AMPLIACIÓN DE PROGRAMACIÓN ENTERA

Pedro Miranda Menéndez

Resumen

Eneste tema estudiaremos algunos problemas de Programación Entera. En primer lugar 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. A continuación veremos el algoritmo de Balas, que permite resolver problemas de Programación Binaria y que está inspirado en el algoritmo de Ramificación y Acotación de P. Entera. Finalmente estudiaremos el problema de asignación, que aunque puede resolverse mediante el algoritmo del simplex o mediante técnicas de Programación Entera o incluso mediante el algoritmo de Balas, 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.

1. Matrices unimodulares

Consideremos un problema de Programación Entera (PE) dado por

$$\begin{array}{lll} \text{mín} & \vec{c}^t\vec{x} & \\ s.a & A\vec{x} & \leq \vec{b} \\ & \vec{x} \geq \vec{0} & \vec{x} \in \mathbb{Z}^n \end{array}$$

y su correspondiente Problema Relajado (PR)

$$\begin{array}{lll} \text{mín} & \vec{c}^t\vec{x} & \\ s.a & A\vec{x} & \leq \vec{b} \\ & \vec{x} \geq \vec{0} & \end{array} .$$

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.

PR	PE
Infactible	Infactible
S. no acotada	S. no acotada Infactible
S. finita	S. finita Infactible

El problema es que la solución óptima 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 B de (A, Id). Entonces, si desarrollamos el determinante por las columnas que provienen de Id se tiene que o bien $|B| = 0$ ó bien

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

donde B' es una submatriz cuadrada de A con lo que $|B'| = 1, -1$ ó 0. 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. ■

Proposición 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.

En general, estudiar la unimodularidad de una matriz a partir de la definición es poco operativo. Veamos ahora un resultado que nos permite concluir de forma más cómoda que una matriz es unimodular.

Proposición 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 algoritmo de Balas

El algoritmo de Balas es un método para resolver problemas de Programación Binaria

$$\begin{array}{lll} \text{mín} & \vec{c}^t\vec{x} & \\ s.a & A\vec{x} & \leq \vec{b}\ . \\ & \vec{x} \in \{0,1\}^n & \end{array}$$

Las variables binarias aparecen con bastante frecuencia en problemas en los que $x_i = 1$ representa que algo sucede o se realiza y $x_i = 0$ significa la no aparición o no realización. Además, al tomar las variables el valor 0 ó 1, el problema no puede tener s. no acotada, lo que lo hace un problema especial. El algoritmo de Balas sirve para resolver problemas en los que *todas las variables involucradas son binarias*. El algoritmo de Balas es una modificación del método de ramificación y acotación en el que se tiene en cuenta que las variables sólo pueden tomar dos valores, 0 y 1. La idea es comenzar por la mejor solución posible; si es factible el problema está resuelto. Si no es factible, ramificamos el problema fijando los valores de una de las variables y hallamos la solución óptima en cada rama. Como veremos al desarrollar el método, la obtención de esta solución es inmediata y sólo es necesario evaluar la factibilidad.

El algoritmo de Balas tiene dos pasos: un paso de preparación del problema y un paso iterativo.

En el Paso de Preparación del problema el problema original se transforma de forma que sea trivial encontrar la solución óptima. Para ello, la idea es que si el problema fuese de minimizar y todas las variables tuviesen coeficiente positivo en la función objetivo, la mejor solución posible sería aquélla en que todas las variables tomasen el valor cero. Los pasos en esta etapa del problema son los siguientes:

- **Transformar el problema original en un problema de minimizar:** Si el problema original es de tipo minimizar no hay que hacer nada; si fuese de maximizar, hay que pasar a minimizar el opuesto de la función objetivo.
- **Conseguir que todas las variables tengan coeficiente positivo en la función objetivo:** Si alguna variable x_i tuviese coeficiente negativo, se considera el cambio de variable $y_i = 1 - x_i$; entonces $x_i = 1 - y_i$ y sustituyendo en la función objetivo *y en las restricciones*, se tiene que y_i tendría coeficiente positivo en la función objetivo. Nótese que y_i es también una variable binaria, con lo que las condiciones del problema no se alteran.
- **Ordenar las variables en orden creciente de su coeficiente en la función objetivo.** Para simplificar la notación, consideraremos que las variables están ordenadas según el orden natural, es decir, la variable con menor coeficiente es la primera (x_1 ó y_1), luego x_2 ó y_2, y así sucesivamente.

