Programación entera para modelos lineales

Daniel Molina Pérez
Centro de Investigaciones Hidráulicas (CIH), Instituto Superior Politécnico José Antonio Echeverría (Cujae). La Habana.
e-mail: danielmolinaperez90@gmail.com

Eric Cabrera Estupiñán
Centro de Investigaciones Hidráulicas (CIH), Instituto Superior Politécnico José Antonio Echeverría (Cujae). La Habana.
Empresa de Ingeniería y Proyectos del Petróleo (EIPP). La Habana.
e-mail: ecabrera@cipp.minbas.cu

RESUMEN
Se presenta una herramienta para resolver modelos de programación lineal entera, lo que significa encontrar el valor óptimo de la función objetivo para variables de decisión enteras. Es una función llamada *nbintprog* (non binary integer programming) creada por los autores en el asistente matemático MATLAB (MatrixLaboratory) y para comodidad del usuario presenta una forma similar a las funciones de optimización propias de MATLAB. La existencia de situaciones donde sólo tiene sentido que las variables de decisión, por su naturaleza, sean números enteros y los inconvenientes que presenta la herramienta que propone MATLAB para su solución constituyen los motivos del trabajo. *Nbintprog* tiene como base matemática una técnica de ramificación y acotación (branch and bound) que junto a estrategias de programación utilizadas presenta características favorables para la solución de dichos modelos.

Palabras clave: MATLAB, nbintprog, optimización, programación entera, programación lineal.

Integer programming for linear models

ABSTRACT
A new tool able to solve integer linear programming problems is presented. The problem arises when one has to find the optimal value of the objective function for integer decision variables. This tool is a function called *nbintprog* (non binary integer programming) created by the authors in MATLAB mathematical assistant. To the user's comfort it is presented in a similar way as any other of the typical MATLAB optimization functions. The existence of situations where it only makes sense that the decision variables are integers and the inconvenience of the existing MATLAB function for its solution became the motive of this work. The function is based on the well-known branch and bound technique which together with other programming strategies may eventually reduce the search time of the solution significantly.

Keywords: MATLAB, nbintprog, optimization, integer programming, linear programming.

recibido: Septiembre 2013 | aprobado: Octubre 2013

INTRODUCCIÓN

Un problema de optimización o programación matemática consiste en maximizar o minimizar una función real dado un dominio definido. De forma general, la optimización se basa en el descubrimiento de los "mejores valores" de alguna función objetivo (FO), existiendo una variedad de diferentes tipos de funciones objetivo y diferentes tipos de dominios (Society 2010). Al procedimiento o algoritmo matemático mediante el cual se resuelve un problema formado por una función objetivo lineal y un sistema de ecuaciones o inecuaciones lineales se le llama programación lineal.

Existen casos de modelos lineales en los que por razones físicas se necesita que el valor óptimo de la FO sea resultado de valores enteros para todas las variables de decisión, el algoritmo capaz de resolver este problema se conoce como "programación lineal entera pura" (PLEP). El hecho de conocer las variables de decisión que den solución a un modelo de programación lineal, no significa necesariamente encontrarse cerca de las variables que den solución a la PLEP del modelo. La aproximación al valor entero más cercano de las variables de decisión no siempre conduce a la solución óptima del modelo y en otros casos provoca la violación de algunas de las restricciones (Marrero 1985). Para demostrar lo expresado se presentan los siguientes ejemplos:

Ejemplo (1)

$$MAX(Z) = 4x + 3y \quad (1)$$
$$2x + y \leq 2 \quad (2)$$
$$3x + 4y \leq 6 \quad (3)$$
$$x \geq 0 \quad (4) \qquad y \geq 0 \quad (5)$$

Donde la ecuación (1) es la FO que, en este caso, se busca su máximo valor (ver figura 1).

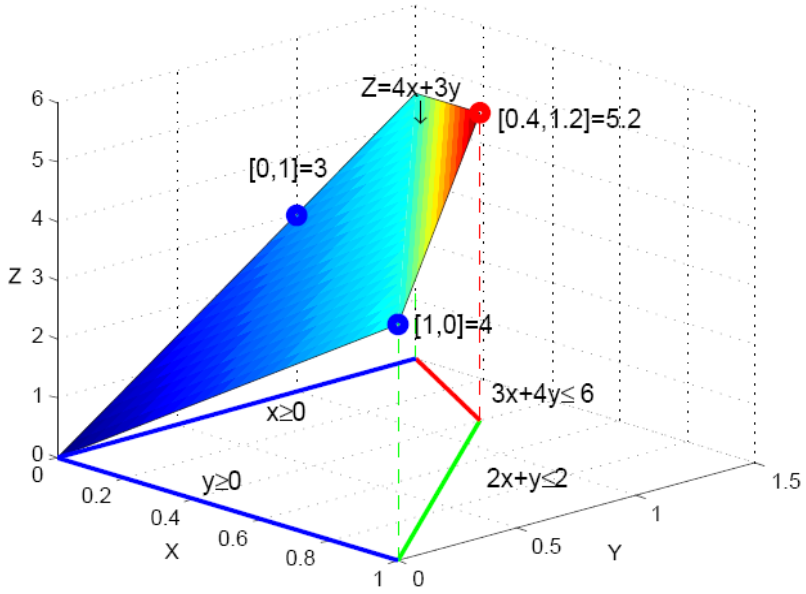

Figura 1. Representación gráfica del modelo del ejemplo 1

Las ecuaciones (2), (3), (4) y (5) son las restricciones del dominio, es decir, delimitan lo que se llama el espacio factible, que se ilustra en la figura 1.

