CAPÍTULO 4

PROGRAMACIÓN ENTERA

4.1. Introducción a la Programación Lineal Entera (PLE)

Los primeros intentos para resolver un problema de *programación lineal entera* surgieron de la metodología utilizada en la resolución de problemas de programación lineal. El primer algoritmo finito fue dado por R. Gomory y se denominó *Método de los planos de corte*.

Los avances teóricos en la resolución de *programación lineal entera* han sido importantes, si bien no se ha visto correspondido en la eficacia del cómputo. Esto es debido a los errores de redondeo cometidos en las sucesivas iteraciones y acumulados en el cómputo que realizan los ordenadores.

Un *problema de Programación Entera* es un problema de programación lineal en el cual algunas de las variables, o todas, tienen que ser números enteros no negativos. El objetivo de la *Programación Lineal Entera* es encontrar el valor de la función que

$$\text{Max (Min) } z = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$$

denominada **función objetivo**.

La función objetivo se encuentra sujeta a una serie de **restricciones**:

$$a_{11} x_1 + a_{12} x_2 + \ldots + a_{1n} x_n \; (\leq, \geq, =) \; b_1$$
$$a_{21} x_1 + a_{22} x_2 + \ldots + a_{2n} x_n \; (\leq, \geq, =) \; b_2$$
$$\ldots.$$
$$a_{m1} x_1 + a_{m2} x_2 + \ldots + a_{mn} x_n \; (\leq, \geq, =) \; b_m$$
$$x_j \geq 0 \; (j=1, 2, ..., n)$$
$$x_j \text{ entero}$$

Cuando se nos presente la resolución de un Problema de Programación Entera, lo resolvemos como un problema de Programación Lineal. Si sus soluciones son enteras, ésta es la solución para el problema de *programación lineal entera.*

En cualquier problema se verifica que la solución óptima

$$z_{op}\,(PL) \geq z_{op}\,(PLE)$$

Ésta relación se cumple siempre porque cualquier solución factible para un problema de PLE es también una solución factible para la su relajación lineal (PL).

Definición 4.1. El problema de programación lineal que se obtiene al omitir todas las restricciones enteras ó variables 0-1 se llama *relajación de programación lineal* para la *programación entera*.

Definición 4.2. **Criterio de optimalidad en un problema de PLE**: Una solución entera factible $\mathbf{x_F}$ es óptima para el problema de PLE si es solución óptima de una relajación lineal. En tal caso se cumple que $z_{op}\,(PL) = z_{op}\,(PLE)=z_F$

Un problema de programación entera en el cual solamente algunas de las variables tienen que ser números enteros, se llama un problema de *programación entera mixta*. Por ejemplo

$$\begin{aligned} \max\ & z = 3x_1 + 2x_2 \\ st\ & x_1 + x_2 \leq 5 \\ & x_1, x_2 \geq 0\ ;\ x_1 \text{ entero} \end{aligned}$$

Un problema de programación entera en el cual todas las variables toman valores 0 ó 1, se denomina problema de *programación entera* 0-1 (programación lineal binaria).

La relajación de programación lineal para la programación mixta del ejemplo anterior es:

$$\begin{aligned} \max\ & z = 3x_1 + 2x_2 \\ st\ & x_1 + x_2 \leq 5 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Por lo tanto, la relajación programación lineal es una versión menos restringida, o más relajada, de la programación entera.

Esto significa que la región factible para cualquier programación entera tiene que estar incluida en la región factible de la relajación programación lineal correspondiente.

4.2. Restricciones

A menudo, cuando tenemos que plantear un problema de programación matemática, se nos presentan enunciados en los que figura la condición:

i) "*o bien*" (una o la otra).
Esto supone una formulación del tipo

$$f(x_1, x_2,\ldots, x_n) \leq 0$$
ó
$$g(x_1, x_2,\ldots, x_n) \leq 0$$

Para implementar esta condición, se toma una variable binaria, 0-1, y un valor M suficientemente grande para asegurar el cumplimiento de las restricciones $f(x_1, x_2,\ldots, x_n) \leq M$ y $g(x_1, x_2,\ldots, x_n) \leq M$, además de las otras restricciones del problema.

$$f(x_1, x_2,\ldots, x_n) \leq My$$

$$g(x_1, x_2,\ldots, x_n) \leq M(1-y)$$

$$y = 0, 1$$

Podemos ver que si y = 0, la restricción $f(x_1, x_2,\ldots, x_n) \leq 0$, se cumple. La restricción $g(x_1, x_2,\ldots, x_n) \leq M$ puede cumplirse, pero no lo sabemos.
Para y= 1, la restricción $f(x_1, x_2,\ldots, x_n) \leq M$ puede cumplirse y $g(x_1, x_2,\ldots, x_n) \leq 0$ se cumple.

ii) "*si entonces*"(condicional)

En este tipo de restricciones se desea estar seguro de que se satisface la restricción $g(x_1, x_2,\ldots, x_n) \geq 0$ si se satisface la restricción $f(x_1, x_2,\ldots, x_n) > 0$.
Para implementar el caso anterior, tomando un valor M suficientemente grande, introducimos una variable binaria, 0-1, y cambiamos los signos en las desigualdades, dando como resultado la formulación siguiente:

$$f(x_1, x_2,\ldots, x_n) \leq M(1-y)$$ (es un artificio esta formulación)

$$- g(x_1, x_2,\ldots, x_n) \leq My$$

$$y = 0, 1$$

Para $f(x_1, x_2,\ldots, x_n) > 0$, y = 0, tenemos

$f(x_1, x_2,\ldots, x_n) \leq M$, se cumple

$-g(x_1, x_2,\ldots, x_n) \leq 0$, se cumple

Para $f(x_1, x_2,\ldots, x_n) > 0$, y = 1, tenemos

$f(x_1, x_2,\ldots, x_n) \leq 0$

$-g(x_1, x_2,\ldots, x_n) \leq M$

Si no se satisface $f(x_1, x_2,\ldots, x_n) > 0$, los x_j (1=1, 2, ..., n) no tienen restricciones y tanto $g(x_1, x_2,\ldots, x_n) < 0$ como $g(x_1, x_2,\ldots, x_n) \geq M$ son posibles.

4.3. Programación lineal general con enteros

La Programación Lineal estándar asume que las variables de decisión son continuas. Sin embargo, en muchas aplicaciones, los valores fraccionarios pueden no tener sentido, por ejemplo 9/2 trabajadores. Los problemas de programación lineal con enteros son más difíciles de resolver que los de programación lineal continua. ¿Por qué no resolver todos los problemas como problemas de programación lineal estándar y redondear las respuestas a los enteros más cercanos? Desafortunadamente, esto genera dos problemas:

- La solución redondeada puede no ser factible.
- El redondeo puede no dar una solución óptima.

Por lo tanto, el redondeo de resultados de programación lineal puede proporcionar respuestas razonables, pero, para garantizar soluciones óptimas, debemos aplicar programación lineal con enteros. Por defecto, el software LINDO de Programación Lineal asume que todas las variables son continuas. Para problemas de Programación Lineal Entera, deberemos utilizar la sentencia de entero general, GIN. GIN, seguida de un nombre de variable, restringe el valor de la variable a los enteros no negativos (0, 1, 2,…). El siguiente ejemplo ilustra el uso de la sentencia GIN.

```
Max 11X1 + 10X2
st 2X1 + X2 ≤ 12
   X1 - 3X2 ≥ 1
END
GIN X1
GIN X2
```

La salida después de siete iteraciones es:

```
        VALOR DE LA FUNCION OBJETIVO

              1)      66.00000

    VARIABLE        VALOR          COSTE REDUCIDO
          X1         6.000000         -11.000000
          X2         0.000000         -10.000000

       FILA   HOLGURA O EXCEDENTE
          2)         0.000000
          3)         5.000000
```

Si no hubiéramos especificado X1 y X2 como enteros generales en este modelo, LINDO no habría hallado la solución óptima de X1 = 6 y X2 = 0. En cambio, LINDO habría considerado a X1 y X2 como continuos y habría llegado a la solución X1 = 5.29 y X2 = 1.43.

Obsérvese, asimismo, que el simple redondeo de la solución continua a los valores enteros más próximos no da la solución óptima en este ejemplo. En general, las soluciones continuas redondeadas pueden no ser las óptimas y, en el peor de los casos, no son factibles. Sobre esta base, uno se puede imaginar que puede requerir mucho tiempo obtener la solución óptima en un modelo con muchas variables de enteros. En general, esto es así, y convendrá utilizar la característica GIN sólo cuando es absolutamente necesario.

