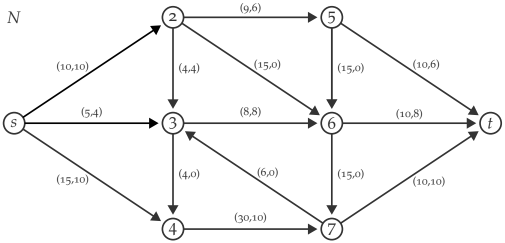
s
2
3
4
5
6
7
t

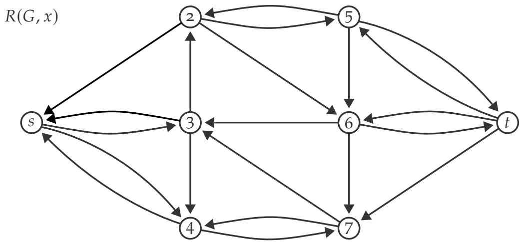

N
2
5
s
3
6
t
4
7
(9,6)
(10,10)
(4,4)
(15,0)
(15,0)
(10,6)
(5,4)
(8,8)
(10,8)
(15,10)
(4,0)
(6,0)
(15,0)
(10,10)
(30,10)

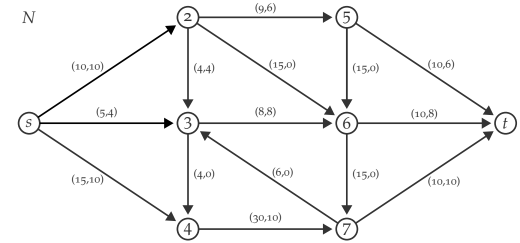

R(G, x)
2
5
s
3
6
t
4
7

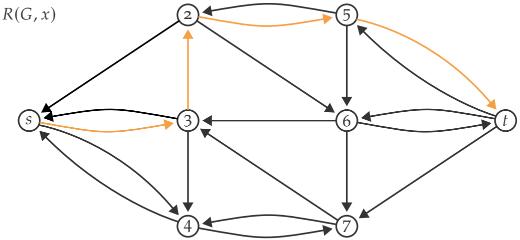

N
(9,6)
(10,10)
(4,4)
(15,0)
(15,0)
(10,6)
(5,4)
(8,8)
(10,8)
(15,10)
(4,0)
(6,0)
(15,0)
(10,10)
(30,10)
2
5
s
3
6
t
4
7


R(G, x)
2
5
s
3
6
t
4
7


N
2
5
s
3
6
t
4
7
(9,6)
(10,10)
(4,4)
(15,0)
(15,0)
(10,6)
(5,4)
(8,8)
(10,8)
(15,10)
(4,0)
(6,0)
(15,0)
(10,10)
(30,10)


R(G, x)
2
5
s
3
6
t
4
7


N
2
5
s
3
6
t
4
7
(9,6)
(10,10)
(4,4)
(15,0)
(15,0)
(10,6)
(5,4)
(8,8)
(10,8)
(15,10)
(4,0)
(6,0)
(15,0)
(10,10)
(30,10)


R(G, x)
2
5
s
3
6
t
4
7

- Dado un camino de aumento P, para cada arco $ij \in P$ definimos

$$\Delta(ij) = \begin{cases} u_{ij} - x_{ij} & \text{si } ij \in A \\ x_{ji} & \text{si } ji \in A \end{cases}$$

- Dado un camino de aumento P, para cada arco $ij \in P$ definimos

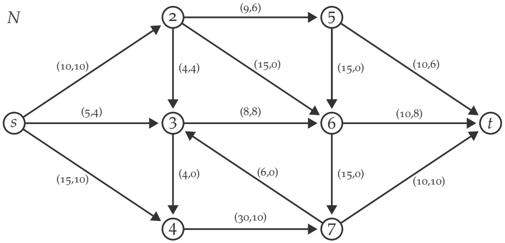
$$\Delta(ij) = \begin{cases} u_{ij} - x_{ij} & \text{si } ij \in A \\ x_{ji} & \text{si } ji \in A \end{cases}$$

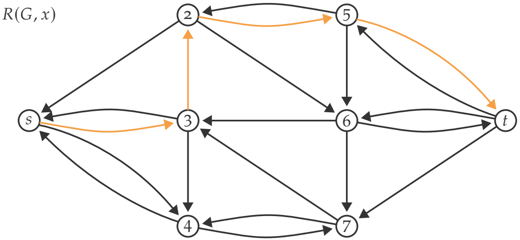
- Definimos además $\Delta(P) = \min_{ij \in P}\{\Delta(ij)\}$.

