QÜESTIIÓ, vol. 26, 3, p. 517-536, 2002

UN ALGORITMO PARA EL PROBLEMA DE BIFLUJO MÁXIMO SIMÉTRICO NO DIRIGIDO

A. SEDEÑO-NODA
C. GONZÁLEZ-MARTÍN
Universidad de La Laguna*

En este trabajo proponemos un algoritmo de $O(nm\log U)$ para resolver el problema de biflujo máximo simétrico en una red no dirigida. Para resolver este problema se introduce un cambio de variable que permite dividir el problema original en dos problemas de flujo máximo. De esta manera se obtiene un algoritmo sencillo y eficiente donde se utilizan las herramientas computacionales propias de la resolución del clásico problema de maximizar un único flujo.

An algorithm for the undirected symmetric maximum biflow problem

Palabras clave: Problema de biflujo máximo simétrico, algoritmo de escalado en las capacidades, redes no dirigidas

Clasificación AMS (MSC 2000): 90C27

* Departamento de Estadística, Investigación Operativa y Computación (DEIOC). Universidad de La Laguna. 38271 La Laguna, Tenerife (España).

– Recibido en agosto de 2001.
– Aceptado en mayo de 2002.

1. INTRODUCCIÓN

Son numerosos los casos reales en los que aparecen problemas de flujos en redes (ver, por ejemplo, Ahuja *et al.* (1993)). Entre estos problemas, unos de los más estudiados son los *problemas de flujo máximo*, cuya versión clásica considera un único flujo sobre una red conexa dirigida con un único nodo fuente y un único nodo sumidero. Ford y Fulkerson (1956) estudian y resuelven este problema proponiendo el conocido Método de los Caminos Incrementales, que toma como base el denominado Teorema de Flujo-Máximo Corte-Mínimo.

Cuando se consideran, simultáneamente, varios tipos de flujos aparecen los correspondientes problemas de flujos múltiples. Ejemplos de estos casos aparecen en problemas de tráfico urbano, problemas de transporte ferroviario, problemas de sistemas de comunicaciones de diverso tipo, etc. (ver, por ejemplo, Ahuja *et al.* (1993)).

Hemos de señalar que, en general, para los problemas en los que se maximizan tres o más flujos no se verifica el equivalente del Teorema de Flujo-Máximo Corte-Mínimo. Sin embargo, para el caso de dos flujos (*problema de biflujo máximo*), Hu (1963) demuestra el Teorema de Biflujo-Máximo Corte-Mínimo. Una demostración del dual del teorema de Hu, es dada por Seymour (1978).

En este trabajo consideramos el *problema de biflujo máximo simétrico no dirigido* (BFMS). Para ilustrar el caso en estudio, consideremos el siguiente ejemplo. Dos ciudades, de características similares, deben estar interconectadas por una red de tráfico terrestre. Señalemos cuatro vértices de la red, distinguiendo los dos vértices por los que se accede a la red de tráfico desde cada una de las dos ciudades y los otros dos vértices por los que se sale de la citada red hacia la ciudad contraria a la de partida. Si, como es lógico, diferenciamos el tráfico desde la primera ciudad hacia la segunda y desde ésta a la primera, tendremos un problema de dos flujos cuya suma, para hacer eficiente la gestión de la red, habrá que maximizar. Teniendo presente, además, que los desplazamientos desde una ciudad a otra deben producirse (dentro de una determinada unidad de tiempo) en sentido contrario, la similitud de características de las dos ciudades impondrá la igualdad de la magnitud de los dos flujos considerados.

Este problema es un caso particular del problema de biflujo máximo no dirigido (BFM). Sin embargo, la resolución de este problema requiere un tratamiento específico ya que, al contrario de lo que ocurre para el problema BFM, su solución no queda caracterizada por el teorema de Hu. Hemos de señalar que, en la literatura que hemos consultado, no existe ningún tipo de estudio referido al problema BFMS.

Recordemos que la formulación del problema BFM se realiza habitualmente sobre una red dirigida donde los flujos pueden tomar valores negativos. Esta formulación aparece en los trabajos de Hu (1963), Rothschild y Whinston (1966), Sakarovitch (1973), Itai (1978), Seymour (1978) y Sedeño-Noda *et al.* (2001). En el presente trabajo, partimos

de una formulación similar para el problema BFMS, realizando un cambio de variable introducido en Sedeño-Noda *et al.* (2001) que posibilita separar el problema original en dos problemas de flujo máximo. Esta forma de actuar permite desarrollar un algoritmo que resuelve el problema estudiado de forma eficiente.

Después de esta introducción, en la segunda sección establecemos la formulación del problema de biflujo máximo simétrico. En la sección tercera, recordamos los conceptos y definiciones asociados al problema de flujo máximo necesarios para el entendimiento del algoritmo propuesto posteriormente. En la cuarta sección, introducimos la formulación equivalente del problema. En la quinta sección presentamos el algoritmo que resuelve el problema estudiado, incluyendo su ejecución sobre un ejemplo, y su complejidad teórica. En la sexta sección, el trabajo finaliza con una serie de conclusiones.

2. NOTACIÓN Y FORMALIZACIÓN DEL PROBLEMA

Sea $G = (V,A)$ una red dirigida, donde V es el conjunto de n nodos y A es el conjunto de m arcos. Distinguimos cuatro nodos especiales en G: los nodos fuentes s^1, s^2y los nodos sumideros t^1, t^2. Al igual que en Sakarovitch (1973), consideraremos que la red es antisimétrica, es decir, si $(i,j) \in A \Rightarrow (j,i) \notin A$. Dado cualquier nodo i, definimos los conjuntos $\text{Pred}(i) = \{j \in V/(j,i) \in A\}$ y $\text{Suc}(i) = \{j \in V/(i,j) \in A\}$ (respectivamente, los vértices predecesores y sucesores del vértice dado). $\forall (i,j) \in A$, $u_{ij} \geq 0$ denota la mayor cantidad de flujo (capacidad) que puede soportar (i,j). Denominaremos U al $\max\{u_{ij}/(i,j) \in A\}$ y $\text{A}(i) = \{(i,j) \in A : j \in \text{Suc}(i)\}$ al conjunto de arcos que salen del nodo i.