Como último comentario, el comando GIN también acepta un argumento de valor entero en lugar de un nombre de variable. El número corresponde al número de variables que uno desea que sean enteros generales. Estas variables deben aparecer primero en la formulación. De tal modo, en este ejemplo sencillo, podríamos haber reemplazado las dos sentencias GIN por la única sentencia GIN 2.

Veamos ahora un ejemplo con restricciones del tipo "*Y-O*".

Supongamos que una panadería vende seis variedades de rosquillas. La preparación de las variedades 1, 2 y 3 implica un proceso bastante complicado, por lo que la panadería decidió que no les conviene hornear estas variedades, a menos que pueda hornear y vender por lo menos 10 docenas de las variedades 1, 2 y 3 de forma conjunta. La capacidad de la panadería limita el número total de rosquillas horneadas a 30 docenas, y el beneficio por unidad de la variedad j es c_j euros. Se supone que la panadería consigue vender todo lo que hornea.

Denotando por x_j, (j = 1, 2, … , 6) el número de docenas de la variedad j que se deben hornear.

$$\text{Max } z = \Sigma\, c_j\, X_j$$
$$\text{st } \Sigma\, x_j \leq 30$$
$$x_1 + x_2 + x_3 = 0, \text{ ó } x_1 + x_2 + x_3 \geq 10$$

Estos problemas se suelen resolver introduciendo una variable binaria, y.

$$\text{Max } z = \Sigma\, c_j\, X_j$$
$$\text{st } \Sigma\, x_j \le 30$$
$$x_1 + x_2 + x_3 - 10y \le 0$$
$$x_1 + x_2 + x_3 - 10y \ge 10$$
$$y = 0, 1; \quad X_j \ge 0 \quad (j=1, 2, ..., 6)$$

Un ejemplo de programación lineal binaria escrito para el programa LINDO:

```
Max 11X1 + 9X2 + 8X3 + 15X4
st 4X1 + 3X2 + 2X3 + 5X4 ≤ 8
END
INT X1
INT X2
INT X3
INT X4
```

Con el comando INT en LINDO se restringe la variable a 0 ó 1. A estas variables con frecuencia se las llama variables binarias. En muchas aplicaciones, las variables binarias pueden ser muy útiles en situaciones de todo o nada. En los ejemplos podemos considerar: asumir un coste fijo, construir una nueva planta o comprar un nivel mínimo de algún recurso para recibir un descuento por cantidad, etc.

La salida que proporciona LINDO da la solución óptima y el valor óptimo después de ocho iteraciones utilizando el método "Branch-and-Bound" (Ramificar y Acotar).

Observe que, en lugar de repetir INT cuatro veces, se puede emplear INT 4. Las primeras cuatro variables aparecieron en la función objetivo.

```
          VALOR DE LA FUNCION OBJETIVO

         1)      24.00000

      VARIABLE      VALOR          COSTE REDUCIDO
         X1          0.000000         -11.000000
         X2          1.000000          -9.000000
         X3          0.000000          -8.000000
         X4          1.000000         -15.000000

       FILA   HOLGURA O EXCEDENTE   PRECIOS DUALES
         2)          0.000000          0.000000
```

Nº. DE ITERACIONES = 8

4.4.- Algoritmos de Gomory para la Programación Lineal Entera

Dado un problema de programación lineal entera

$$\begin{aligned} &\text{Max } z = c^T X \\ &\text{st} \\ &\quad AX \le b \\ &\quad X \ge 0 \\ &\quad X \text{ entero} \end{aligned}$$

Estudiaremos dos métodos para su resolución: el método fraccional y el entero de Gomory.

4.4.1. Algoritmo fraccional de Gomory

Antes de iniciar su desarrollo, es necesario que todos los coeficientes que aparecen en las restricciones sean enteros, esto se consigue multiplicando cada restricción por el mínimo común múltiplo de los denominadores de los coeficientes a_{ij} y b_i correspondientes a cada restricción. Esto está motivado porque en el desarrollo del algoritmo no se hace distinción entre las variables de decisión y de holgura.

La función objetivo, escrita en función de las variables básicas y no básicas, es

$$z = \sum_{i=1}^{m} c_i y_i + \sum_{j=1}^{n} c_j w_j \qquad (4.1)$$

Las variables básicas y_i (i=1, 2, ..., m) de cualquier tabla del simplex se pueden escribir en la forma:

$$x_k = y_k + \sum_{j=1}^{n} \alpha_{k\,j} w_j \qquad \text{(tomando la fila k-ésima de la tabla del simplex)}$$

$$y_k = x_k - \sum_{j=1}^{n} \alpha_{k\,j} w_j \qquad (4.2)$$

siendo y_k variable básica y w_j variable no básica en la tabla del simplex, $\alpha_{k\,j}$, los coeficientes de la j-ésima variable no básica en la k-ésima fila y x_k, la solución para y_k.

Un número real puede expresarse como suma de su parte entera y su parte fraccionaria

$$a = [a] + f_a, \quad x_k = [x_k] + f_k, \quad \alpha_{i\,j} = [\alpha_{i\,j}] + f_{ij}$$

donde [a], parte entera, es el mayor entero menor o igual que a y f_a verifica $0 \le f_a < 1$.

Si x_k no es entero y aplicamos la descomposición anterior a la ecuación (4.2), tenemos:

$$y_k = ([x_k] + f_k) - \sum_{j=1}^{n} ([\alpha_{kj}] + f_{kj}) w_j$$

de donde

$$f_k - \sum_{j=1}^{n} f_{kj} w_j = y_k - [x_k] - \sum_{j=1}^{n} [\alpha_{kj}] w_j \Rightarrow$$

Teniendo en cuenta que todas las variables w_j e y_k deben ser enteras, el segundo miembro de la igualdad ha de ser un número entero y, por tanto, el primero también lo será.

Operando, vemos que

$$f_k - \sum_{j=1}^{n} f_{kj} w_j \le f_k$$

ya que $f_{kj} \ge 0$, $w_j \ge 0$.

Como $0 \le f_k < 1$ y $f_k - \sum_{j=1}^{n} f_{kj} w_j$ ha de ser un número entero, al tener que ser menor que uno, debe verificarse que

$$f_k - \sum_{j=1}^{n} f_{kj} w_j \le 0 \Rightarrow f_k \le \sum_{j=1}^{n} f_{kj} w_j$$

Introduciendo una variable de holgura h_k, se obtiene la restricción

$$f_k + h_k = \sum_{j=1}^{n} f_{kj} w_j \qquad \Rightarrow \qquad -f_k = -\sum_{j=1}^{n} f_{kj} w_j + h_k \Rightarrow$$

que se denomina *plano de corte*.

Los planos de corte serán restricciones que nos permiten llegar a la solución entera óptima dentro de la región factible.

Dada la última tabla del simplex de un problema de PL que proporciona el programa LINDO escriba el plano de corte asociado a la variable x1:

ROW	(BASIS)	X1	X2	SLK 2	SLK 3	SLK 4	
1	ART	0.000	0.000	0.500	0.000	0.250	7.750
2	X1	1.000	0.000	**1.5**	0.000	**-0.25**	**2.250**
3	SLK 3	0.000	0.000	-2.000	1.000	0.500	0.500
4	X2	0.000	1.000	-0.500	0.000	0.250	2.750

Solución: $-1/4=0.25=h_{x1}-1/5h_1-3/4h_3$

$9/4=2+1/4$; $f1=1/4$
$3/2=1+1/2$; $f_{a12}=1/2$
$-1/4=-1+3/4$; $f_{a14}=3/4$

4.4.2. Algoritmo entero de Gomory

El algoritmo entero de Gomory surge como alternativa al algoritmo fraccional en un intento de evitar los errores de redondeo presentados por éste. Para ello, los coeficientes de los planos de corte que se construyan deberán ser enteros. Para desarrollar este método, no es necesario resolver el problema de programación lineal, se partirá de una tabla dual factible. Las variables no factibles y_i (i=1, 2, ..., m) de cualquier tabla del simplex, en cualquier iteración, se pueden escribir en la forma:

$$x_k = y_k + \sum_{j=1}^{n} \alpha_{k\,j}\, w_j \qquad \text{(tomando la fila k-ésima de la tabla del simplex)}$$

$$y_k = x_k - \sum_{j=1}^{n} \alpha_{k\,j}\, w_j \qquad (4.3)$$

siendo y_i variable básica y w_j variable no básica en la tabla del simples, $\alpha_{k\,j}$, los coeficientes de la j-ésima variable no básica en la k-ésima fila y x_k, el valor de y_k en la solución no factible, es decir $x_k < 0$.