.

Como muestra dicha figura 1 la solución óptima de la PL en este ejemplo es Z(x,y)=5,2 para [x=0,4; y=1,2]. Una aproximación de las variables a sus valores enteros más cercanos sería [x=0; y=1] resultando en Z(x,y)=3. Sin embargo, véase que el punto [x=1; y=0] tiene Z(x,y)=4 lo que prueba que dicha aproximación no garantiza la solución del modelo de PLEP .

Ejemplo (2)

$$MAX\ (Z) = 8x + 10y \quad (6)$$
$$4x + 6y \leq 24 \quad (7)$$
$$8x + 3y \leq 24 \quad (8)$$
$$x \geq 0;\ \ y \geq 0 \quad (9)$$

La solución óptima en este ejemplo es Z(x,y)=[128/3] para [x=2; y=8/3]. Una aproximación de las variables a sus valores enteros más cercanos queda [x=2; y=3] la cual se encuentra fuera del dominio del modelo incumpliendo con una de las restricciones, ver figura 2.

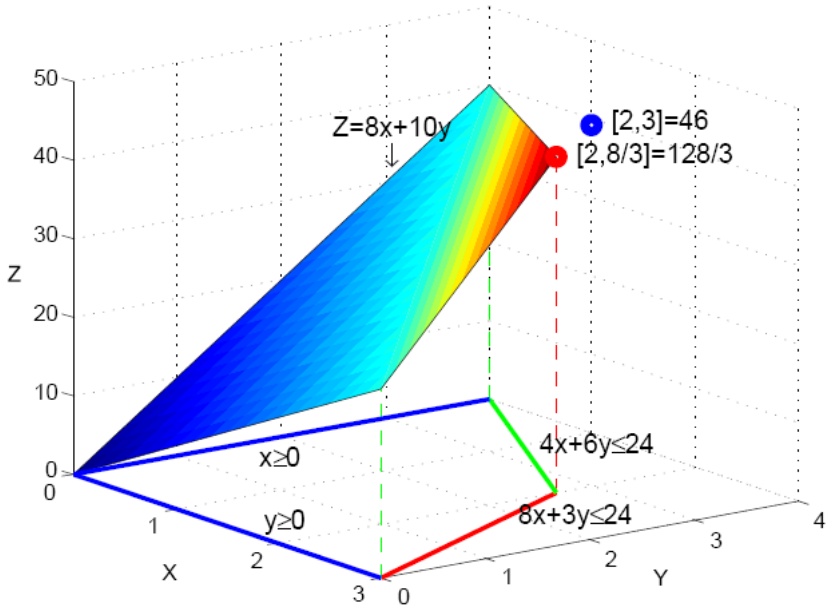

Figura 2. Representación gráfica del modelo del ejemplo 2

Por estas razones para obtener la solución óptima como resultado de variables enteras (PLEP) ha surgido la necesidad de complementar algoritmos utilizados en la PL con estrategias de PLEP. Una de las más eficaces estrategias es el método de ramificación y acotación (branch and bound). Este artículo propone una función programada en MATLAB capaz de resolver modelos de PLEP utilizando dicho método como base matemática.

Dicha herramienta, además de agilizar el proceso manual, extiende sus objetivos a ofrecer notables ventajas sobre otras estrategias como se verá más adelante.

MÉTODO DE RAMIFICACIÓN Y ACOTACIÓN

Se llamará *problema relajado* (PR) de un problema de PLEP al problema resuelto solamente mediante programación lineal (PL), es decir, sin que existan condiciones de integridad sobre las variables (Wolsey and Nemhauser 1988). El método comienza por la solución del PR del modelo inicial.

La *ramificación* consiste en dividir cada problema en dos nuevos subproblemas obtenidos mediante la imposición de restricciones excluyentes que dividen el dominio del problema original en dos partes, pero eliminando en ambas partes la solución no entera del problema original (Troncoso 2005), es decir, si xbi es una variable de decisión no entera, entonces se generan dos restricciones xi ≤ [xbi] (siendo [xbi] la parte entera por defecto de xbi) y xi ≥ [xbi]+1, que añadidas cada una por separado al problema original, da lugar a dos nuevos subproblemas. En el ejemplo 2, y=8/3=2,67. Por tanto se introducirán las siguientes restricciones: y **≤** 2; y **≥** 3, de forma que se ha eliminado una porción del dominio donde no hay soluciones enteras, figura 3.

