Estudio sobre las redes de flujo s–t y el algoritmo de Ford–Fulkerson para hallar $f_{\text{máx}}$

POR RONALDO GLIGAN

rgomes21@alumno.uned.es

Resumen

En numerosas ocasiones, surge la necesidad de modelar el flujo de diferentes entidades, sean estos líquidos, tráfico de vehículos, telecomunicaciones, carga eléctrica u otros agentes, a través de un medio determinado. En estos casos, suele aparecer el problema de encontrar la ruta de mayor caudal para transportar un agente desde un punto inicial de la red hasta otro punto de la misma, lo que nos conduce a la siguiente pregunta: ¿Cómo hallar de manera sistemática la ruta óptima para lograr el máximo transporte dentro de la red?

En este texto, introduciremos los conceptos necesarios para plantear el problema de manera rigurosa y luego presentaremos un algoritmo que selecciona la ruta de *flujo máximo*.

Definición 1. RED DE FLUJO s–t.

*Una **red de flujo s–t** es un grafo dirigido $G=(V,E)$ con dos vértices característicos: uno llamado **fuente** y denotado por s, y el otro llamado **sumidero** e indicado por t.*

*Toda arista $uv \in E$ posee una **capacidad** $c(uv) \in \mathbb{R}_{\geqslant 0}$, la cual indica el flujo máximo que la arista uv permite. Si no existe la arista uv, entonces su capacidad es cero, $c(uv)=0$.*

Ejemplo 2. La Figura 1 representa una red de flujo.

Ejemplo 3. En la Figura 1, $c(ac)=3$, mientras que $c(cb)=1$. Implícitamente, $c(ca)=0$.

Objetivo. Dada una red de flujo $G=(V,E)$ con fuente s, sumidero t y capacidades $c(uv)$, $\forall uv \in E$, hallar el camino de extremos s,t que permite el mayor flujo.

Para ello, consideraremos la aplicación flujo f,

$$f : E \longrightarrow \mathbb{R}_{\geqslant 0},$$

que indica cuán saturada se encuentra una arista cualsea de E. Es natural afirmar que $\forall uv \in E$ se debe dar que el flujo sea estrictamente positivo y que

$$f(uv) \leqslant c(uv).$$

Esto es, que el flujo nunca puede ser mayor que la capacidad pues, si así fuera, estaríamos sobresaturando la arista uv. También, damos por hecho que los vértices no pueden retener flujo y que, consecuentemente, su capacidad es nula.

$$\forall v \in V \ : \ c(v)=0.$$

Por conveniencia, definiremos para cada vértice $v \in V$, dos conjuntos asociados: los flujos entrantes y los flujos salientes de v.

Definición 4. Conjunto de Flujos Entrantes y Salientes.

Definimos $F(v)$ para cada $v \in V$ como el conjunto de ***flujos entrantes*** *de v.*

$$F(v) = \{u \in V \mid \exists uv \in E\}$$

Análogamente, $F'(v)$ es el conjunto de ***flujos salientes*** *de v.*

$$F'(v) = \{w \in V \mid \exists vw \in E\}$$

Ejemplo 5. En la Figura 1, $F(c) = \{a, b\}$ y $F'(c) = \{b, t\}$.

Nota 6. El lector podrá encontrar en otros textos, comúnmente en inglés, que $F(v) = \text{In}(v)$ y $F'(v) = \text{Out}(v)$.

Nota 7. En nuestra exposición, estableceremos $F(s) = \varnothing$ y $F'(t) = \varnothing$ y en una sección posterior hablaremos sobre redes *multifuente* y *multisumidero*. Esto significa que no habrá flujo entrante hacia la fuente ni flujo saliente desde el sumidero. Se realiza esta decisión por dos motivos: uno lógico; pues en muchas ocasiones no tiene sentido que se escape flujo del sumidero, y otro para simplificar las demostraciones, aunque se podría abordar el problema permitiendo la existencia de aristas entrantes a s y salientes de t.

Como los vértices tienen capacidad nula, entonces todo flujo entrante debe ser igual al saliente (Figura 3). Es decir, el flujo se conserva con excepción de los vértices s y t (Figura 2, Figura 4), que tienen flujo entrante nulo y flujo saliente nulo respectivamente.

$$\forall v \notin \{s, t\} \ : \ \sum_{u \in F(v)} f(uv) \ = \ \sum_{w \in F'(v)} f(vw)$$

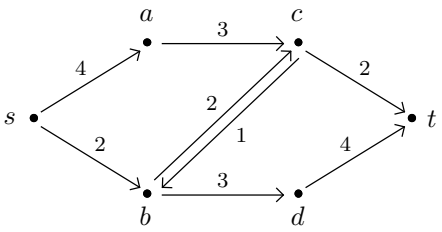


Figura 1. Red de flujo con fuente s y sumidero t. Cada arista tiene anotada su capacidad.

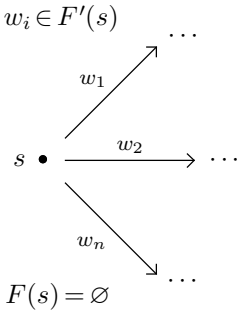


Figura 2. s no tiene flujo entrante.

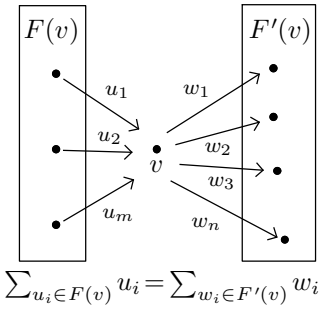


Figura 3. Conservación del flujo.

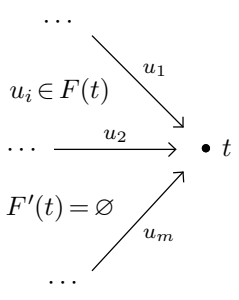


Figura 4. t no tiene flujo saliente.

Así, resumimos las propiedades de f en la Definición 8:

Definición 8. Función de Flujo.

