Flujo Máximo

Miguel Pascual Domínguez

16 de marzo de 2017

1. Introducción

Al igual que el algoritmo de Dijkstra nos ayuda a encontrar el camino más corto desde un vértice a otro, existen algoritmos que nos proporcionan suma de afluencia de los "mejores" caminos de un vértice a otro, entendiendo como mejor, a los caminos con mayor afluencia sin superar la capacidad del camino en sí, es decir, la capacidad de sus aristas.

Este problema sirve por ejemplo para indicar el camino que debe recorrer un líquido, a través de una serie de tuberías desde el lugar en el que se produce hasta el lugar en el que se consume, procurando que la cantidad exportada sea máxima.

2. Redes de Flujo

2.1. Red de flujo, Flujo y sus Propiedades

Como es de suponer, el problema del flujo máximo no es aplicable a todo grafo dirigido valorado. Sino que dicho grafo debe tener algunas características especiales.

Llamamos *red de flujo* $G = (V, E)$ a un grafo dirigido valorado cuyas aristas tienen una capacidad positiva es decir $c(u, v) \geq 0$, siendo $c(u, v)$ una función que dada una arista te devuelve la capacidad de esa arista que va de u a v. Por tanto si alguna $(u, v) \notin E$ suponemos que $c(u, v) = 0$. Además se deben distinguir dos vértices especiales, que son la *fuente* **s** (source) y el *sumidero* **t** (sink). La fuente es un vértice que solo tiene aristas salientes es decir $\forall (u, v) \in E, s \neq v$, y el sumidero solo tiene aristas entrantes es decir $\forall (u, v) \in E, t \neq u$.

Podemos suponer que las redes de flujo con las que se trabaja son conexas y por tanto $\forall\ u \in V$ existe un camino que va desde s a t pasando por u, sea o no sea máximo en capacidad. Entonces podemos observar que $|E| \geq |V| - 1$, ya que para que un grafo sea conexo, como mínimo debe tener $|V| - 1$ aristas.

Ahora que ya hemos definido correctamente el concepto de red de flujo, sumidero, y fuente, podemos definir flujo. Dada una red de flujo $G = (V, E)$ con una función de capacidad c, llamaremos *flujo* en G a una función $f : V \times V \longrightarrow \mathbb{R}$ que cumpla las siguientes características :

1. Restricción de capacidad: $\forall u, v \in V, f(u, v) \leq c(u, v)$.
2. Simetría de inclinación: $\forall u, v \in V, f(u, v) = -f(v, u)$.
3. Conservación de flujo: $\forall u \in V - \{s, t\}\ \sum_{v \in V} f(u, v) = 0$.

Como podemos observar la propiedad 3, es igual que la ley de Kirchhoff de las corrientes alternas; la suma de las corrientes entrantes en un nodo debe ser igual a la suma de las corrientes salientes.

La siguiente figura 1 presenta una red de flujo conexa, con 6 vértices y 10 aristas, las cuales están valoradas con su correspondiente capacidad en el dibujo, y en el caso de las aristas de 1 a 2 y de 2 a 1 están colocados a la izquierda y derecha, respectivamente, además de estar indicado en la figura. La fuente de esta red, es el vértice s coloreado de azul y el sumidero es el vértice t coloreado de amarillo.

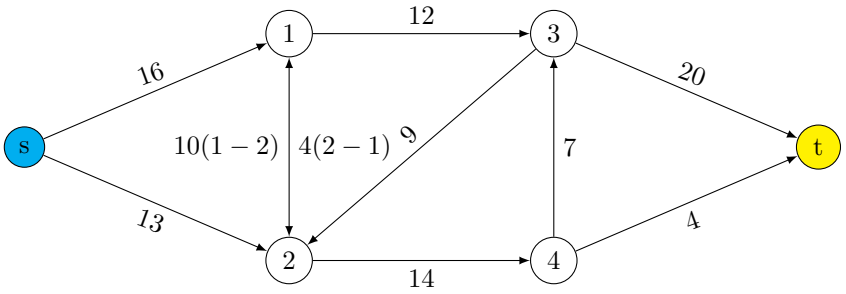

Figura 1: Red de Flujo.

Añadamos ahora a este dibujo, véase figura 2, una función de flujo, indicando en cada arista su capacidad y su cantidad de flujo correspondiente(capacidad:flujo); que puede ser positiva, negativa o cero, pero solo se mostrará las positivas. Se puede observar la conservación de flujo, basta comprobarlo realizando unas sumas y restas, para cada vértice exceptuando el s y el t. Se define también el *valor de flujo* de f como $|f| = \sum_{v \in V} f(s, v)$, es decir la cantidad de flujo que sale de s. En este caso $|f| = 19 = 11 + 8$.

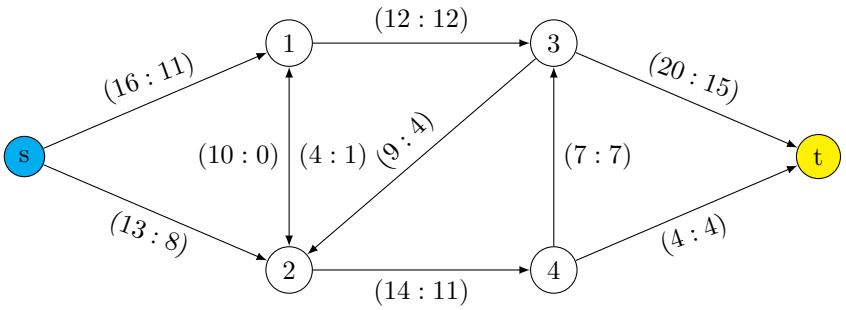

Figura 2: Red de Flujo y función de Flujo.

Indaguemos un poco en las 3 propiedades. Como se debe cumplir la simetría de inclinación, se puede redefinir la conservación de flujo como $\sum_{u \in V} f(u, v) = 0$ $\forall v \in V - \{s, t\}$, esto implica que no solo se debe cumplir la conservación fijado un vértice de salida, sino también fijado el vértice de entrada. Otra propiedad curiosa que es fácil de ver es que si $(u, v) \notin E$ y $(v, u) \notin E$ entonces $f(u, v) = f(v, u) = 0$, para demostrar esto; sabemos que $f(v, u) \leq c(v, u) = 0$, entonces suponemos que $f(v, u) < 0$ y por lo tanto $-f(v, u) > 0$ pero $-f(v, u) =$

$f(u,v) \geq c(u,v) = 0$ y llegamos a una contradicción que nos da el resultado que buscábamos.

