AMPLIACIÓN DE PROGRAMACIÓN ENTERA

Pedro Miranda Menéndez

Resumen

En este tema estudiaremos el problema de asignación, que aunque pueden resolverse mediante el algoritmo del simplex o mediante técnicas de Programación Entera, tiene un algoritmo especial que permite resolverlo con mayor rapidez. El problema de asignación aparece con mucha frecuencia en situaciones prácticas, lo que explica el interés en obtener un algortimo específico para su resolución. Previamente veremos algunos resultados sobre matrices unimodulares, que son matrices tales que permiten asegurar que el problema de Programación Lineal Relajado asociado a un problema de Programación Entera tiene todas sus S.B.F. enteras.

1. Matrices unimodulares

Consideremos un problema de Programación Entera (PE) y su correspondiente Problema Relajado (PR). Ya sabemos que la infactibilidad de PR conlleva la infactibilidad de PE; por otra parte, si PR tiene solución no acotada, el problema PE tiene solución no acotada o es infactible.

Supongamos entonces que el problema PR tiene s.o. finita. En este caso PE es infactible o tiene solución óptima finita. El problema es que la solución de PR no será en general factible para PE. De hecho, no podemos asegurar que la solución factible para PE más cercana a la solución óptima de PR sea la solución óptima de PE. En esta sección nos centraremos en buscar condiciones para asegurar que la s.o. de PR sea factible para el problema de PE.

Como la s.o. de PR se alcanza en una S.B.F., basta encontrar condiciones en las que todas las S.B.F. de PR sean de coordenadas enteras.

Definición 1. *Sea $A \in \mathcal{M}_{m\times n}$. Diremos que A es una matriz* **totalmente unimodular** *(o* **unimodular** *para aligerar la terminología) si cualquier submatriz de A que sea cuadrada tiene determinante 0, 1 ó -1.*

En particular, los elementos de la matriz, que son las submatrices de orden 1, tienen que ser 1, -1 ó 0.

Supongamos que tenemos un problema en forma canónica. Entonces las restricciones de PR son

$$A\vec{x} \leq \vec{b}.$$

Si añadimos las variables de holgura para poner el problema en forma estándar las restricciones quedan

$$A\vec{x} + Id\vec{h} = \vec{b} \Leftrightarrow (A, Id) \begin{pmatrix} \vec{x} \\ \vec{h} \end{pmatrix} = \vec{b}.$$

Lema 1. *Sea $A \in \mathcal{M}_{m\times n}$. Entonces A es unimodular si y sólo si (A, Id) es unimodular.*

Demostración: $\Rightarrow$) Consideremos una submatriz cuadrada de (A, Id). Entonces, si desarrollamos el determinante por las columnas que provienen de Id se tiene que

$$|B| = \pm 1|B'|, 0$$

donde B' es una submatriz cuadrada de A. En consecuencia, si A es unimodular, entonces el determinante de B será 1, -1 ó 0. Es decir, que (A, Id) es una matriz unimodular.

$\Leftarrow$) Si (A, Id) es unimodular, entonces cualquier submatriz cuadrada de A es una submatriz cuadrada de (A, Id), por lo que su determinante es 1, -1 ó 0. ■

Proposicíon 1. *Sea PR y supongamos que A es unimodular y $\vec{b} \in \mathbb{Z}^m$. Entonces todas las S.B.F. son de coordenadas enteras.*

Demostración: Recordemos que una S.B. viene dada por una base B contenida en (A, Id). Por ser una base, B es una submatriz cuadrada invertible de orden $m \times m$ y, al ser (A, Id) unimodular concluimos que $|B| = \pm 1$. Por otra parte, la S.B.F. viene dada por $\vec{x}_B = B^{-1}\vec{b}$ y $\vec{x}_N = \vec{0}$. Si resolvemos el sistema

$$B\vec{x}_B = \vec{b}$$

mediante el método de Cramer se tiene que

$$x_i = \frac{|B_i|}{|B|}, \forall i = 1, ..., m,$$

donde B_i es la matriz B en la que la columna i-ésima ha sido sustituida por $\vec{b}$. Entonces, como A es unimolular y $\vec{b}$ tiene sus coordenadas enteras, B_i tiene todos sus elementos enteros y, en consecuencia, su determinante es entero. Como A es unimodular, se tiene que $|B| = \pm 1$, con lo que $x_i \in \mathbb{Z}, \forall i$. ■

Por supuesto, es posible que todas las S.B.F. de PR sean enteras sin estar en las condiciones de la proposición anterior.