Un *biflujo* es un par $x = (x^1, x^2)$ donde $x^k : A \longrightarrow R$ con $k = 1,2$, que satisface:

$$\sum_{j \in \text{Suc}(i)} x_{ij}^k - \sum_{j \in \text{Pred}(i)} x_{ji}^k = \begin{cases} f^k, & \text{si} \quad i = s^k \\ 0, & \text{si} \quad i \in V - \{s^k, t^k\} \\ -f^k, & \text{si} \quad i = t^k \end{cases} \tag{1.1}$$

$$\left|x_{ij}^1\right| + \left|x_{ij}^2\right| \leq u_{ij}, \qquad \forall (i,j) \in A \tag{1.2}$$

para algún $f = \left(f^1, f^2\right)$, con $f^k \geq 0$ para $k = 1,2$. El problema de biflujo máximo (BFM) consiste en encontrar un biflujo x tal que la suma $f^1 + f^2$ sea máxima. En otras palabras, el problema BFM consiste en enviar simultáneamente la mayor cantidad posible de flujo de la fuente s^1 al sumidero t^1 y de la fuente s^2 al sumidero t^2, satisfaciendo las restricciones de capacidades (1.2) y las de conservación del flujo (1.1). Notar que las restricciones (1.2) implican que no se satisfaga la propiedad de unimodularidad exhibida en los problemas de redes normalizadas en las que circula un único flujo. Por lo tanto,

el óptimo del *patrón de biflujo* puede ser un vector no entero. Evidentemente, $s^1 \neq t^1$ y $s^2 \neq t^2$. Asumiremos, sin pérdida de generalidad, que $s^1 \notin \{s^2, t^2\}$ y $t^1 \notin \{s^2, t^2\}$.

El problema de biflujo máximo simétrico (BFMS) se plantea en los mismos términos que el problema BFM, pero con la restricción añadida de que la cantidad enviada entre los nodos s^1 y t^1 debe coincidir con la que se envíe entre los nodos s^2 y t^2, es decir, se ha de cumplir que $f^1 = f^2$.

Debemos notar que las restricciones (1.2) permiten la posibilidad de valores negativos en el biflujo. Esto significa que si el valor del *k*-ésimo flujo x_{ij}^k es negativo para el arco $(i, j) \in A$, entonces el flujo atraviesa el arco en sentido opuesto.

3. CONCEPTOS Y PROPIEDADES BÁSICOS

Los siguientes conceptos y propiedades son conocidos para los problemas en los que se maximiza un único flujo entre la fuente s y el sumidero t de una red dirigida G. Un *corte* es una partición del conjunto de nodos V en dos subconjuntos Sy $\bar{S} = V - S$. Un corte define un subconjunto en A denotado por $\left[S, \bar{S}\right]$: el conjunto de arcos con un extremo en S y el otro extremo en $\bar{S}$. La *capacidad de un corte* viene dada por la suma de las capacidades de los arcos que salen del conjunto S y llegan a $\bar{S}$, es decir, los arcos del conjunto que denotaremos por $(S, \bar{S})$. Esta capacidad es igual a $u\left[S, \bar{S}\right] = \sum_{(i,j)\in(S,\bar{S})} u_{ij}$. Debemos notar que $\left[S, \bar{S}\right] = (S, \bar{S}) \cup (\bar{S}, S)$.

Dado un flujo x, la *capacidad residual* de cualquier arco $(i, j) \in A$ es la máxima cantidad adicional de flujo que se puede enviar desde el nodo i al nodo j mediante los arcos (i, j) y (j, i), cuyo valor es $r_{ij} = u_{ij} - x_{ij} + x_{ji}$. La red $R(x)$ que contiene los arcos con capacidades residuales positivas se denomina *red residual*. Por tanto, la red residual $R(x)$ tiene el mismo conjunto de nodos V y, por cada arco $(i, j) \in A$ se introducen los arcos (i, j) y (j, i) con capacidades residuales r_{ij} y r_{ji} respectivamente.

Un *camino incremental* en $R(x)$ es un camino dirigido de s a t en $R(x)$. Definimos la *capacidad residual* de un camino como la menor de las capacidades residuales de los arcos en el camino. El resultado que indica que *un flujo x es máximo si la red residual $R(x)$ no contiene caminos incrementales* es ampliamente conocido y constituye la regla de parada del ya clásico *Algoritmo de Caminos Incrementales* de Ford y Fulkerson (1956). Estos autores demuestran que *la mayor cantidad de flujo que se puede enviar desde el nodo fuente s al nodo sumidero t coincide con la menor de las capacidades de todos los $s-t$ cortes de la red* (Teorema del *Flujo-Máximo Corte-Mínimo*).

Entenderemos que $g = (V, E)$ es la red no dirigida obtenida cuando no se consideran direcciones en los arcos de G y que $G' = (V, A')$ es la versión dirigida de g, donde

$A' = \{(i,j),(j,i)/\{i,j\} \in E\}$. Necesitaremos particularizar algunos de los conceptos anteriores para la red no dirigida g. Así, debemos significar que la capacidad de un corte en g, $u^*\left[S,\bar{S}\right]$[1], coincide con la suma de las capacidades de los arcos $(S,\bar{S})$ y $(\bar{S},S)$ en la red dirigida antisimétrica G, es decir, $u^*\left[S,\bar{S}\right] = \sum_{(i,j)\in(S,\bar{S})} u_{ij} + \sum_{(i,j)\in(\bar{S},S)} u_{ij}$.

Para el problema de biflujo máximo, Hu (1963) da el siguiente resultado para el caso de un grafo no dirigido:

Teorema 1. *(Teorema del Biflujo-Máximo Corte-Mínimo). El valor Máximo de* $f^1 + f^2 = \min\left(u\left[S^1,\bar{S^1}\right], u\left[S^2,\bar{S^2}\right]\right)$, *donde* $\left[S^1,\bar{S^1}\right]$ *recorre los posibles cortes tales que* $s^1,s^2 \in S^1$ *y* $t^1,t^2 \in \bar{S^1}$ *y* $\left[S^2,\bar{S^2}\right]$ *recorre los posibles cortes tales que* $s^1,t^2 \in S^2 y t^1,s^2 \in \bar{S^2}$ *en la red no dirigida g.*

4. FORMULACIÓN EQUIVALENTE DEL PROBLEMA DE BIFLUJO MÁXIMO SIMÉTRICO

La resolución del problema BFMS parte de una formulación equivalente del problema 1, realizando un cambio de variables que permite separarlo en dos problemas de flujo máximo con un único bien. Este cambio de variables, introducido por Sedeño-Noda *et al.* (2001), es el siguiente:

$$\begin{aligned} x^1_{ij} + x^2_{ij} &= z^1_{ij} - z^1_{ji} \\ x^1_{ij} - x^2_{ij} &= z^2_{ij} - z^2_{ji} \end{aligned}$$

Por tanto:

$$x^1_{ij} = \frac{z^1_{ij} - z^1_{ji} + z^2_{ij} - z^2_{ji}}{2}; \quad x^2_{ij} = \frac{z^1_{ij} - z^1_{ji} - z^2_{ij} + z^2_{ji}}{2}$$

Realizando los cambios de variables, y sumando y restando las restricciones de conservación de flujo (1.1) se obtiene para $k = 1,2$:

$$\sum_{j\in \mathrm{Suc}(i)} \left(z^k_{ij} - z^k_{ji}\right) - \sum_{j\in \mathrm{Pred}(i)} \left(z^k_{ji} - z^k_{ij}\right) = \begin{cases} f^1, & \text{si} \quad i = s^1 \\ (-1)^{k-1} f^2, & \text{si} \quad i = s^2 \\ 0, & \text{si} \quad i \in V - \{s^1,s^2,t^1,t^2\} \\ -f^1, & \text{si} \quad i = t^1 \\ (-1)^k f^2, & \text{si} \quad i = t^2 \end{cases}$$