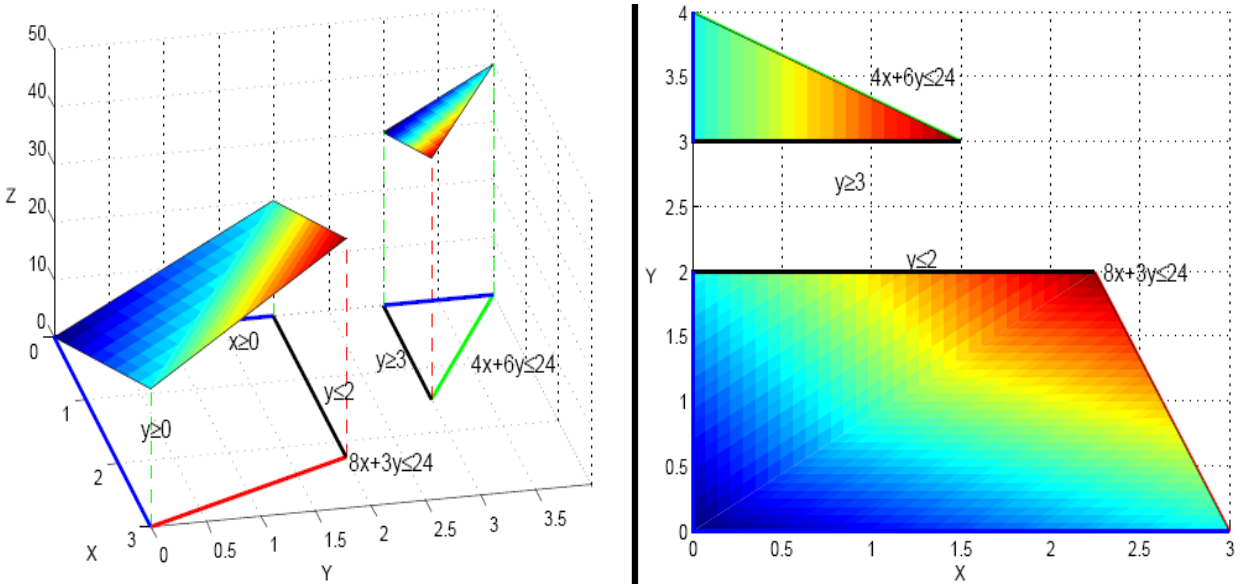

Figura 3. Representación gráfica de la ramificación del modelo del ejemplo 2

Una vez que se ramifica se resuelve cada subproblema o nodo, (como se le llamará en lo adelante) mediante PL lineal, es decir, se resuelve el PR de cada uno. Mientras que en ambos nodos exista alguna variable no entera se ramificará el nodo que tenga mejor valor de FO (VFO). El nodo no ramificado pasa a formar parte de una lista de nodos vivos. Una vez encontrado el primer nodo con todas sus variables enteras se tiene el primer VFO de pivote y se confrontará con el VFO de cada nodo vivo. Será ramificado el nodo vivo que tenga mejor VFO que el pivote ya que puede encontrarse una solución entera mejor por dicha rama, de ser así esta se convertiría en el nuevo pivote. Cuando existen varios nodos vivos mejores que el pivote existen los criterios FIFO (first input first output), LIFO (last input first output) y el criterio BIFO (best input first output) donde se ramifica primero el nodo vivo de mejor VFO (Lahoz 2009). Este último criterio será el empleado en la función que se presenta. Por otro lado, el nodo que tenga peor VFO que el pivote se cerrará por *acotación* y deja de formar parte de la lista de nodos vivos. Garantizando así no explorar todas las soluciones del modelo, fenómeno que puede hacer que la búsqueda de la

solución aumente de minutos a años ya que el número de soluciones es grande incluso en pequeños problemas como se muestra:

$$0 \le x \le 150 \; ; \quad 0 \le y \le 200$$
posibles soluciones= 151 x 201 =30 351

Ejemplo (3)
Se resuelve el siguiente modelo mediante ramificación y acotación

$$MAX\ (Z) = 4x + 5y \qquad (10)$$
$$x \le 1 \qquad (11)$$
$$3x + 4y \le 6 \qquad (12)$$
$$x \ge 0 \qquad (13) \qquad y \ge 0 \qquad (14)$$

Donde la solución del PR es Z(x,y)=[7,75] para [x=1; y=3/4], ver figura 4a.

Primera ramificación (figura 4b):

Nodo 1
MAX $(Z) = 4x + 5y$
$x \le 1$
$3x + 4y \le 6$
$x \ge 0 \; ; \quad y \ge 0$
$\boldsymbol{y \le 0}$

Solución: [x=1; y=0]; Z(x,y)=[4]

Nodo 2
MAX $(Z) = 4x + 5y$
$x \le 1$
$3x + 4y \le 6$
$x \ge 0 \; ; \quad y \ge 0$
$\boldsymbol{y \ge 1}$

Solución: [x=0,67; y=1]; Z(x,y)=[7,67]