Con esto podemos definir también el *flujo total positivo entrante* de un vértice $v \in V$ como $\sum_{u \in V, f(u,v) \geq 0} f(u,v)$. Como cabe esperar, también existe su definición simétrica el *flujo total positivo saliente* de un vértice v, que basta cambiar $f(u,v)$ por $f(v,u)$ en el sumatorio. Además se introduce el concepto de *flujo total neto* como la resta entre el flujo total positivo saliente menos en entrante, que salvo en los vértices s y t, será 0. En inglés hay una expresión para esta propiedad "flow in equals flow out".

Ejemplo 2.2.1

Cojamos la red de flujo dibujada en la figura 2 con la función de flujo ya asignada. Supongamos que en cada vértice tenemos una ciudad por ejemplo s=Galicia, 1=León, 2=Valladolid, 3=Segovia, 4=Soria, y t=Madrid. Y sabemos que en Galicia, se fabrican pelotas de fútbol, y queremos que todas las que se fabriquen en un día, salgan ese mismo día hacía Madrid y que al final del día no queden existencias en Galicia. Entre cada ciudad circula un tipo de camión que tiene una capacidad determinada, y esas capacidades no se pueden modificar, entonces lo que queremos es maximizar el número de pelotas de fútbol que se van a fabricar para mandarlos todos sin sobrepasar las capacidades de los camiones y sin que queden existencias. Para ello es lógico planear los envíos como una red de flujo, porque se deben cumplir las propiedades de la función de flujo y por ejemplo los envíos que vienen de un solo camión, tienen que haber partido de la ciudad correspondiente, y las cantidades que salen de cualquier ciudad que no sea Galicia deben haber llegado en algún momento, sino se estarían enviando existencias que no se han fabricado.

El problema que surge es que en la vida real es más complejo ya que no se pueden tratar los envíos exactamente igual que un flujo en una red de flujo. Ya que se pueden enviar balones de León (1) a Valladolid (2) pero también de Valladolid (2) a León (1). Y por tanto si por ejemplo tratamos como flujos los envíos de Valladolid a León que pueden ser 3 balones y de León a Valladolid que pueden ser 7, es decir $f(2,1) = 3$ *y* $f(1,2) = 7$ entonces nos encontramos que $-3 = -f(2,1) \neq f(1,2) = 7$ y por tanto se viola una de las propiedades del flujo, la de simetría de inclinación. A pesar de ello, esto se puede arreglar de una manera muy sencilla, *cancelando* el valor de flujo menor, es decir como $f(2,1) = 3 < f(1,2) = 7$, tenemos en cuenta el mayor pero extrayéndole lo del menor y así hacemos que $f(1,2) = 4 = 7 - 3$ y por la simetría $f(2,1) = 3 - 7 = -4$ de esta manera se respetan las propiedades del flujo, en los dibujos solo se mostraran los flujos positivos. Al modificar esto, puede producir una pequeña confusión porque dada una red de flujo con una función de flujo, no se puede determinar el valor real de los envíos o transacciones, porque dado un valor de flujo $f(u,v) = 8$ no se puede saber con exactitud si en realidad es un 8 porque se enviaron 8 cantidades de u a v y 0 de v a u, o si se enviaron 10 cantidades de u a v y 2 de v a u.

Finalmente, definamos el problema del flujo máximo.

Dada una red de flujo con s y t ya asignados, el *problema de flujo máximo* consiste en encontrar una función de flujo f tal que el valor de flujo sea máximo, es decir que $|f|$ sea máximo.

2.2. Redes con múltiples fuentes y sumideros

Como uno se puede imaginar un problema de flujo no tiene por que tener una sola fuente ni un solo sumidero, sino que pueden tener varias de ambos, varias de uno y uno del otro, o uno de los dos. Este sencillo problema de flujo máximo con varias fuentes y varios sumideros se puede reducir a un problema que tenga una sola fuente y un solo sumidero. Para ello se añaden una superfuente, que viene a ser un vértice nuevo s que esta conectado al resto de fuentes s_i mediante una arista $(s, s_i) \in E$ que tiene $c(s, s_i) = \infty$ para cada $i = 1, 2, ..., m$ siendo m el número de fuentes; y un supersumidero, que es lo simétrico, es decir, se añade un vértice t que esta conectado con una arista $(t_i, t) \in E$ que tiene $c(t_i, t) = \infty$ para cada $i = 1, 2, ..., n$ siendo n el número de fuentes. La superfuente dará tanto flujo como sea necesario a cada fuente por eso $c(s, s_i) = \infty$; y el supersumidero recibirá tanto flujo como sea necesario de cada sumidero por eso $c(t_i, t) = \infty$. Ilustremos esto con una imagen.

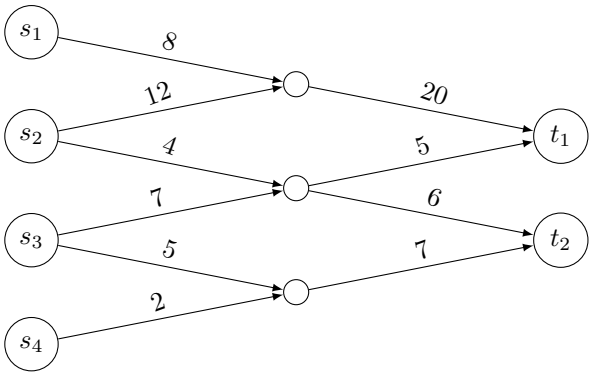

Figura 3: Red de Flujo con varias fuentes y varios sumideros.

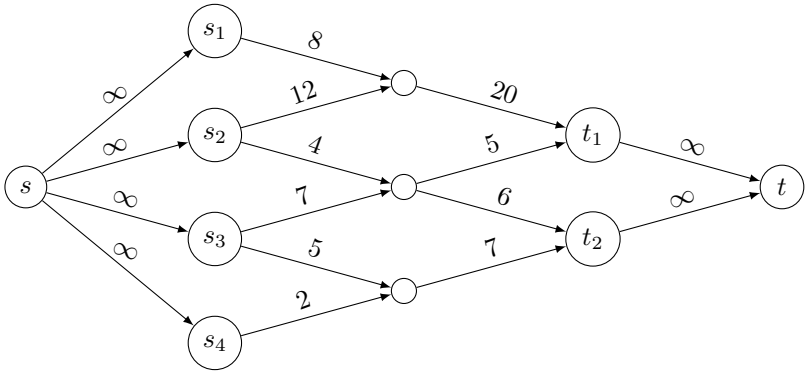

Figura 4: Red de Flujo con varias fuentes y varios sumideros y un supersumidero y una superfuente.