Sea $G=(V,E)$ una red de flujo s–t. La aplicación

$$f : E \longrightarrow \mathbb{R}_{\geqslant 0}$$

denota el flujo que transita en cada vértice. Esta cumple las siguientes propiedades:

Restricción de capacidad. $\forall uv \in E: \quad 0 \leqslant f(uv) \leqslant c(uv)$.

Conservación del flujo. $\forall v \in V - \{s,t\}: \quad \sum_{u \in F(v)} f(uv) = \sum_{w \in F'(v)} f(vw)$.

De la Definición 1, podemos deducir que si una arista no existe, entonces su capacidad es cero. Si una arista uv no existe, entonces $c(uv)=0$ y de la conservación del flujo de la Definición 8: $0 \leqslant f(uv) \leqslant c(uv) \Longrightarrow 0 \leqslant f(uv) \leqslant 0$. Ergo, el flujo de una arista inexisteste es nulo.

Ahora podemos concretar un poco más nuestro objetivo. Este es hallar el valor máximo de f, que llamaremos $f_{\text{máx}}$, y luego un camino de s hasta t que tenga flujo $f_{\text{máx}}$. El flujo $f_{\text{máx}}$ se caracterizará en el Teorema 22.

Redes residuales

Con el propósito de optimizar el flujo, definiremos la noción de *capacidad residual* como la cantidad de flujo que aún puede circular por una arista.

Definición 9. Capacidad Residual.

*Sea $G=(V,E)$ un red s–t. Para toda arista $uv \in E$, se define su **capacidad residual** como la diferencia de la capacidad $c(uv)$ menos el flujo $f(uv)$ que recorre uv.*

$$c_f(uv) = c(uv) - f(uv), \quad \forall uv \in E.$$

Asimismo, definimos una *red residual* como la red pareja de una red de flujo, idéntica en vértices y en aristas, pero con una única diferencia; sólo se consideran las capacidades residuales. Véase la Figura 24.

Definición 10. Red Residual.

*Dada una red s–t $G=(V,E)$, se define su **red residual** correspondiente $G_f(V,E_f)$ como la red s–t con los mismos vértices y aristas, solo que se considera la capacidad residual c_f en vez de c.*

Un *camino incremental* es un camino del grafo G_f tal que ninguna capacidad residual sea nula.

Definición 11. Camino Incremental.

*Sea $G=(V,E)$ una red s–t. Un **camino incremental** es un camino $(u_1u_2,\ldots,u_{k-1}u_k)$, $u_iu_{i+1} \in E_f$, $c_f(u_iu_{i+1})>0$, en la red residual $G_f=(V,E_f)$.*

El *cuello de botella* de un camino P es la máxima capacidad que puede fluir por P, que es la mínima capacidad residual de P.

Definición 12. Cuello de Botella.

Sea $P = (u_1\, u_2, \ldots, u_{k-1}\, u_k)$ *un camino incremental de* G_f. *Se define el cuello de botella de* P, botella(P), *como*

$$\text{botella}(P) = \min\{c_f(uv) \mid uv \in P\}.$$

Más adelante, veremos que una red es de flujo máximo si y solo si no existen caminos incrementales en su red residual.

Cortes

Ahora presentaremos el concepto de *corte* en una red s–t. Intuitivamente, hacer un corte s–t, como el de la Figura 5, es eliminar aristas de tal manera que se obtengan dos conjuntos disjuntos denominados V_s y V_t, que se cumpla que $s \in V_s$, $t \in V_t$ y que no quede ninguna arista de ningún vértice de V_s a V_t ni viceversa. Las aristas eliminadas que se «encontraban» entre V_s y V_t, estarán contenidas en $V_s\, V_t$ y aquellas con dirección V_t a V_s en $V_t\, V_s$.

Definición 13. Corte s–t.

Sea $G = (V, E)$ *una red de flujo* s–t. *Un* ***corte s–t*** *es un par* (V_s, V_t) *donde se cumplen las siguientes propiedades*

1. $V_s, V_t \subset V$,
2. $V_s \cap V_t = \varnothing, \quad V_s \cup V_t = V$,
3. $s \in V_s, \quad t \in V_t$,
4. $V_s\, V_t,\ V_t\, V_s \subset E$,
5. $V_s\, V_t = \{uv \mid u \in V_s \wedge v \in V_t\}, \qquad V_t\, V_s = \{vu \mid u \in V_s \wedge v \in V_t\}$.

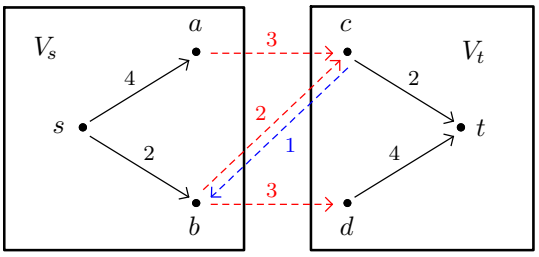


Figura 5. $(V_s, V_t) = (\{s, a, b\}, \{c, d, t\})$, $V_s\, V_t = \{ac, bc, bd\}$ (en rojo) y $V_t\, V_s = \{cb\}$ (en azul) es un posible corte s–t de la Figura 1.

Proposición 14. *$F'(s)$ y $F(t)$ forman cortes s–t.*

Demostración de 14. Basta comprobar que $F'(s)$ y $F(t)$ cumplen con la definición de corte s–t. Sea $G=(V,E)$ una red de flujo s–t.

Corte s–t creado por $F'(s)$ Este es el corte (V_s, V_t, X) con

$$V_s=\{s\}, \quad V_t=V-\{s\}$$

que cumple con la definición de corte: $s\in\{s\}=V_s$, $t\in V-\{s\}=V_t$, $V_s\cap V_t=\{s\}\cap(V-\{s\})=\varnothing$ y $V_s\cup V_t=\{s\}\cup(V-\{s\})=V$.

Corte s–t creado por $F(t)$ Tal corte implicaría que

$$V_s=V-\{t\}, \quad V_t=\{t\}$$

y se procedería de igual manera que en el caso de $F'(s)$. □

Consideremos ahora la capacidad que existe entre V_s y V_t al hacer un corte s–t en la red. Naturalmente, esta capacidad se define como la suma de las capacidades de los vértices que atraviesan el corte, es decir, las capacidades entre las partes cortadas.