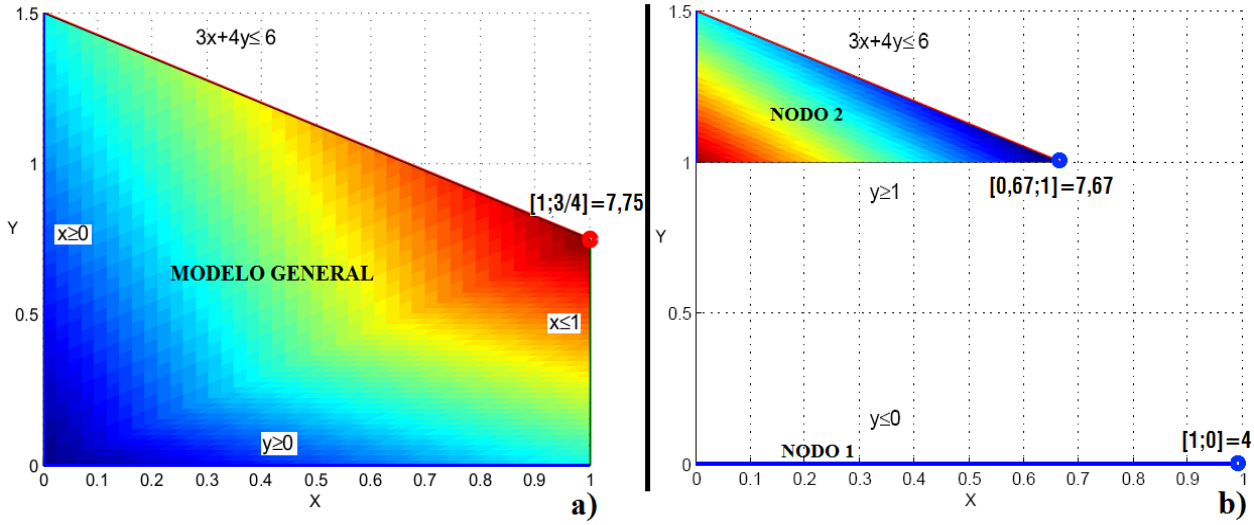

Figura 4. a) Representación gráfica del modelo del ejemplo 3 b) Primera ramificación

En el Nodo1 se encuentra el primer VFO resultante de variables enteras (VFO de pivote), pero no se puede garantizar que sea la solución ya que existe una solución no entera en el nodo 2 con mayor VFO (nodo vivo), lo que hace necesario ramificar dicho nodo.
Segunda ramificación (figura 5a):

Nodo 2-1
MAX (Z) = $4x + 5y$
$x \leq 1$
$3x + 4y \leq 6$
$x \geq 0$; $y \geq 0$
$\boldsymbol{y \geq 1}$
$\boldsymbol{x \leq 0}$

Solución: [x=0; y=1,5]; Z(x,y)=[7,5]

Nodo 2-2
MAX (Z) = $4x + 5y$
$x \leq 1$
$3x + 4y \leq 6$
$x \geq 0$; $y \geq 0$
$\boldsymbol{y \geq 1}$
$\boldsymbol{x \geq 1}$

Solución: no factible

El Nodo2-2 carece de variables que puedan satisfacer todas sus restricciones por lo que no tiene solución factible y se cierra, sin embargo el Nodo2-1 tiene un VFO mayor que el VFO de pivote por lo que se tiene que ramificar.

Tercera ramificación (figura 5b):

Nodo 2-1-1
MAX (Z) = $4x + 5y$
$x \leq 1$
$3x + 4y \leq 6$
$x \geq 0$; $y \geq 0$
$\boldsymbol{y \geq 1}$; $\boldsymbol{x \leq 0}$
$\boldsymbol{y \leq 1}$

Solución: [x=0; y=1]; Z(x,y)=[5]

Nodo 2-1-2
MAX (Z) = $4x + 5y$
$x \leq 1$
$3x + 4y \leq 6$
$x \geq 0$; $y \geq 0$
$\boldsymbol{y \geq 1}$; $\boldsymbol{x \leq 0}$
$\boldsymbol{y \geq 2}$