Dividiendo (4.3) por un número real $\lambda \neq 0$ se tiene

$$\frac{y_k}{\lambda} = \frac{x_k}{\lambda} - \sum_{j=1}^{n} \frac{\alpha_{kj}}{\lambda}\, w_j$$

Teniendo en cuenta que todo número real puede expresarse como suma de su parte entera y su parte fraccionaria

$$a = [a] + f_a, \quad x_k = [x_k] + f_k, \quad \alpha_{ij} = [\alpha_{ij}] + f_{ij}$$

la expresión anterior se expresa

$$\left(\left[\frac{1}{\lambda}\right] + f_{1/\lambda}\right) y_k = \frac{x_k}{\lambda} - \sum_{j=1}^{n}\left(\left[\frac{\alpha_{kj}}{\lambda}\right] + f_{kj}\right) w_j$$

operando

$$\left[\frac{1}{\lambda}\right] y_k + \sum_{j=1}^{n}\left[\frac{\alpha_{kj}}{\lambda}\right] w_j \le \frac{x_k}{\lambda}$$

Teniendo en cuenta que todas las variables w_j e y_k deben ser enteras, el primer miembro de la igualdad ha de ser un número entero y, por tanto,

$$\left[\frac{1}{\lambda}\right] y_k + \sum_{j=1}^{n}\left[\frac{\alpha_{kj}}{\lambda}\right] w_j \le \left[\frac{x_k}{\lambda}\right] \qquad (4.4.)$$

Multiplicando la expresión (4.3) por $\left[\frac{1}{\lambda}\right]$ tenemos

$$\left[\frac{1}{\lambda}\right] y_k + \sum_{j=1}^{n}\left[\frac{1}{\lambda}\right] \alpha_{kj} w_j \le \left[\frac{1}{\lambda}\right] x_k \qquad (4.5)$$

Restando (4.5)-(4.4), se obtiene

$$\sum_{j=1}^{n}\left(\left[\frac{1}{\lambda}\right] \alpha_{kj} - \left[\frac{\alpha_{kj}}{\lambda}\right]\right) w_j \ge \left[\frac{1}{\lambda}\right] x_k - \left[\frac{x_k}{\lambda}\right]$$

Tomando $\lambda > 1$, $\left[\frac{1}{\lambda}\right] = 0$, e introduciendo una variable de holgura, h_k, se tiene la restricción

$$\sum_{j=1}^{n}\left[\frac{\alpha_{kj}}{\lambda}\right] w_j + h_k = \left[\frac{x_k}{\lambda}\right]$$

y el plano de corte que deberemos incluir en las restricciones es

$$h_k = \sum_{j=1}^{n} \left[\frac{\alpha_{kj}}{\lambda} \right] (-w_j) + \left[\frac{x_k}{\lambda} \right]$$

Podemos observar que, tal como hemos construido el plano de corte, toda solución entera del problema verifica dicha restricción.

4.5. Algoritmo de ramificar y acotar

En la práctica, la mayoría de los problemas de programación entera se resuelven mediante el uso de la técnica de ramificar y acotar.

Los métodos de ramificar y acotar encuentran la solución óptima para un problema de programación entera mediante la enumeración eficiente de los puntos en la región factible.

Antes de explicar cómo funciona la ramificación y el acotamiento, es necesario hacer la siguiente observación: *si se resuelve un problema de programación entera mediante la relajación de un problema de programación lineal y se obtiene una solución en la cual todas las variables son números enteros, entonces la solución óptima de la relajación de programación lineal será también la solución óptima de programación entera*.

Le método de *ramificar y acotar* consiste en descomponer el problema original en una sucesión de subproblemas hasta identificar la solución óptima.

Para darse cuenta de la validez de esta observación, considérese el siguiente problema de programación entera:

$$\begin{aligned} Max \quad & z = 3x_1 + 2x_2 \\ st \quad & 2x_1 + x_2 \le 6 \end{aligned}$$

$x_1, x_2 \ge 0$; x_1, x_2 entero

La solución óptima, considerándolo un problema de programación lineal para esta programación entera, es

$$x_1 = 0, \quad x_2 = 6, \quad z = 12$$

Esta solución da valores enteros a cada variable, la observación anterior implica que

$$x_1 = 0, \quad x_2 = 6, \quad z = 12$$

también es la solución óptima para el problema de programación entera.

Obsérvese que la región factible para la programación entera es un subconjunto de todos los puntos en la región factible de la relajación por programación lineal.

Así, el valor óptimo de *z* para la programación entera no puede ser mayor que el valor óptimo de *z* para la relajación de programación lineal, es decir,

$$z_{op}\,(PL) \geq z_{op}\,(PLE)$$

Esto significa que el valor óptimo de *z* para la programación entera debe ser $z \leq 12$.

Pero el punto $x_1 = 0, \quad x_2 = 6, \quad z = 12$ es factible para la programación entera y tiene $z = 12$. Por lo tanto, $x_1 = 0, \quad x_2 = 6, \quad z = 12$ debe ser óptimo para la programación entera.

En forma resumida, los pasos del algoritmo *Ramificar y Acotar*, para un problema de maximización, son:

1.- Definimos z como la cota inferior de la solución entera óptima del problema de *programación lineal entera*. Hacemos inicialmente z = - ∞ e i = 0 (subproblema i-ésimo).

2.- *Estudio y ramificación*. Seleccione SPi (subproblema i-ésimo) como el próximo subproblema por investigarse. Resuelva el SPi y trate de estudiarlo utilizando las condiciones apropiadas.

Si el SPi se ha estudiado (solución inferior, no factible o entera), actualice la cota inferior z si se encuentra una mejor solución del *programación lineal entera*; si no es así, seleccione un nuevo subproblema 1 y repita el paso 2. Si todos los subproblemas se han investigado, deténgase; la solución óptima del problema de PLE está asociada con la última cota inferior z, en caso de que ésta exista.

Si no es así, si el SPi no está estudiado, siga con el paso 3 para efectuar la ramificación del SPi.

3.- *Ramificación*. Seleccione una de las variables x_j. cuyo valor óptimo x^*_j en la solución del SPi no satisfaga la restricción de valor entero. Elimine la región $[x^*_j] <$

$x_j < [x^*_j] + 1$ (donde [a] define al mayor entero ≤ a), creando dos subproblemas SP que correspondan a las dos siguientes restricciones mutuamente excluyentes:

$x_j < [x^*_j]$ y $x_j > [x^*_j] + 1 \Rightarrow$Volver al paso 2.

Resulta más conveniente explicar los fundamentos del algoritmo de *ramificar y acotar* mediante un ejemplo numérico.

Considere el siguiente problema de *programación lineal entera*:

$$z = 5x_1 + 4x_2$$
st
$$x_1 + x_2 \leq 5$$
$$10x_1 + 6x_2 \leq 45$$
$$x_1, x_2 \geq 0 \text{ y enteras}$$

El espacio de soluciones de PL asociado, SP0, se define por cancelación de las restricciones enteras. La solución óptima SP0 da

$x_1 = 3.75$, $x_2 = 1.25$ y $z = 23.75$

El procedimiento *ramificar y acotar* se basa en tratar sólo con el problema PL. Como la solución óptima no satisface la necesidad de valores enteros, el algoritmo de *ramificar y acotar* exige "*modificar*" el espacio de soluciones PL en forma tal que nos permita identificar, finalmente, la solución óptima del problema de *programación lineal entera*. Primero, seleccionamos una de las variables cuyo valor corriente en la solución óptima SP0 infringe el requisito de valor entero. Seleccionando $x_1 = 3.75$ arbitrariamente, observamos que la región $(3 < x_1 < 4)$ del espacio de soluciones SP0 no puede, por definición, incluir ninguna solución factible del problema de *programación lineal entera*. Entonces, podemos modificar el espacio de soluciones de PL eliminando esta región no prometedora, lo que, en realidad, es equivalente a reemplazar el espacio original SP0 por dos subproblemas de *programación lineal*, los SP1 y SP2, definidos de la manera siguiente:

1. Espacio SP1 = espacio SP0 + $(x_1 \leq 3)$

2. Espacio SP2 = espacio SP0 + $(x_1 \geq 4)$

Los dos espacios contienen los mismos puntos enteros factibles del modelo *programación lineal entera*. Esto significa que, desde el punto de vista del problema original de *programación lineal entera*, tratar con SP1 y SP2 es igual que tratar con el subproblema original SP0. La diferencia principal es que la selección de las nuevas restricciones de acotamiento ($x_1 \leq 3$ y $x_1 \geq 4$) mejorarán la

oportunidad de forzar a los puntos extremos óptimos de SP1 y SP2 hacia la satisfacción del requisito de valor entero. Además, el hecho de que las restricciones de acotamiento estén en la "vecindad inmediata" del óptimo continuo del SP0 incrementará las posibilidades de producir "buenas" soluciones enteras.