Ejemplo 15. En el corte de la Figura 5, las aristas rojas representan los elementos de V_sV_t con capacidad $c(V_sV_t)=c(ac)+c(bc)+c(bd)+c(ca)=3+2+3+1=9$.

El flujo de un corte es el flujo de V_sV_t —el saliente— menos el flujo V_tV_s —que regresa. En la Figura 5, si enviamos un flujo por V_sV_t puede darse el caso de que, por ejemplo, parte del flujo enviado por ac regrese por cb. Así, para ajustarlo, se debe restar.

Definición 16. Capacidad y Flujo de un Corte s–t.

*Sea $G=(V,E)$ una red de flujo s–t y (V_s,V_t) un corte s–t cualquiera. Se define la **capacidad de un corte** como la suma de todas las capacidades de V_sV_t.*

$$c(V_sV_t)=\sum_{uv\in V_sV_t} c(uv).$$

*Asimismo, el **flujo a través del corte** es la suma de todos los flujos de V_sV_t menos aquellos que regresan por V_tV_s.*

$$f(V_sV_t)=\sum_{uv\in V_sV_t} f(uv)-\sum_{uv\in V_tV_s} f(uv).$$

Ahora procederemos a demostrar tres resultados que nos permitirán realizar una caracterización entre cortes y flujo máximo.

Observación 17. La restricción de capacidad de la Definición 8 también se cumple en cortes s–t arbitrarios: el flujo de un corte está acotado superiormente por la capacidad de dicho corte. Esto queda reflejado en la Proposición 18:

Proposición 18. Restricción de Capacidad en Cortes s–t.

Sea (V_s,V_t) un corte s–t. Entonces

$$0\leqslant f(V_sV_t)\leqslant c(V_sV_t).$$

Demostración de 18.

$$\begin{aligned}
0 \leqslant f(V_s V_t) &= \sum_{uv \in V_s V_t} f(uv) - \sum_{uv \in V_t V_s} f(uv) && \text{(Definición 16)} \\
&\leqslant \sum_{uv \in V_s V_t} f(uv) && \text{(El sustraendo no es negativo)} \\
&\leqslant \sum_{uv \in V_s V_t} c(uv) && \text{(Consecuencia de la Definición 8)} \\
&= c(V_s V_t). && \text{(Definición 16)}
\end{aligned}$$

$\therefore 0 \leqslant f(V_s V_t) \leqslant c(V_s V_t).$ □

Proposición 19. *Sea (V_s, V_t) un corte s–t y v un vértice de V_t. Entonces el corte s–t (V_s', V_t') con $V_s' = V_s \cup \{v\}$ y $V_t' = V_t - \{v\}$ tiene el mismo flujo que el corte (V_s, V_t).*

$$f(V_s V_t) = f(V_s' V_t') \quad \textit{con } V_s' = V_s \cup \{v\},\ V_t' = V_t - \{v\}.$$

Demostración de 19. Considere los flujos entrantes y salientes de v: $F(v)$ y $F'(v)$ respectivamente. El corte s–t no solo separa los vértices, sino que segrega $F(v)$ y $F'(v)$ dependiendo de hacia dónde apunten las aristas. Sean

$$F_s(v) = \{uv \in F(v) \mid u \in V_s\}, \quad F_t(v) = \{uv \in F(v) \mid u \in V_t\},$$

$$F_s'(v) = \{vw \in F'(v) \mid w \in V_s\}, \quad F_t'(v) = \{vw \in F'(v) \mid w \in V_t\}.$$

Nótese que $F_s(v) \cup F_t(v) = F(v)$ y $F_s'(v) \cup F_t'(v) = F'(v)$.

El flujo del corte $(V_s' = V_s \cup \{v\}, V_t' = V_t - \{v\})$ será el flujo de $V_s V_t$ menos la «contribución» de v, pero ese flujo se «irá» a V_s'. Por ello,

$$\begin{aligned}
f(V_s' V_t') &= f((V_s \cup \{v\})(V_t - \{v\})) && \text{(Por hipótesis)} \\
&= \left[\begin{array}{rl} f(V_s V_t) & - \left(\displaystyle\sum_{u \in F_s(v)} f(uv) + \sum_{u \in F_t(v)} f(uv)\right) \\ & + \left(\displaystyle\sum_{w \in F_s'(v)} f(vw) + \sum_{w \in F_t'(v)} f(vw)\right) \end{array}\right] && \text{(«Contribución» de } v \text{ «cambiada», antes explicado)} \\
&= f(V_s V_t) - \left(\sum_{u \in F(v)} f(uv)\right) + \left(\sum_{w \in F'(v)} f(vw)\right) && \left(\begin{array}{l} \because F_s(v) \cup F_t(v) = F(v), \\ \because F_s'(v) \cup F_t'(v) = F'(v) \end{array}\right) \\
&= f(V_s V_t). && \text{(Conservación del flujo, Definición 8)}
\end{aligned}$$

□

Si en una red de flujo cualsea (V, E) tenemos el corte $(V_s = \{s\}, V_t = (V - \{s\}))$ y $v \in V_t$ (esto es, $v \in V \wedge v \neq s$), entonces, gracias a la Proposición 19, podemos asegurar que

$$f((\{s\})(V - \{s\})) = f((\{s\} \cup V_s')(V - (\{s\} \cup V_s'))) \qquad \forall V_s' \subset V - \{t\}.$$

Informalmente, mover los vértices que se quiera de V_t a V_s no afecta al flujo, este es siempre igual al de $(\{s\}, (V - \{s\}))$. Esto es muy importante.

Como en todos los cortes el flujo es el mismo, esto nos motiva a definir el concepto de *valor* del flujo, denotado por $|f|$, de dos maneras posibles: 1) como el flujo saliente de s ó 2) como el flujo entre cualquier corte s–t, ambas recogidas en la Definición 20.

Definición 20. Valor del Flujo.