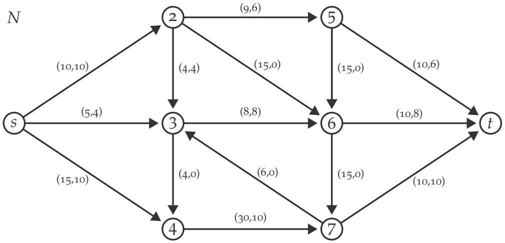
- Dado un camino de aumento P, para cada arco $ij \in P$ definimos

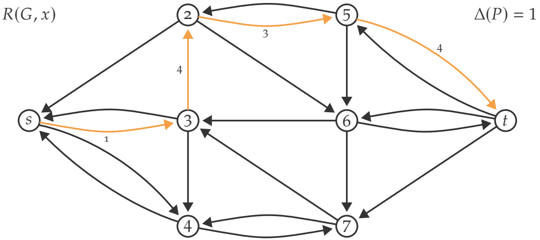
$$\Delta(ij) = \begin{cases} u_{ij} - x_{ij} & \text{si } ij \in A \\ x_{ji} & \text{si } ji \in A \end{cases}$$

- Definimos además $\Delta(P) = \min_{ij \in P}\{\Delta(ij)\}$.
- Podemos encontrar un camino de aumento P en la red residual en $O(m)$, y calculamos $\Delta(P)$ en $O(n)$.


N
2
5
s
3
6
t
4
7
(9,6)
(10,10)
(4,4)
(15,0)
(15,0)
(10,6)
(5,4)
(8,8)
(10,8)
(15,10)
(4,0)
(6,0)
(15,0)
(10,10)
(30,10)


R(G, x)
2
5
s
3
6
t
4
7


N
(9,6)
(10,10)
(4,4)
(15,0)
(15,0)
(10,6)
(5,4)
(8,8)
(10,8)
(4,0)
(6,0)
(15,0)
(15,10)
(10,10)
(30,10)
s
2
3
4
5
6
7
t


R(G, x)
Δ(P) = 1
3
4
4
1
s
2
3
4
5
6
7
t

Proposición

Sea x un flujo definido sobre una red N con valor F y sea P un camino de aumento en $R(G, x)$. Entonces el flujo $\bar{x}$, definido por

$$\bar{x}(ij) = \begin{cases} x_{ij} & \text{si } ij \notin P \\ x_{ij} + \Delta(P) & \text{si } ij \in P \\ x_{ij} - \Delta(P) & \text{si } ji \in P \end{cases}$$

es un flujo factible sobre N con valor $\bar{F} = F + \Delta(P)$.

Teorema

Sea x un flujo definido sobre una red N. Entonces x es un flujo máximo $\iff$ no existe camino de aumento en $R(G, x)$.

Teorema (max flow-min cut)

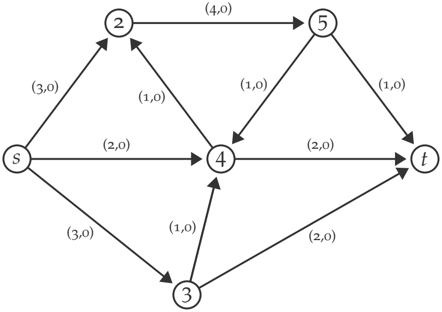
Dada una red N, el valor del flujo máximo es igual a la capacidad del corte mínimo.