Las figuras 3 y 4 representan exactamente la misma red de flujo, a excepción de la superfuente y el supersumidero, y por lo tanto toda función de flujo que se le pueda asignar a la red de flujo de la figura 3 también se le podrá asignar a la red de flujo de la figura 4, y viceversa.

2.3. Flujo en conjuntos de Vértices

Al igual que se puede estudiar el flujo involucrado en solo dos vértices, el saliente y el entrante de la correspondiente arista, también se puede estudiar el flujo en un conjunto de vértices de V. Entonces, sean X e Y conjuntos de vértices, se define el flujo de X sobre Y como $f(X,Y) = \sum_{x\in X}\sum_{y\in Y} f(x,y)$.

La propiedad de la conservación de flujo antes mencionada difiere un poco, ya que se da como la condición de que $f(u,V) = 0\ \forall u \in V - \{s,t\}$. En algunos casos se cometerá un abuso de notación ya que es fácil ver por la definición de que $|f| = f(s,V) = f(s,V-s)$ dando por hecho que $V = V - \{s\}$.

Expondremos ahora un lema que la demostración se dejará pensarla al lector ya que basta usar las propiedades de simetría de inclinación y de la conservación de flujo expuestas al principio de la sección.

Lema 2.3.1

Dada una red de flujo $G = (V,E)$, y f un flujo de G. Entonces se cumplen las siguientes propiedades:

1. $\forall X \subseteq V$, $f(X,X) = 0$.
2. $\forall X,Y \subseteq V$, $f(X,Y) = -f(Y,X)$.
3. $\forall X,Y,Z \subseteq V$ tal que $X \cap Y = \emptyset$ entonces $f(X\cup Y,Z) = f(X,Z) + f(Y,Z)$ y $f(Z,X\cup Y) = f(Z,X) + f(Z,Y)$.

Después de haber introducido este lema, es sencillo observar que se puede obtener también el valor del flujo no solo con el de la fuente sino con el del sumidero es decir $|f| = f(s,V) = f(V,t)$. Para probar esto consiste en ir aplicando las propiedades del lema, en las siguientes igualdades.

Demostración. $|f| = f(s,V) = f(V,V) - f(V-s,V) = -f(V-s,V) = f(V,v-s) = f(V,t) + f(V,V-s-t) = f(V,t)$

Comentemos un poco la demostración para aclarar lo que se ha realizado. La primera es por definición. En la segunda igualdad se ha hecho uso del punto 3 del lema, es decir $f(V,V) = f(s,V) + f(V-s,V)$, y despejando $f(s,V)$ se obtiene lo aplicado. En la tercera se ha hecho uso del primer punto $f(V,V) = 0$. En la cuarta, la simetría (punto 2) $-f(V-s,V) = f(V,V-s)$. En la quinta, de nuevo el punto 3. En la sexta, la propiedad de conservación de flujo es decir $f(V,V-s-t) = 0$.

□

3. Método de Ford-Fulkerson

En esta sección comentaremos el método de Ford-Fulkerson para la resolución del problema del flujo máximo. Se denomina método y no algoritmo porque del método se pueden extraer varias implementaciones distintas con distintos tiempos de ejecución. Este método tiene tres importantes conceptos en los que se basa, que son las *redes residuales*, las *cadenas de aumento* y los *cortes*. Explicaremos posteriormente una implementación del algoritmo y analizaremos su tiempo de ejecución.

El Método de Ford-Fulkerson es un método iterativo, que consiste en empezar con una función de flujo $|f| = 0$ es decir que $\forall (u,v) \in E$, $f(u,v) = 0$. A medida que vaya ejecutándose el método, irá encontrando cadenas de aumento

que pueden ser vistas por el momento como un camino que va de s a t y que añade flujo. Y terminará cuando ya no encuentre una cadena de aumento.

Este sería el pseudocódigo correspondiente con el método:

function FORD-FULKERSON-METHOD(G,s,t)
  $f \leftarrow 0$
  while *Haya una cadena de aumento* p **do**
    Aumentar f *con* p
  end while
  return f
end function

3.1. Red Residual, Cadenas de Aumento y Cortes

Definamos ahora los conceptos necesarios. Comencemos con *red residual*.

Dada una red de flujo $G = (V, E)$ con fuente s y sumidero t. Entonces dado un flujo f, la *capacidad residual* se define como la cantidad de flujo adicional que puede pasar a través de una arista $(u, v) \in E$ sin sobrepasar su capacidad es decir $c_f(u, v) = c(u, v) - f(u, v)$. Si $f(u, v) < 0$ entonces $c_f(u, v) = c(u, v) + |f(u, v)|$, es decir los negativos también se tienen en cuenta. Definimos ahora la *red residual* como $G_f = (V, E_f)$ con $E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$. Por tanto, toda arista de E_f soporta que se transmita flujo por ella.

Jugando con las aristas de E_f, podemos llegar a la conclusión de que $\forall (u, v) \in E_f$ es o bien porque $(u, v) \in E$ *o bien* $(v, u) \in E$. Ya que si ninguna de las dos perteneciese a E, $c(u, v) = c(v, u) = f(u, v) = f(v, u) = 0$ y entonces $c_f(u, v) = c_f(v, u) = 0$. Por tanto obtenemos que $|E_f| \leq 2|E|$. Observemos también que G_f es una red de flujo con las capacidades residuales c_f.

Lema 3.1.1

Sea $G = (V, E)$ una red de flujo con fuente s y sumidero t, sea un flujo f en G. Además tenemos que G_f es la red residual, y que f' es un flujo de G_f. Entonces el flujo definido como $(f + f')(u, v) = f(u, v) + f'(u, v)$ es un flujo en G con valor de flujo $|f + f'| = |f| + |f'|$.

Demostración. 1. Para la restricción de capacidad notemos que como f' es flujo de G_f entonces por definición $f'(u, v) \leq c_f(u, v) = c(u, v) - f(u, v) \forall u, v \in V$.

Entonces $(f + f')(u, v) = f(u, v) + f'(u, v) \leq f(u, v) + (c(u, v) - f(u, v)) = c(u, v)$.

2. Simetría de inclinación $(f + f')(u, v) = f(u, v) + f'(u, v) = -f(v, u) - f'(v, u) = -(f(v, u) + f'(v, u)) = -(f + f')(u, v)$.