Ejemplo 1. *Consideremos el problema*

$$\begin{array}{rll} \text{máx} & 4x_1 + 10x_2 - 7x_3 + 5x_4 & \\ \text{sujeto a} & -x_1 - 3x_2 + 2x_3 - 5x_4 & \leq 1 \\ & -2x_1 + x_2 + 2x_3 - 2x_4 & \geq 0 \\ & x_1, x_2, x_3, x_4 \in \{0, 1\} & \end{array}$$

Pasamos primeramente el problema a minimizar

$$\begin{array}{rll} \text{mín} & -4x_1 - 10x_2 + 7x_3 - 5x_4 & \\ \text{sujeto a} & -x_1 - 3x_2 + 2x_3 - 5x_4 & \leq 1 \\ & -2x_1 + x_2 + 2x_3 - 2x_4 & \geq 0 \\ & x_1, x_2, x_3, x_4 \in \{0, 1\} & \end{array}$$

A continuación, hacemos los cambios de variable para las variables con coeficiente negativo: $y_1 = 1 - x_1$, $y_2 = 1 - x_2$, $y_4 = 1 - x_4$, *obteniéndose*

$$\begin{array}{rll} \text{mín} & -4(1 - y_1) - 10(1 - y_2) + 7x_3 - 5(1 - y_4) & \\ \text{sujeto a} & -(1 - y_1) - 3(1 - y_2) + 2x_3 - 5(1 - y_4) & \leq 1 \\ & -2(1 - y_1) + 1 - y_2 + 2x_3 - 2(1 - y_4) & \geq 0 \\ & y_1, y_2, x_3, y_4 \in \{0, 1\} & \end{array}$$

que, haciendo las cuentas, conduce al problema equivalente

$$\begin{array}{rll} \text{mín} & 4y_1 + 10y_2 + 7x_3 + 5y_4 & \\ \text{sujeto a} & y_1 + 3y_2 + 2x_3 + 5y_4 & \leq 10 \\ & 2y_1 - y_2 + 2x_3 + 2y_4 & \geq 3 \\ & y_1, y_2, x_3, y_4 \in \{0, 1\} & \end{array}$$

Finalmente, ordenamos las variables de menor a mayor.

$$\begin{array}{rll} \text{mín} & 4y_1 + 5y_4 + 7x_3 + 10y_2 & \\ \text{sujeto a} & y_1 + 5y_4 + 2x_3 + 3y_2 & \leq 10 \\ & 2y_1 + 2y_4 + 2x_3 - y_2 & \geq 3 \\ & y_1, y_2, x_3, y_4 \in \{0, 1\} & \end{array}$$

Pasemos ahora al Paso Iterativo. En el problema resultante del Paso de Preparación, la solución óptima es que todas las variables tomen el valor 0. Si esta solución es factible entonces tiene que ser la s. óptima; y deshaciendo el cambio de variable se obtiene la solución del problema original.

Supongamos que esta solución no es factible. Entonces ramificamos por la primera variable (x_1 ó y_1) que llamaremos z_1; entonces, tenemos dos nuevos subproblemas: uno con $z_1 = 0$ y otro con $z_1 = 1$. Nótese que ahora tenemos una variable menos. Se trata de resolver estos dos nuevos problemas. Pero la soluciones vuelven a ser triviales, pues serán $(0, ..., 0)$ y $(1, 0, ..., 0)$.

Nótese sin embargo que la primera solución coincide con la del problema anterior, que ya sabemos que no es factible. Para evitar comprobar la factibilidad de soluciones que ya hemos probado, tomaremos en estos casos la mejor solución excluida ésta; y esta nueva solución es otra vez muy sencilla de encontrar, pues será $(0, 1, 0, ..., 0)$.