Proposicíon 2. *Sea A una matriz cuyas elementos son todos 1, 0 ó -1 de forma que a lo sumo hay dos elementos no nulos en cada columna. Si existe una partición de las filas $\{F, F^c\}$ tal que*

$$\sum_{i\in F} a_{ij} - \sum_{i\in F^c} a_{ij} = 0, \forall j = 1, ..., n,$$

entonces A es una matriz unimodular.

Demostración: Sea $A' \subseteq A$ y veamos que $|A'| = 0, 1$ ó -1. Lo demostraremos por inducción en la dimensión de A'.

Si $A' \in \mathcal{M}_{1\times 1}$, entonces A' es un número de A y vale -1, 0 ó 1, con lo que el resultado es cierto.

Sea $k > 1$ y supongamos cierto el resultado hasta $k-1$. Consideremos una matriz $A' \in \mathcal{M}_{k\times k}$ contenida en A y estudiemos sus columnas. Nótese que en cada columna es tal que el número de valores distintos de cero es cero, uno o dos. Consideremos cada caso por separado:

- Si alguna columna sólo tiene ceros, entonces $det(A') = 0$ y el resultado es cierto.
- Si hay alguna fila con un solo valor distinto de cero, entonces podemos desarrollar el determinante por esa columna, obteniendo $det(A') = \pm 1 det(A_{k-1})$, donde $A_{k-1} \in \mathcal{M}_{k-1\times k-1}$ contenida en A. Aplicando inducción se tiene que $det(A_{k-1}) \in \{0, 1, -1\}$, con lo que $det(A') \in \{0, 1, -1\}$.
- Si todas las columnas tienen dos valores distintos de cero, entonces esto implica que si consideramos la suma de las filas de A' que son parte de las filas de F y la suma de las filas de A' correspondiente a las filas de F^c, estas dos sumas coinciden, con lo que $det(A') = 0$. □

2. El problema de asignación

2.1. Planteamiento

Vamos a comenzar con un ejemplo, que será el hilo conductor de esta sección.

Ejemplo 1. *Una empresa de alquiler de coches tiene que enviar cuatro coches a cuatro clientes. Los coches están situados en distintas partes de la ciudad, así como los clientes. Por ello el coste cambia. A continuación se da la tabla de precios en euros. ¿Cuál debe ser la asignación que minimiza el coste?*

Coche\Cliente	*1*	*2*	*3*	*4*
1	*1*	*4*	*6*	*3*
2	*9*	*7*	*10*	*9*
3	*4*	*5*	*11*	*7*
4	*8*	*7*	*8*	*5*

En un problema de asignación general, tenemos *m trabajadores* y *n tareas* a realizar. Cada trabajador O_i sólo puede realizar una tarea a lo sumo y cada tarea D_j necesita un trabajador para poder ser realizada. Además, se conoce el coste de asignar un trabajador O_i a la tarea D_j, que denotaremos c_{ij}. El objetivo es minimizar el coste total de realizar las tareas. El problema de asignación es un caso particular de problema de transporte, en el que los orígenes ofrecen una unidad y los destinos demandan una unidad. Si denotamos por x_{ij} las variables binarias que representan que el trabajador i realice la tarea j, el problema de asignación puede escribirse como

$$\begin{aligned} \text{mín} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\ s.a. \quad & \sum_{j=1}^{n} x_{ij} \leq 1, i = 1, ..., m. \\ & \sum_{i=1}^{m} x_{ij} \geq 1, i = 1, ..., n. \\ & x_{ij} \in \{0, 1\}. \end{aligned}$$

Por lo tanto, este problema puede resolverse mediante técnicas generales de Programación Entera o Binaria, y también aplicando el algoritmo del problema de transporte. Sin embargo, vamos a ver un procedimiento que explota la estructura especial del problema de asignación y que es más eficiente que los algoritmos anteriores.

Se supone en principio que es posible asignar cualquier trabajador a cualquier tarea. En la práctica es posible que esto no sea así; en ese caso asignaremos valor M al correspondiente c_{ij}, de forma que si al resolver el problema obtenemos que el coste depende de M, esto implicará que el problema es infactible.

El procedimiento que vamos a desarrollar necesita que el problema esté *balanceado*, que en este problema particular implica que el número de trabajadores y de tareas es el mismo. Si el problema no está balanceado, siempre podemos balancearlo añadiendo trabajadores o tareas ficticias y asignando costes nulos. Nótese que, si tenemos n trabajadores y n tareas, en la solución final sólo tenemos n variables con valor 1; si aplicamos el algoritmo de transporte, en cada S.B.F. tenemos $2n - 1$ variables básicas; por lo tanto, si aplicamos el problema de transporte para resolver el problema de asignación tendremos una solución altamente degenerada. De la misma forma, las S.B.F. intermedias serán degeneradas y esto hace que el algoritmo de transporte sea poco eficiente. Más aun, en cualquier asignación hay una y sólo una variable con valor 1 por fila y por columna.