3. Conservación de flujo $\forall u \in V - \{s, t\}$ $\sum_{v \in V}(f + f')(u, v) = \sum_{v \in V}(f(u, v) + f'(u, v)) = \sum_{v \in V} f(u, v) + \sum_{v \in V} f'(u, v) = 0 + 0 = 0$.

Y por último, $|f + f'| = \sum_{v \in V}(f + f')(s, v) = \sum_{v \in V}(f(s, v) + f'(s, v)) = \sum_{v \in V} f(s, v) + \sum_{v \in V} f'(s, v) = |f| + |f'|$. □

Veamos ahora lo que es una cadena de aumento. Sea una red de flujo $G = (V, E)$ y f un flujo, se denomina *cadena de aumento* p como un camino de s a t en G_f. Se denomina la capacidad residual de la cadena en aumento como el mínimo de las capacidades residuales de las aristas del camino, $c_f(p) = min\{c_f(u, v) : (u, v) \in p\}$. A esto también se le suele denominar como el máximo aumento con el que se puede incrementar el flujo f. Debido al

siguiente lema podemos crear un nuevo flujo f' a partir de la cadena en aumento y del flujo f.

Lema 3.1.2 Sea $G = (V, E)$ red de flujo, f flujo en G, y p una cadena de aumento en G_f. Definimos $f_p : V \times V \longrightarrow \mathbb{R}$ como

$$f_p(u,v) = \begin{cases} c_f(p) & si \quad (u,v) \in p \\ -c_f(p) & si \quad (v,u) \in p \\ 0 & en\ otro\ caso \end{cases}$$

Entonces f_p es un flujo en G_f con $|f_p| = c_f(p) > 0$.

Corolario 3.1.3 Usando los dos lemas anteriores podemos determinar una función de flujo f' en $G = (V, E)$ ya que dado una red de flujo G, con función de flujo f, una cadena de aumento p de la red residual G_f. Entonces $f' = f + f_p$, f_p la definida en el lema anterior y f cualquier función de flujo de G, es una función de flujo en G con además mayor valor de flujo ya que $|f'| = |f| + |f_p| > |f|$.

Por último veamos lo que son los *cortes*.

Sea $G = (V, E)$ red de flujo, llamaremos corte a una partición de V en S y $T = V - S$ tal que $s \in S$ y $t \in T$ siendo s la fuente y t el sumidero. Teniendo un flujo$f$ de G, entonces el flujo neto del corte es $f(S, T)$ tal y como lo definimos antes en el flujo en conjuntos de vértices, es decir, $f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v)$. Y la capacidad del corte de la misma manera $c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v)$.

Diremos que el corte es un corte mínimo cuando su capacidad es mínima con respecto al resto de cortes posibles en la red de flujo.

Observemos que en el flujo neto de un corte pueden aparecer flujos negativos como en la siguiente figura 5. Tomamos el corte como $S = \{los\ verdes\}$ y $T = \{los\ blancos\}$. Observamos que $f(S, T) = f(1, 3) + f(2, 3) + f(2, 4) = 12 + (-4) + 11 = 19$ y que $c(S, T) = c(1, 3) + (2, 4) = 12 + 14 = 26$.

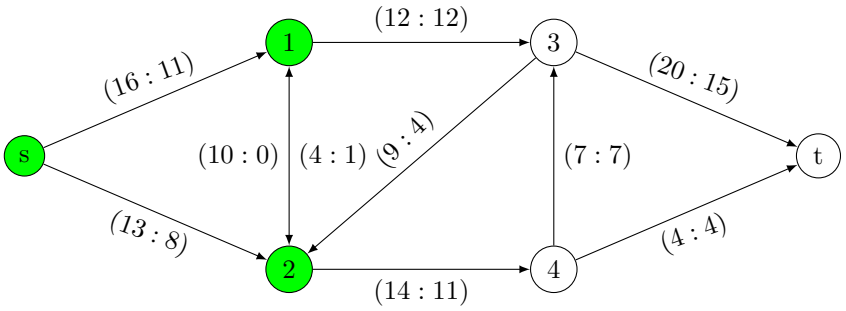

Figura 5: Red de Flujo y función de Flujo.

Lema 3.1.4

Sea f flujo en $G = (V, E)$ una red de flujo y sea (S, T) un corte en G. Entonces $f(S, T) = |f|$.

Demostración. Teniendo en cuenta que $f(S - s, V) = 0$ debido a la conservación

de flujo. Tenemos que

$f(S,T) = f(S,V) - f(S,S) = f(S,V) = f(s,V) - (S-s,V) = f(s,V) = |f|$.

Observamos que los pasos que se siguen son muy parecidos en los seguidos en la demostración de $|f| = f(s,V)$ realizada anteriormente. □

Corolario 3.1.5

La capacidad de cualquier corte en una red de flujo $G = (V,E)$ acota al valor de flujo $|f|$ de cualquier flujo f de G.

Demostración. $|f| = f(S,T) = \sum_{u\in S}\sum_{v\in T} f(u,v) \leq \sum_{u\in S}\sum_{v\in T} c(u,v) = c(S,T)$. □

3.2. Teorema del flujo máximo y del corte mínimo

Teorema 3.2.1 (Flujo Máximo y Corte Mínimo).

Sea f flujo en una red de flujo $G = (V,E)$, con fuente s y sumidero t entonces las siguientes afirmaciones son equivalentes:

1. f es un flujo máximo en G.
2. La red residual G_f no tiene cadenas de aumento.
3. $|f| = c(S,T)$ para algún corte de G.

Demostración. (1) ⇒ (2): Supongamos que f es flujo máximo y que la red residual tiene una cadena en aumento. Entonces la función $f' = f + f_p$ siendo f_p el flujo de la cadena en aumento cumple que $|f'| > |f|$ lo cual es una contradicción ya que $|f|$ es máximo.

(2) ⇒ (3): Supongamos que G_f no tiene cadenas de aumento, es decir que no existen caminos de s a t. Definamos $S = \{v \in V que\ existe\ un\ camino\ entre\ s\ y\ v\ en\ G_f\}$. y $T = V - S$ es obvio que $t \notin S$ ya que no existe cadena de aumento. Para todo par de vértices u y v tales que $u \in S$ y $v \in T$ $f(u,v) = c(u,v)$ ya que si existiera un camino de u a v, v estaría en S. Entonces por el lema 3.1.4 $|f| = f(S,T) = c(S,T)$.