*Sea $G=(V,E)$ una red de flujo. El **valor del flujo** de la red G es*

$$\begin{aligned} |f| &= \sum_{w\in F'(s)} f(sw) \\ &= f((\{s\})(V-\{s\})) \\ &= f((V_s)(V-V_s)), \quad \forall V_s \subset V-\{t\} \wedge s\in V_s. \end{aligned}$$

De la Proposición 18 y la Definición 20 se puede deducir fácilmente la Proposición 21: el valor del flujo nunca es mayor que la capcidad de cualquier corte.

Proposición 21. Valor del Flujo Limitado por Capacidad de un Corte.

Sea $G=(V,E)$ una red de flujo s–t y (V_s,V_t) un corte cualsea de G. Se cumple que

$$|f| \leqslant c(V_s,V_t).$$

Demostración de 21.

$$\begin{aligned} f(V_sV_t) &= |f| && \text{(Definición 20)} \\ f(V_sV_t) &\leqslant c(V_sV_t) && \text{(Proposición 18)} \\ \therefore \quad |f| &\leqslant c(V_sV_t). \end{aligned}$$

□

Análizando la Proposición 21, podemos ver la capacidad de cualquier corte es *al menos* el valor del flujo de la red G. El Teorema 22 que presentaremos afirma la existencia de un corte que tiene capacidad mínima. Esto quiere decir, que $c(V_sV_t)=|f|$. Entonces, $f_{\text{máx}}$ es la capacidad del corte con capacidad mínima.

Relacción entre flujos y cortes

El siguiente teorema es de suma importancia. Este es el *Max–flow min–cut theorem*[1] (Teorema 22).

Teorema 22. *«Max-Flow Min-Cut Theorem».*

Sea $G=(V,E)$ una red de flujo s–t. Entonces, las siguientes proposiciones son equivalentes:

1. *f es $f_{\text{máx}}$.*
2. *G_f no tiene caminos incrementales.*
3. *Existe un corte (V_s,V_t) tal que $c(V_sV_t)=|f|$.*

1. En castellano: «El flujo máximo es igual al corte s–t con la menor capacidad».

Esto es, hallar el corte con mínima capacidad es encontrar la mayor cantidad de flujo desde s hasta t ($3 \implies 1$).

Demostración de 22. (Esta demostración proviene de la referencia bibliográfica [3]).

($1 \implies 2$). Sea $f = f_{\text{máx}}$ y sea G_f tal que existe un camino incremental, entonces podemos aumentar $|f|$ al «caminar» por dicho camino, lo que contradice que $f = f_{\text{máx}}$. Entonces, si el flujo f es máximo, entonces no puede exisitr ningún camino incremental en G_f.

($2 \implies 3$). Construiremos un corte (V_s, V_t) tal que $c(V_s V_t) = |f|$. Sea V_s el conjunto de todos los vértices que son accesibles desde s; si existe un camino incremental desde s hasta v, entonces $v \in V_s$. Por hipótesis, G_f no tiene caminos incrementales de s hasta t, lo que implica que $t \notin V_s$. Sea $V_t = V - V_s$. Entonces, (V_s, V_t) es un corte s–t válido. Consideraremos ahora el flujo a través de este corte (V_s, V_t).

Para cualquier arista $uv \in V_s V_t$, se debe dar que su capacidad residual sea nula, $c_f(uv) = 0$, pues, de lo contrario, existiría un camino incremental y $v \in V_s$, lo que sería una contradicción. Por la Definición 9 de capacidad residual, deducimos fácilmente que, si $c_f(uv) = 0$, entonces $c(uv) = f(uv) = 0$. Tenemos, pues

$$\begin{aligned} |f| &= f(V_s V_t) && \text{(Definición 20)} \\ &= \sum_{uv \in V_s V_t} f(uv) - \sum_{uv \in V_t V_s} f(uv) && \text{(Definición 16)} \\ &= \sum_{uv \in V_s V_t} c(uv) && (\because c(uv) = f(uv)) \\ &= c(V_s V_t). && \text{(Definición 20)} \end{aligned}$$

Por lo tanto, si G_f no tiene caminos incrementales, entonces necesariamente $|f| = c(V_s V_t)$ para algún corte (V_s, V_t).

($3 \implies 1$). (V_s, V_t) es un corte tal que $|f| = c(V_s V_t)$, entonces, por la Proposición 21, se puede concluir que f es $f_{\text{máx}}$. □

Método para encontrar caminos s–t de flujo máximo

El algoritmo de Ford–Fulkerson[2] es capaz de encontrar el camino de flujo $f_{\text{máx}}$ mediante el siguiente proceso: «Si existe algún camino incremental en G_f, mandar flujo por este. Hacer así con todos las posibles aristas y quedarse con la que mayor flujo permita. Continuar este proceso hasta que no queden más caminos sin recorrer.» Este método asegura que, al final de su ejecución, no queden caminos incrementales en G_f lo que implica que, gracias al Teorema 22 ($2 \implies 1$), el flujo máximo de Ford–Fulkerson sea de hecho el flujo máximo.

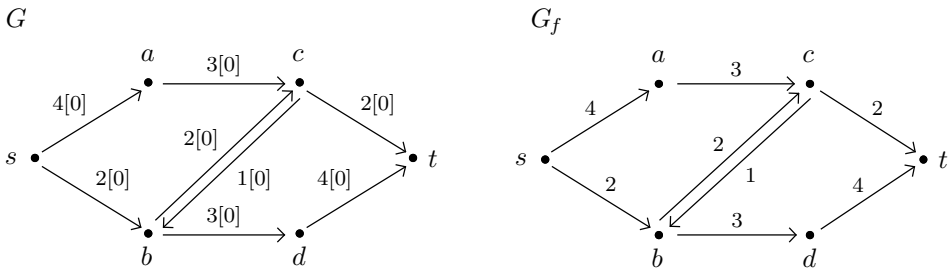
Algoritmo de Ford–Fulkerson

Ford–Fulkerson$(G = (V, E), c)$ con $s, t \in V$, $c : E \longrightarrow \mathbb{R}_{\geqslant 0}$
  $f(uv) \leftarrow 0 \quad \forall uv \in E$, (Al principio, consideramos nulos los flujos)
  Mientras exista un camino P_i de s a t en G_f tal que $c_f(uv) \neq 0 \quad \forall uv \in P_i$, *hacer* (Se puede ejecutar BFS u otro algoritmo de búsqueda para encontrar estos caminos)
    $c_f(P_i) \leftarrow \min\{c_f(uv) \mid uv \in P_i\}$, (Hallar el cuello de botella)
    Para cada vértice $uv \in P_i$, *hacer*
      $f(uv) \leftarrow f(uv) + c_f(P_i)$, (Enviamos flujo hacia v)
      $f(vu) \leftarrow f(vu) - c_f(P_1)$, (Flujo que regresa)
  Devolver f.