Solución: no factible

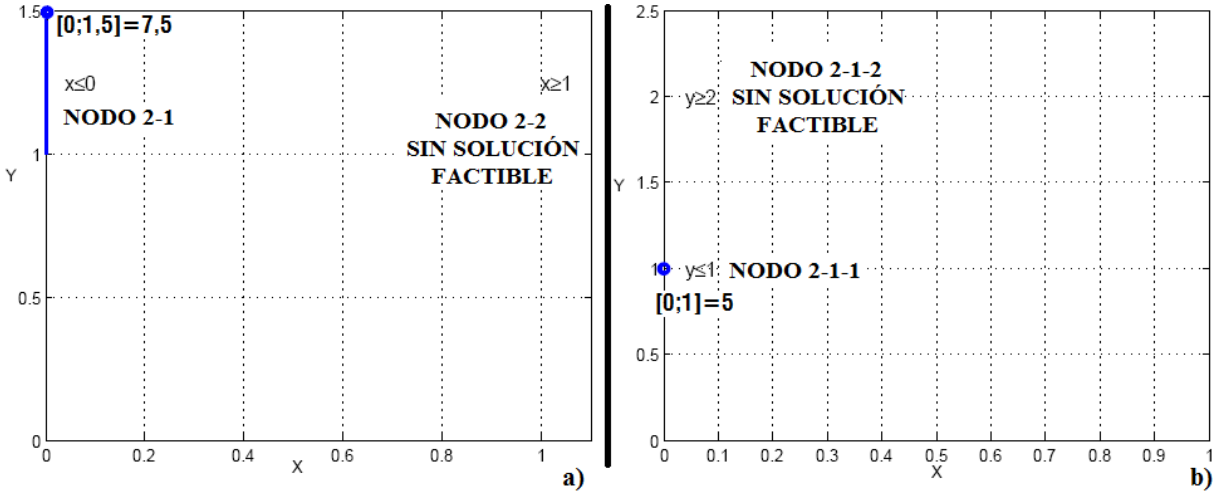

Figura 5. a) Segunda ramificación b) Tercera ramificación

El Nodo 2-1-1 tiene todas sus variables enteras y su VFO es mayor que el VFO de pivote por tanto se convierte en la nueva FO de pivote y al no existir nodos vivos, [x=0; y=1]; Z(x,y)=[5] es la solución óptima del modelo lineal entero. En la figura 6 se muestra el árbol de ramificaciones y nodos.

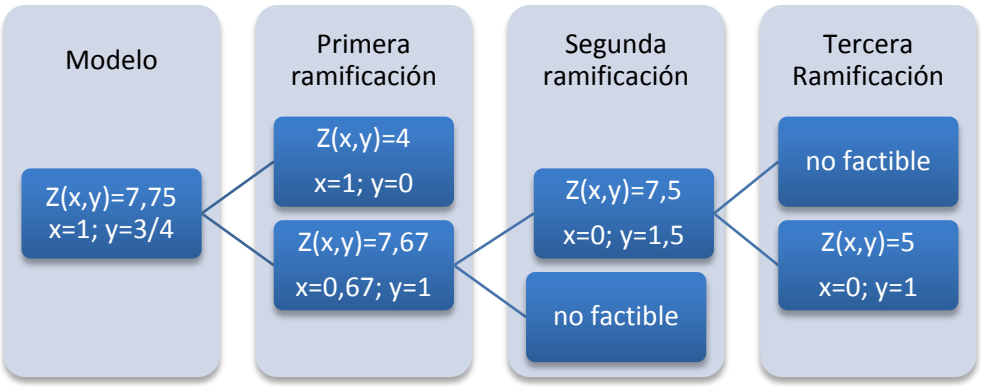

Figura 6. Árbol de ramificaciones y nodos del modelo

NBINTPROG

La función propuesta atiende al nombre de *nbintprog* y para su llamada debe encontrarse su ruta como carpeta actual (Current Folder) de MATLAB. En esta sección se describe el modo de trabajo de la herramienta.

Declarando vectores y matrices

La función *nbintprog* resuelve problemas de programación lineal entera del tipo:

$$\text{MIN } Z = f \times x \qquad \text{Sujeto a: } A \times x \leq b$$
$$Aeq \times x = beq$$
$$lb \leq x \leq ub$$

Donde:

f: Vector de coeficientes que multiplican a las variables de decisión en la FO
x: Vector de variables de decisión (solución)
b: Vector de términos de la derecha de las restricciones de desigualdad
beq: Vector de términos de la derecha de las restricciones de igualdad
lb: Vector que define la frontera inferior del dominio
ub: Vector que define la frontera superior del dominio
A: Matriz de coeficientes que multiplican a las variables en las restricciones de desigualdad
Aeq: Matriz de coeficientes que multiplican a las variables en las restricciones de igualdad

Es importante el hecho de que *nbintprog* al igual que las funciones propias de MATLAB solo minimizan la FO, por tanto si se quiere maximizar se debe multiplicar por (-1) toda la FO como se muestra en la figura 7. De igual forma todas las restricciones de desigualdad las considera como (≤), por tanto las que sean (≥) se deben multiplicar por (-1) para cambiar el sentido de la desigualdad, tal transformación se realiza en las ecuaciones (13) y (14) del ejemplo 3 resultando como se muestran en la figura 7.

$$MAX\ (Z) = -4x - 5y \qquad f = [-4\ -5]$$
$$1x + 0y \leq 1$$
$$3x + 4y \leq 6$$
$$-1x + 0y \leq 0$$
$$0x - 1y \leq 0$$
$$\Rightarrow\ A = \begin{bmatrix} 1+0 \\ 3+4 \\ -1+0 \\ 0-1 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 6 \\ 0 \\ 0 \end{bmatrix}$$

Declaración en MATLAB

```
f= [-4  -5]
A= [1 0 ; 3 4 ;-1 0 ;0 -1]
b= [1; 6 ;0 ;0]
```

Figura 7. Declaración de matrices y vectores del ejemplo 3

Declarando datos de entrada (inputs) datos de salida (outputs)

Anteriormente se declararon las matrices y vectores que necesita *nbintprog*, ahora se procede a hacer un llamado a la función y la forma es como se muestra en la ecuación (15):

$$[output_1, output_2, \dots, output_n] = nbintprog(input_1, input_2, \dots, input_n) \quad (15)$$

Los inputs o datos de entrada que se encuentran a la derecha separados por coma son los datos que necesita el programa, los cuales se declararon anteriormente, un aspecto muy importante es que el orden de los inputs debe ser el específico de cada función. Se puede prescindir de algunos inputs, su utilización depende de los tipos de restricciones que existan en el modelo. Sin embargo si no se necesita un input intermedio, en su posición se debe declara una matriz nula para no alterar el orden de los demás, ver figura 8.