Las nuevas restricciones $x_1 \leq 3$ y $x_1 \geq 4$ son mutuamente excluyentes, SP1 y SP2 deben tratarse como dos problemas de programación lineal separados. Esto da lugar al *método de ramificar y acotar*. En efecto, ramificar significa subdividir un espacio de soluciones en subespacios mutuamente excluyentes. Las ramas asociadas se definen por las restricciones $x_1 \leq 3$ y $x_1 \geq 4$, donde x_1 se denomina variable de ramificación. La solución óptima de la *programación lineal entera* debe encontrarse en el SP1 o en el SP2. Sin embargo, en ausencia del espacio gráfico de soluciones, no se sabe dónde puede encontrarse la solución óptima, por lo que la única opción es investigar ambos problemas. Se hace esto trabajando con un problema SP1 o SP2. Escojamos arbitrariamente al SP1, asociado con $x_1 \leq 3$. En efecto, debemos resolver el siguiente problema:

$$
\begin{aligned}
&\text{Max } z = 5x1 + 4x2 \\
&\quad \text{st} \\
&\qquad x_1 + x_2 \leq 5 \\
&\qquad 10x_1 + 6x_2 \leq 45 \\
&\qquad\quad x_1 \leq 3 \\
&\qquad\quad x_1, x_2 \geq 0
\end{aligned}
$$

Como se indicó antes, el SP1 es el mismo que el SP0 con la restricción adicional de acotamiento superior, $x_1 \leq 3$. Así, podemos aplicar el método simplex de acotamiento superior para resolver el problema. Esto da la nueva solución óptima: $x_1 = 3$, $x_2 = 2$ y $z = 23$.

Como esta solución satisface el requisito de valor entero, se dice que el SP1 está agotado, lo que significa que el SP1 no puede producir ninguna solución mejor de la *programación lineal entera* y no necesita investigarse más a fondo.

Determinar una solución factible entera en una etapa temprana de los cálculos es crucial para incrementar la eficiencia del algoritmo Ramificar y Acotar. Tal solución fija una cota inferior al valor objetivo óptimo del problema *programación lineal entera*, que, a su vez, se puede usar para descartar automáticamente cualesquiera subproblemas no explorados (como el SP2) que no dan una mejor solución entera. En términos de nuestro ejemplo, el SP1 produce la cota inferior $z = 23$. Esto significa que cualquier solución entera mejorada debe tener un valor de z mayor que 23. Sin embargo, como la solución óptima del problema SP0 (original) tiene $z = 23.75$ y como todos los coeficientes de la función objetivo son enteros, se infiere que ningún subproblema que proceda del SP0 puede producir un valor de z mejor que 23. En consecuencia, sin ulterior investigación, podemos descartar al

SP2. En este caso se dice que el SP2 está agotado porque no puede dar una mejor solución entera.

Del análisis anterior vemos que un subproblema está agotado si se satisface una de las siguientes condiciones:

i) El subproblema da una solución factible entera del problema *programación lineal entera*.
ii) El subproblema no puede dar una mejor solución que la mejor cota inferior disponible (valor de z) del problema *programación lineal entera*. (Un caso especial de esta condición es que el subproblema no tendrá ninguna solución factible)

Si la función objetivo ha de minimizarse, el procedimiento es el mismo, excepto que se emplean cotas superiores. Así, el valor de la primera solución entera se vuelve una cota superior para el problema y se eliminan los SPi cuando sus valores z de primera aproximación son mayores que la cota superior actual.

4.5.1. Consideraciones para los cálculos

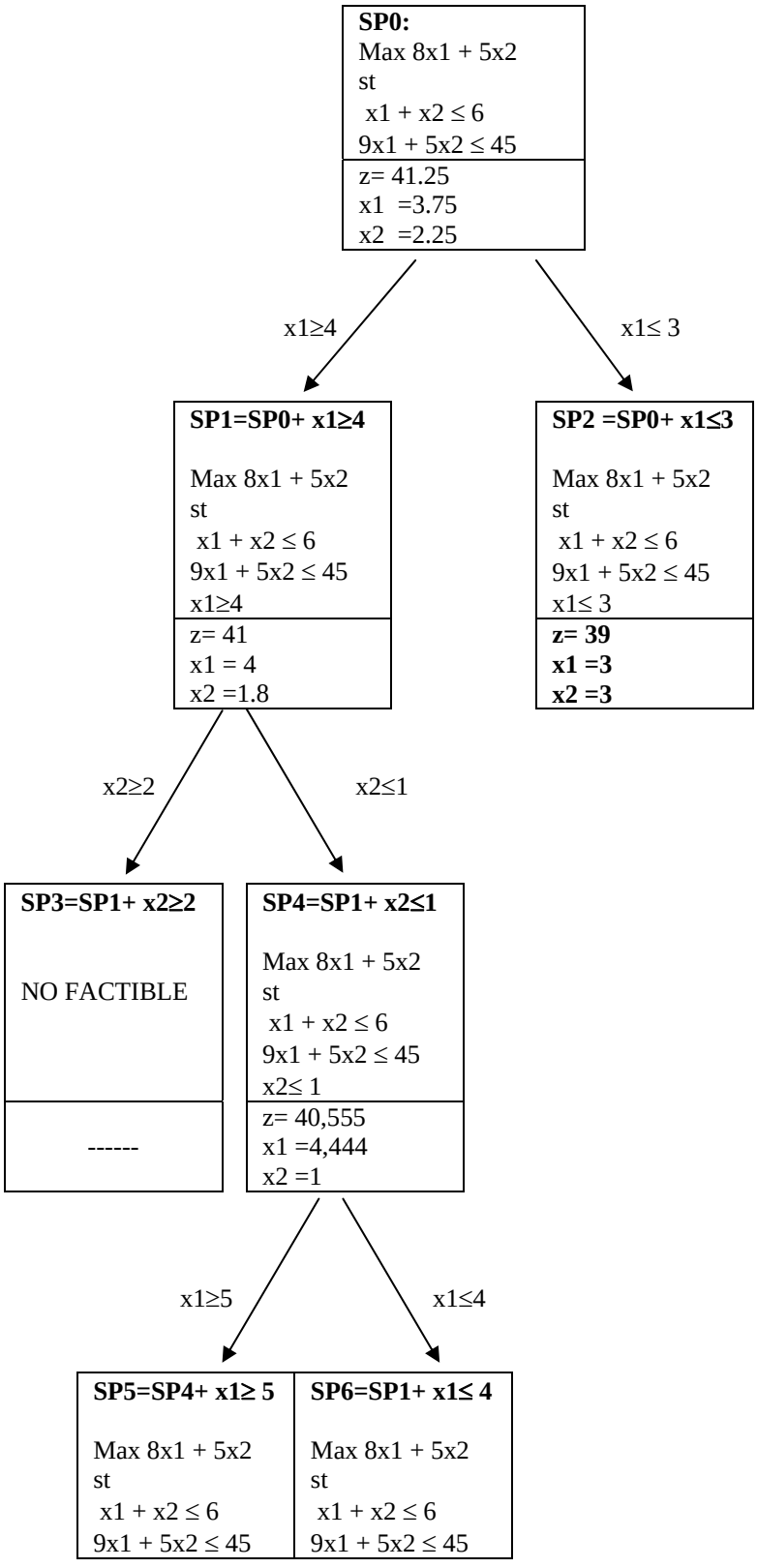
Siempre se realizan las bifurcaciones a partir de aquel programa que parece estar más cerca del valor optimo. Cuando existen varios candidatos para continuar las bifurcaciones, se selecciona aquel que tenga el mayor valor z si se va a maximizar la función objetivo, o aquel que tenga el menor valor z si se va a minimizar la función objetivo.

Las restricciones adicionales se agregan una a una. Si una primera aproximación incluye a más de una variable no entera, las nuevas restricciones se imponen a aquella variable que está más lejos de ser un entero; esto es, aquella variable cuya parte fraccionaria está más cerca de 0.5. En caso de empate, se selecciona arbitrariamente una de las variables.

Finalmente, es posible que un programa entero o un programa lineal tengan más de una solución optima.