[1]Denotaremos u^* a la capacidad del corte en la red no dirigida g

Recordemos que, por ser G una red antisimétrica, si $j \in \mathrm{Suc}(i)$ entonces $j \notin \mathrm{Pred}(i)$ y, si $j \in \mathrm{Pred}(i)$, entonces $j \notin \mathrm{Suc}(i)$. Por tanto, si queremos determinar los vértices predecesores y sucesores de uno dado sobre la red $G' = (V, A')$, encontramos que $\mathrm{Pred}'(i) = \mathrm{Suc}'(i) = \mathrm{Pred}(i) \cup \mathrm{Suc}(i)$. Por tanto, para esta última red podemos rescribir para $k = 1, 2$, las restricciones de conservación de flujo de la manera siguiente:

$$\sum_{j \in \mathrm{Suc}'(i)} z_{ij}^k - \sum_{j \in \mathrm{Pred}'(i)} z_{ji}^k = \begin{cases} f^1, & \text{si} \quad i = s^1 \\ (-1)^{k-1} f^2, & \text{si} \quad i = s^2 \\ 0, & \text{si} \quad i \in V - \{s^1, s^2, t^1, t^2\} \\ -f^1, & \text{si} \quad i = t^1 \\ (-1)^k f^2, & \text{si} \quad i = t^2 \end{cases}$$

Por otro lado, veamos cómo se modifican las restricciones (1.2) cuando se introducen los cambios de variables anteriores. Para ello, podemos escribir las restricciones (1.2) de la manera siguiente:

$$(1.2') \qquad \left|x_{ij}^1\right| + \left|x_{ij}^2\right| \leq u_{ij} \quad \Leftrightarrow \quad \begin{cases} -u_{ij} \leq x_{ij}^1 + x_{ij}^2 \leq u_{ij} \\ -u_{ij} \leq x_{ij}^1 - x_{ij}^2 \leq u_{ij} \end{cases}$$

Debemos notar que $-u_{ij} \leq z_{ij}^k - z_{ji}^k \leq u_{ij}$ para $k = 1, 2$ y que esas restricciones se pueden escribir como $0 \leq z_{ij}^k \leq u_{ij}$ y $0 \leq z_{ji}^k \leq u_{ij}$ (solución no solapada). Es decir, si $z_{ij}^k - z_{ji}^k = \alpha \geq 0$, hacemos $z_{ij}^k = \alpha$ y $z_{ji}^k = 0$; y, si $z_{ij}^k - z_{ji}^k = \alpha < 0$, hacemos $z_{ij}^k = 0$ y $z_{ji}^k = -\alpha$. Las variables z_{ij}^k se identifican usando las capacidades residuales r_{ij}^k tal que $z_{ij}^k = \text{máx}\left(0, u_{ij} - r_{ij}^k\right)$.

Por conveniencia en la notación, sea u'_{ij} la capacidad del arco $(i, j) \in A'$ definida por:

$$u'_{ij} = \begin{cases} u_{ij}, & \text{si} \quad (i, j) \in A \\ u_{ji}, & \text{si} \quad (j, i) \in A \end{cases}$$

En definitiva, para $k = 1, 2$, tenemos las cotas $0 \leq z_{ij}^k \leq u'_{ij}$ para todo arco $(i, j) \in A'$.

Obtenemos, por tanto, la siguiente formulación alternativa de (1):

$$\begin{aligned} &\text{maximizar} \quad f^1 + f^2 \\ &\text{sujeto a:} \\ (2.1a) \qquad &\sum_{j \in \mathrm{Suc}'(i)} z_{ij}^1 - \sum_{j \in \mathrm{Pred}'(i)} z_{ji}^1 = \begin{cases} f^1, & \text{si} \quad i = s^1 \\ f^2, & \text{si} \quad i = s^2 \\ 0, & \text{si} \quad i \in V - \{s^1, s^2, t^1, t^2\} \\ -f^1, & \text{si} \quad i = t^1 \\ -f^2, & \text{si} \quad i = t^2 \end{cases} \end{aligned}$$

$$\sum_{j\in \mathrm{Suc}'(i)} z^2_{ij} - \sum_{j\in \mathrm{Pred}'(i)} z^2_{ji} = \begin{cases} f^1, & \text{si} \quad i = s^1 \\ -f^2, & \text{si} \quad i = s^2 \\ 0, & \text{si} \quad i \in V - \{s^1, s^2, t^1, t^2\} \\ -f^1, & \text{si} \quad i = t^1 \\ f^2, & \text{si} \quad i = t^2 \end{cases} \qquad \textbf{\textit{P2}} \tag{2.1b}$$

$$0 \le z^k_{ij} \le u'_{ij}, k = 1,2 \quad \forall (i,j) \in A' \tag{2.2}$$

$$f^1 = f^2 \tag{2.3}$$

En el problema *P2* se puede observar que las restricciones pueden ser separadas en aquellas que únicamente afectan a cada una de las variables z^k con $k = 1,2$. Sin embargo, el valor de la función objetivo no es separable. Separaremos estos dos problemas, llamando *P2a* al problema con las restricciones que únicamente contiene las variables z^1 y denominando *P2b* al relacionado con las variables z^2.

maximizar $f^1 + f^2$

sujeto a:

$$\sum_{j\in \mathrm{Suc}'(i)} z^1_{ij} - \sum_{j\in \mathrm{Pred}'(i)} z^1_{ji} = \begin{cases} f^1, & \text{si} \quad i = s^1 \\ f^2, & \text{si} \quad i = s^2 \\ 0, & \text{si} \quad i \in V - \{s^1, s^2, t^1, t^2\} \\ -f^1, & \text{si} \quad i = t^1 \\ -f^2, & \text{si} \quad i = t^2 \end{cases}$$

$$0 \le z^1_{ij} \le u'_{ij}, \forall (i,j) \in A'$$

$$f^1 = f^2$$

P2a

maximizar $f^1 + f^2$

sujeto a:

$$\sum_{j\in \mathrm{Suc}'(i)} z^2_{ij} - \sum_{j\in \mathrm{Pred}'(i)} z^2_{ji} = \begin{cases} f^1, & \text{si} \quad i = s^1 \\ -f^2, & \text{si} \quad i = s^2 \\ 0, & \text{si} \quad i \in V - \{s^1, s^2, t^1, t^2\} \\ -f^1, & \text{si} \quad i = t^1 \\ f^2, & \text{si} \quad i = t^2 \end{cases}$$

$$0 \le z^2_{ij} \le u'_{ij}, \forall (i,j) \in A'$$

$$f^1 = f^2$$

P2b

En el problema *P*2*a* se maximiza un único flujo entre las fuentes s^1, s^2 y los sumideros t^1, t^2 con la restricción adicional de que $f^1 = f^2$. De manera similar, en el problema *P*2*b* se maximiza un único flujo desde las fuentes s^1, t^2 a los sumideros t^1, s^2 verificando que $f^1 = f^2$. Evidentemente, si la solución óptima de estos dos problemas es la misma, hemos solucionado el problema *P*2.