[…] = ***nbintprog*** **(*f, A, b*)** resuelve el modelo sujeto a $A \times x \leq b$

[…] = ***nbintprog*** **(*f, A, b, Aeq, beq*)** resuelve el problema de arriba satisfaciendo además las restricciones de igualdad del tipo $Aeq \times x = beq$

[…] = ***nbintprog*** **(*f, A, b, Aeq, beq, lb, ub*)** define además, la frontera inferior y superior de todas las variables de decisión, de modo que la solución siempre queda en el rango $lb \leq x \leq ub$

[…] = ***nbintprog*** **(*f, A, b, Aeq, beq, lb, ub, x0*)** se define además, un punto inicial de búsqueda

[…] = ***nbintprog*** **(*f, A, b, Aeq, beq, lb, ub, x0, options_int*)** *options_int* es una estructura de datos que permite entrar al programa diferentes parámetros para la búsqueda de la solución; a continuación se muestra cuáles son dichos parámetros y posteriormente cómo declararlos:

- Tolerancia con la cual las variables de decisión son consideradas enteras (*TolXInteger*):

$$TolXInteger = valor\ absoluto[redondeo(xbi) - xbi] \quad (16)$$

Esta tolerancia es necesaria debido a que MATLAB guarda cada número con una precisión de 16 lugares decimales, por lo que buscar una combinación de variables realmente enteras elevaría el número de ramificaciones y de tiempo, buscando una precisión innecesaria. *Nbintprog* al igual que las funciones propias de MATLAB tiene una *TolXInteger* por defecto de 10^{-8}.

- Tolerancia de parada de la FO (*TolFun*):

$$TolFun = (VFO\ de\ pivote - mejor\ VFO de\ los\ nodos\ vivos)/abs(VFO\ de\ pivote + 1) \quad (17)$$

En resumen, *TolFun* tiene en cuenta la proximidad del VFO de pivote con el mejor VFO de los nodos vivos, de forma que cuando esta proximidad es muy pequeña el programa se detiene y ofrece como solución la FO de pivote actual. Esta herramienta puede ser muy útil ya que existen problemas que presentan muchos nodos vivos extremadamente cercanos al VFO de pivote, y una vez que se ramifican resultan valores peores que dicho pivote lo que trae implícito un consumo de tiempo innecesario. El hecho de utilizar la estrategia de selección BIFO hace que en cada iteración disminuya esta tolerancia. *Nbintprog* al igual que las funciones propias de MATLAB tiene una *TolFun* por defecto de 10^{-3}.

- Máximo número de iteraciones(MaxIter)

Para no exceder un determinado número de iteraciones. La solución que devuelve es la correspondiente al pivote actual, tiene por defecto un valor de 1000 iteraciones.

- Máximo número de ramificaciones (MaxNodes)

Para no exceder un determinado número de nodos ramificados. La solución que devuelve es la correspondiente al pivote actual, tiene por defecto un valor de 5000 ramificaciones.

- Opción de visualización de los resultados del proceso iterativo de *nbintprog* (Display)

Para observar los resultados de cada iteración es necesario llevar el parámetro Display de su estado apagado (por defecto) a estado (iter) posteriormente se muestra la manera de hacerlo.

[…] = ***nbintprog*** (***f, A, b, Aeq, beq, lb, ub, x0, options_int, options***) *options* son una serie de opciones aplicables sólo a la función *linprog* que constituye una herramienta de apoyo para *nbintprog*, útil para la solución del PR, podría darse el caso especial en que se necesite la modificación de esta herramienta y *options* es la vía, para más información sobre *options* buscar en la ayuda de MATLAB la función *linprog*.

Para modificar los valores de los parámetros por defecto de *options_int* u *options* se utiliza la función *optimset* como se muestra en la ecuación (18). Es muy importante que los parámetros se escriban como aquí se indica incluyendo sus mayúsculas, ver su aplicación en el ejemplo 4.

$$options\_int = optimset('parametro1', valor1, 'parametro2', valor2, \dots) \qquad (18)$$

Los datos de salida (outputs), donde MATLAB devolverá los resultados obtenidos, se declaran entre corchetes en la parte izquierda de la función separados por coma, teniendo en cuenta la importancia del orden, como se observa a continuación:

x = ***nbintprog*** (…) devuelve las variables de decisión que dieron solución al problema

[*x, Fo*] = ***nbintprog*** (…) devuelve además, el valor de la FO de la solución

[*x, Fo, exit*] = ***nbintprog*** (…) devuelve además, la razón de parada del programa