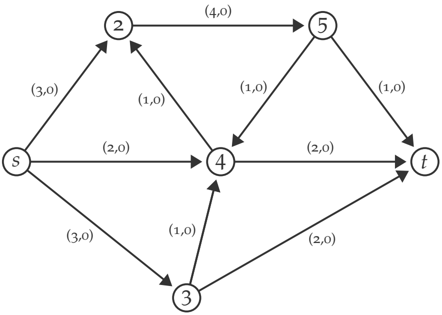
Lester Ford
(1927–2017)

Delbert Fulkerson
(1924–1976)

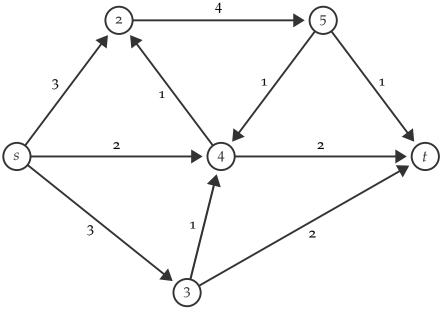
- El algoritmo de Ford y Fulkerson (1956) obtiene un flujo máximo con complejidad $O(nmU)$, donde $U = \max_{ij \in A} u_{ij}$.

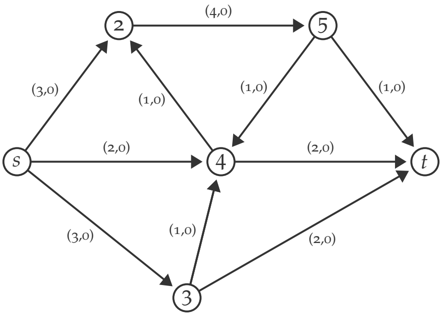
Definir un flujo inicial en N (por ejemplo, $x = 0$)
mientras exista $P :=$ camino de aumento en $R(G, x)$ **hacer**
  para cada arco $ij \in P$ **hacer**
    si $ij \in A$ **entonces**
      $x_{ij} := x_{ij} + \Delta(P)$
    si no $(ji \in A)$
      $x_{ji} := x_{ji} - \Delta(P)$
    fin si
  fin para
fin mientras


(4,0)
2
5
(3,0)
(1,0)
(1,0)
(1,0)
s
(2,0)
4
(2,0)
t
(3,0)
(1,0)
(2,0)
3

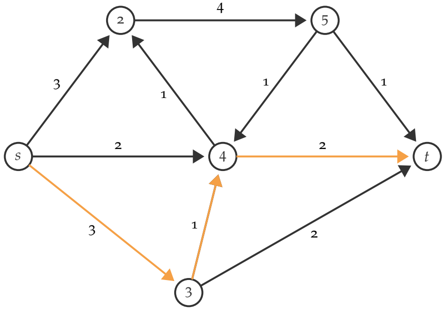


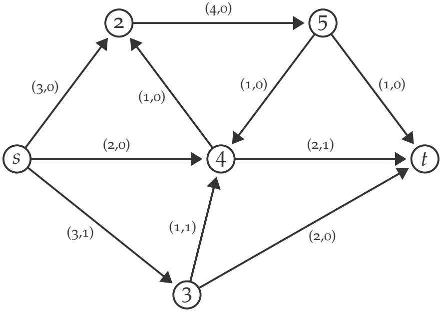
$R(G, x)$



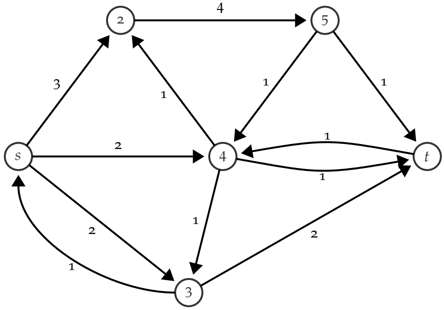


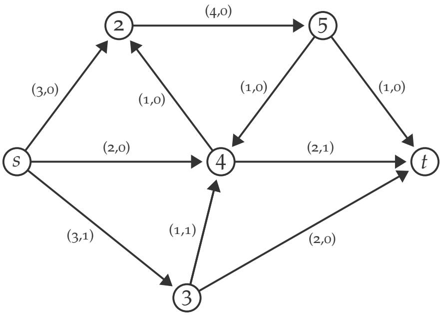
$R(G, x)$



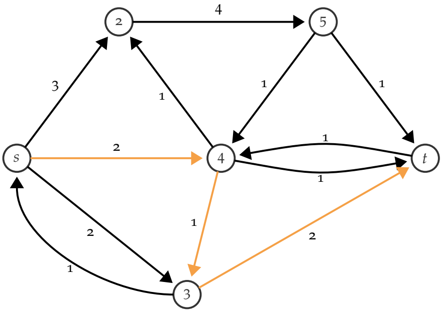


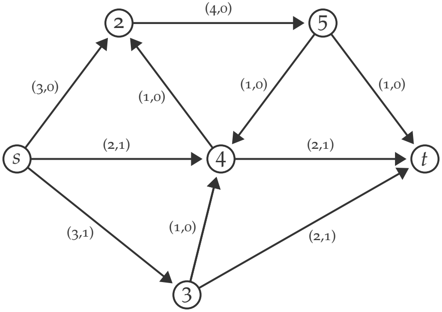
R(G, x)