Veamos, mediante un ejemplo, el método de *ramificar y acotar*.

$$\begin{aligned} \text{Max } z &= 8x1 + 5x2 \\ \text{st} & \\ x_1 + x_2 &\leq 6 \\ 9x_1 + 5x_2 &\leq 45 \\ x_1, x_2 &\geq 0 \end{aligned}$$


SP0:
Max 8x1 + 5x2
st
x1 + x2 ≤ 6
9x1 + 5x2 ≤ 45
z= 41.25
x1 =3.75
x2 =2.25
x1≥4
x1≤ 3
SP1=SP0+ x1≥4
Max 8x1 + 5x2
st
x1 + x2 ≤ 6
9x1 + 5x2 ≤ 45
x1≥4
z= 41
x1 = 4
x2 =1.8
SP2 =SP0+ x1≤3
Max 8x1 + 5x2
st
x1 + x2 ≤ 6
9x1 + 5x2 ≤ 45
x1≤ 3
z= 39
x1 =3
x2 =3
x2≥2
x2≤1
SP3=SP1+ x2≥2
NO FACTIBLE

SP4=SP1+ x2≤1
Max 8x1 + 5x2
st
x1 + x2 ≤ 6
9x1 + 5x2 ≤ 45
x2≤ 1
z= 40,555
x1 =4,444
x2 =1
x1≥5
x1≤4
SP5=SP4+ x1≥ 5
Max 8x1 + 5x2
st
x1 + x2 ≤ 6
9x1 + 5x2 ≤ 45
SP6=SP1+ x1≤ 4
Max 8x1 + 5x2
st
x1 + x2 ≤ 6
9x1 + 5x2 ≤ 45

x1≥ 5	x1≤ 4
z^*= 40	**x= 37**
x1 =5	**x1 =4**
x2 =0	**x2 =1**

dejando a un lado el requerimiento de que las variables sean enteras, se obtiene x_1= 3.75, x_2 =2.25, con z = 41.25, como la solución al problema de programación lineal asociado. Dado que x_1 está más alejada de un valor entero que x_2, la empleamos para generar los subproblemas SP1 y SP2: $x_1 \geq 4$ y $x_1 \leq 3$. Seguiremos ramificando los subproblamas que nos han dado soluciones no enteras hasta agotar las posibles ramificaciones. Todos aquellos subproblemas que nos den soluciones enteras entrarán a formar parte del conjunto de soluciones candidatas a la solución óptima, en nuestro caso las soluciones candidatas nos las ofrecen los subproblemas: SP2 (z=39, x1=3, x2=3), SP5 (z=40, x1=5, x2=0) y SP6 (z=37, x1=4, x2=1); de todas ellas tomaremos la que mayor valor tenga en la función objetivo SP5 (z=40, x1=5, x2=0).

Resuelto el ejercicio anterior mediante LINDO, tenemos:

```
Max 8x1 + 5x2
st
 x1 + x2 <= 6
9x1 + 5x2 <= 45

END
GIN 2

LP OPTIMUM FOUND AT STEP      2
 OBJECTIVE VALUE =   41.2500000

 SET      X1 TO >=    4 AT    1, BND=   41.00    TWIN=  39.00       7
 SET      X2 TO <=    1 AT    2, BND=   40.56    TWIN=-0.1000E+31   9 (no factible)
 SET      X1 TO >=    5 AT    3, BND=   40.00    TWIN=  37.00      12

 NEW INTEGER SOLUTION OF    40.0000000     AT BRANCH      3 PIVOT      12
 BOUND ON OPTIMUM:  40.00000
 DELETE       X1 AT LEVEL     3
 DELETE       X2 AT LEVEL     2
 DELETE       X1 AT LEVEL     1
 ENUMERATION COMPLETE. BRANCHES=     3 PIVOTS=      12

 LAST INTEGER SOLUTION IS THE BEST FOUND
 RE-INSTALLING BEST SOLUTION...

      OBJECTIVE FUNCTION VALUE

      1)      40.00000
```

```
 VARIABLE       VALUE        REDUCED COST
     X1          5.000000      -8.000000
     X2          0.000000      -5.000000

    ROW   SLACK OR SURPLUS     DUAL PRICES
     2)       1.000000         0.000000
     3)       0.000000         0.000000

 NO. ITERATIONS=      12
 BRANCHES=    3 DETERM.=  1.000E    0
```

Las expresiones:
SET X1 TO >= 4 AT 1, BND= 41.00 TWIN= 39.00 7 significa que la mejor solución para $x1 \geq 4$ es 41 y de 39 para $x1 \leq 3$.
NEW INTEGER SOLUTION OF 40.00 AT BRANCH 3 PIVOT 12, nos indica que encontró la solución optima en la tercera ramificación (SET X1 TO >= 5 AT 3, BND= 40.00 TWIN= 37.00 12).

Veamos otro ejemplo:

$$\text{Max } z = 3x_1 + 4x_2$$
$$\text{st } 2x_1 + x_2 <= 6$$
$$2x_1 + 3x_2 <= 9$$
$$x_1, x_2 \text{ no negativas y enteras}$$

```
 LP OPTIMUM FOUND AT STEP      2

  OBJECTIVE FUNCTION VALUE

  1)     12.75000

  VARIABLE       VALUE        REDUCED COST
  X1         2.250000         0.000000
  X2         1.500000         0.000000

  ROW   SLACK OR SURPLUS     DUAL PRICES
  2)         0.000000         0.250000
  3)         0.000000         1.250000
```

Planteando ahora los subproblemas, por el método de ramificar y acotar, tenemos:

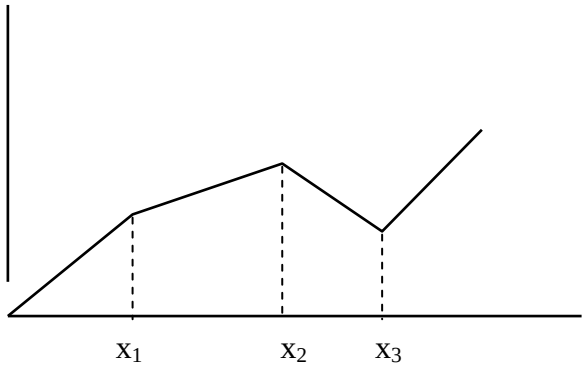
SP1	SP2
Max z = $3x_1 + 4x_2$ st $2x_1 + x_2 <= 6$ $2x_1 + 3x_2 <= 9$ $x_2 < = 1$ x_1 y x_2 no negativas y enteras	Max z = $3x_1 + 4x_2$ st $2x_1 + x_2 \leq 6$ $2x_1 + 3x_2 < 9$ $x_2 => 2$ x_1 y x_2 no negativas y enteras
LP OPTIMUM FOUND AT STEP 2 OBJECTIVE FUNCTION VALUE 1) 11.50000 VARIABLE VALUE REDUCED COST X1 2.500000 0.000000 X2 1.000000 0.000000 ROW SLACK OR SURPLUS DUAL PRICES 2) 0.000000 1.500000 3) 1.000000 0.000000 4) 0.000000 2.500000 NO. ITERATIONS= 2	LP OPTIMUM FOUND AT STEP 1 OBJECTIVE FUNCTION VALUE 1) 12.50000 VARIABLE VALUE REDUCED COST X1 1.500000 0.000000 X2 2.000000 0.000000 ROW SLACK OR SURPLUS DUAL PRICES 2) 1.000000 0.000000 3) 0.000000 1.500000 4) 0.000000 -0.500000 NO. ITERATIONS= 1

SP3	SP4
Max z = $3x_1 + 4x_2$ st $2x_1 + x_2 \leq 6$ $2x_1 + 3x_2 < 9$ $x_2 < = 1$ $x_1 < = 2$ x_1 y x_2 no negativas y enteras	Max z = $3x_1 + 4x_2$ st $2x_1 + x_2 < 6$ $2x_1 + 3x_2 < 9$ $x_2 =>2$ $x_1 <= 2$ x_1 y x_2 no negativas y enteras
LP OPTIMUM FOUND AT STEP 1 OBJECTIVE FUNCTION VALUE 1) 10.00000 VARIABLE VALUE REDUCED COST X1 2.000000 0.000000 X2 1.000000 0.000000 ROW SLACK OR SURPLUS DUAL PRICES 2) 1.000000 0.000000 3) 2.000000 0.000000 4) 0.000000 4.000000 5) 0.000000 3.000000 NO. ITERATIONS= 1	LP OPTIMUM FOUND AT STEP 1 OBJECTIVE FUNCTION VALUE 1) 12.50000 VARIABLE VALUE REDUCED COST X1 1.500000 0.000000 X2 2.000000 0.000000 ROW SLACK OR SURPLUS DUAL PRICES 2) 1.000000 0.000000 3) 0.000000 1.500000 4) 0.000000 -0.500000 5) 0.500000 0.000000 NO. ITERATIONS= 1