***exit* = 1**: El programa convergió a la solución
***exit* = 2**: La *TolFun* existente fue menor que la *TolFun* admisible
***exit* = 3**: El número de iteraciones sobrepasó a MaxIter
***exit* = 4**: El número de ramificaciones sobrepasó a MaxNodes

[*x, Fo, exit, output*] = ***nbintprog*** (…) *output* es una estructura de datos que devuelve el tiempo empleado en encontrar la solución y la cantidad de iteraciones realizadas

Ejemplo (4)

Plantear y resolver el modelo del ejemplo 3 mediante *nbintprog* con una *TolFun*=0, pues se desea ramificar todos los nodos vivos con mejor VFO que el pivote, por muy cerca que se encuentren de este, una *TolXInteger* = 0,00001 y también se quiere visualizar la tabla de resultados, declarar las ecuaciones (13) y (14) como frontera inferior.

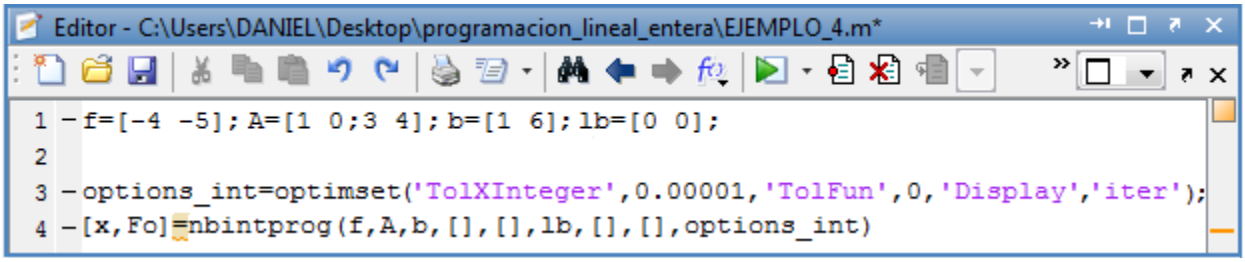

Figura 8. Planteamiento del modelo en el script

En la tabla 1 se observan los resultados de cada iteración. En la iteración 2 aparece un asterisco en la izquierda de la tabla, esto ocurre cada vez que hay un cambio de FO de pivote. Posteriormente *nbintprog* muestra la solución, véase que x es un vector que contiene a los valores de [x ; y]

Tabla 1. Resultados iterativos del modelo

Número de iteración	Nodos ramificados	Valor de la FO de pivote	Nodos vivos no ramificados	Menor VFO de nodos vivos	Tolerancia de FO existente (TolFun)
1	1	-4	1	-7,67	73 %
* 2	3	-5	0	-	-
x = [0,0000;1,0000] Fo =-5,0000					

PROBLEMA DE PRODUCCIÓN DE ASPERSORES

Un taller de montaje de aspersores consta de tres secciones A, B y C. En cada sección se pueden ensamblar tres tipos diferentes de aspersores A-10, A-12 y A-15. La capacidad de producción de cada sección varía dependiendo del tipo de aspersor a ensamblar como indica la tabla 2. Acorde con una orientación, se necesita que por cada aspersor A-10, se fabrique un aspersor A-12 y dos A-15. Determinar y resolver el modelo matemático que maximice el número de aspersores a fabricar.

Tabla 2. Secciones del taller con sus capacidades de producción diaria de aspersores

	Capacidad de producción diaria		
Sección	A-10	A-12	A-15
A	85	85	70
B	75	65	55
C	75	70	80

Formulación estándar del modelo

Variables de decisión:

$XA_{A\text{-}10}$ = Cantidad de aspersores A-10 a producir por día en la sección A.
$XA_{A\text{-}12}$ = Cantidad de aspersores A-12 a producir por día en la sección A.
$XA_{A\text{-}15}$ = Cantidad de aspersores A-15 a producir por día en la sección A.
$XB_{A\text{-}10}$ = Cantidad de aspersores A-10 a producir por día en la sección B.
$XB_{A\text{-}12}$ = Cantidad de aspersores A-12 a producir por día en la sección B.
$XB_{A\text{-}15}$ = Cantidad de aspersores A-15 a producir por día en la sección B.
$XC_{A\text{-}10}$ = Cantidad de aspersores A-10 a producir por día en la sección C.
$XC_{A\text{-}12}$ = Cantidad de aspersores A-12 a producir por día en la sección C.
$XC_{A\text{-}15}$ = Cantidad de aspersores A-15 a producir por día en la sección C.