Los siguientes resultados, y sus correspondientes demostraciones, aparecen en Sedeño-Noda *et al.* (2001). Aunque en dicha referencia están planteados para el problema de biflujo máximo sin considerar la restricción de simetría $f^1 = f^2$, son también válidos para el caso simétrico en la forma en que se exponen a continuación:

Lema 1. *El valor óptimo de P2 coincide con el mínimo de los valores óptimos de P2a y P2b.*

Teorema 1. *El valor óptimo de P*2*a*(*P*2*b*) *coincide con la menor de las capacidades de cualquier corte* $\left[S^1, \overline{S^1}\right]$ $\left(\left[S^2, \overline{S^2}\right]\right)$ *en la red no dirigida g tal que* $s^1, s^2 \in S^1$ $\left(s^1, t^2 \in S^2\right)$ *y* $t^1, t^2 \in \overline{S^1}$ $\left(t^1, s^2 \in \overline{S^2}\right)$.

Teorema 2. *(Biflujo-Máximo Corte-Mínimo). El máximo biflujo en una red G coincide con la menor de las capacidades de los cortes en la red no dirigida g de la forma* $\left[S^1, \overline{S^1}\right]$ *y* $\left[S^2, \overline{S^2}\right]$, *donde* $s^1, s^2 \in S^1$ *y* $s^1, t^2 \in S^2$. *Es decir,* $f^1 + f^2 = \text{mín}\left(u\left[S^1, \overline{S^1}\right],\right.$ $\left. u\left[S^2, \overline{S^2}\right]\right)$ *en g.*

Supongamos que un algoritmo envía la mayor cantidad de flujo posible, que denotamos por f^{1*}, entre los nodos s^1 y t^1. Sea entonces $f^{2'}$ la mayor cantidad de flujo que se puede enviar de s^2 a t^2 cuando se envían f^{1*} unidades de flujo entre los nodos s^1 y t^1. Evidentemente, $\left(f^{1*}, f^{2'}\right)$ se corresponde con una solución óptima del problema de biflujo máximo.

Observemos que, al ser un caso particular, podemos establecer la siguiente relación entre las soluciones óptimas del problema de biflujo máximo simétrico y del problema de biflujo máximo:

Lema 2. *Sea* $\left(f^1, f^2\right)$ *el biflujo máximo simétrico, entonces se tiene que* $f^1 + f^2 \leq f^{1*} + f^{2'}$.

Demostración

Es evidente por el teorema de Biflujo-Máximo Corte-Mínimo, que cualquier biflujo $\left(f^1, f^2\right)$ tal que $f^1 = f^2$ debe satisfacer $f^1 + f^2 \leq f^{1*} + f^{2'}$.

□

Entonces, cabe plantear si a partir del biflujo máximo $\left(f^{1*}, f^{2'}\right)$ es posible obtener el biflujo máximo simétrico.

Teorema 3. *Dado el biflujo máximo* $\left(f^{1*}, f^{2'}\right)$ *se puede obtener un biflujo máximo simétrico* $\left(f^1, f^2\right)$.

Demostración

Los posibles casos que se pueden dar son:

a) Si $f^{1*} = f^{2'}$, hemos encontrado trivialmente el biflujo máximo simétrico con $f^k = f^{1*}$ con $k = 1,2$. En este caso, se tiene que $f^1 + f^2 = f^{1*} + f^{2'}$. $\left(f^1, f^2\right)$ es máximo por el teorema 2.

b) Si $f^{1*} < f^{2'}$, tenemos que enviar $-\left(f^{2'} - f^{1*}\right)$ unidades de flujo s^2 a t^2. De esta manera $f^k = f^{1*}$ con $k = 1,2$. Por tanto, $f^1 + f^2 = 2f^{1*} < f^{1*} + f^{2'}$. $\left(f^1, f^2\right)$ es máximo, pues $f^1 = f^{1*}$ es el máximo valor de flujo que se puede enviar de s^1 a t^1.

c) Si $f^{1*} > f^{2'}$, tenemos que enviar $-\left(f^{1*} - f^{2'}\right)/2$ unidades de flujo s^1 a t^1. Por el teorema de Biflujo-Máximo Corte-Mínimo, a lo sumo se pueden enviar $\left(f^{1*} - f^{2'}\right)/2$ unidades de flujo s^2 a t^2. Existen dos posibles subcasos:

*c*1) Se pueden enviar las $\left(f^{1*} - f^{2'}\right)/2$ de s^2 a t^2. Con lo que se obtiene un biflujo máximo simétrico tal que $f^k = \left(f^{1*} - f^{2'}\right)/2$ con $k = 1,2$. En este caso se tiene que $f^1 + f^2 = f^{1*} + f^{2'}$. $\left(f^1, f^2\right)$ es máximo por el teorema 2.

*c*2) Se pueden enviar ϕ unidades de s^2 a t^2 con $\phi < \left(f^{1*} - f^{2'}\right)/2$. En este caso, con el fin de obtener un biflujo máximo simétrico se han de enviar $-\left(\left(f^{1*} - f^{2'}\right)/2 - \phi\right)$ de s^1 a t^1. De esta forma se obtiene un biflujo máximo simétrico tal que $f^k = f^{2'} + \phi$ con $k = 1,2$. Además, se tiene que $f^1 + f^2 = 2f^{2'} + 2\phi < f^{1*} + f^{2'}$. Entonces, $\left(f^1, f^2\right)$ es máximo, pues $f^2 = f^{2'} + \phi$ es la mayor cantidad de flujo que se puede enviar de s^2 a t^2 cuando $f^1 = f^{2'} + \phi$.

□

5. ALGORITMO PARA OBTENER UN BIFLUJO MÁXIMO SIMÉTRICO

En los problemas *P*2*a* y *P*2*b* se impone el envío de la mayor cantidad de flujo posible desde s^1 a t^1. Parece lógico que, en una primera instancia, se trate de obtener el flujo

máximo entre estos nodos utilizando el adecuado algoritmo de flujo máximo. Una vez resuelto este primer paso, consideraremos dos redes residuales cada una de ellas asociadas a las variables z^k con $k = 1,2$. Sean $R^1(z^1)$ y $R^2(z^2)$ las correspondientes redes residuales una vez se han enviado las f^1 unidades de flujo. Para obtener la solución del problema de biflujo, necesitamos enviar la misma cantidad de flujo f^2 desde s^2 a t^2 en $R^1(z^1)$ y desde t^2 a s^2 en $R^2(z^2)$, teniendo en cuenta los posibles casos señalados en el teorema 3. Por tanto, debemos identificar caminos incrementales en ambas redes, determinar sus correspondientes capacidades y enviar el mínimo de ambas a través de los correspondientes caminos. Esto último conlleva una actualización de las capacidades residuales de los arcos de ambos caminos.