(3) ⇒ (1): Por el corolario anterior sabemos que para cualquier corte que se coja, $|f| \leq c(S,T)$, entonces si $|f| = c(S,T)$ implica que el flujo es máximo.

□

3.3. Algoritmo general de Ford-Fulkerson

Presentaremos ahora una implementación del método de Ford-Fulkerson basándose en ir aumentado poco a poco la función flujo añadiéndole las cadenas en aumento obtenidas de la red residual. El pseudocódigo sería el siguiente:

```
function FORD-FULKERSON-METHOD(G,s,t)
    for para cada (u,v) ∈ E[G] do
        f[u,v] ← 0
        f[v,u] ← 0
    end for
    while Haya un camino p de s a t en G_f do
        c_f(p) ← min{c_f(u,v) : (u,v) ∈ p}
        for para cada (u,v) ∈ p do
            f[u,v] ← f[u,v] + c_f(p)
            f[v,u] ← −f[u,v]
        end for
```

end while
return f
end function

En este pseudocódigo se puede apreciar todo lo explicado antes, ya que en las 4 primeras lineas se inicializa el flujo a 0, y en el resto, se busca una cadena de aumento p en la red residual G_f y se modifica la red de flujo f sacando las operaciones del resultado del corolario 3.1.3, cuando finalmente no se encuentra una cadena de aumento p, sale del while y se termina el algoritmo porque gracias al teorema de flujo máximo y del corte mínimo tenemos que f es máximo.

Cabe destacar que este método solo funciona correctamente con números enteros pero en el caso de que las capacidades sean racionales, se realiza un cambio de escala para pasarlos a enteros y solucionado el problema. En cambio si las capacidades son números irracionales no se puede hacer nada al respecto, pero también es sabido que un número irracional no se puede guardar con una precisión exacta en un ordenador, por tanto es altamente improbable que el problema tenga capacidades irracionales.

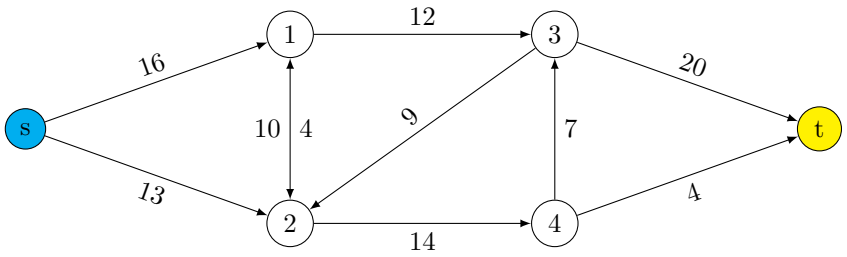

Figura 6: Red de Flujo.

Esta figura nos muestra la red de flujo inicial en la que se muestra el problema (En caso de duda de la capacidad de alguna arista es la misma que la figura 1). En las siguientes imágenes, se mostrará la alteración de la red de flujo con el flujo f provocada por la ejecución del algoritmo, que es añadirle a f el correspondiente f_p de la cadena de aumento de esa iteración. Inicialmente la función de flujo $f = 0$, las cadenas en aumento se obtienen de las redes residuales.

1ª Iteración del bucle: La cadena en aumento es el camino formado por (s,1,3,2,4,t) en ese orden. Cuyo valor mínimo de capacidad es 4, realizando las cuentas f queda como en el dibujo.

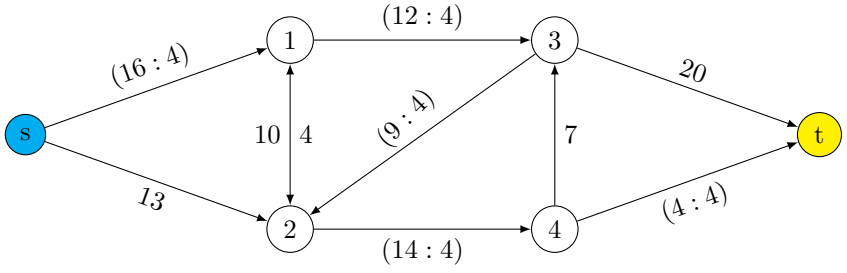

Figura 7: 1ª Iteración del bucle.

2ª Iteración del bucle: La cadena en aumento es el camino formado por (s,1,2,4,3,t) en ese orden. Cuyo valor mínimo de capacidad es 7, realizando las cuentas f queda como en el dibujo, que es exactamente sumar 7 a las aristas correspondientes a la cadena en aumento.

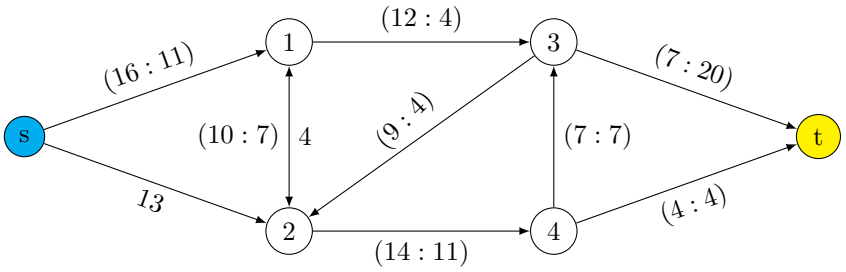

Figura 8: 2ª Iteración del bucle.

3ª Iteración del bucle: La cadena en aumento es el camino formado por (s,2,1,3,t) en ese orden, cuyo valor mínimo de capacidad es 8. f queda como en el dibujo, sumando 8 a las aristas correspondientes a la cadena en aumento, salvo que ya hubiese un valor de flujo en dirección contraria a esos vértices (Como en el caso de 1 y 2), en ese caso como el flujo que mostramos es el positivo $(8 - 7 = 1 > 0)$ pues mostramos el que va de 2 a 1.

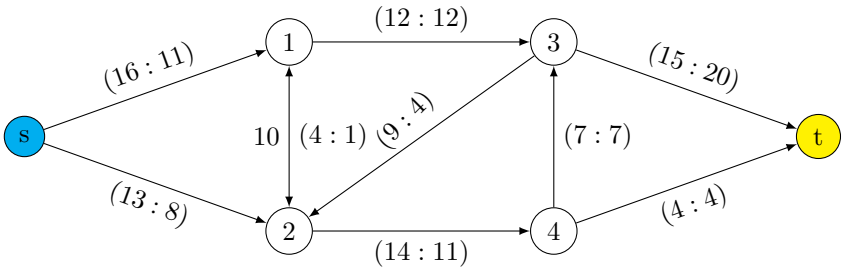

Figura 9: 3ª Iteración del bucle.