Supongamos que tenemos un problema de asignación balanceado; así, todos los trabajadores realizan un trabajo y se realizan todos los trabajos. Además, todas las desigualdades pasan a ser igualdades y el problema queda

$$\begin{aligned} \text{mín} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\ s.a. \quad & \sum_{j=1}^{n} x_{ij} = a_i, \; i = 1, ..., m. \\ & \sum_{i=1}^{m} x_{ij} = b_j, \; j = 1, ..., n. \\ & x_{ij} \in \{0, 1\}. \end{aligned}$$

Consideremos la forma matricial del problema. En este caso la matriz de restricciones $A \in \mathcal{M}_{(n+n)\times n*n}$ es

$$
A = \begin{array}{c}
\begin{array}{ccccccccccc} x_{11} & \dots & x_{1n} & x_{21} & \dots & x_{2n} & \dots & x_{n1} & \dots & x_{nn} \end{array} \\
\left(\begin{array}{ccccccccccc}
1 & \dots & 1 & 0 & \dots & 0 & \dots & 0 & \dots & 0 \\
0 & \dots & 0 & 1 & \dots & 1 & \dots & 0 & \dots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
0 & \dots & 0 & 0 & \dots & 0 & \dots & 1 & \dots & 1 \\
1 & \dots & 0 & 1 & \dots & 0 & \dots & 1 & \dots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
0 & \dots & 1 & 0 & \dots & 1 & \dots & 0 & \dots & 1
\end{array} \right)
\end{array}
$$

$$
=: \quad (a^{11}...a^{1n}a^{21}...a^{2n}...a^{n1}...a^{nn}).
$$

Proposicíon 3. *La matriz A es totalmente unimodular, es decir, para toda $A' \subseteq A$ que sea cuadrada, $det(A') = 0, 1$ ó -1.*

Demostración: Basta tener en cuenta que la suma de las n primeras filas coincide con la suma de las n últimas y aplicar la Proposición 2. □

Como A es unimodular y el vector de términos independientes es entero y vale uno, esto implica que la solución es entera y que los valores de las variables en una S.B.F. son cero o uno. Por lo tanto, el problema puede escribirse como

$$
\begin{aligned}
\text{mín} \quad & \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\
s.a. \quad & \sum_{j=1}^{n} x_{ij} = a_i, \; i = 1, ..., m. \\
& \sum_{i=1}^{m} x_{ij} = b_j, \; j = 1, ..., n. \\
& x_{ij} \geq 0
\end{aligned}
$$

y resolverse aplicando el algoritmo del simplex.

2.2. El algoritmo húngaro

Para resolver el problema de asignación vamos a utilizar el conocido como algoritmo húngaro. Veamos el funcionamiento de este algoritmo. En primer lugar, el problema de asignación tiene la siguiente propiedad.

Proposicíon 4. *Si los valores $c_{ij}, i = 1, ..., m, j = 1, ..., n$ se modifican de forma que a cada c_{ij} pasa a ser $c_{ij} + p_i + q_j, i = 1, ..., m, j = 1, ..., n$, entonces se obtiene un problema de asignación equivalente.*

Demostración: En efecto, lo único que varía en el nuevo problema es la función objetivo. Denotemos por z' esta nueva función y por z la función objetivo antigua. Entonces, para una solución factible $x_{ij}, i = 1, ..., m, j = 1, ..., n$, se tiene:

$$z' = \sum_{i,j}(c_{ij}+p_i+q_j)x_{ij} = \sum_{i,j} c_{ij}x_{ij} + \sum_i p_i \sum_j x_{ij} + \sum_j q_j \sum_i x_{ij} = z + \sum_i p_i \sum_j x_{ij} + \sum_j q_j \sum_i x_{ij}.$$

Ahora bien, $\sum_i x_{ij} = 1 = \sum_j x_{ij}$, por las restricciones del problema de asignación. Luego,

$$z' = z + \sum_i p_i + \sum_j q_j = z + cte,$$

con lo que el óptimo se encuentra para los mismos valores de $x_{ij}, i = 1, ..., m, j = 1, ..., n$ en los dos problemas. □

La idea del algoritmo húngaro es la siguiente: Sumaremos o restaremos cantidades por filas y columnas hasta que todos los $c_{ij}, i, j = 1, ..., n$ sean no negativos. Entonces, si consiguiésemos una asignación de valor cero sería la solución. Se trata de encontrar las constantes a restar o sumar de forma que sea posible encontrar una asignación en esas condiciones.