Para identificar los respectivos caminos, proponemos el uso de dos etiquetas distancias (Goldberg (1985)). Sea una *función distancia* $d : V \longrightarrow Z^+ \cup \{0\}$ con respecto a las capacidades residuales r_{ij} de una red residual $R(x)$. Se dice que es *válida* si satisface las siguientes dos condiciones:

$$a)\ d(t) = 0$$

$$b)\ d(i) \leq d(j) + 1\ ,\ \forall (i,j) \in A,\ r_{ij} > 0$$

donde t es el nodo sumidero (s es el nodo fuente) del correspondiente problema de flujo máximo. Llamaremos *etiqueta distancia* de un nodo i a $d(i)$. Mediante inducción, se puede demostrar que $d(i)$ es una cota inferior de la longitud del camino mínimo desde el nodo i al nodo t en $R(x)$, donde la longitud del camino viene dada por el número de arcos que este utiliza. Si para cada nodo i, $d(i)$ coincide con la longitud del camino mínimo desde el nodo i al nodo t en $R(x)$, entonces diremos que las etiquetas distancias son *exactas*. Si $d(s) \geq n$, entonces no hay camino incremental en $R(x)$.

Un arco (i,j) en la red residual se denomina *admisible* si satisface $d(i) = d(j) + 1$. Un camino incremental en $R(x)$ que consiste de arcos admisibles es denominado *camino admisible*. Un camino admisible es un camino incremental mínimo de s a t.

Denotaremos por $d^1(i)$ a la etiqueta distancia del nodo i en $R^1(z^1)$ y por $d^2(i)$ a la etiqueta distancia del mismo nodo en $R^2(z^2)$.

Con el fin de no tener que calcular las capacidades residuales de los caminos incrementales y, posteriormente, actualizar las capacidades residuales de sus arcos, proponemos escalar las capacidades.

Para definir el escalado en las capacidades introducimos un parámetro Δ y nos referimos a la red Δ-*residual*, $R(\Delta)$, como a la red que únicamente contiene arcos cuya capacidad residual es al menos Δ. De esta manera, todos los caminos incrementales en $R(\Delta)$ tienen una capacidad residual de al menos Δ.

Como hemos introducido una escala, diremos que un arco (i, j) es Δ-*admisible* si $d(i) = d(j) + 1$ y $r_{ij} \geq \Delta$. Por lo tanto, en una fase Δ se identificarán caminos incrementales Δ-*admisibles* en $R^1(z^1)$ y en $R^2(z^2)$ de manera simultánea.

Los procedimientos que proponemos trabajan en fases de escalado. Cada fase tiene un Δ fijo y, cuando finaliza una fase, comienza la siguiente con $\Delta = \Delta/2$. Al principio $\Delta = 2^{\lceil \log 2U \rceil}$, es decir, es igual al menor entero potencia de dos que es mayor o igual que $2U$. La última fase ocurre cuando $\Delta = 1$. En esta última fase $R^k(\Delta) = R^k(z^k)$ con $k = 1, 2$. De esta manera, cada procedimiento realiza $\lceil \log 2U | \rceil$ fases de escalado.

Debemos tener en cuenta que, cuando tengamos que devolver flujo, nos plantearemos identificar caminos de t^1 a s^1. Las anteriores ideas permiten desarrollar el siguiente algoritmo:

```
Algoritmo Biflujo_Máximo_Simétrico;
  begin
  Sea Z¹ = Z² :=0; ∀(i, j) ∈ A Sea r¹ij = r²ij := uij y r¹ji = r²ji := uij ;
  Obtener el flujo máximo f¹ entre s¹ y t¹ ;
  f² = 0; φ := f¹ ; Actualizar_flujo( s², t², t², s², R¹, R², φ );
  if φ >0 then {En cualquier otro caso tenemos el biflujo máximo simétrico}
     begin
     φ := φ /2; θ = φ ;
     Actualizar_flujo( t¹, s¹, t¹, s¹, R¹, R², θ ); f¹ := f¹ - φ ; (caso c)
     Actualizar_flujo( s², t², t², s², R¹, R², φ );
     if φ >0 then; Actualizar_flujo( t¹, s¹, t¹, s¹, R¹, R², φ ); f¹ := f¹ - φ (caso c2)
     end;
  f² := f¹
  end.
```

```
Procedure Avanzar(i,d,R,θ ,pred);
   Sea (i,j) un arco Δ -admisible;
   rij = rij − θ ; rji = rji + θ ;
   pred(j) := i; i := j
end

Procedure Retroceder(i,d,R,θ ,pred,s);
   d(i) := min{d(j) + 1 : (i, j) ∈ A'(i) y rij ≥ Δ};
   if i ≠ s then
      begin
      j := i; i := pred(i);
      rij = rij + θ ;rji = rji − θ
      end;
```

```
Procedure Restablecer(i,s,R,θ ,pred);
   while (i≠s) do
      begin
      j := i; i := pred(i);
      rij = rij + θ ;rji = rji − θ
      end;
```

```
Procedure Actualizar_flujo(p,q,v,w, R^1 , R^2 , φ );
 begin
 Δ := 2^⌈log 2U⌉ ;
 while ( Δ ≥ 1) and (φ >0) do
  begin
  d^1(q) := 0 ; Obtener d^1 mediante un recorrido en amplitud hacia atrás en R^1 a
  partir de q; Sea i^1 := p;
  d^2(w) := 0 ; Obtener d^2 mediante un recorrido en amplitud hacia atrás en R^2 a
  partir de w; Sea i^2 := v;
  while ( d^1(p) < n ) and ( d^2(v) < n ) and (φ >0) do
   begin
   θ :=min( φ , Δ );
   if ( i^1 ≠ q) then
    if ( i^1 tiene un arco Δ -admisible) then Avanzar( i^1 , d^1 , R^1 , θ , pred^1 )
    else Retroceder( i^1 , d^1 , R^1 , θ , pred^1 ,p);
   if ( i^2 ≠ w) then
    if ( i^2 tiene un arco Δ -admisible) then Avanzar( i^2 , d^2 , R^2 , θ , pred^2 )
    else Retroceder( i^2 , d^2 , R^2 , θ , pred^2 ,v);
   if ( i^1 =q) and ( i^2 =w) then i^1 := p; i^2 := v; φ :=φ - θ
   end;
  if ( i^1 ≠ p) then Restablecer( i^1 ,p, R^1 , θ , pred^1 );
  if ( i^2 ≠ v) then Restablecer ( i^2 ,v, R^2 , θ , pred^2 );
  Δ := Δ / 2
  end
 end;
```