4ª Iteración del bucle: La cadena en aumento es el camino formado por (s,2,3,t) con valor mínimo de capacidad 4; realizando las cuentas f queda como en el dibujo, que es exactamente sumar 4 a las aristas correspondientes a la cadena en aumento, en este caso con la arista de 2 a 3 se cancela con su opuesta y queda flujo cero, por eso el valor del flujo en esa arista "desaparece" ya que no es estrictamente positivo y no se muestra.

Finalmente al realizar la 5ª iteración no se encuentra una cadena en aumento entonces el flujo es máximo con $|f| = 11 + 12 = 23$ que es la cantidad de flujo que sale de s.

La complejidad de este algoritmo viene estrictamente determinada por la manera en la que se implemente la condición del while que encuentra una cadena de aumento p. Si la manera de encontrar las cadenas en aumento es con una búsqueda en anchura la complejidad pasa a ser en tiempo polinómico.

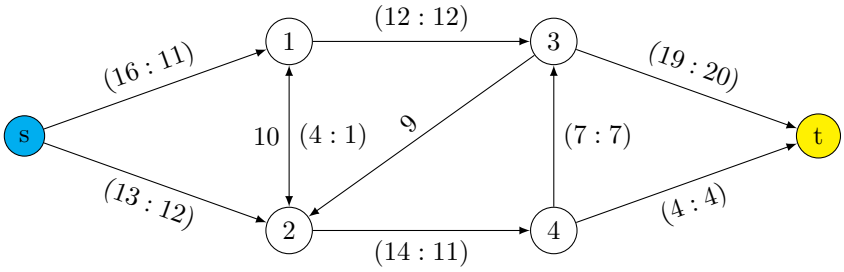

Figura 10: 4ª Iteración del bucle.

Observemos que si suponemos que la condición del while es constante, el tiempo de ejecución del algoritmo es de $O(E|f^*|)$, siendo f^* la función de flujo máximo, ya que el primer for se ejecuta exactamente E veces y el while como mucho se ejecuta $|f^*|$ suponiendo que las cadenas de aumento solo aumentan la función de flujo en una unidad. Pero si tuviésemos solamente esto en cuenta, la complejidad real sería O($\max\{E, |f^*|\}$). Para ver que no es así observemos que tanto la condición, como lo de dentro del while tiene como mínimo coste E.

Si tomamos que la estructura de datos que se utiliza para guardar la información en este algoritmo es óptima, es decir usar una representación óptima para la red de flujo $G = (V, E)$, la cual vamos a suponer que es un grafo dirigido valorado $G' = (V, E')$ con $E' = \{(u, v) : (u, v) \in E \ o \ (v, u) \in E\}$. Obviamente las aristas de G están G'. Y es fácil ver que las aristas de G_f siendo f un flujo de G, son todas aquellas aristas de G' que cumplen que $c(u, v) - f(u, v) \neq 0$. Por lo que el tiempo de búsqueda de un camino en G_f, sea cual sea el camino buscado es de $O(V + E') = O(E)$, independientemente si se usa el recorrido en profundidad o el recorrido en anchura. Por tanto el coste de ejecución del algoritmo es de $O(E|f^*|)$.

Una posible implementación de la condición del while, basada en un recorrido DFS es la siguiente:

```
function HAY-UNA-CADENA-EN-AUMENTO(G,s,t)
    marcados = bool[G.V()]
    aristaDe = aristas[G.V()] //Un vector con aristas
    q = Cola < int > ()
    marcados[s] = true
    q.meter(s)
    while q no este vacia do
        v ← q.cogerPrimero()
        q.borrarPrimero()
        for para cada e ∈ G.adj(v) do
            w ← e.destino()
            if e.capacidadResidual() > 0 and !marcados[w] then
                marcados[w] = true
                aristaDe[w] = e
                q.meter(w)
            end if
        end for
    end while
```

return *marcados*[*t*]
end function

4. Algoritmo de Edmonds-Karp

Este algoritmo está basado en el método descrito anteriormente pero la parte de la condición del while viene determinada por un recorrido en anchura, teniendo en cuenta que la cadena en aumento encontrada, debe ser la más corta desde s a t cuyas aristas en G_f tienen capacidad 1. Cuando la condición del método se resuelve de esa manera, el algoritmo recibe el nombre de Edmonds-Karp que tiene un tiempo de ejecución de $O(VE^2)$; probémoslo.

Lema 4.1
Dada una red de flujo $G = (V, E)$, en la que se ejecuta el algoritmo de Edmonds-Karp, entonces $\forall v \in V - \{s, t\}$ el camino de menor distancia $\delta_f(s, v)$ en la red residual G_f aumenta monótonamente (solo crece) con cada aumento del flujo.

Demostración. Supongamos que $\exists v \in V - \{s, t\}$, tal que hay una cadena de aumento que al aumentar el flujo f provoca que la distancia de s a v decrezca. Denotemos f al flujo anterior y f' al flujo aplicándole esa cadena de aumento. Entonces $\delta_{f'}(s, v) < \delta_f(s, v)$, sea p el camino más corto de s a u tal que $(u, v) \in E_{f'}$ y que $\delta_{f'}(s, u) = \delta_{f'}(s, v) - 1$,es decir que $p \cup (u, v)$ es el camino de s a v.

Además sabemos, por como hemos elegido u y v, la distancia de s a u no disminuye, al cambiar el flujo. Por tanto $\delta_{f'}(s, u) \geq \delta_f(s, u)$. Podemos también suponer que $(u, v) \notin E_f$, porque de no ser así tendríamos que $\delta_f(s, v) \leq \delta_f(s, u) + 1 \leq \delta_{f'}(s, u) + 1 = \delta_{f'}(s, v)$. Y entonces habría una contradicción.

Veamos que ocurre si $(u, v) \notin E_f$. Tal y como está definido el algoritmo de Edmonds-Karp el flujo de v a u ha tenido que aumentar y por lo tanto en la red residual G_f tiene como última arista (v, u), entonces $\delta_f(s, v) = \delta_f(s, u) - 1 \leq \delta_{f'}(s, u) - 1 = \delta_{f'}(s, v) - 2$ por tanto llegaríamos a la misma contradicción que antes por lo que v no podría existir. □

Teorema 4.2
Dada una red de flujo $G = (V, E)$, en la que se ejecuta el algoritmo de Edmonds-Karp, entonces el número de veces que se aumenta el flujo es de $O(VE)$.