Para resolver el problema de asignación consideraremos la tabla de costes, al igual que en el problema de transporte. Los pasos del algoritmo húngaro son los siguientes:

Paso 1. Inicialización. Se trata de conseguir al menos un cero en cada fila y en cada columna. Para ello se hace lo siguiente:

Paso 1.1. Restar a los elementos de cada fila el mínimo de esa fila. De esta manera obtenemos al menos un cero por fila y que todos los valores de la matriz sean no negativos.

Paso 1.2. Restar a los elementos de cada columna el mínimo de esa columna. De esta manera obtenemos al menos un cero por columna.

Ejemplo 2. *(Continuación del Ejemplo 1) En nuestro caso ya tenemos el problema balanceado. Aplicando el paso de inicialización se obtiene la siguiente matriz.*

$$\begin{pmatrix} 1 & 4 & 6 & 3 \\ 9 & 7 & 10 & 9 \\ 4 & 5 & 11 & 7 \\ 8 & 7 & 8 & 5 \end{pmatrix} \sim \begin{pmatrix} 0 & 3 & 5 & 2 \\ 2 & 0 & 3 & 2 \\ 0 & 1 & 7 & 3 \\ 3 & 2 & 3 & 0 \end{pmatrix} \sim \begin{pmatrix} 0 & 3 & 2 & 2 \\ 2 & 0 & 0 & 2 \\ 0 & 1 & 4 & 3 \\ 3 & 2 & 0 & 0 \end{pmatrix}.$$

Paso 2. Asignación. En este paso se trata de buscar una solución factible que tenga valor 0, teniendo en cuenta que una asignación consiste en elegir una variable por fila y por columna. Para hallar el máximo número de ceros que podemos asignar en la matriz actual usaremos el *algoritmo de emparejamiento máximo en grafos bipartitos*. Para ello necesitamos introducir una serie de conceptos sobre grafos.

Definición 2. *Dado un grafo no dirigido $G = (V, E)$, se llama un* **emparejamiento** *M a un subconjunto de aristas tal que cada uno de los vértices de G es incidente a lo sumo con una arista de M.*

Dado un emparejamiento, tenemos entonces un subconjunto de vértices que no inciden con ninguna de las aristas de M. Estos vértices se llaman *vértices expuestos*.

Definición 3. *Un* **cubrimiento** *es un conjunto de vértice $R \subseteq V$ tal que cualquier arista incide en algún vértice de R.*

Los problemas de encontrar el emparejamineto M tal que $|M|$ máximo y de ecotrar un cubrimiento R tal que $|R|$ mínimo están relacionados por el siguiente resultado:

Lema 2. *En un grafo no orientado, dados un emparejamiento M y un cubrimiento R cualesquiera, se tiene que $|R| \geq |M|$.*

Demostración: Dado $|M| = k$, entonces $M = \{\{i_1, f_1\}, ..., \{i_k, f_k\}\}$ y como M es un emparejamiento todos estos vértices son diferentes. Ahora, como R es un cubrimiento, tiene que incidir en todas las aristas de M, con lo que necesariamente $|R| \geq k$.

Definición 4. *Un grafo no dirigido se dice* **bipartito** *si existe una partición $\{V_1, V_2\}$ de los vértices del grafo de forma que cualquier arista tenga un vértice en V_1 y el otro en V_2.*

Entonces, en el caso del problema de asignación, se puede construir un grafo bipartito, en el que los vértices de V_1 son los orígenes, los vértices de V_2 los destinos y tomando como aristas las conexiones entre un origen y un destino si el valor de esta asignación es cero. Entonces, el problema de encontrar el máximo número de ceros de forma que se mantenga la factibilidad es equivalente a encontrar un emparejamiento en ese grafo que tenga cardinalidad máxima.

La idea del algoritmo es usar lo que se conoce como cadena de aumento.

Definición 5. *Dado un emparejamiento M, diremos que una secuencia de aristas $v_0 - v_1 - ... - v_p$ con p impar es un* **cadena de aumento** *si cumple:*

- *v_0 es un vértice expuesto.*
- *Las aristas del tipo $v_i - v_{i+1}$ no están en M cuando i es par.*
- *Las aristas del tipo $v_i - v_{i+1}$ están en M cuando i es impar.*
- *v_p es un vértice expuesto.*

La idea subyacente bajo el concepto de cadena de aumento es que se puede construir un emparejamiento con una arista más que M. Para ello se comienza con un vértice expuesto v_0. Este vértice se conecta con un vértice v_1 que no está expuesto; entonces la arista $v_0 - v_1$ se añade a M; como consecuencia de esto, M deja de ser un emparejamiento, ya que v_1 incide en dos aristas. Para que vuelva a ser un emparejamiento, quitamos la arista $v_1 - v_2$ de M. A continuación v_2 se une con un vértice v_3 que incide en M; por construcción de M, no puede ser que $v_2 - v_3$ esté en M. Entonces añadimos esta arista a M y el proceso se repite. El punto final es cuando se llega desde v_{p-1} a un vértice que no está expuesto. Entonces, como p es impar, hemos añadido una arista más de las que hemos quitado y así hemos encontrado un emparejamiento con una arista más.