SP5	SP6
Max z = $3x_1 + 4x_2$ st $2x_1 + x_2 < 6$ $2x_1 + 3x_2 < 9$ $x_2 < = 1$ $x_1 >= 3$ x_1 y x_2 no negativas y enteras	Max z = $3x_1 + 4x_2$ st $2x_1 + x_2 < 6$ $2x_1 + 3x_2 < 9$ $x_2 < = 2$ $x_1 >= 3$ x_1 y x_2 no negativas y enteras
OBJECTIVE FUNCTION VALUE 1) 9.000000 VARIABLE VALUE REDUCED COST X1 3.000000 0.000000 X2 0.000000 0.000000 ROW SLACK OR SURPLUS DUAL PRICES 2) 0.000000 4.000000 3) 3.000000 0.000000 4) 1.000000 0.000000 5) 0.000000 -5.000000 NO. ITERATIONS= 3	OBJECTIVE FUNCTION VALUE 1) 12.50000 VARIABLE VALUE REDUCED COST X1 1.500000 0.000000 X2 2.000000 0.000000 ROW SLACK OR SURPLUS DUAL PRICES 2) 1.000000 0.000000 3) 0.000000 0.500000 4) 0.000000 -1.500000 5) -1.500000 -1.000000 NO. ITERATIONS= 2

4.6. Programación lineal entera aplicada a la función lineal por trozos

En este apartado veremos cómo utilizar la programación entera binaria para modelar problemas de optimización en los que intervienen funciones lineales por trozos.



Supongamos que una *función lineal a trozos*, f(x), tiene *n* puntos de ruptura x_1, x_2,…, x_n. Para algún k (k=1, 2, ..., n-1), $x_k \leq x \leq x_{k+1}$, podemos formar una combinación lineal

$$x = \alpha_k x_k + (1- \alpha_k) x_{k+1}$$

$$f(x) = \alpha_k f(x_k)+ (1- \alpha_k) f(x_{k+1})$$

$$f(x) = \sum_{i=1}^{n} \alpha_i f(x_i) = z$$

Por tanto, si tenemos f(x) con *n* puntos de ruptura podemos plantear el problema en el campo de la *programación lineal entera* de la siguiente forma:

$$x = \sum_{i=1}^{n} \alpha_i x_i \Rightarrow f(x) = \sum_{i=1}^{n} \alpha_i f(x_i)$$

$$st \sum_{i=1}^{n} \alpha_i = 1$$

Introducimos una variable binaria 0-1, $y_i = (0, 1)$, tal que

$$\sum_{i=1}^{n-1} y_i = 1$$

$$\alpha_1 \leq y_1$$

$$\alpha_2 \leq y_1 + y_2$$

$$\alpha_3 \leq y_2 + y_3$$

$$\ldots$$

$$\alpha_{n-1} \leq y_{n-2} + y_{n-1}$$

$$\alpha_n \leq y_{n-1}$$

$$\sum_{i=1}^{n} \alpha_i \leq 2 \sum_{i=1}^{n-1} y_i$$

4.7. Problemas importantes que hacen uso de la programación lineal

Dentro de este apartado estudiaremos problemas de programación lineal muy conocidos en este campo:

i) Problema de Transporte
ii) Problema de Asignación
iii) Problema de Transbordo

4.7.1. Formulación del Modelo de Transporte

La programación lineal es un campo tan amplio que se extiende a tipos de problemas para los cuales existen métodos de solución especiales. Uno de estos se conoce como *problema de transporte*. El método simplex de programación lineal puede servir para resolver estos problemas. Pero se han desarrollado métodos más sencillos que aprovechan ciertas características de los problemas. Entonces, el *método del transporte* son sólo técnicas especiales para resolver ciertos tipos de problemas de programación lineal.

El transporte desempeña un papel importante en la economía y en las decisiones administrativas. Con frecuencia la disponibilidad de transporte económico es crítica para la supervivencia de una empresa.

¿Qué significa problema de transporte? Supóngase que un fabricante tiene tres plantas que producen el mismo producto. Estas plantas, a su vez, mandan el producto a cuatro almacenes. Cada planta puede mandar productos a todos los almacenes, pero el coste de transporte varía con las diferentes combinaciones. El problema es determinar la cantidad que cada planta debe mandar a cada almacén con el fin de minimizar el coste total de transporte.

La manera más fácil de reconocer un problema de transporte es por su naturaleza o estructura "*de-hacia*": de un origen hacia un destino, de una fuente hacia un usuario, del presente hacia el futuro, de aquí hacia allá. Al afrontar este tipo de problemas, la intuición dice que *debe* haber una manera de obtener una solución. Se conocen las fuentes y los destinos, las capacidades y demandas y los costes de cada trayectoria. Debe haber una combinación óptima que minimice el coste (o maximice la ganancia). La dificultad estriba en el gran número de combinaciones posibles.

Puede formularse un problema de transporte como un problema de programación lineal y aplicarse el método simplex. Si lo hiciéramos, encontraríamos que los problemas de transporte tienen características matemáticas únicas.

Consideremos un conjunto de lugares que llamaremos orígenes y otro que llamaremos destinos.

El problema de transporte, con *m* orígenes y *n* destinos, se puede dar en una tabla:

	Destino (b_j)			
Origen (a_i)	b_1	b_2	….	b_n
a_1	c_{11} /x_{11}	c_{12} /x_{12}	…	c_{1n} /x_{1n}
a_2	c_{21} /x_{21}	c_{22} /x_{22}	…	c_{2n} /x_{2n}
.	.	.	…	.
.	.	.	…	.
a_m	c_{m1} /x_{m1}	c_{m2} /x_{m2}		c_{mn} /x_{mn}

Los pasos a seguir para formular un problema de trasporte son:

1. Definición de variables:

x_{ij} = "número de unidades que deseamos transportar del origen i-ésimo (oferta) al destino j-ésimo (demanda)"

c_{ij} = "coste de enviar una unidad del origen i-ésimo (oferta) al destino j-ésimo (demanda) "

2. Formulación del problema de transporte:

$$\text{Min } z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}\, x_{ij}$$

st

$$\text{oferta i} : \sum_{j=1}^{n} x_{ij} \le a_i$$

$$\text{demanda j} : \sum_{i=1}^{m} x_{ij} \ge b_j$$

$$x_{ij} \ge 0 \qquad \forall\ i, j\ (i = 1, 2, ..., m;\ j = 1, 2, ..., n)$$

Cada modelo tiene tantas restricciones de oferta como el número de orígenes (m) que existan y tantas restricciones de demanda como el número de destinos (n) que existan.

Las restricciones de oferta garantizan que no se transportará más de la cantidad disponible en los orígenes y las restricciones de demanda garantizan que las cantidades demandas serán satisfechas.

Siendo m el número de restricciones de oferta y n el número de restricciones de demanda, en un Modelo de Transporte existirá siempre, m x n variables en totaly existirá siempre m+n -1 variables básicas y (mxn)- (m+n-1) no básicas.

Dentro de este planteamiento podemos encontrar tres situaciones:

1.- *Problema Balanceado(ofertas igual a las demandas)*: $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j = A$

2.- *Las ofertas superan a las demandas*: $\sum_{i=1}^{m} a_i \geq \sum_{j=1}^{n} b_j$

3.- *Las demandas superan a las ofertas*: $\sum_{i=1}^{m} a_i \leq \sum_{j=1}^{n} b_j$

Pasemos a estudiar cada uno de los casos expuestos:

1.- El primer caso es el estándar (*balanceado*) al que tenderemos siempre.

2.- En el caso 2 *las ofertas superan a las demandas*: $\sum_{i=1}^{m} a_i \geq \sum_{j=1}^{n} b_j$

Para conseguir que el problema se convierta en un problema balanceado, crearemos un destino ficticio.

Para el origen i-ésimo la oferta es: $a_i = \sum_{j=1}^{n} x_{ij} + x_{i0}$

La oferta total es $A = \sum_{i=1}^{m} a_i = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} + \sum_{i=1}^{m} x_{i0}$

Operando en la expresión anterior, tenemos

$$\sum_{i=1}^{m} x_{i0} = b_0 = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j$$

donde x_{i0} es el número de unidades que deseamos transportar desde el origen i-ésimo (oferta) al destino ficticio, b_0.