El Paso Iterativo se reitera hasta que se cierren todas las ramas, ya sea por llegar a una solución óptima factible o por llegar a un problema infactible, ya sea por acotación. Si tenemos varias ramas por estudiar comenzaremos por la rama con mejor valor, pues es la que nos podría llevar a poder acotar otras ramas y reducir el número de ramas que se estudian.

Finalmente, se comparan las soluciones obtenidas en las ramas que se han cerrado por obtener una s. óptima. De entre estas soluciones, elegimos la que dé lugar al menor valor. Si ninguna rama se ha cerrado por esta razón, entonces el problema es infactible.

Para hallar la solución del problema original tenemos que deshacer el cambio de variable. El valor óptimo de la función objetivo podría hallarse también deshaciendo los cambios, pero es más rápido calcularlo a partir de los valores óptimos de las variables en la función objetivo original.

Ejemplo 2. *(Continuación del Ejemplo 1) El problema que se había obtenido en el Paso de Preparación era*

$$\begin{array}{rll} \text{mín} & 4y_1 + 5y_4 + 7x_3 + 10y_2 & \\ \text{sujeto a} & y_1 + 5y_4 + 2x_3 + 3y_2 & \leq 10 \\ & 2y_1 + 2y_4 + 2x_3 - y_2 & \geq 3 \\ & y_1, y_2, x_3, y_4 \in \{0, 1\} & \end{array}$$

La solución inicial $(0, 0, 0, 0)$ *es infactible. Ramificamos por* y_1.

Para el subproblema con $y_1 = 0$ *(P1) la solución es* $(0, 1, 0, 0)$ *con* $z = 5$ *y no verifica la segunda restricción. Para el subproblema con* $y_1 = 1$ *(P2) la solución es* $(1, 0, 0, 0)$ *con* $z = 4$ *y no verifica la segunda restricción. Por acotación, ramificamos el segundo de estos problemas por* y_4.

Para el subproblema con $y_4 = 0$ *(P2.1) la solución es* $(1, 0, 1, 0)$ *con* $z = 11$ *y es factible. Para el subproblema con* $y_4 = 1$ *(P2.2) la solución es* $(1, 1, 0, 0)$ *con* $z = 9$ *y también es factible.*

Falta entonces ramificar P1 por y_4. *Para el subproblema con* $y_4 = 0$ *(P1.1) la solución es* $(0, 0, 1, 0)$ *con* $z = 7$ *y no verifica la segunda restricción. Para el subproblema con* $y_4 = 1$ *(P1.2) la solución es* $(0, 1, 1, 0)$ *con* $z = 12$ *y es factible.*

Falta entonces ramificar P1.1 por x_3. *Para el subproblema con* $x_3 = 0$ *(P1.1.1) la solución es* $(0, 0, 0, 1)$ *con* $z = 10$ *y no verifica la segunda restricción. Para el subproblema con* $y_3 = 1$ *(P1.1.2) la solución es* $(0, 0, 1, 1)$ *con* $z = 17$ *y no verifica la segunda restricción. Sin embargo, ambas ramas quedan cerradas por acotación.*

De esta forma, todas las ramas quedan cerradas y el algoritmo finaliza. Entre las soluciones de las ramas, la mejor es $z = 9$, *que se alcanza para* $y_1 = 1, y_4 = 1, x_3 = 0, y_2 = 0$. *Deshaciendo el cambio de variable obtenemos como solución*

$$x_1 = 0, x_4 = 0, x_3 = 0, x_2 = 1,$$

que sustituyendo en la función objetivo original nos proporciona un valor $z = 10$.

El proceso seguido en la resolución puede verse en la Figura 1.

3. El problema de asignación

3.1. Planteamiento

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

Ejemplo 3. *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. 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.