Consideremos entonces el problema de asignación y lo pasamos a un problema de emparejamiento máximo en un grafo bipartito. El algortimo para hallar un emparejamiento máximo es el siguiente:

Paso 2.1. Sea un emparejamiento inicial (posiblemente $M = \emptyset$).

Paso 2.2. Se etiquetan con * los vértices de V_1 que estén expuestos.

Paso 2.3. Se selecciona un vértice de V_1 que esté etiquetado y no estudiado. Si no existen vértices en estas condiciones, el algoritmo ha terminado.

Paso 2.4 Sea i un vértice de V_1 que está etiquetado. Entonces, se asigna la etiqueta i a todos los vértices de V_2 que no tengan etiqueta y tales que $\{i, j\} \in E \backslash M$. Si no hay tales vértices se vuelve al Paso 2.3.

Paso 2.5. Sea $j \in V_2$ con etiqueta.

- Si j no está expuesto, se busca el vértice k de V_1 tal que $\{k, j\} \in M$ y se asigna a k la etiqueta j. Se vuelve al Paso 2.3.
- Si j está expuesto, entonces se ha encontrado una cadena de aumento y se cambia M por el nuevo emparejamiento. Volver al Paso 2.1.

Sea V_1^+, V_2^+ los vértices que están etiquetados al final del algoritmo. Entonces, se tiene lo siguiente:

Proposicíon 5. *Al finalizar el algoritmo:*

1. $R := V_1^- \cup V_2^+$ *es un cubrimiento de las aristas del grafo.*
2. $|R| = |M|$ *y* M *es un emparejamiento máximo.*

Demostración: Haremos la demostración en tres pasos.

1. Veamos en primer lugar que $R := V_1^- \cup V_2^+$ es un cubrimiento de las aristas del grafo. Para ello, supongamos que existe una arista (i, j) entre V_1^+ y V_2^-.
   a) Si fuese una arista de M, entonces el vértice i de V_1^+ no estaría expuesto inicialmente, y la única forma de incluirlo en V_1^+ es que hubiese una arista $(i, j) \in M$ con $j \in V_2^+$, lo que es una contradicción, puesto que entonces hay dos aristas de M que inciden en i.
   b) Si no fuese una arista de M, entonces como $i \in V_1^+$, se tiene que $j \in V_2^+$ según el tercer paso del algoritmo.
2. Sea $j \in V_2^+$. Entonces:
   a) j está en una arista de M. Si no fuese así, entonces j estaría expuesto y como $j \in V_2^+$, es que hay un $i \in V_1^+$ tal que $(i, j) \notin M$. Pero entonces se tendría una cadena de mejora y el algoritmo no habría terminado.
   b) Si $(i, j) \in M$, entonces $i \in V_1^+$, puesto que $j \in V_2^+$, y entonces i se habría marcado en el cuarto paso del algoritmo.
3. Sea $i \in V_1^-$. Entonces:
   a) i está en una arista de M. Si no fuese así, i estaría expuesto y entonces $i \in V_1^+$ según el primer paso del algoritmo.
   b) Si $(i, j) \in M$, entonces $j \in V_2^-$ puesto que si $j \in V_2^+$, esto llevaría a que $i \in V_1^+$ según el tercer paso del algoritmo.

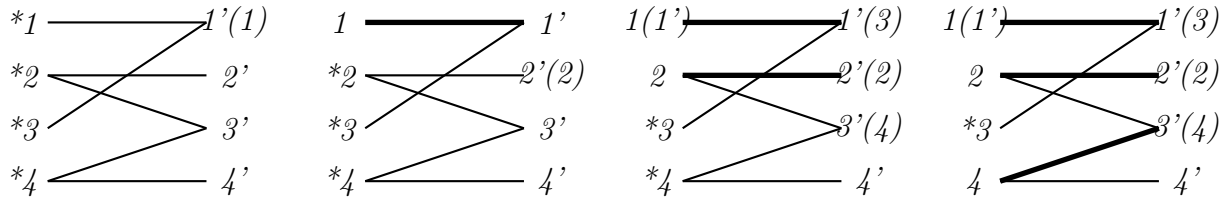
En definitiva,

$$\left.\begin{array}{l} |M| \geq |V_1^-| + |V_2^+| \\ |M| \leq |V_1^-| + |V_2^+| \end{array}\right\} \Rightarrow |M| = |V_1^-| + |V_2^+|$$

y entonces M es una asignación óptima. □

Si conseguimos un emparejamiento de cardinal n, entonces ya hemos encontrado una asignación óptima.