El algoritmo para el biflujo máximo simétrico, comienza obteniendo el flujo máximo entre s^1 y t^1 enviando $f^1 = f^{1*}$. A continuación, en la primera llamada al procedimiento *Actualizar_flujo*, se intentan enviar $f^2 = f^1$ unidades de flujo de s^2 a t^2 en $R^1(z^1)$ y de t^2 a s^2 en $R^2(z^2)$. Esta forma de proceder impide que el caso $f^1 < f^2$ pueda darse. Por ello, en la aplicación del algoritmo únicamente se pueden dar los casos *a*) y *c*), nunca el *b*) del teorema 3. En consecuencia, después de la primera llamada a *Actualizar_flujo*, o bien es $f^1 = f^2$ (con lo que se ha obtenido el biflujo máximo simétrico) o bien f^1 sigue siendo mayor que f^2. En este último caso, al realizar la segunda llamada a *Actualizar_flujo*, $\phi = \left(f^1 - f^2\right)/2$ unidades de flujo son devueltas desde t^1 a s^1 tanto en $R^1(z^1)$ como en $R^2(z^2)$. Luego, cuando se llama por tercera vez al procedimiento *Actualizar_flujo*, se intentan enviar esas ϕ unidades de flujo de s^2 a t^2 en $R^1(z^1)$ y de t^2 a s^2 en $R^2(z^2)$. Si se han enviado, el algoritmo termina y $f^1 = f^2$; en otro caso, todavía se tiene que $f^1 > f^2$ y se envían $\phi = f^1 - f^2$ unidades desde t^1 a s^1, tanto en $R^1(z^1)$ como en $R^2(z^2)$. Esta operación se realiza en la cuarta llamada a *Actualizar_flujo*, obteniendo, en su caso, el biflujo máximo simétrico.

Para identificar simultáneamente un camino Δ-*admisible* en cada red residual $R^k(\Delta)$ con $k = 1,2$, se procede de la manera siguiente. En cada fase de escalado se calculan las etiquetas distancias exactas d^k con $k = 1,2$. El algoritmo realiza, iterativamente, pasos *Avanzar* o *Retroceder* sobre el último nodo de cada camino admisible parcial. Si del

nodo actual i^k sale un arco admisible (i^k, j^k), entonces realiza un paso *Avanzar* y añade este arco al camino parcial actual. Cada paso *Avanzar* actualiza las capacidades residuales del arco Δ-*admisible* (i^k, j^k) por $r_{ij} = r_{ij} - \theta$ y $r_{ji} = r_{ji} + \theta$, es decir, θ unidades de flujo son enviadas a través de dicho arco. Si no se identifica un arco Δ-*admisible*, se realiza un paso *Retroceder* que incrementa la etiqueta distancia del nodo i^k, haciendo que el arco $\left(\text{pred}^k(i^k), i^k\right)$ sea no admisible y, por lo tanto, retrocediendo un arco sobre el camino parcial $\text{pred}^k(i^k)$ almacena el nodo anterior al nodo i^k en el actual camino Δ-*admisible* de $R^k(\Delta)$). Además, se deshace el envío de θ unidades de flujo anterior. Así, si (i^k, j^k) está en un camino Δ-*admisible*, ya hemos enviado las θ unidades de flujo y, si retrocedemos sobre él, deshacemos ese envío. Si en este cometido se alcanzan los respectivos nodos sumideros, el proceso se repite comenzando en las respectivas fuentes. Por la forma en que se desarrolla el algoritmo, es claro que esto únicamente sucede si se han detectado, a la vez, un camino Δ-*admisible* en $R^1(\Delta)$ y en $R^2(\Delta)$, aumentando f^2, ó disminuyendo f^1, en θ unidades de flujo. Una fase de escalado acaba cuando la etiqueta distancia de al menos un nodo fuente sea mayor o igual que n. Si esto ocurre, al menos un i^k puede ser distinto de su nodo fuente, lo que significa que se han enviado θ unidades de flujo a lo largo del camino parcial identificado para la correspondiente red residual. El procedimiento *Restablecer* se encarga de devolver esas θ unidades de flujo. Finalmente, para obtener el patrón de biflujo máximo se deshace el cambio de variable, es decir:

$$\forall (i,j) \in A, \quad z_{ij}^k - z_{ji}^k = u_{ij} - r_{ij}^k \quad \Rightarrow \quad x_{ij}^1 = \frac{z_{ij}^1 + z_{ij}^2}{2} - u_{ij} x_{ij}^2 = \frac{z_{ij}^1 - z_{ij}^2}{2}$$

5.1. Ejemplo

A continuación aplicaremos el algoritmo al ejemplo de la figura 1. El proceso para obtener el patrón de biflujo máximo simétrico dado en la figura 2, se desarrolla en la figura 3.