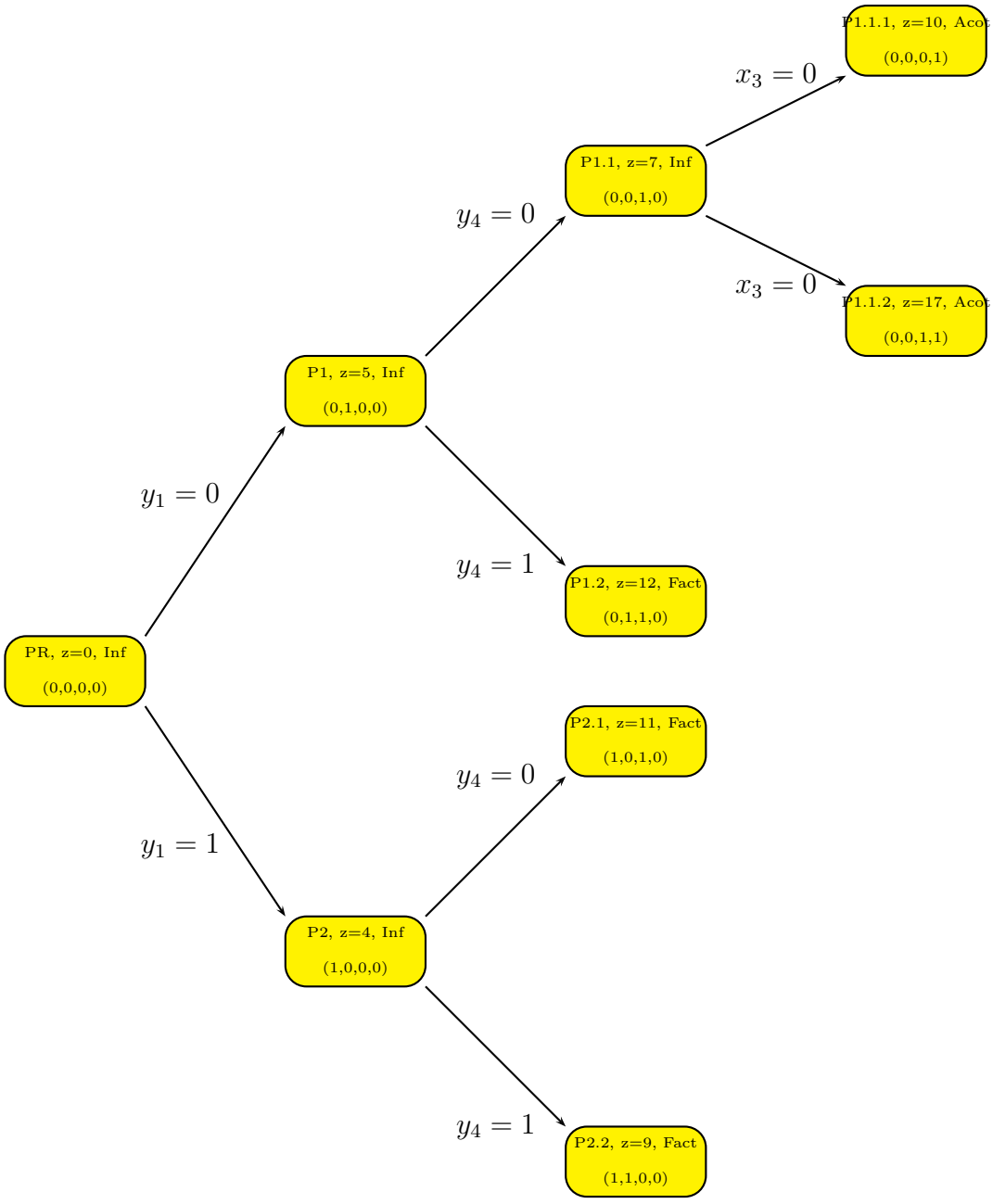


Figura 1: Diagrama en árbol de resolución.

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 significa 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 sin embargo la diferencia entre estas dos situaciones. Si añadimos tareas el problema original es factible, mientras que si añadimos trabajadores el problema original es infactible. En este último caso, el algoritmo nos dice cómo asignar los trabajadores de forma que los costes sean mínimos aunque no se realicen todas las tareas.

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

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

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

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

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

Proposición 3. *La matriz A es totalmente unimodular.*

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 óptima 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

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

y resolverse aplicando el algoritmo del simplex. Nótese sin embargo que, si tenemos n trabajadores y n tareas, en la solución final sólo tenemos n variables con valor 1; si aplicamos el algoritmo del simplex, en cada S.B.F. tenemos $2n-1$ variables básicas (una de las restricciones es redundante); por lo tanto, si aplicamos el simplex para resolver el problema de asignación tendremos una solución óptima altamente degenerada. De la misma forma, las S.B.F. intermedias serán degeneradas y esto hace que el algoritmo del simplex sea poco eficiente.