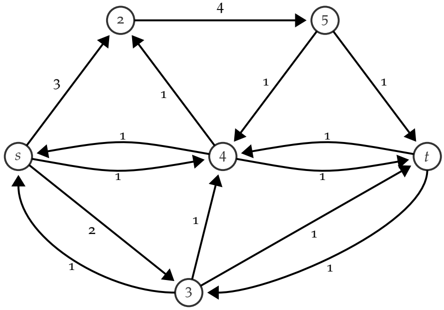


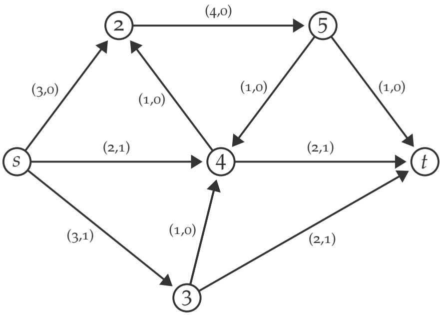
$R(G, x)$



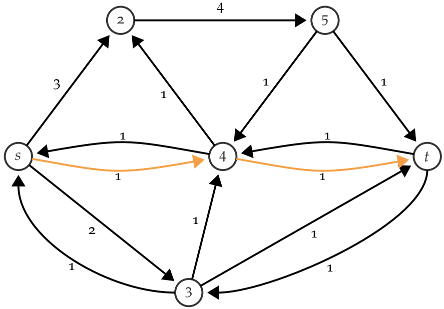


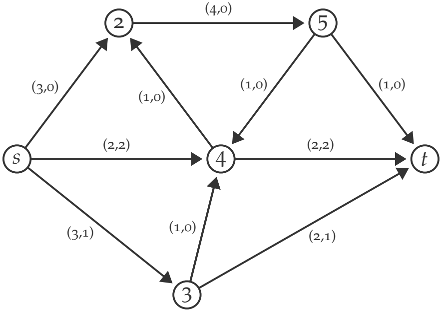
$R(G, x)$



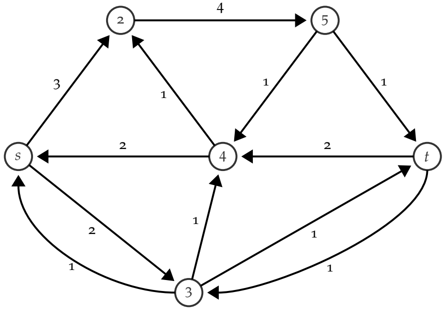


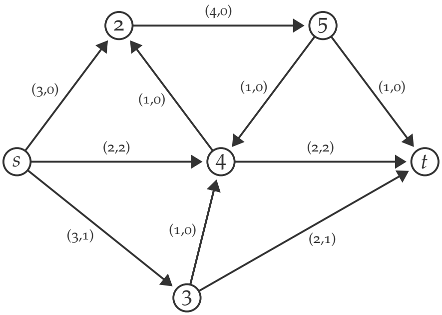
$R(G, x)$



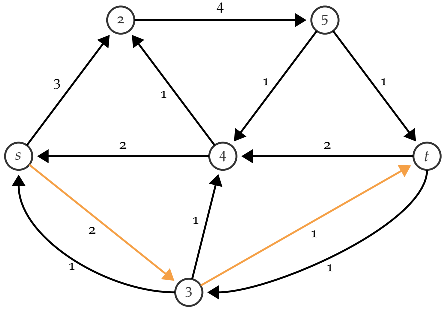


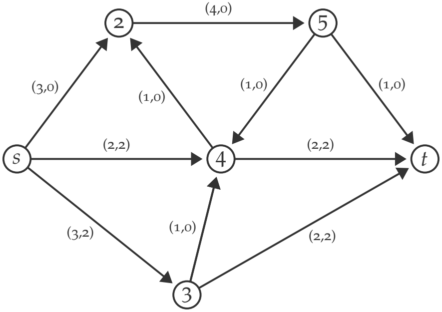
$R(G, x)$



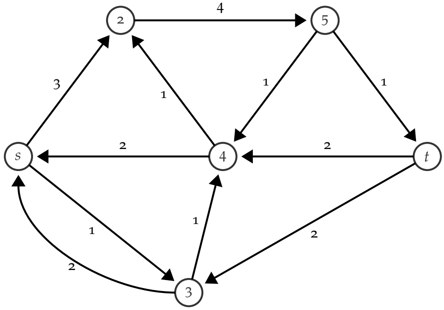


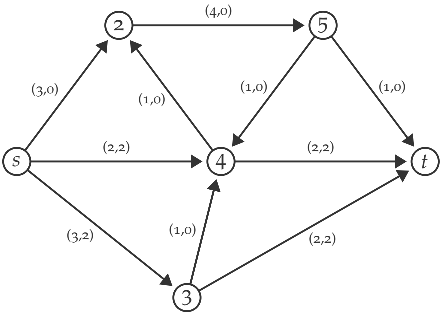