Demostración. Decimos que una arista $(u, v) \in G_f$ es crítica en una cadena de aumento p si $c_f(p) = c_f(u, v)$. Es obvio que por la manera en la que se elige $c_f(p)$ entonces existe al menos una arista crítica en p y que además esa arista al aumentar el flujo va a "desaparecer"de la red residual. Veamos que para cada arista como mucho puede ser crítica $|V|/2 - 1$ veces.

Cojamos dos vértices u y v que estén conectados y además cuya arista (u, v) es crítica por primera vez en una cadena de aumento, entonces tenemos que $\delta_f(s, v) = \delta_f(s, u) + 1$.

Una vez que ya se ha añadido el flujo de esa arista crítica, esa arista ya no puede volver a aparecer en la red residual salvo que en el último aumento de flujo, la arista en sentido contrario (v, u), apareciese en la cadena de aumento. Y en tal caso $\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1$ siendo f' el flujo en ese momento.

Como sabemos por el lema que acabamos de demostrar las distancias no pueden disminuir sino que crecen monótonamente por lo que $\delta_{f'}(s,u) = \delta_{f'}(s,v) + 1 \geq \delta_f(s,v) + 1 = \delta_f(s,u) + 2$.

Por lo tanto la distancia de s a u como mucho aumenta en 2 y como se trata de los caminos más cortos desde s a u entonces entre medias no puede estar ni s, ni u, ni t. Entonces, la distancia desde s hasta v es como mucho de $|V|-2$ entonces esa arista como mucho puede ser crítica $(|V|-2)/2 = |V|/2-1$. Como hay E aristas que pueden pertenecer a la red residual, y cada cadena de aumento tiene por lo menos una arista crítica, entonces tenemos que el número de aumentos de flujo en el algoritmo de Edmonds-Karp es de $O(VE)$. □

Podemos observar que si la cadena de aumento se encuentra con el recorrido en anchura y la parte interna del bucle en el método de Ford-Fulkerson es de coste E, entonces el algoritmo de Edmonds-Karp tiene de coste $O(VE^2)$.

5. Problemas relacionados

En esta sección trataremos un problema muy usado en la investigación operativa, ya que el algoritmo que lo resuelve se usa mucho en un problema que se llama Problema de Asignación. Además probaremos que el cardinal máximo de un emparejamiento en un grafo bipartito coincide con el valor del flujo máximo en ese mismo grafo añadiéndole una superfuente y un supersumidero. También comentaremos que existen dos tipos de algoritmos distintos que resuelven el problema del flujo máximo con una complejidad distinta a la demostrada anteriormente, pero que el estudio de dichos algoritmos requieren una exhaustiva explicación que no son objeto este trabajo.

5.1. Problema de emparejamiento máximo en grafo bipartitos

Aquí desarrollaremos lo que se llama el problema de emparejamiento máximo en grafos bipartitos, que consiste en dado un grafo bipartito encontrar un conjunto de aristas de cardinal máximo que cumplan unas ciertas características.

Dado un grafo (asumiremos en esta sección que es no dirigido) $G = (V, E)$, se denomina *emparejamiento* M a un conjunto de aristas tal que $M \subseteq E$, para cualquier vértice de V como mucho tiene una arista incidente en el emparejamiento. Se dice que un vértice es *saturado* cuando tiene aristas incidentes y se dice *no saturado* cuando no tiene aristas incidentes. Se dice que el emparejamiento es máximo si el cardinal lo es, es decir que para cualquier otro emparejamiento M' del mismo grafo, se cumple que $|M| \geq |M'|$.

Asumiremos que los grafos $G = (V, E)$ son bipartitos a partir de ahora en este apartado, es decir $\exists L, R \subset V$ (L=left(izquierda), R=Right(derecha)) tales que $V = L \cup R$ y $L \cap R = \emptyset$ y que todas las aristas de E van de un vértice de L a uno de R. Un ejemplo de un grafo bipartito es el mostrado en la figura 11.

Con el método de Ford-Fulkerson es posible encontrar un emparejamiento máximo en un grafo bipartito en tiempo polinómico en $|V|$ y $|E|$. El truco consiste en construir una red de flujo $G' = (V', E')$ a partir del grafo bipartito. La manera de realizar esa construcción es la siguiente, se toma el conjunto de vértices V y se le añaden dos vértices s y t, $V' = V \cup \{s \cup t\}$, que harán de

superfuente y supersumidero aunque tales conceptos no existen como tal, ya que al ser no dirigido no se pueden tratar como tal. Además se cogen esos dos nuevos vértices y se unen los vértices de L con s y los vértices de R con t y así encontramos la red de flujo que buscábamos, $E' = E \cup \{(s,u) : u \in L\} \cup \{(v,t) : v \in R\}$. Queda ilustrado en la figura 12.

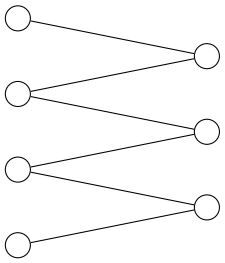

Figura 11: Grafo Bipartito.

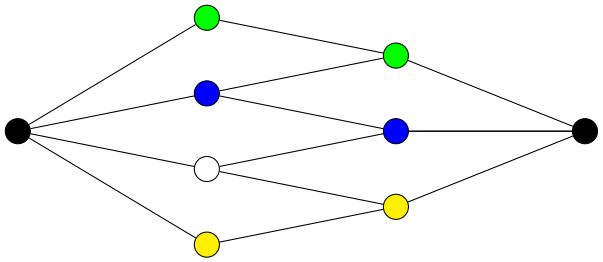

Figura 12: Grafo Bipartito con un supersumidero y una superfuente.

Por último como una red de flujo necesita una función capacidad, para terminar esta construcción se le asigna a todas las aristas una capacidad unitaria.

Se puede observar en el siguiente lema, que un emparejamiento en G corresponde directamente a un flujo f en la red de flujo G'. Diremos que f es de valor entero si $f : V \times V \longrightarrow \mathbb{Z}$ en vez de estar definida en $\mathbb{R}$.

Lema 5.1.1 Sea $G = (V, E)$ un grafo bipartito, y sea $G' = (V, E)$ su correspondiente red de flujo. Si tenemos que M es un emparejamiento, entonces existe f flujo de valor entero en G' tal que $|f| = |M|$. Y si tenemos un flujo f' de valor entero en G' entonces existe un emparejamiento M' tal que $|f'| = |M'|$.