3.2. 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.

Proposición 4. *Si los valores $c_{ij}, i = 1, ..., n, j = 1, ..., n$ se modifican de forma que cada c_{ij} pasa a ser $c_{ij} + p_i + q_j, i = 1, ..., n, j = 1, ..., n$, 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 $x_{ij}, i = 1, ..., n, j = 1, ..., n$, 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 $x_{ij}, i = 1, ..., n, j = 1, ..., n$ 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 trivialmente la solución del problema.

Se trata por tanto de encontrar las constantes a restar o sumar de forma que sea posible encontrar una asignación en esas condiciones. Nótese también que como las variables no nulas valen 1, entonces podemos identificar la variable x_{ij} con la casilla $\{i, j\}$ de la matriz de asignación. Por ello, el problema se reduce a identificar n casillas de forma que haya una casilla seleccionada por fila y por columna y que esta selección tenga coste mínimo. Nótese que hay $n!$ posibles elecciones y como este valor crece muy rápido no será eficiente evaluarlos todos y escoger la mejor selección.

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 4. *(Continuación del Ejemplo 3) En nuestro caso ya tenemos el problema balanceado. Aplicando el paso de inicialización se obtiene la siguiente matriz.*

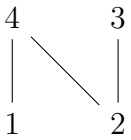
$$\begin{pmatrix} 1 & 4 & 6 & 3 \\ 9 & 7 & 10 & 9 \\ 4 & 5 & 11 & 7 \\ 8 & 7 & 8 & 5 \end{pmatrix} \sim \begin{pmatrix} 0 & 3 & 5 & 2 \\ 2 & 0 & 3 & 2 \\ 0 & 1 & 7 & 3 \\ 3 & 2 & 3 & 0 \end{pmatrix} \sim \begin{pmatrix} 0 & 3 & 2 & 2 \\ 2 & 0 & 0 & 2 \\ 0 & 1 & 4 & 3 \\ 3 & 2 & 0 & 0 \end{pmatrix}.$$

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.*

Ejemplo 5. *Dado el siguiente grafo, puede verse que $M = \{(1, 4), (2, 3)\}$ es un emparejamiento. También es un emparejamiento $M' = \{(4, 2)\}$.*

Figura 2: Ejemplo de emparejamiento en un grafo.



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**.

Ejemplo 6. *(Cont. del Ejemplo 5) Para M, tenemos que no hay vértice expuestos. Para M', los vértices $1, 3$ están expuestos.*

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

Ejemplo 7. *(Cont. del Ejemplo 5) Para el grafo de este ejemplo, los vértices $R = \{2, 4\}$ son un cubrimiento del grafo. También es un cubrimiento $R' = \{1, 2, 4\}$.*

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

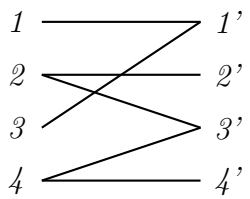
Lema 2. *En un grafo no orientado, dados un emparejamiento M y un cubrimiento R cualesquiera, se tiene que $|R| \geq |M|$.*

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 i_j ó f_j están en R, y así $|R| \geq k$. □

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 tiene 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.

Ejemplo 8. *(Continuación del Ejemplo 3) En nuestro ejemplo, dado la tabla que se obtiene al inicializar el problema, tendríamos el grafo bipartito siguiente:*



Entonces, el problema de encontrar el máximo número de ceros de forma que se mantenga la factibilidad (un cero a lo sumo por fila y por columna) 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.*

Ejemplo 9. *(Continuación del Ejemplo 3) Para el ejemplo anterior, y dado el emparejamiento* $M = \emptyset$*, tenemos la cadena de aumento* $1 - 1'$*. Si ahora consideramos el emparejamiento* $M' = \{(1, 1'), (2, 3')\}$*, entonces* $4 - 3' - 2 - 2'$ *es una cadena de aumento.*

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, pues en ese caso M dejaría de ser un emparejamiento. 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 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.