$R(G, x)$



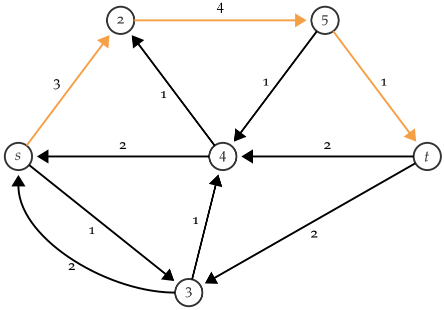


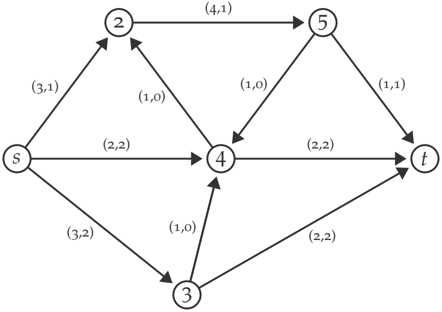
R(G, x)



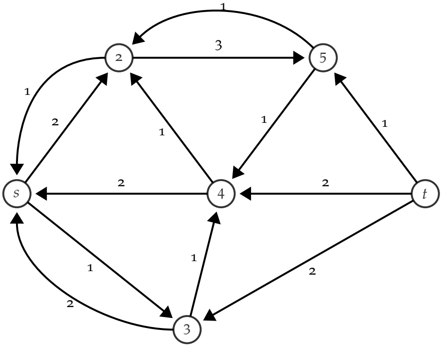


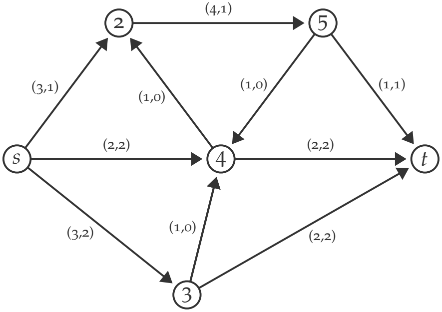
$R(G, x)$



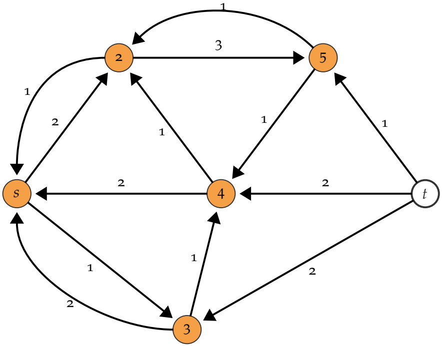


$R(G, x)$





$R(G, x)$



Teorema

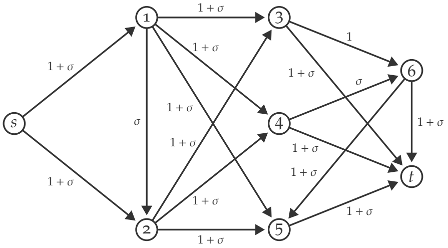
Si las capacidades de los arcos de la red son enteras, entonces el problema de flujo máximo tiene un flujo máximo entero.