Ejemplo 3. *(Continuación del Ejemplo 1) En nuestro ejemplo vamos a proceder de la siguiente manera, dado el grafo bipartito siguiente:*



Iteración 1 *Iteración 2* *Iteración 3* *Iteración 4*

*Comenzamos con $M = \emptyset$. Entonces todos los vértices de V_1 están etiquetados con *. Si empezamos con 1, entonces tenemos la arista 1-1' y como 1' está expuesto, entonces tenemos una cadena de aumento, con lo que hacemos $M = \{\{1, 1'\}\}$ y volvemos a empezar.*

En la segunda iteración todos los vértices de V_1 están etiquetados salvo 1. Si escogemos 2, entonces tenemos la arista 2-2' y como 2' está expuesto, entonces tenemos una cadena de aumento, con lo que hacemos $M = \{\{1, 1'\}, \{2, 2'\}\}$ y volvemos a empezar.

*En la tercera iteración los vértices de V_1 que están etiquetados con * son 3 y 4. Si escogemos 3, entonces tenemos la arista 3-1' y como 1' no está expuesto, entonces cogemos 1'-1 (etiqueta 1' para 1). Como no es posible seguir, no tenemos una cadena de aumento de esta forma. Pasamos a coger el vértice 4, entonces tenemos la arista 4-3' y como 3' está expuesto, entonces tenemos una cadena de aumento, con lo que hacemos $M = \{\{1, 1'\}, \{2, 2'\}, \{4, 3'\}\}$ y volvemos a empezar.*

*En la tercera iteración el único vértice de V_1 que está etiquetados con * es 3. Si escogemos 3, entonces tenemos la arista 3-1' y como 1' no está expuesto, entonces cogemos 1'-1. Como no es posible seguir, no tenemos una cadena de aumento de esta forma. Y ya no se puede seguir con lo que el algoritmo termina. La matriz de asignación quedaría de la siguiente manera:*

$$\begin{pmatrix} \mathbf{0} & 3 & 2 & 2 \\ 2 & \mathbf{0} & \mathit{0} & 2 \\ \mathit{0} & 1 & 4 & 3 \\ 3 & 2 & \mathbf{0} & \mathit{0} \end{pmatrix}.$$

donde los ceros en negrita expresan la mejor asignación de ceros que podemos hacer con esos ceros, y los demás ceros aparecen en cursiva.

Como ha sucedido en el ejemplo anterior, en muchas ocasiones no seremos capaces de obtener una asignación completa. La idea entonces del algoritmo húngaro es restar otras cantidades siguiendo la Proposición 4 de manera que obtengamos una nueva matriz de términos no negativos equivalente a la anterior con ceros colocados en otras posiciones. Si no se ha obtenido una asignación de valor cero, el siguiente paso del algoritmo húngaro es hallar el mínimo número de líneas para cubrir todos los ceros de la matriz de asignación. Esto se consigue de la siguiente manera:

Paso 3. Determinar el número mínimo de líneas que cubren todos los ceros.

Paso 3.1. Marcar con "x" toda fila que no contenga un cero encuadrado.

Paso 3.2. Marcar con "x" toda columna con un cero tachado en una fila marcada.

Paso 3.3. Marcar con "x" toda fila con un cero encuadrado en una columna tachada.

Paso 3.4. Reiterar los pasos 3.2 y 3.3 mientras sea posible.

Paso 3.5. Pasar líneas a través de las filas no marcadas y las columnas marcadas.

La idea de este algoritmo es la siguiente: Consideremos una fila que no tiene ceros encuadrados (paso 3.1). Entonces necesariamente tiene que tener algún cero tachado (paso 3.2). Esto implica que en la columna de ese cero hay un cero encuadrado (paso 3.3). Ahora consideraremos la fila con el cero encuadrado como una fila sin ceros encuadrados (paso 3.4), con lo que volvemos al paso 3.2 y el proceso vuelve a empezar. Ahora en el paso 3.5 pasamos una línea por filas con ceros encuadrados que no hayan aparecido en el proceso anterior. Y se tachan las columnas marcadas, con lo que se eliminan los ceros tachados y encuadrados que aparecen en el proceso anterior.