Proposición 5. *Sea* M *un emparejamiento que no admite ninguna cadena de aumento. Entonces* M *es un emparejamiento de cardinal máximo.*

Demostración: Sea M en las condiciones del enunciado y supongamos que no es un emparejamiento de cardinal máximo. Entonces exite otro emparejamiento M' tal que $|M'| > |M|$. Sea entonces el grafo $G' = (V, (M \cup M') \backslash (M \cap M'))$. Entonces, como M y M' son emparejamientos, todos los vértices tienen grado de incidencia 0, 1 ó 2. Como consecuencia, las componentes conexas de este grafo son cadenas o ciclos.

Si la componente conexa es un ciclo, dado un vértice inicial, las aristas de la componente conexa tienen que alternar aristas de M y M', puesto que ambos son emparejamientos. Y como es un ciclo y no puede empezar y acabar en una arista de M (o M'), el número de aristas es par.

Si tomamos una cadena, entonces dado uno de los vértices con grado de incidencia 1, nos ocurrirá que irán alternando las aristas de M y M'. Como $|M'| > |M|$, tiene que exitir una cadena con un número impar de aristas y en el que hay una arista más de M' que de M. Esta cadena es una cadena de aumento para M. ■

Consideremos entonces el problema de asignación y lo pasamos a un problema de emparejamiento máximo en un grafo bipartito. El algoritmo 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.

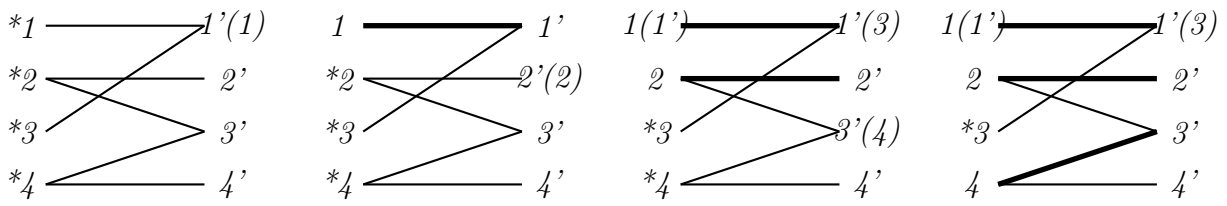
Paso 2.3. Se selecciona un vértice de V_1 que esté etiquetado y no estudiado. Si no existen vértices en estas condiciones, el algoritmo ha terminado.

Paso 2.4 Sea i un vértice de V_1 que está etiquetado y no estudiado. 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.

Ejemplo 10. *(Continuación del Ejemplo 3) 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:*

$$\begin{pmatrix} \boxed{0} & 3 & 2 & 2 \\ 2 & \boxed{0} & \not{0} & 2 \\ \not{0} & 1 & 4 & 3 \\ 3 & 2 & \boxed{0} & \not{0} \end{pmatrix}.$$

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

Proposición 6. *Al finalizar el algoritmo anterior:*

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 $(i, k) \in M$ con $k \in V_2^+$, 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 cuarto 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 esta arista es $(i, j) \in M$, entonces $i \in V_1^+$, puesto que $j \in V_2^+$, y entonces i se habría marcado en el quinto paso del algoritmo.

   En definitiva, (i, j) va desde V_1^+ a V_2^+.
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 esta arista es $(i, j) \in M$, entonces $j \in V_2^-$ puesto que si $j \in V_2^+$, esto llevaría a que $i \in V_1^+$ según el cuarto paso del algoritmo.

   En resumen, (i, j) va desde V_1^- a V_2^-.

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.

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. Entonces tendremos que sumar o restar cantidades por filas y columnas, pero manteniendo siempre la no negatividad de las cantidades que aparecen en la tabla. Para ello, en primer lugar vamos a cubrir todos los ceros de la matriz con líneas; el objetivo es usar el menor

número de líneas posible y esto se consigue identificando los trabajadores con sus filas y las tareas con sus columnas correspondientes. Por lo visto anteriormente, sabemos que $R := V_1^- \cup V_2^+$ es un cubrimiento de las aristas del grafo o, equivalentemente, si tachamos las filas correspondientes a V_1^- y las columnas correspondientes a V_2^+, entonces habremos cubierto todos los ceros de la tabla. Además, el número de líneas utilizado será el mismo que el número de ceros encuadrados y, por tanto, será menor que n (puesto que la solución no es óptima).