Teorema

Si los valores del flujo inicial y las capacidades de los arcos de la red son enteras, entonces el método de Ford y Fulkerson realiza a lo sumo nU iteraciones, donde U es una cota superior finita para el valor de las capacidades.

Si las capacidades o el flujo inicial son números irracionales, el método de Ford y Fulkerson puede no parar (es decir, realizar un número infinito de pasos).

$\sigma = (\sqrt{5} - 1)/2$



Iteración	Camino de aumento
$6k + 1$	$s, 1, 2, 3, 6, t$
$6k + 2$	$s, 2, 1, 3, 6, 5, t$
$6k + 3$	$s, 1, 2, 4, 6, t$
$6k + 4$	$s, 2, 1, 4, 6, 3, t$
$6k + 5$	$s, 1, 2, 5, 6, t$
$6k + 6$	$s, 2, 1, 5, 6, 4, t$

Algoritmo de Edmonds y Karp

Jack Edmonds
(1934–)

Richard Karp
(1935–)

- La modificación de Edmonds y Karp (1972) a este algoritmo consiste en usar BFS para buscar caminos de aumento.
- Resuelve el problema con complejidad $O(nm^2)$.

- Algoritmo de Ford y Fulkerson (1956): $O(nmU)$.
- Algoritmo de Edmonds y Karp (1972): $O(nm^2)$.

Problema de flujo máximo

- Algoritmo de Ford y Fulkerson (1956): $O(nmU)$.
- Algoritmo de Edmonds y Karp (1972): $O(nm^2)$.
- Algoritmo de Dinic (1970): $O(n^2m)$.
- Algoritmo de Malhotra, Kumar y Maheshwari (1978): $O(n^3)$.
- Algoritmo de Cheriyan y Maheshwari (1988): $O(n^2\sqrt{m})$.
- Algoritmo de Goldberg y Tarjan (1988): $O(nm\log\frac{n^2}{m})$.
- Algoritmo de King, Rao y Tarjan (1994): $O(nm\log_{\frac{m}{n\log n}} n)$.

Problema de flujo máximo

- Algoritmo de Ford y Fulkerson (1956): $O(nmU)$.
- Algoritmo de Edmonds y Karp (1972): $O(nm^2)$.
- Algoritmo de Dinic (1970): $O(n^2m)$.
- Algoritmo de Malhotra, Kumar y Maheshwari (1978): $O(n^3)$.
- Algoritmo de Cheriyan y Maheshwari (1988): $O(n^2\sqrt{m})$.
- Algoritmo de Goldberg y Tarjan (1988): $O(nm \log \frac{n^2}{m})$.
- Algoritmo de King, Rao y Tarjan (1994): $O(nm \log_{\frac{m}{n \log n}} n)$.
- Algoritmo de Orlin (2013): $O(nm)$.

Problema de flujo máximo

- Algoritmo de Ford y Fulkerson (1956): $O(nmU)$.
- Algoritmo de Edmonds y Karp (1972): $O(nm^2)$.
- Algoritmo de Dinic (1970): $O(n^2m)$.
- Algoritmo de Malhotra, Kumar y Maheshwari (1978): $O(n^3)$.
- Algoritmo de Cheriyan y Maheshwari (1988): $O(n^2\sqrt{m})$.
- Algoritmo de Goldberg y Tarjan (1988): $O(nm \log \frac{n^2}{m})$.
- Algoritmo de King, Rao y Tarjan (1994): $O(nm \log_{\frac{m}{n \log n}} n)$.
- Algoritmo de Orlin (2013): $O(nm)$.
- Algoritmo de Gao, Liu y Peng (2021): $O(m^{\frac{3}{2}-\frac{1}{328}} \log U)$.
- Algoritmo de Chen, Kyng, Liu, Gutenberg y (2022): $O(m^{1+O(1)} \log U)$.

- Un matching o correspondencia entre los vértices de G, es un conjunto $M \subseteq E$ de aristas de G tal que para todo $v \in V$, v es incidente a lo sumo a una arista de M.