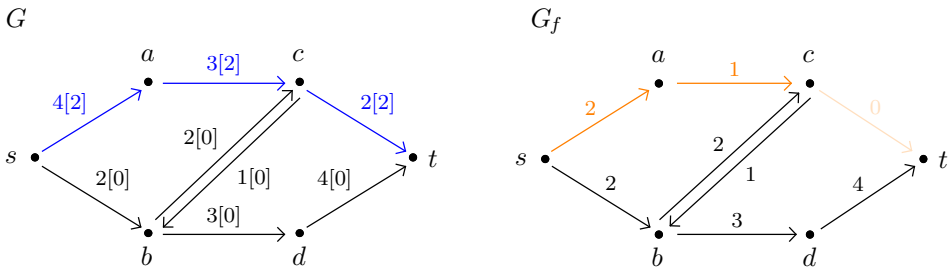
2. Llamado así en honor de sus autores: Lester Randolph Ford Jr. y Delbert Ray Fulkerson.

Convenio 23. En la representación de las redes de flujo, usaremos la notación «capacidad[flujo]» en cada arista. Por ejemplo, en la Figura 6, la arista dt, aquella que va del vétice d al vértice t, tiene capacidad 4 y flujo 3.

Ejemplo 24. Hallaremos la red de flujo máximo de la Figura 1 con el algoritmo de Ford–Fulkerson. Empezamos inicializando todos los flujos a cero. A la derecha tenemos G_f con las capacidades residuales que son las mismas que las capacidades, pues los flujos son cero. Estas irán cambiando cada vez que actualicemos los flujos en G.

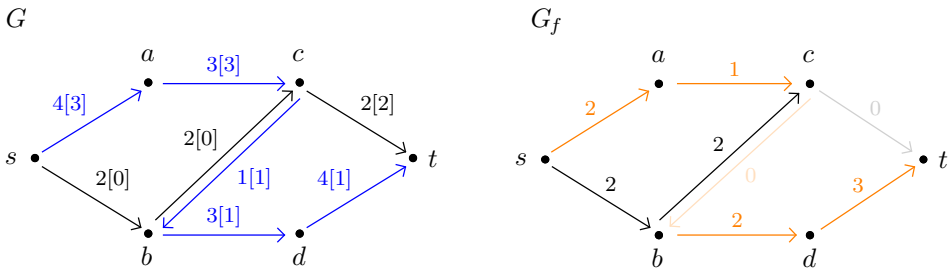


Buscamos los posibles caminos de s hasta t en G_f con cualquier algoritmo de exploración de grafos, aunque a la hora de ejecutarlo en grandes redes s–t conviene usar un algoritmo eficiente (más adelante presentamos el algoritmo de Edmonds–Karp, qué suele ser más rápido que Ford–Fulkerson). Uno de estos caminos es $P_1 = (sa, ac, ct)$ (en azul). Buscamos el cuello de botella[3] de P_1, que es botella$(P_1) = 2$, se lo restamos a cada una de las capacidades residuales en G_f y finalmente actualizamos los flujos.



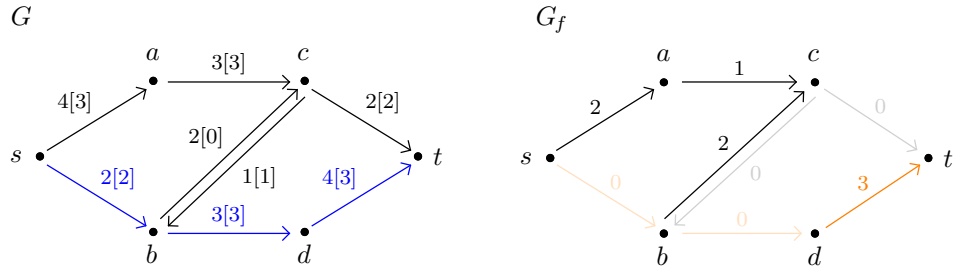
Observe como la arista cd de G_f ha desaparecido. Esto es porque al restarle botella(P_1) —el cuello de botella es exactamente dicha arista— se anula y la dejamos de considerar.

Continuamos con otro posible camino en G_f. Observe que en el estado actual no puede existir ningún camino que pase por ct, puesto que $c(ct) = 0$. Sea $P_2 = (sa, ac, cb, bd, dt)$ el siguiente camino que consideraremos. Entonces, botella$(P_2) = 1$.



Se continúa de igual manera. Vemos que hemos agotado los caminos válidos que pasan por sa. Tomaremos $P_3 = (sb, bd, dt)$ como siguiente camino. Es fácil comprobar que botella$(P_3) = 2$. Con esto, actualizamos los flujos que pasan por P_3 en G y restamos botella(P_3) en los flujos residuales.

3. Como dijo el filósofo Thomas Reid: *«Una cadena es tan fuerte como su eslabón más débil»*.



Observe que no quedan más caminos válidos (aquellos en los que ninguna de las aristas es cero) de s hasta t. El algoritmo ha terminado: hemos encontrado el flujo máximo. Vemos que $f_{\text{máx}} = \sum_{v \in F'(s)} f(sv) = f(sa) + f(sb) = 3 + 2 = 5$ (Definición 20). Además, en G queda «impregnado» el camino con $f_{\text{máx}}$ (Figura 6).