Demostración. Dividamos la demostración en dos partes. En la primera parte demostraremos que un emparejamiento M en G corresponde con un flujo f en G' de valor $|f| = |M|$, y después mostraremos el recíproco es decir que un flujo corresponde con un emparejamiento.

Primera parte: Dado un emparejamiento M, definamos el siguiente flujo f, tal que si $(u,v) \in M$ entonces $f(s,u) = f(u,v) = f(v,t) = 1$ y $f(u,s) = f(v,u) = f(t,v) = -1$ Para todas aquellas aristas que no estén en M tenemos que $f(u,v) = 0$.

Observamos que $\forall (u,v) \in M$ corresponde con 1 unidad de flujo en G' además como estamos en un grafo bipartito, las aristas del emparejamiento van de un

grupo de vértices a otro por lo que el flujo neto en el corte formado por $(L \cup \{s\}, R \cup \{t\})$ nos muestra que $f(L \cup \{s\}, R \cup \{t\}) = |M| = |f|$ la última igualdad se da por el lema 3.1.4.

Segunda parte: Dado un flujo f en G' de valor-entero y sea $M = \{(u, v) : u \in L,\ v \in R,\ f(u, v) > 0\}$. Entonces por tal y como hemos definido G' al principio de esta sección, sabemos que los vértices de L tienen como mucho cada uno una arista entrante que viene de s. Entonces por ser f flujo, por la conservación de flujo, solo tienen una arista saliente con flujo positivo. Además como hemos supuesto que es de valor entero, sabemos que de esos vértices tienen solo una arista saliente como mucho. Por tanto las que tengan una entrante deben tener una saliente de los vértices de L. Pero es cierto que este razonamiento se puede hacer de manera simétrica para vértices de V por tanto, las aristas de M forman un emparejamiento. Ahora sabemos por la primera parte que con el corte de (L, R) tenemos que $f(L, R) = |M|$ veamos que también es igual a $|f|$ para ello desarrollemos esa ecuación. Como las intersecciones de cualquiera de estos cuatro conjuntos L, R, s y t, entre sí son vacíos por el lema 2.3.1 podemos escribir esa ecuación como:

$|M| = f(L, R) = f(L, V') - f(L, L) - f(L, s) - f(L, t)$, ahora por conservación de flujo sabemos que $f(L, V') = 0$, además sabemos que $f(L, L) = 0$ y como no hay aristas de L a t, $f(L, t) = 0$, entonces la ecuación queda como $|M| = f(L, R) = -f(L, s) = f(s, L) = f(s, V') = |f|$, la penúltima igualdad se da porque todas las aristas que salen de s van a parar a L por lo tanto se puede extender a todo V'. Y con eso ya tenemos el resultado buscado. □

Teorema 5.1.2

En una red de flujo $G = (V, E)$ si la función de capacidad toma solo valores enteros, entonces el flujo máximo f producido con el método de Ford-Fulkerson tiene un valor de flujo $|f|$ que es también un número entero. Además $f(u, v) \in \mathbb{Z}\ \forall u, v \in V$.

La demostración consiste en probarlo por inducción en el número de iteraciones del bucle.

Corolario 5.1.3 El cardinal del emparejamiento máximo de un grafo bipartito G coincide con el valor del flujo máximo en su red de flujo G' asociada.

Demostración. Supongamos que existe un emparejamiento M de cardinal máximo en G pero que su flujo f asociado en G' no es máximo. Entonces existe un flujo f' en G' que si es máximo y por el teorema 5.1.2 sabemos que f' es de valor entero. Entonces $|M| = |f| < |f'| = |M'|$ siendo M' el emparejamiento asociado a f' pero entonces llegamos a la contradicción de que M no es máximo. La demostración dado un flujo máximo y suponiendo que el emparejamiento no lo es, es análogo. □

En la figura 12 podemos observar un emparejamiento máximo que es igual que el flujo máximo, aunque es fácil ver que no es único, ya que en este caso esta formado por las aristas que unen los vértices del mismo color, y se pueden cambiar de color el vértice azul con el blanco o el blanco con el amarillo y entonces quedaría otro emparejamiento pero de igual cardinal; los vértices negros solo se deben tener en cuenta si se considera el flujo ya que no pertenecían al grafo bipartito original si no solamente a la red.

5.2. Algoritmos de Push-Relabel y de Relabel-To-Front

Introduzcamos un poco qué son cada uno de estos tipos de algoritmos.

Los algoritmos Push-Relabel resuelven el problema del flujo máximo, pero el enfoque que se toma es ligeramente distinto al usado en el método de Ford-Fulkerson ya que en vez de buscar en toda la red residual del flujo f, se centran en un vértice cada vez mirando solo en sus vecinos. El problema es que, durante la ejecución de este tipo de algoritmos no se respeta la conservación de flujo, sino que se mantiene otro tipo de flujo denominado de preflujo, que consiste en una misma función de flujo que respeta las dos primeras propiedades de los flujos y una tercera que consiste en que $f(V, u) \geq 0 \ \forall u \in V - \{s\}$. Estos algoritmos tienen un tiempo de ejecución la mayoría de $O(V^2E)$.

Por último los algoritmos Relabel-To-Front son una variación de los anteriores, que consiguen un tiempo de ejecución de $O(V^3)$. Este tipo de algoritmos tiene una lista de vértices de la red de flujo. Entonces se va recorriendo la lista y se cogen aquellos vértices tales que $f(V, u) > 0 \ \forall u \in V - \{s, t\}$ y se "descargan" ejecutando el algoritmo de push-relabel hasta que $f(V, u) = 0$. Y después de que el vértice sea "descargado" se coloca al principio de la lista y se empieza la búsqueda de nuevo.

6. Conclusión

En conclusión podemos observar que el método de Ford-Fulkerson y el algoritmo de Edmonds-Karp son buenos algoritmos para resolver este problema, sobretodo cuando la red de flujo con la que se esta realizando es una red de flujo que está entre ser dispersa y ser densa como un lugar intermedio, además de que históricamente fueron de los primeros.

Mientras que los algortimos de push-relabel se usan más bien para redes de flujo muy dispersas y en cambio los algoritmos relabel-front son sobretodo para redes de flujo muy densas, ya que el coste no depende de las aristas sino del número de vértices.

7. Bibliografía

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Capítulo 26 "Maximum Flow".

Robert Sedgewick, Kevin Wayne. *Algorithms*, Fourth Edition. Pearson Education, Inc. Capítulo 6.4 "Network-flow algorithms".