- Un matching o correspondencia entre los vértices de G, es un conjunto $M \subseteq E$ de aristas de G tal que para todo $v \in V$, v es incidente a lo sumo a una arista de M.
- El problema de matching máximo consiste en encontrar un matching de cardinal máximo entre todos los matchings de G.

- Un matching o correspondencia entre los vértices de G, es un conjunto $M \subseteq E$ de aristas de G tal que para todo $v \in V$, v es incidente a lo sumo a una arista de M.
- El problema de matching máximo consiste en encontrar un matching de cardinal máximo entre todos los matchings de G.
- El problema de matching máximo es resoluble en tiempo polinomial para grafos en general (Edmonds, 1961–1965).

- Un matching o correspondencia entre los vértices de G, es un conjunto $M \subseteq E$ de aristas de G tal que para todo $v \in V$, v es incidente a lo sumo a una arista de M.
- El problema de matching máximo consiste en encontrar un matching de cardinal máximo entre todos los matchings de G.
- El problema de matching máximo es resoluble en tiempo polinomial para grafos en general (Edmonds, 1961–1965).
- Pero en el caso de grafo bipartitos, podemos enunciar un algoritmo más simple transformándolo en un problema de flujo máximo en una red.

Dado el grafo bipartito $G = (V_1 \cup V_2, E)$ definimos la siguiente red $N = (V', E')$:

Dado el grafo bipartito $G = (V_1 \cup V_2, E)$ definimos la siguiente red $N = (V', E')$:

- $V' = V_1 \cup V_2 \cup \{s, t\}$, con s y t dos vértices ficticios representando la fuente y el sumidero de la red.

Dado el grafo bipartito $G = (V_1 \cup V_2, E)$ definimos la siguiente red $N = (V', E')$:

- $V' = V_1 \cup V_2 \cup \{s, t\}$, con s y t dos vértices ficticios representando la fuente y el sumidero de la red.
- $$E' = \{(i, j) : i \in V_1, j \in V_2, ij \in E\} \cup \{(s, i) : i \in V_1\} \cup \{(j, t) : j \in V_2\}.$$

Dado el grafo bipartito $G = (V_1 \cup V_2, E)$ definimos la siguiente red $N = (V', E')$:

- $V' = V_1 \cup V_2 \cup \{s, t\}$, con s y t dos vértices ficticios representando la fuente y el sumidero de la red.
- $E' = \{(i, j) : i \in V_1, j \in V_2, ij \in E\}$
  $\cup \{(s, i) : i \in V_1\}$
  $\cup \{(j, t) : j \in V_2\}$.
- $u_{ij} = 1$ para todo $ij \in E$.

Matching máximo en grafos bipartitos

Dado el grafo bipartito $G = (V_1 \cup V_2, E)$ definimos la siguiente red $N = (V', E')$:

- $V' = V_1 \cup V_2 \cup \{s, t\}$, con s y t dos vértices ficticios representando la fuente y el sumidero de la red.
- $$\begin{aligned} E' = &\{(i, j) : i \in V_1, j \in V_2, ij \in E\} \\ &\cup \{(s, i) : i \in V_1\} \\ &\cup \{(j, t) : j \in V_2\}. \end{aligned}$$
- $u_{ij} = 1$ para todo $ij \in E$.

El cardinal del matching máximo de G será igual al valor del flujo máximo en la red N.

Flujo de costo mínimo

Datos de entrada

1. Un grafo dirigido $G = (N, A)$.
2. Imbalance $b : N \to \mathbb{Z}$ de cada nodo.
3. Capacidad $u : A \to \mathbb{Z}_+$ de cada arco.
4. Costo unitario $c : A \to \mathbb{Z}$ para cada arco.

Problema

Encontrar un flujo que respete el imbalance de cada nodo y las cotas de cada arco, con el menor costo posible.

1. Para cada nodo $i \in N$, debemos tener $b_i = \sum_{j \in N^+(i)} x_{ij} - \sum_{j \in N^-(i)} x_{ji}$.
2. La cantidad x_{ij} enviada por el arco $ij \in A$ debe cumplir $0 \leq x_{ij} \leq u_{ij}$.
3. El costo del flujo es $C = \sum_{ij \in A} c_{ij}\, x_{ij}$.