Ejemplo 4. *(Continuación del Ejemplo 1) En nuestro caso el proceso es el siguiente:*

0	*3*	*2*	*2*	
2	**0**	*0*	*2*	
0	*1*	*4*	*3*	
3	*2*	**0**	*0*	

0	*3*	*2*	*2*	
2	**0**	*0*	*2*	
0	*1*	*4*	*3*	×
3	*2*	**0**	*0*	

				×
0	*3*	*2*	*2*	
2	**0**	*0*	*2*	
0	*1*	*4*	*3*	×
3	*2*	**0**	*0*	

				×
0	*3*	*2*	*2*	×
2	**0**	*0*	*2*	
0	*1*	*4*	*3*	×
3	*2*	**0**	*0*	

Luego se obtiene

				×
0*(M)*	*3*	*2*	*2*	×
2(M)	**0***(M)*	*0(M)*	*2(M)*	
0(M)	*1*	*4*	*3*	×
3(M)	*2(M)*	**0***(M)*	*0(M)*	

Proposicíon 6. *El algoritmo anterior cubre todos los ceros.*

Demostración: Supongamos que hay un cero no cubierto. Entonces ese cero tiene que estar en una fila marcada y en una columna no marcada.

- Si es un cero encuadrado entonces, como está en una fila marcada, es porque está en una columna marcada, lo que es una contradicción.
- Si es un cero tachado entonces, como está en una columna no marcada, todos los ceros tachados de esa columna no están en filas marcadas por el Paso 3.2; en particular, este cero no está en una fila marcada, lo que es una contradicción.

Esto finaliza la demostración. □

Proposicíon 7. *El número de líneas que se obtienen utilizando el algoritmo anterior coincide con el cardinal del emparejamiento máximo.*

Demostración: Haremos la demostración en tres pasos:

1. Hay más líneas que ceros encuadrados. Esto es así porque el algoritmo cubre todos los ceros, con lo que por cada cero encuadrado pasa al menos una línea. Y no es posible que una línea cubra dos ceros encuadrados porque entonces estarían en la misma fila o en la misma columna, lo que contradice que se tenga un emparejamiento.
2. Veamos ahora que no hay líneas que no cuabran ningún cero encuadrado. Supongamos que hubiese una línea en estas condiciones; como las filas no marcadas tienen un cero encuadrado, esto implica que esta línea tiene que ser una línea que cubra una columna, es decir, que corresponde a una columna marcada. Por otra parte, esta columna no se puede marcar en el Paso 3.2, puesto que en este caso tendría un cero encuadrado. Por ello, necesariamente se ha cubierto a partir del Paso 3.3. Por ello, corresponde a un cero no encuadrado en una fila marcada; y además no hay ningún cero encuadrado en esa columna. Por otra parte, en esa fila hay un cero encuadrado en una columna marcada. Y si esa columna está marcada es porque hay un cero no encuadrado en esa columna que está en una fila marcada. Reiterando este proceso se llegará a una columna marcada debido a que hay un cero tachado en una fila sin ceros encuadrados. Entonces podemos encuadrar los cero tachados de las columna que estamos considerando y desencuadrar los ceros encuadrado en las filas que estamos considerando, de forma que tendríamos un cero encuadrado más. Pero esto contradice que nuestra familia de ceros encuadrados es óptima.
3. Finalmente, nótese que por un cero encuadrado no pueden pasar dos líneas, ya que tendrían que ser líneas en su fila y en su columna, y si su columna está marcada, entonces se marca la fila, con lo que no pasaría línea por esa fila.

Esto finaliza la demostración. □

Paso 4. Actualización de la matriz de asignación.

Consideremos ahora los elementos que no han sido tachados. Todos esos elementos son positivos, pues todos los ceros están cubiertos. Sea r el mínimo de todos estos elementos. Definimos F los subíndices de las filas marcadas y por G los subíndices de las columnas no marcadas; es decir, F y G denotan los índices de las filas y columnas no tachadas. Denotemos por c'_{ij} los nuevos valores de la matriz de asignación. Estos valores vienen definidos por:

$$c'_{ij} := \begin{cases} c_{ij} - r & \text{si i} \in \text{F}, \text{j} \in \text{G} \\ c_{ij} & \text{si i} \notin \text{F}, \text{j} \in \text{G} \\ c_{ij} & \text{si i} \in \text{F}, \text{j} \notin \text{G} \\ c_{ij} + r & \text{si i} \notin \text{F}, \text{j} \notin \text{G} \end{cases}$$

Nótese que estamos ante un problema de asignación equivalente, porque la definición anterior consiste en restar r a las filas no tachadas y sumar r a las columnas tachadas.

Tenemos ahora que comprobar que la nueva matriz de asignación tiene todos sus elementos no negativos; así, volvería a ser óptima una asignación de valor cero. Pero esto es cierto, puesto que los elementos que son menores que en la matriz anterior son los que no están tachados y se les está restando el mínimo valor entre todos ellos.