Ejemplo 11. *(Continuación del Ejemplo 3) En nuestro caso, ya hemos visto que en la última iteración tenemos $V_1^- = \{2, 4\}$ y $V_2^+ = \{1\}$, con lo que se obtendría la siguiente tabla:*

$\times$				
$\boxed{0}(M)$	3	2	2	$\times$
$2(M)$	$\boxed{0}(M)$	$\emptyset(M)$	$2(M)$	
$\emptyset(M)$	1	4	3	$\times$
$3(M)$	$2(M)$	$\boxed{0}(M)$	$\emptyset(M)$	

Paso 3. 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. Denotamos por F y G los índices de las filas y columnas tachadas. Denotemos por c'_{ij} los nuevos valores de la matriz de asignación. Estos valores vienen definidos por:

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

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 de entre todos ellos.

Paso 4. Reiterar los pasos 2 y 3 hasta obtener una asignación de valor cero.

Proposición 7. *El algoritmo de asignación termina en un número finito de pasos.*

Demostración: Veremos que la suma de coeficientes en la nueva matriz de asignación es menor que en la anterior. Como todos son no negativos, veremos que 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 en cada iteración de forma que se tenga una sucesión que no converja a cero, ni que esa convergencia se haga 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. □

Nótese finalmente que para volver a aplicar el Paso 2 no es necesario volver a empezar con el emparejamiento vacío, sino que se puede empezar con el emparejamiento óptimo conseguido en la iteración anterior. Para que esto sea posible se necesita que todos los ceros encuadrados en la iteración anterior sigan siendo ceros en la nueva matriz; y ello será así si no hay dos líneas que pasen por su posición. Pero esto es cierto ya que si pasasen dos líneas, esto implicaría que el cero en posición (i, j) es tal que $i \in V_1^-, j \in V_2^+$ y además $(i, j) \in M$, el emparejamiento óptimo obtenido al aplicar el Paso 2. Pero por otra parte, hemos visto en la demostración de la Proposición 6 que si $i \in V_1^-$ y además $(i, j) \in M$, entonces $j \in V_2^-$; y también que si $j \in V_2^+$ y además $(i, j) \in M$, entonces $i \in V_1^+$; llegamos entonces a contradicción y así no hay dos líneas que pasen por (i, j).

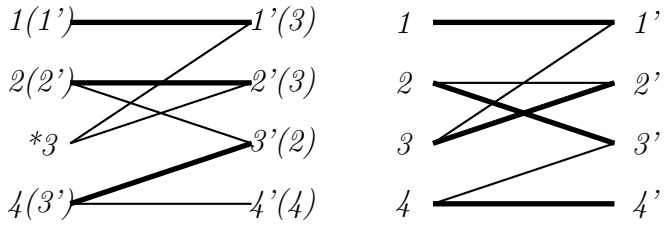
Ejemplo 12. *(Continuación del Ejemplo 3) Ya habíamos obtenido anteriormente la matriz*

	$\times$			
$\boxed{0}(M)$	*3*	*2*	*2*	$\times$
2(M)	$\boxed{0}(M)$	$\emptyset(M)$	*2*(M)	
$\emptyset(M)$	*1*	*4*	*3*	$\times$
3(M)	*2*(M)	$\boxed{0}(M)$	$\emptyset(M)$	

Entonces el mínimo valor no marcado 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, por lo 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, tenemos 4-4', con lo que etiquetamos 4' con (4). Como 4' está expuesto, hemos conseguido una cadena de aumento. La mejora 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:*

$$\begin{pmatrix} \boxed{0} & 2 & 1 & 1 \\ 3 & \not{0} & \boxed{0} & 2 \\ \not{0} & \boxed{0} & 3 & 2 \\ 4 & 2 & \not{0} & \boxed{0} \end{pmatrix},$$

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.