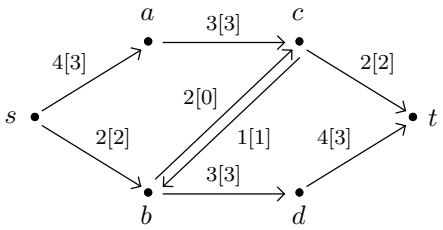


Figura 6. Red de la Figura 1 con flujos máximos.

Edmonds–Karp: generalización de Ford–Fulkerson e implementación en código

Nota 25. Es muy importante remarcar que el algoritmo de Ford–Fulkerson, tal y como lo hemos descrito arriba, no siempre llega a una conclusión. Para resolver este problema hay que tener el cuenta el flujo que regresa, que en Ejemplo 24 no se tomó en cuenta, en G_f. También, para encontrar los caminos incrementales más cortos, hace falta usar una variación de este algoritmo, llamado Edmonds–Karp, que utiliza la búsqueda BFS; de anchura.

En esencia, el algoritmo de Edmonds–Karp es una modificación del de Ford–Fulkerson que intenta hallar los caminos incrementales más cortos con esperanza acabar antes.

Algoritmo de Edmonds–Karp (Haskell)

```
-- Importamos la librería que contiene `Graph`, que representa grafos dirigidos
-- finitos mediante una matriz de adyacencia y una implementación del algoritmo
-- BFS
import Data.Graph
import Data.Graph.Inductive.Query.BFS

--             Red s-t, Fuente,   Sumidero, Flujo máximo
edmondsKarp :: Graph -> Vertex -> Vertex -> Int
edmondsKarp grafo fuente sumidero =
  let padres = replicate (length grafo) (-1) -- Este array es llenado por BFS y
almacena el camino
      flujoMaximo = 0 -- Inicialmente no hay flujo
```

```
      resultadoBFS = bfs grafo [fuente] -- Ejecutamos BFS en el grafo
comenzando desde la fuente
  in
    -- Aumentamos el flujo mientras haya un camino desde la fuente hasta el
sumidero
    if sumidero `elem` resultadoBFS then
      let flujoCamino = encontrarFlujoCamino grafo padres fuente sumidero --
Encontrar la capacidad residual mínima a lo largo del camino
          grafoActualizado = actualizarCapacidadesResiduales grafo padres
fuente sumidero flujoCamino -- Actualizar las capacidades residuales de las
aristas
          padresActualizados = map (\v -> if v == -1 then -1 else padres !! v)
[0..(length padres - 1)] -- Actualizar el arreglo de padres
      in
        flujoMaximo + flujoCamino + edmondsKarp grafoActualizado fuente
sumidero -- Llamamos recursivamente a la función con el grafo actualizado
    else
      flujoMaximo -- No hay más caminos incrementales, devolvemos el flujo
máximo

encontrarFlujoCamino :: Graph -> [Vertex] -> Vertex -> Vertex -> Int
encontrarFlujoCamino grafo padres fuente sumidero =
  let flujoCamino = foldl (\flujo v -> min flujo (capacidad grafo (padres !! v)
v)) maxBound (reverse (obtenerCamino padres fuente sumidero []))
  in
    flujoCamino

actualizarCapacidadesResiduales :: Graph -> [Vertex] -> Vertex -> Vertex -> Int
-> Graph
actualizarCapacidadesResiduales grafo padres fuente sumidero flujoCamino =
  let grafoActualizado = foldl (\g v -> establecerCapacidad g (padres !! v) v
((capacidad g (padres !! v) v) - flujoCamino)) grafo (reverse (obtenerCamino
padres fuente sumidero []))
      grafoActualizado' = foldl (\g v -> establecerCapacidad g v (padres !!
v) ((capacidad g v (padres !! v)) + flujoCamino)) grafoActualizado (reverse
(obtenerCamino padres fuente sumidero []))
  in
    grafoActualizado'

obtenerCamino :: [Vertex] -> Vertex -> Vertex -> [Vertex] -> [Vertex]
obtenerCamino padres fuente sumidero camino
  | sumidero == fuente = sumidero:camino
  | otherwise = obtenerCamino padres fuente (padres !! sumidero)
(sumidero:camino)

capacidad :: Graph -> Vertex -> Vertex -> Int
capacidad grafo u v =
  case lab grafo v of
    Just c -> c
    Nothing -> 0

establecerCapacidad :: Graph -> Vertex -> Vertex -> Int -> Graph
establecerCapacidad grafo u v c =
  let grafoActualizado = delLEdge (u, v, capacidad grafo u v) grafo
  in
    insEdge (u, v, c) grafoActualizado
```

Para una implementación formal de Edmonds–Karp, véase el lector la referencia bibliográfica [4].

Nota 26. Tenga en cuenta que el algoritmo de Ford–Fulkerson tiene una complejidad de $O(|E| \cdot f_{\text{máx}})$ para una red s–t $G = (V, E)$, mientras que Edmonds–Karp tiene $O(|V| \cdot |E|^2)$, que suele ser más rápido en la mayoría de casos prácticos. El algoritmo de Dinitz, una mejora de Edmonds–Karp, se ejecuta en $O(|V|^2 \cdot |E|)$.

Redes con más de una fuente y un sumidero

Toda red semejante a una s–t se puede transformar en una red s–t equivalente. Considere la red no s–t de la Figura 7.

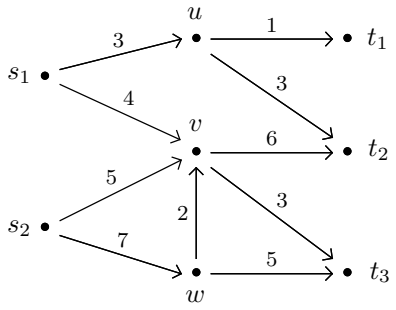


Figura 7. Red que no es s–t, pues tiene dos fuentes y tres sumideros.