3.-En el caso 3 (*las demandas superan a las ofertas*): $\sum_{i=1}^{m} a_i \leq \sum_{j=1}^{n} b_j$, en este caso generamos un a_0 (origen ficticio).

Para el destino j-ésimo la demanda es: $b_j = \sum_{i=1}^{m} x_{ij} + x_{0j}$

donde x_{0j} es el número de unidades que deseamos transportar desde el origen ficticio al destino j-ésimo.

$$\sum_{j=1}^{n} b_j = \sum_{j=1}^{n} \sum_{i=1}^{m} x_{i\,j} + \sum_{j=1}^{n} x_{0j}$$

El origen ficticio oferta la cantidad

$$\sum_{j=1}^{n} x_{0j} = a_0 = \sum_{j=1}^{n} b_j - \sum_{i=1}^{m} a_i$$

Teorema 6.1. Todo problema de transporte admite una solución factible acotada.

Demostración

Consideraremos el caso balanceado, ya que todos se reducen a éste.

Definamos la variable: $x_{i\,j} = \frac{a_i b_j}{A}$

Oferta i-ésima : $\sum_{j=1}^{n} \frac{a_i b_j}{A} = \frac{a_i}{A} \sum_{j=1}^{n} b_j = \frac{a_i}{A} A = a_i$

Demanda j-ésima : $\sum_{i=1}^{m} \frac{a_i b_j}{A} = \frac{b_j}{A} \sum_{i=1}^{m} a_i = \frac{b_j}{A} A = b_j$

Por tanto, admite una solución factible que está acotada, $x_{i\,j} \le \min \{ a_i , b_j \}$.

4.7.2. Métodos para encontrar soluciones factibles

Al iniciar, todos los renglones de los orígenes y las columnas de destinos de la tabla simplex de transporte se toman en cuenta para proporcionar una variable básica (asignación).

1. Se selecciona la siguiente variable básica (asignación) entre los renglones y columnas en que todavía se puede hacer una asignación de acuerdo a algún criterio.
2. Se hace una asignación lo suficientemente grande como para que use el resto de los recursos en ese renglón o la demanda restante en esa columna (cualquiera que sea la cantidad más pequeña).
3. Se elimina ese renglón o columna (la que tenía la cantidad más pequeña en los recursos o demanda restantes) para las nuevas asignaciones. (*Si la fila y la columna tienen la misma cantidad de recursos y demanda restante, entonces arbitrariamente se elimina el renglón. La columna se usará después para proporcionar una variable básica degenerada, es decir, una asignación con cero unidades.*)
4. Si sólo queda un renglón o una columna dentro de las posibilidades, entonces el procedimiento termina eligiendo como básicas cada una de las variables *restantes* (es decir, aquellas variables que no se han elegido ni se han eliminado al quitar su renglón o columna) asociadas con ese renglón o columna que tiene la única asignación posible. De otra manera se regresa al paso 1.

Con el uso del ordenador, todos estos problemas se resuelven mediante el software apropiado. Por ello mencionaremos, de forma resumida, dos métodos para resolver el problema del transporte: el *método de la esquina noroeste y el método de aproximación de Vogel.*

El *Método de la Esquina Noroeste:* la primera elección es x_{11} (es decir, se comienza en la esquina noroeste de la tabla simplex de transporte). De ahí en adelante, si x_{ij} fue la última variable básica seleccionada, la siguiente elección es $x_{i,j+1}$ (es decir, se mueve una columna a la *derecha*) si quedan recursos en el origen *i*. De otra manera, se elige $x_{i+1,j}$ (es decir, se mueve un renglón hacia *abajo*).

Lo primero que debemos hacer al resolver cualquier problema de transporte es comprobar que esté balanceado; si no lo estuviera, agregamos un origen o un destino artificial según sea el caso para conseguir que el problema quede balanceado y podamos comenzar a resolverlo.

El *Método de Aproximación de Vogel*: para cada renglón y columna que queda bajo consideración, se calcula su **diferencia**, que se define como la *diferencia aritmética entre el coste unitario más pequeño (c_{ij}) y el que le sigue, de los que quedan en ese renglón o columna.* (Si se tiene un empate para el coste más pequeño de los restantes de un renglón o columna, entonces la *diferencia* es 0). En el renglón o columna que tiene la *mayor diferencia* se elige la variable que tiene el *menor coste unitario que queda*. (Los empates para la mayor de estas diferencias se pueden romper de manera arbitraria).

Iniciamos el método calculando las primeras diferencias para cada renglón y columna.

El método de aproximación de Vogel ha sido el más popular durante muchos años, en parte porque es relativamente fácil hacerlo a mano. Este criterio toma en cuenta los costes unitarios en forma efectiva, ya que la *diferencia* representa el mínimo coste adicional en que se incurre por no hacer una asignación en la celda que tiene el menor coste en esa columna o renglón.

Podemos decir que el método de aproximación de Vogel proporciona una mejor *solución inicial* que el criterio de la esquina noroeste, en otras palabras es más cualitativo.

La prueba de optimalidad estándar del método simplex para el problema de transporte se puede reducir de la siguiente manera:

Una solución básica factible es óptima si y sólo si $c_{ij}-u_i-v_j \geq 0$ para toda (i,j) tal que x_{ij} es no básica.

Así, lo único que hay que hacer para realizar esta prueba es obtener los valores de u_i y v_j para la solución básica factible actual y después calcular los valores $c_{ij}-u_i-v_j$ según se describe enseguida. Como el valor de $c_{ij}-u_i-v_j$ debe ser cero si x_{ij} es una variable básica, u_i y v_j satisfacen el conjunto de ecuaciones:

$$c_{ij} = u_i + v_j \qquad \text{para cada (i,j) tal que } x_{ij} \text{ es básica.}$$

Existen m+n–1 variables básicas y, por tanto, hay m+n–1 ecuaciones de este tipo. Como el número de incógnitas (las u_i y v_j) es m+n, se puede asignar un valor arbitrario a cualquiera de estas variables sin violar las ecuaciones. La elección de esta variable y su valor no afecta al valor de ningún $c_{ij}-u_i-v_j$, aun cuando x_{ij} sea no básica, por lo que la única diferencia (menor) estriba en la facilidad para resolver estas ecuaciones. Una elección conveniente para lograr esto es seleccionar la u_i que tiene el *mayor número de asignaciones en su renglón* (los empates se rompen de manera arbitraria) y asignarle un valor de cero. Gracias a la sencilla estructura de estas ecuaciones, resulta muy fácil obtener algebraicamente los valores del resto de las variables.

Con el uso del ordenador, todos estos problemas se resuelven mediante el software apropiado.

4.7.3. Problema de Asignación

El *problema de asignación* es un caso particular del problema del transporte. En general, es un problema de transporte balanceado. Trabajaremos bajo distintos supuestos acerca del número de sujetos y puestos de trabajo.

Definimos la variable $x_{ij} = \begin{cases} 1 & \textit{si a la persona } i \textit{ se le asigna el puesto } j \\ 0 & \textit{en caso contrario} \end{cases}$

Supuesto 1: m = n.

$$\text{Min } z = \sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij}\, x_{ij}$$

s.t.

$$\text{sujeto i-ésimo : } \sum_{j=1}^{m} x_{ij} = 1$$

$$\text{puesto j-ésimo : } \sum_{i=1}^{m} x_{i\,j} = 1$$

$$x_{ij} \geq 0 \quad \forall\ i\ j$$

Supuesto 2: Asignación múltiple

Sea a_i = "número de tareas, como máximo, que puede desarrollar el i-ésimo individuo"

$$\text{Min } z = \sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij}\, x_{ij}$$

s.t.

$$\text{sujeto i : } \sum_{j=1}^{m} x_{ij} \leq a_i$$

$$\text{puesto j : } \sum_{i=1}^{m} x_{i\,j} = 1$$

$$x_{ij} \geq 0 \ , \forall\ i\ j$$

Supuesto3. m≠n. Analicemos el caso: m > n

$$\text{Min } z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}\, x_{ij}$$

s.t.

$$\text{sujeto i : } \sum_{j=1}^{n} x_{ij} \leq 1, \ (i=1,2,..., m)$$

$$\text{puesto j : } \sum_{i=1}^{m} x_{i\,j} = 1, \ (j=1,2,..., n)$$

$$x_{ij} \geq 0 \ , \forall\ i\ j$$

Para m < n

$$\text{Min } z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}\, x_{ij}$$

s.t.

$$\text{sujeto i} : \sum_{j=1}^{n} x_{ij} = 1,\ (i=1,2,\ldots, m)$$

$$\text{puesto j} : \sum_{i=1}^{m} x_{ij} \le 1,\ (j=1,2,\ldots, n)$$

$$x_{ij} \ge 0\ ,\ \forall\ i\ j$$

Veamos unos ejemplos que ilustren lo anterior.