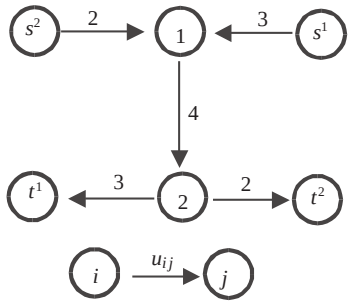

Figura 1

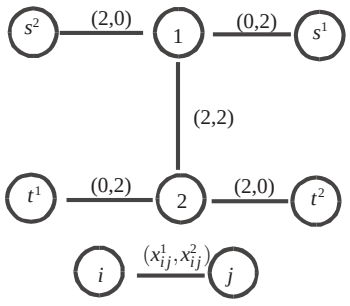

Figura 2

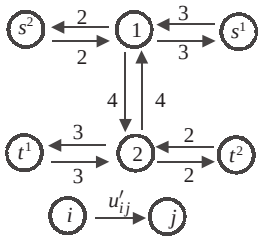

Fig. 3*a*

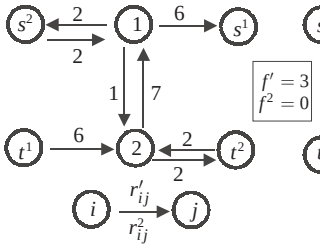

Fig. 3*b*

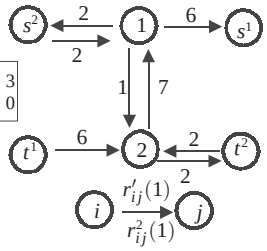

Fig. 3*c*

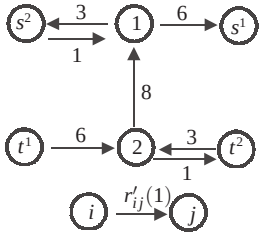

Fig. 3*d*

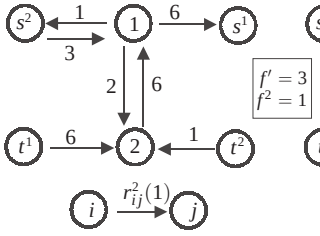

Fig. 3*e*

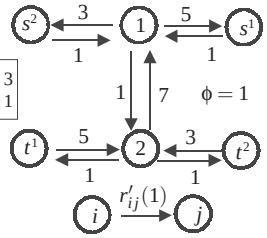

Fig. 3*f*

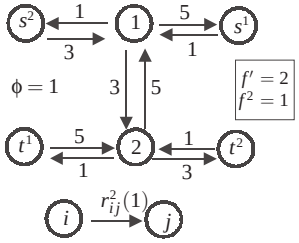

Fig. 3*g*

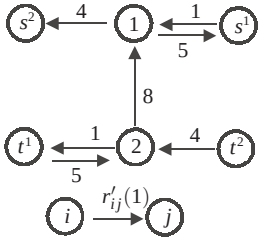

Fig. 3*h*

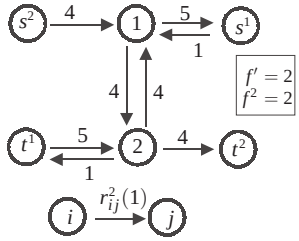

Fig. 3*i*

La figura 3*a*, representa la red G' con las capacidades u'. La figura 3*b*, muestra la red residual $R^1(z^1)$ que coincide con $R^2(z^2)$ cuando se han enviado $f^1 = 3$ unidades de flujo desde s^1 a $t^1 (f^2 = 0)$. Se puede observar en esta figura que únicamente hay un camino de s^2 a t^2 en $R^1(z^1)$ de capacidad 1 y otro de t^2 a s^2 en $R^2(z^2)$ de capacidad 2. Por ello, al llamar por primera vez a *Actualizar_flujo*, se identifican ambos caminos cuando el procedimiento se encuentre en la fase $\Delta = 1$ (figura 3*c*). Tras enviar 1 unidad de flujo a través de estos caminos en sus respectivas redes incrementales (figuras 3*d* y 3*e*), se puede observar que en $R^1(1)$ no hay camino incremental (aunque si lo hay en $R^2(1)$). En este punto tenemos que $f^1 = 3$ y $f^2 = 1$. Por lo tanto, $f^1 > f^2$, lo que implica que $\phi = (f^1 - f^2)/2 = 1$. Entonces tenemos que la segunda llamada al procedimiento

Actualizar_flujo envía 1 unidad de flujo de t^1 a s^1, tanto en $R^1(z^1)$ y en $R^2(z^2)$. El resultado de esta operación se muestra en las figuras 3*f* y 3*g* respectivamente. Ahora tenemos que $f^1 = 2$ y $f^2 = 1$, pero lo importante es que podemos enviar 1 unidad de flujo de s^2 a t^2 en $R^1(z^1)$ y de t^2 a s^2 en $R^2(z^2)$. Esta operación se realiza cuando se llama por tercera vez al procedimiento *Actualizar_flujo*. El resultado de esta operación se muestra en las figuras 3*h* y 3*i*, de donde se tiene que $f^1 = 2$ y $f^2 = 2$. En este punto no es necesario llamar de nuevo al procedimiento *Actualizar_flujo*. Finalmente, el patrón de biflujo máximo simétrico se obtiene deshaciendo los cambios de variables (Figura 2).

5.2. Complejidad del algoritmo

En esta sección demostraremos que la complejidad del algoritmo en el caso peor es $O(nm\log U)$. El algoritmo se inicia con la obtención del flujo máximo entre s^1 y t^1. Por tanto, se debe elegir un método de flujo máximo con la complejidad computacional adecuada para realizar este cometido.

Por otra parte, el algoritmo anterior ejecuta entre una y cuatro veces el procedimiento *Actualizar_flujo*. Ambos procedimientos requieren el mismo esfuerzo computacional. Cada uno de ellos realiza $\lceil \log 2U \rceil$ fases de escalado. Cada fase de escalado Δ es llevada a cabo hasta que alguna de las etiquetas distancia de las correspondientes fuentes en $R^1(\Delta)$ ó en $R^2(\Delta)$ es mayor o igual que n.

Lema 3. *Cada fase de escalado de Actualizar_flujo requiere un esfuerzo computacional de O(nm).*

Demostración

Supongamos que nos fijamos en la etiqueta distancia $d^1(p)$ en $R^1(\Delta)$ y que ésta es la que determina la condición de parada de cada fase de escalado cuando $d^1(p) \geq n$. Así, el procedimiento actualiza la etiqueta distancia $d^1(i)$ en $R^1(\Delta)$ de un nodo i a lo sumo n veces ya que, si un nodo tiene una distancia mayor o igual que n, este nodo no es alcanzable mediante un camino elemental desde la fuente. Además, en el peor de los casos, la etiqueta distancia de un nodo puede incrementarse cada vez en una unidad. Por tanto, el número de operaciones *Retroceder* está acotado por $O(n^2)$, lo que implica que el esfuerzo computacional necesario para encontrar arcos admisibles y actualizar las etiquetas de los vértices es $O(mn)$.

Veamos ahora cuál es el número de envíos de flujo. Para ello diremos que un envío es Δ-*saturante* a través de (i, j) si, después de realizarlo, la capacidad residual del arco (i, j) es estrictamente menor que Δ. Un arco puede ser Δ-*saturado* a lo sumo $n/2$ veces. Esto es así debido a que entre dos envíos Δ-*saturantes* consecutivos, a través de (i, j), las

etiquetas distancias de los nodos i y j deben haber incrementado en al menos dos unidades. Así, sumando para todos los arcos, se tiene que el número de envíos Δ-*saturantes* es $O(nm)$. Esto implica que el número de envíos de flujo es a lo sumo $O(nm)$. Como el esfuerzo computacional de cada envío de flujo es $O(1)$, el esfuerzo total vuelve a ser $O(nm)$. El número de llamadas a *Avanzar* está acotado por el número de envíos de flujo más el número de operaciones *Retroceder*, es decir, $O(nm+n^2)$. Finalmente, la operación *Restablecer* es llamada a lo sumo una vez, requiriendo un esfuerzo de $O(n)$. Por lo tanto, la complejidad de una fase, considerando que la etiqueta distancia $d^1(p)$ en $R^1(\Delta)$ determina la condición de parada, es $O(nm)$.

La misma complejidad en el peor caso se obtendría si se considerara que es la etiqueta distancia $d^2(v)$ en $R^2(\Delta)$ la que determina la condición de parada. Como, en cada fase, alguna de las dos posibilidades debe ocurrir, la complejidad de cada fase coincide con el máximo de las complejidades de ambas. Por lo tanto, la complejidad es $O(nm)$.

□

Teorema 4. *El algoritmo tiene una complejidad* $O(nm\log U)$.