El procedimiento para transformar una red no s–t en una que sí lo es es sencillo. Dada una red no s–t $G = (V, E)$, se deben crear dos nuevos vértices en V', la superfuente s_0 y el supersumidero t_0 con $s_0\,s_1, \ldots, s_0\,s_m$ y $t_1\,t_0, \ldots, t_n\,t_0$ aristas en E' tal que se cumplan las siguientes propiedades:

1. $G = (V, E)$ no es s–t, $G' = (V', E')$ sí es una red s–t.
2. $F(s_0) = \varnothing$ y $F'(t_0) = \varnothing$, como se comenta en la Nota 7.
3. $\exists s_0\,s_1, \ldots, s_0\,s_m \in E'$ y $\exists t_0\,t_1, \ldots, t_0\,t_n \in E'$.
4. Para cada $k, 1 \leqslant k \leqslant m$, $c(s_0\,s_k) = \sum_{v \in F'(s_k)} c(s_k\,v)$.
5. De igual manera, para cada $k, 1 \leqslant k \leqslant n$, $c(t_k\,t_0) = \sum_{v \in F(t_k)} c(v\,t_k)$.

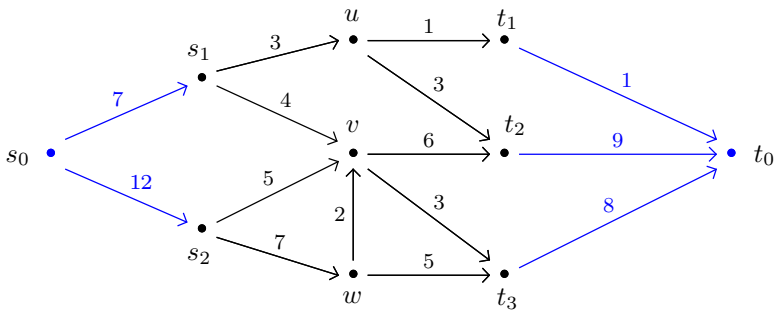


Figura 8. Red de la Figura 7 transformada en una red s–t.

En palabras, primero añadimos dos nuevos vértices en el grafo, s_0 y t_0. Si hay m fuentes y n sumideros, entonces se crearán m aristas de s_0 a cada una de las s_i y lo mismo con los sumideros. Consideremos, luego, la capacidad de cada arista nueva; esta no puede ser menor que la capacidad saliente de cada s_i pues, de lo contrario, no se trataría de la misma red s–t. Semejantemente, las capacidades entrantes de cada t_i no pueden ser menor que el flujo $c(t_i\, t_0)$.

Aplicaciones

La optimización del flujo máximo es de mucho interés para muchas empresas e instituciones, pues puede conllevar mejorar el tránsito de su producto o servicio, abaratar costes, evitar gastos innecesarios de construir infraestructura irrelevante o incluso reforzar esta donde más flujo halla.

Aerolíneas. Considerando cada aeropuerto como un vértice del grafo y cada arista como un posible viaje o escala, y teniendo en cuenta las limitaciones de plazas (y costes y tiempos de espera si se quisiera) de cada aeropuerto. En esencia, se trata de una red de flujo s–t.

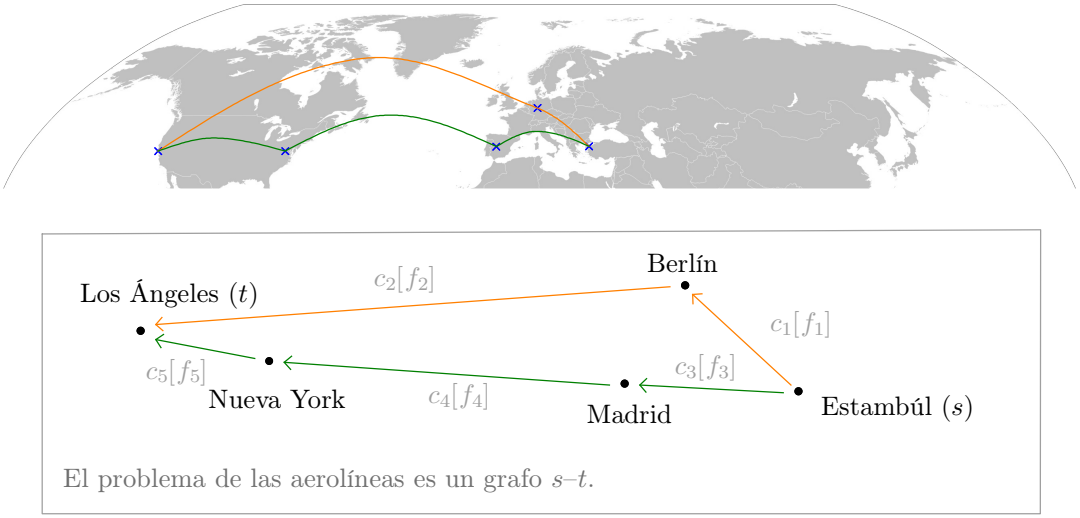


Figura 9. Viajes de avión desde Estambúl hasta Los Ángeles con posibles escalas por Berlin ó Madrid y Nueva York son un problema de redes de flujo s–t.

Ferrocarriles. En este área surgió por primera vez en 1954 el problema del flujo máximo, estudiado por el matemático americano T. E. Harris. Este extracto originalmente en inglés de la referencia bibliográfica [6] es el problema que Harris pronunció:

> «*Considere una red ferroviaria que conecta dos ciudades a través de una serie de ciudades intermedias, donde cada enlace de la red tiene asignado un número que representa su capacidad. Suponiendo estado estacionario, encuentre el flujo máximo de una ciudad a la otra.*»

Observe cómo este enunciado es, en esencia, el problema del flujo máximo en una red s–t. El artículo de 1955 de Harris sirvió de inspiración a Ford y Fulkerson para describir su algoritmo. Específicamente, Harris estudio las redes ferroviarias soviéticas en su obra *Fundamentals of a Method for Evaluating Rail Net Capacities*, que estaba escrita para US Air Force y, por tanto, era confidencial pero Ford y Fulkerson tuvieron acceso prematuro a ella antes de su desclasificación en 1999. La Figura 10 fue realizada por Harris.