Un buffet de abogados ha aceptado 5 nuevos casos, cada uno de los cuales puede ser llevado adecuadamente por cualquiera de los cinco asociados más recientes. Debido a la diferencia de experiencia y práctica, los abogados emplearon distintos tiempos en los casos. Uno de los asociados más experimentados ha estimado las necesidades de tiempo en horas como sigue:

Abogados	Caso 1	Caso 2	Caso 3	Caso 4	Caso 5
1	145	122	130	95	115
2	80	63	85	48	78
3	121	107	93	69	85
4	118	83	116	80	105
5	97	75	120	80	111

Determine la forma óptima de asignar los casos a los abogados de manera que cada uno de ellos se dedique a un caso diferente y que el tiempo total de los empleados sea mínimo.

Definimos la variable $x_{ij} = \begin{cases} 1 & \textit{si al abogado } i \textit{ se le asigna el caso } j \\ 0 & \textit{en caso contrario} \end{cases}$

$$\text{Min } z = 145x_{11} + 122x_{12} + 130x_{13} + 95x_{14} + 130x_{15} + 80x_{21} + 63x_{22} + 85x_{23} + 48x_{24} + 78x_{25} + 121x_{31} + 107x_{32} + 93x_{33} + 69x_{34} + 85x_{35} + 118x_{41} + 83x_{42} + 116x_{43} + 80x_{44} + 105x_{45} + 97x_{51} + 75x_{52} + 120x_{53} + 80x_{54} + 111x_{55}$$

st

$$x_{11} + x_{12} + x_{13} + x_{14} + x_{15} = 1$$
$$x_{21} + x_{22} + x_{23} + x_{24} + x_{25} = 1$$
$$x_{31} + x_{32} + x_{33} + x_{34} + x_{35} = 1$$
$$x_{41} + x_{42} + x_{43} + x_{44} + x_{45} = 1$$
$$x_{51} + x_{52} + x_{53} + x_{54} + x_{55} = 1$$

$$x_{11} + x_{21} + x_{31} + x_{41} + x_{51} = 1$$
$$x_{12} + x_{22} + x_{32} + x_{42} + x_{52} = 1$$
$$x_{13} + x_{23} + x_{33} + x_{43} + x_{53} = 1$$
$$x_{14} + x_{24} + x_{34} + x_{44} + x_{54} = 1$$
$$x_{15} + x_{25} + x_{35} + x_{45} + x_{55} = 1$$

x_{ij} binarias (i, j=1, 2, ..., 5)

Llevado el planteaminto del problema a LINDO, tenemos:

```
LP OPTIMUM FOUND AT STEP     16
 OBJECTIVE VALUE =   443.000000

 FIX ALL VARS.(   16)  WITH RC >   2.00000

 NEW INTEGER SOLUTION OF  443.000000  AT BRANCH   0 PIVOT   16
 BOUND ON OPTIMUM:  443.0000
LAST INTEGER SOLUTION IS THE BEST FOUND
 RE-INSTALLING BEST SOLUTION...

       OBJECTIVE FUNCTION VALUE

       1)      443.0000

  VARIABLE        VALUE          REDUCED COST
       X11         0.000000        145.000000
       X12         0.000000        122.000000
       X13         1.000000        130.000000
       X14         0.000000         95.000000
       X15         0.000000        130.000000
       X21         0.000000         80.000000
       X22         0.000000         63.000000
       X23         0.000000         85.000000
       X24         1.000000         48.000000
       X25         0.000000         78.000000
       X31         0.000000        121.000000
       X32         0.000000        107.000000
       X33         0.000000         93.000000
       X34         0.000000         69.000000
       X35         1.000000         85.000000
       X41         0.000000        118.000000
       X42         1.000000         83.000000
       X43         0.000000        116.000000
       X44         0.000000         80.000000
       X45         0.000000        105.000000
       X51         1.000000         97.000000
       X52         0.000000         75.000000
       X53         0.000000        120.000000
       X54         0.000000         80.000000
       X55         0.000000        111.000000
```

En negrita se ha resaltado la solución de la asignación.

Otro ejemplo. Una tienda de autoservicio que funciona las 24 horas tiene los siguientes requerimientos mínimos para los cajeros:

Periodo	*1*	*2*	*3*	*4*	*5*	*6*
Hora del día	3-7	7-11	11-15	15-19	19-23	23-3
Nº Mínimo	7	20	14	20	10	5

El período 1 sigue inmediatamente a continuación del período 6. Un cajero trabaja 8 horas consecutivas, empezando al inicio desde los 6 períodos. Determínese qué grupo diario de empleados satisface las necesidades con el mínimo de personal.

Definimos la variable x_i = número de empleados que comienzan su labor al inicio del período i-ésimo (i=1, 2, ..., 5)

$$\text{Min } z = x_1 + x_2 + x_3 + x_4 + x_5 + x_6$$

st

$$x_1 + x_2 \geq 20$$
$$x_2 + x_3 \geq 14$$
$$x_3 + x_4 \geq 20$$
$$x_4 + x_5 \geq 10$$
$$x_5 + x_6 \geq 5$$
$$x_1 + x_6 \geq 7$$
$$x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$$

```
        OBJECTIVE FUNCTION VALUE

        1)      45.00000

  VARIABLE         VALUE          REDUCED COST
        X1          6.000000          0.000000
        X2         14.000000          0.000000
        X3          0.000000          0.000000
        X4         20.000000          0.000000
        X5          4.000000          0.000000
        X6          1.000000          0.000000

       ROW   SLACK OR SURPLUS     DUAL PRICES
         2)          0.000000        -1.000000
         3)          0.000000         0.000000
         4)          0.000000        -1.000000
         5)         14.000000         0.000000
         6)          0.000000        -1.000000
         7)          0.000000         0.000000
```

La solución la recogemos en la tabla siguiente:

Periodo	*1*	*2*	*3*	*4*	*5*	*6*
Hora del día	3-7	7-11	11-15	15-19	19-23	23-3
Nº Mínimo	6	14	0	20	4	1

4.7.4. Problema de Transbordo

El problema de Transbordo es un caso más general que el problema de transporte. Los *puntos de transbordo* son aquellos puntos de paso que cumplen la condición de que lo que llega a un nodo tiene que salir de él.

El planteamiento del problema se hace atendiendo a lo anteriormente expuesto.

$$\text{Min } z = \sum_{j=1}^{n}\sum_{i=1}^{m} c_{ij}\, x_{ij} \quad \text{(se suman todos los arcos)}$$

st

orígenes: $\sum x_{ij} - \sum x_{ij} \le a_i$ (arcos que salen - arcos que entran)

transbordo: $\sum x_{ij} - \sum x_{ij} = 0$ (puntos de transbordo)

destinos: $\sum x_{ij} - \sum x_{ij} \ge b_j$ (arcos que entran - arcos que salen)

4.7.5. El problema de la mochila

Existen multitud de variantes del problema de la mochila. En este caso, trataremos con una variante del problema que se denomina *Problema de la Mochila 0-1*.

Supóngase que se tiene n objetos distintos y una mochila. Cada uno de los objetos tiene asociado un peso positivo w_i y un valor positivo v_i para i=1, 2, …, n. La mochila puede llevar un peso que no sobrepase la cantidad W. Teniendo en cuenta estos datos, el objetivo del problema es llenar la mochila de tal forma que se maximice el valor de los objetos transportados en ella. Hay que tener en cuenta que la suma de los pesos de los objetos seleccionados no debe superar la capacidad máxima W de la mochila. Por lo tanto, para obtener la solución deseada, se debe decidir para cada uno de los objetos, si se introduce o no en la mochila. Cada objeto tiene asociado una variable x_i que toma el valor 1 si el objeto se mete en la mochila y 0 en caso contrario. Cualquier objeto se puede introducir en la mochila si hay espacio para él, pero lo que no se puede hacer es transportar en ella una fracción o parte de un objeto. A este tipo de problemas en los que los objetos no pueden ser fraccionados se les llama *Problemas de la Mochila 0-1 o Entera*.

Matemáticamente el problema se puede formular como se muestra a continuación:

$$\text{Max } z = \sum_{i=1}^{n} v_i x_i$$

st

$$\sum_{i=1}^{n} w_i\, x_i \le W$$

$$v_i,\ w_i > 0,\ x_i \in \{0,1\},\ i=1, 2, \ldots, n$$