- Definimos la red residual G_x de un flujo $x : A \longrightarrow \mathbb{R}_+$ reemplazando cada arco $ij \in A$ por dos arcos ij y ji.

- Definimos la red residual G_x de un flujo $x : A \longrightarrow \mathbb{R}_+$ reemplazando cada arco $ij \in A$ por dos arcos ij y ji.

  1. El arco ij tiene costo c_{ij} y capacidad residual $r_{ij} = u_{ij} - x_{ij}$.
  2. El arco ji tiene costo $-c_{ij}$ y capacidad residual $r_{ji} = x_{ij}$.

- La red residual consiste solamente de los arcos con capacidad residual positiva.

- Definimos la red residual G_x de un flujo $x : A \to \mathbb{R}_+$ reemplazando cada arco $ij \in A$ por dos arcos ij y ji.

  1. El arco ij tiene costo c_{ij} y capacidad residual $r_{ij} = u_{ij} - x_{ij}$.
  2. El arco ji tiene costo $-c_{ij}$ y capacidad residual $r_{ji} = x_{ij}$.

- La red residual consiste solamente de los arcos con capacidad residual positiva.

Teorema

Una solución factible x es óptima si y sólo si la red residual G_x no contiene ningún ciclo (dirigido) de costo negativo.

Morton Klein (1926–2001)

Algoritmo de cancelación de ciclos (Klein, 1967)

A partir de un flujo factible, mientras exista un ciclo de costo negativo en la red residual aumentar el flujo a lo largo de ese ciclo.

Algoritmo de cancelación de ciclos

1. Establecer un flujo x factible.
2. **Mientras** G_x contenga un ciclo negativo W **hacer**
   - Definir $\delta := \min\{r_{ij} : ij \in W\}$.
   - Aumentar δ unidades de flujo a lo largo del ciclo W y actualizar x.
3. **Fin mientras**

Algoritmo de cancelación de ciclos

1. Establecer un flujo x factible.
2. **Mientras** G_x contenga un ciclo negativo W **hacer**
    - Definir $\delta := \min\{r_{ij} : ij \in W\}$.
    - Aumentar δ unidades de flujo a lo largo del ciclo W y actualizar x.
3. **Fin mientras**

- Cómo obtenemos el flujo inicial factible?

Teorema

Si todos los imbalances y capacidades son enteros, entonces el problema de flujo de costo mínimo tiene una solución óptima entera.

- Cuál es la complejidad computacional de este algoritmo?

Teorema

Si todos los imbalances y capacidades son enteros, entonces el problema de flujo de costo mínimo tiene una solución óptima entera.

- Cuál es la complejidad computacional de este algoritmo?

    1. $C := \max\{c_{ij} : ij \in A\}$.
    2. $U := \max\{u_{ij} : ij \in A\}$.

Teorema

Si todos los imbalances y capacidades son enteros, entonces el problema de flujo de costo mínimo tiene una solución óptima entera.

- Cuál es la complejidad computacional de este algoritmo?

  1. $C := \max\{c_{ij} : ij \in A\}$.
  2. $U := \max\{u_{ij} : ij \in A\}$.

- El costo del flujo inicial no puede ser superior a mCU y el costo final no puede ser inferior a cero. Luego, el algoritmo realiza a lo sumo mCU iteraciones y su complejidad total es $O(nm^2CU)$.

Flujo de costo mínimo

Teorema

Si todos los imbalances y capacidades son enteros, entonces el problema de flujo de costo mínimo tiene una solución óptima entera.

- Cuál es la complejidad computacional de este algoritmo?

  1. $C := \max\{c_{ij} : ij \in A\}$.
  2. $U := \max\{u_{ij} : ij \in A\}$.

- El costo del flujo inicial no puede ser superior a mCU y el costo final no puede ser inferior a cero. Luego, el algoritmo realiza a lo sumo mCU iteraciones y su complejidad total es $O(nm^2CU)$.

- Si en cada paso se selecciona un ciclo de costo promedio mínimo (y se puede hacer en $O(nm)$), entonces este algoritmo realiza a lo sumo $O(\min\{nm\log(nC), nm^2\log n\})$ iteraciones (Goldberg y Tarjan, 1988).