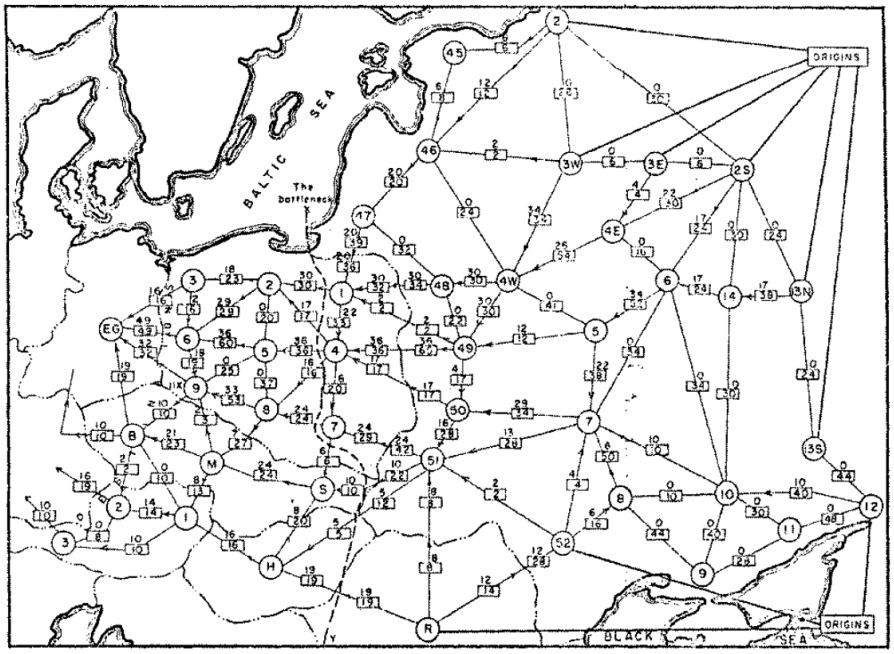


Figura 10. Problema del flujo máximo en la red de ferrocarriles soviética en 1955, estudiada por Ted Harris. Fuente bibliográfica [6].

Telecomunicaciones e Internet. Los *datacenters* (vértices) conectan las distintas regiones geográficas del planeta mediante cables (aristas). Los paquetes IP, *Internet Protocol*, se pueden considerar como el flujo. Al querer estudiar el flujo máximo entre dos datacenters, se puede hallar mediante los métodos descritos en este texto.

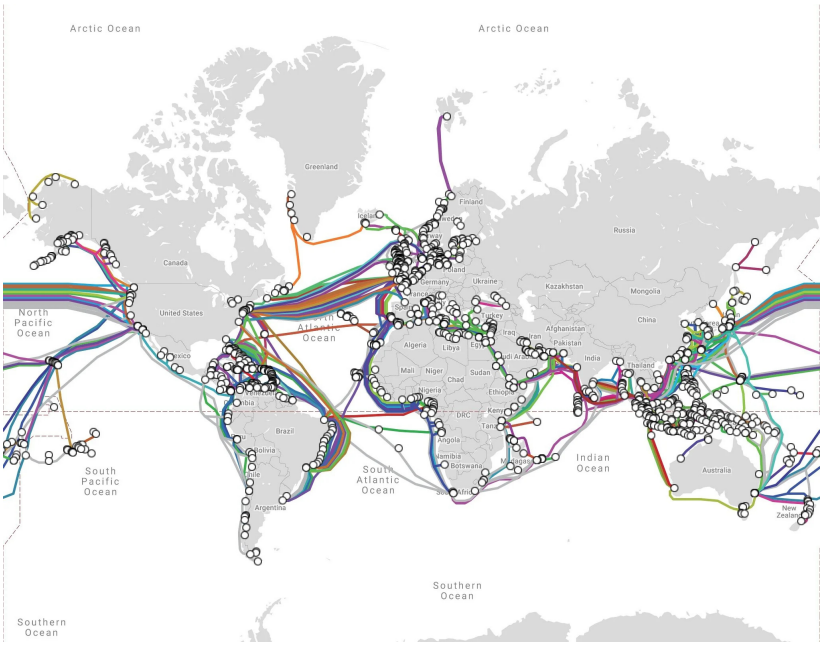


Figura 11. El tráfico de paquetes IP mediante los cables submarinos entre datacenters representa una red s–t.

Los grafos desempeñan un papel fundamental en la comprensión y optimización de sistemas complejos. Su presencia omnipresente en diversos contextos resalta la necesidad de familiarizarse con conceptos como redes s-t, flujo máximo y el algoritmo de Ford-Fulkerson, junto con sus variantes. Estos conocimientos resultan de vital importancia para abordar eficazmente los desafíos de optimización en tales sistemas y aprovechar al máximo su potencial. Además, explorar otras técnicas y enfoques relacionados con los grafos puede ampliar aún más nuestras capacidades de análisis y resolución de problemas en entornos complejos.

Comentario. No existe un consenso en la notación usada respecto a las redes de flujo s–t. Por ejemplo, en la referencia bibliográfica [2] se usa una notación textual en inglés.

Comentario. Si el lector está interesado en implementar un algoritmo de flujo máximo, debería leer *Dinitz's Algorithm: The Original Version and Even's Version* por Yefim Dinitz, el mismo autor.

Bibliografía

[1] Srinivas Devadas. Incremental improvement: max flow, min cut. MIT OpenCourseWare (YouTube), 2015.
[2] Mark Anderson Jonathan L. Gross, Jay Yellen. *Graph Theory and Its Applications*. CRC Press, third edition.
[3] Lalla Mouatadid. Network flows: the max flow/min cut theorem. Universidad de Toronto, 2016. Diseño, análisis y complejidad de algoritmos (CSC 373).
[4] S. Reza Sefidgar Peter Lammich. Formalizing the edmonds-karp algorithm. *Technische Universität München*.
[5] Silvia Flores Ramos. Problemas de redes y flujos. 2021.
[6] Alexander Schrijver. On the history of the transportation and maximum flow problems.