Paso 5. Reiterar los pasos 3 y 4 hasta obtener una asignación de valor cero.

Veamos finalmente que con este método se termina en un número finito de pasos.

Para ello, veremos que la suma de coeficientes en la nueva matriz de asignación es menor que en la anterior. Como todos son no negativos, esto implica que en algún momento llegaríamos a una matriz completa de ceros.

La diferencia en la suma de coeficientes es:

$$\sum_{ij} c_{ij} - \sum_{ij} c'_{ij} = \sum_{i\in F, j\in G} r - \sum_{i\notin F, j\notin G} r.$$

Ahora bien, $\sum_{i\in F, j\in G} r = |F||G|r$. Por otra parte, $|F| + |G| = k$, el número de ceros encuadrados. Entonces,

$$\sum_{i\notin F, j\notin G} r = (n - |F|)(n - |G|)r = (n^2 - n(|F| + |G|) + |F||G|)r.$$

Luego,

$$\sum_{ij} c_{ij} - \sum_{ij} c'_{ij} = (n^2 - n(|F| + |G|))r = n(n - k)r > 0,$$

puesto que no teníamos una asignación completa de ceros encuadrados.

Nótese que no es posible que cada vez se vaya reduciendo el valor en menor cuantía de forma que se tenga una sucesión que converja a cero en un número infinito de iteraciones, porque estamos hablando de diferencias entre un número finito de valores, con lo que el valor r sólo puede tomar un número finito de valores.

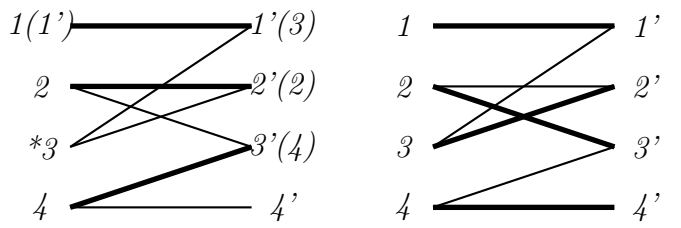
Ejemplo 5. *(Continuación del Ejemplo 1) Ya habíamos obtenido anteriormente la matriz*

×				
0*(M)*	*3*	*2*	*2*	×
2(M)	**0***(M)*	*0(M)*	*2(M)*	
0(M)	*1*	*4*	*3*	×
3(M)	*2(M)*	**0***(M)*	*0(M)*	

Entonces el mínimo valor no marcada es 1 y la nueva matriz de asignación será

$$\begin{pmatrix} 0 & 2 & 1 & 1 \\ 3 & 0 & 0 & 2 \\ 0 & 0 & 3 & 2 \\ 4 & 2 & 0 & 0 \end{pmatrix}.$$

Reiterando el Paso 3 se tiene



Iteración 1 *Iteración 2*

Comenzamos con $M = \{\{1, 1'\}, \{2, 2'\}, \{4, 3'\}\}$. *De esta forma, el único vértice de* V_1 *que está etiquetados con * es 3. Si escogemos 3, entonces tenemos la arista 3-1' y como 1' no está expuesto, entonces cogemos 1'-1. Como no es posible seguir, no tenemos una cadena de aumento de esta forma.*

El otro vértice de V_2 *etiquetado con (3) es 2'. En este caso, como 2' no está expuesto, tenemos la arista 2'-2, porlo que etiquetamos 2 con (2'); ahora partiendo de 2 tenemos 2-3', con lo que 3' se etiqueta con (2); como 3' no está expuesto, tenemos 3'-4, por lo que etiquetamos 4 con (3'); finalmente, tenmos 4-4', con lo que etiquetamos 4' con (4). Como 4' está expuesto, hemos conseguido una cadena de aumento. Esta cadena consiste en incluir en M las aristas en posición impar y sacar las que están en posición par. Por lo tanto,* $M = \{\{1, 1'\}, \{2, 3'\}, \{3, 2'\}, \{4, 4'\}\}$. *Y ya no se puede seguir, con lo que el algoritmo termina. La matriz de asignación quedaría de la siguiente manera:*

$$\begin{pmatrix} \mathbf{0} & 2 & 1 & 1 \\ 3 & \mathit{0} & \mathbf{0} & 2 \\ \mathit{0} & \mathbf{0} & 3 & 2 \\ 4 & 2 & \mathit{0} & \mathbf{0} \end{pmatrix},$$

que ya nos proporciona una asignación de ceros. Por lo tanto, la solución óptima del problema es

Coche	*1*	*2*	*3*	*4*
Cliente	*1*	*3*	*2*	*4*

El valor de la función objetivo es 21.