Demostración

Por el lema 3, cada fase de escalado emplea un tiempo de $O(nm)$, como el número de fases de escalado es $\lceil \log 2U \rceil$, obtenemos que cada procedimiento se ejecuta en un tiempo $O(nm\log U)$. Como, a lo sumo, se ejecutan cuatro veces el procedimiento *Actualizar_flujo*, el algoritmo requiere un tiempo $O(nm\log U)$.

□

6. CONCLUSIONES

En este trabajo, estudiamos el problema de Biflujo Máximo Simétrico sobre redes no dirigidas. Este problema no ha sido considerado previamente en la literatura que hemos consultado. Para su resolución, introducimos una nueva formulación, resultante de la consideración de cambios de variables, que permiten resolverlo utilizando las herramientas clásicas desarrolladas para problemas de flujos de un solo tipo. Después de exponer los conceptos y propiedades necesarios, introducimos un algoritmo que resuelve eficientemente el problema con una complejidad computacional teórica de $O(nm\log U)$.

AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado a través del proyecto 180221024 de la Universidad de La Laguna.

7. REFERENCIAS

Ahuja, R., Magnanti, T. and Orlin, J. B. (1993). *Network Flows*. Prentice-Hall, Inc.

Ford, L. R. and Fulkerson, D. R. (1956). «Maximal Flow through a Network», *Canadian Journal of Mathematics*, 8, 399-404.

Goldberg, A. V. (1985). «A New Max-Flow Algorithm», *Technical Report MIT/LCS/TM-291*, Laboratory for Computer Science, MIT, Cambridge, MA.

Hu, T. C. (1963). «Multi-Commodity Network Flows», *Operations Research*, 11, 344-360.

Itai, A. (1978). «Two-Commodity flow», *J. Assoc. Comput. Mach.*, 25 (4), 596-611.

Rothschild, B. and Whinston, A. (1966). «On Two Commodity Network Flows», *Operations Research*, 14, 377-387.

Sakarovitch, M. (1973). «Two Commodity Network Flows and Linear Programming», *Mathematical Programming*, 4, 1-20.

Sedeño-Noda, A., González-Martín, C. and Alonso-Rodríguez, S. (2001). «The Maximum Biflow Problem: A study of the undirected case», en proceso de referee en *Naval Research Logistics*.

Seymour, P. D. (1978). «A Two Commodity cut Theorem», *Discrete Mathematics*, 23, 177-181.

ENGLISH SUMMARY

AN ALGORITHM FOR THE UNDIRECTED SYMMETRIC MAXIMUM BIFLOW PROBLEM

A. SEDEÑO-NODA
C. GONZÁLEZ-MARTÍN
Universidad de La Laguna*

In this paper, an $O(nm\log U)$ algorithm to solve the maximum undirected symmetric biflow problem is proposed. We introduce changes of variables in the initial mathematical formulation of the problem, which let us split this problem in two maximum flow problems. In this way, we obtain an easy algorithm that uses the computational tools, which are associated with the resolution of the maximum flow problem.

Keywords: Symmetric Maximum Biflow problem, Capacity Scaling Algorithm, Undirected Networks

AMS Classification (MSC 2000): 90C27

* Departamento de Estadística, Investigación Operativa y Computación (DEIOC). Universidad de La Laguna. 38271 La Laguna, Tenerife (España).

– Received August 2001.
– Accepted Mai 2002.

Many realistic problems are solved using network flows modelling (see, for example, Ahuja *et al.* (1993)). Among them, one of the network flows most intensely studied is the maximum flow problem. The original version, with a single commodity, is studied and solved in Ford and Fulkerson (1956), through the well-known *Augmenting Path* algorithm, based on the max-flow minimum-cut theorem. From this starting point, several works have improved the computational complexity given.

When different kinds of commodities over a same network are considered, multi-commodity flow problems arise. In general, for three or more kinds of flows, the analogous max-flow minimum-cut theorem is not verified. However, for the two-commodity problem in undirected case, Hu (1963) proves the max-biflow min-cut theorem. An analogous result is given by Seymour (1978).

In this work, we consider the symmetric maximum biflow (SMBF) problem on undirected network. This problem is a particular case of the maximum biflow (MBF) problem. The formulation of the MBF problem uses a directed network where the quantities of flow can be negative. This formulation is common with Hu (1963), Rothschild and Whinston (1966), Sakarovitch (1973), Itai (1978), Seymour (1978) and Sedeño-Noda *et al.* (2001).

The MBF problem is stayed in the next way: Given a directed network $G = (V,A)$, let V be the set of n nodes and A be the set of m arcs. We distinguish four special nodes in G: the source nodes s^1, s^2 and the sink nodes t^1, t^2. We assume that the network is antisymmetric, that is, if $(i,j) \in A \Rightarrow (j,i) \notin A$. Given any node i, we define the sets $\text{Pred}(i) = \{j \in V/(j,i) \in A\}$ and $\text{Suc}(i) = \{j \in V/(i,j) \in A\}$. Let $u_{ij} \geq 0$ be the capacity associated with the arc $(i,j) \in A$ and $U = \text{máx}\{u_{ij}/(i,j) \in A$. For each node i, we also define the *arc adjacency list* $\text{A}(i) = \{(i,j) \in A : j \in \text{Suc}(i)\}$.

A *biflow* in G is a pair $x = (x^1, x^2)$ that satisfies:

$$\sum_{j \in \text{Suc}(i)} x_{ij}^k - \sum_{j \in \text{Pred}(i)} x_{ji}^k = \begin{cases} f^k, & \text{if} \quad i = s^k \\ 0, & \text{if} \quad i \in V - \{s^k, t^k\} \\ -f^k, & \text{if} \quad i = t^k \end{cases} \qquad k = 1,2 \tag{1.1}$$

$$\left|x_{ij}^1\right| + \left|x_{ij}^2\right| \leq u_{ij}, \qquad \forall (i,j) \in A \tag{1.2}$$

for some $f = (f^1, f^2)$ with $f^k \geq 0$, for $k = 1,2$. The MBF problem consists in finding a biflow x such that the summation $f^1 + f^2$ is maximal. In other words, in MBF we will try to send the maximum amount of flow from the source s^1 to the sink t^1, and from the source s^2 to the sink t^2, verifying the constraints (1.1) and (1.2). Note that constraints (1.2) allow the values of the biflow to be negative. This means that if the value of the kth flow x_{ij}^k is negative for the arc $(i,j) \in A$, we will interpret that the flow travels in

the opposite direction. This corresponds with the undirected case of the MBF problem. The SMBF problem equals the MBF problem with the additional constraint $f^1 = f^2$.

In these conditions, we introduce new changes of variables that let us split the problem in two maximum flow problems with the additional constraint $f^1 = f^2$. Furthermore, this new development leads us to build an efficient algorithm to solve the problem. The changes of variables used in this paper, allow all lower bounds to be zero and the resolution of the problems using classic tools. Then, the implementation of the proposed algorithm results simple and with a complexity of $O(nm \log U)$.