Función objetivo:

$$\boldsymbol{max\, Z = XA_{A-10} + XA_{A-12} + XA_{A-15} + XB_{A-10} + XB_{A-12} + XB_{A-15} + XC_{A-10} + XC_{A-12} + XC_{A-15}} \quad (19)$$

-Restricciones de capacidad de producción

$$\boldsymbol{\frac{XA_{A-10}}{85} + \frac{XA_{A-12}}{85} + \frac{XA_{A-15}}{70} \leq 1} \quad (20)$$

$$\boldsymbol{\frac{XB_{A-10}}{75} + \frac{XB_{A-12}}{65} + \frac{XB_{A-15}}{55} \leq 1} \quad (21)$$

$$\boldsymbol{\frac{XC_{A-10}}{75} + \frac{XC_{A-12}}{70} + \frac{XC_{A-15}}{80} \leq 1} \quad (22)$$

-Otras restricciones adicionales.

$$\boldsymbol{XA_{A-10} + XB_{A-10} + XC_{A-10} - XA_{A-12} - XB_{A-12} - XC_{A-12} = 0} \quad (23)$$

$$\boldsymbol{XA_{A-10} + XB_{A-10} + XC_{A-10} - 0.5 \cdot XA_{A-15} - 0.5 \cdot XB_{A-15} - 0.5 \cdot XC_{A-15} = 0} \quad (24)$$

-Restricciones de no negatividad:

$$\boldsymbol{XA_{A-10},\ XA_{A-12},\ XA_{A-15}, XB_{A-10}, XB_{A-12}, XB_{A-15}, XC_{A-10}, XC_{A-12}, XC_{A-15} \geq 0} \quad (25)$$

Adecuación del modelo para bintprog

A continuación se resuelve el problema utilizando una función propia de MATLAB llamada *bintprog*. Dicha función está concebida para resolver modelos de programación lineal entera binaria. De modo que las variables de decisión solo pueden tomar valores cero-uno (0-1), útil para problemas de grafos y redes (Cabrera 2012). Dicha función puede resolver modelos de PLEP aplicando un artificio matemático (MathWorks 2012). La idea fundamental es representar cada variable de decisión (X) como un conjunto de variables binarias $(z_1, z_2, \ldots z_n)$, el cual tiene la forma:

$$\sum_{n=1}^{n} \left(2^{n-1} z_{i,n}\right) \quad (26)$$

Por ejemplo la variable $XA_{A\text{-}10}$ tiene una capacidad de producción diaria de hasta 85 aspersores y se representa con el conjunto mostrado en la ecuación (27). Esta forma permite que la variable $XA_{A\text{-}10}$ pueda tomar todos los valores enteros desde 0 hasta 127, rango que encierra los posibles valores que puede alcanzar dicha variable.

Aplicación a cada variable de decisión:

$$XA_{A-10} = z_{1,1} + 2z_{1,2} + 4z_{1,3} + 8z_{1,4} + 16z_{1,5} + 32z_{1,6} + 64z_{1,7} \quad i = 1 \quad (27)$$

$$XA_{A-12} = z_{2,1} + 2z_{2,2} + 4z_{2,3} + 8z_{2,4} + 16z_{2,5} + 32z_{2,6} + 64z_{2,7} \quad i = 2 \quad (28)$$

$$XA_{A-15} = z_{3,1} + 2z_{3,2} + 4z_{3,3} + 8z_{3,4} + 16z_{3,5} + 32z_{3,6} + 64z_{3,7} \quad i = 3 \quad (29)$$

$$XB_{A-10} = z_{4,1} + 2z_{4,2} + 4z_{4,3} + 8z_{4,4} + 16z_{4,5} + 32z_{4,6} + 64z_{4,7} \quad i = 4 \quad (30)$$

$$XB_{A-12} = z_{5,1} + 2z_{5,2} + 4z_{5,3} + 8z_{5,4} + 16z_{5,5} + 32z_{5,6} + 64z_{5,7} \quad i = 5 \quad (31)$$

$$XB_{A-15} = z_{6,1} + 2z_{6,2} + 4z_{6,3} + 8z_{6,4} + 16z_{6,5} + 32z_{6,6} \quad i = 6 \quad (32)$$

$$XC_{A-10} = z_{7,1} + 2z_{7,2} + 4z_{7,3} + 8z_{7,4} + 16z_{7,5} + 32z_{7,6} + 64z_{7,7} \quad i = 7 \quad (33)$$

$$XC_{A-12} = z_{8,1} + 2z_{8,2} + 4z_{8,3} + 8z_{8,4} + 16z_{8,5} + 32z_{8,6} + 64z_{8,7} \quad i = 8 \quad (34)$$

$$XC_{A-15} = z_{9,1} + 2z_{9,2} + 4z_{9,3} + 8z_{9,4} + 16z_{9,5} + 32z_{9,6} + 64z_{9,7} \quad i = 9 \quad (35)$$

Función objetivo binaria:

$$\begin{aligned}\min Z = &-z_{1,1} - 2z_{1,2} - 4z_{1,3} - 8z_{1,4} - 16z_{1,5} - 32z_{1,6} - 64z_{1,7} - z_{2,1} - 2z_{2,2} - 4z_{2,3} - 8z_{2,4} \\ &- 16z_{2,5} - 32z_{2,6} - 64z_{2,7} - z_{3,1} - 2z_{3,2} - 4z_{3,3} - 8z_{3,4} - 16z_{3,5} - 32z_{3,6} - 64z_{3,7} \\ &- z_{4,1} - 2z_{4,2} - 4z_{4,3} - 8z_{4,4} - 16z_{4,5} - 32z_{4,6} - 64z_{4,7} - z_{5,1} - 2z_{5,2} - 4z_{5,3} - 8z_{5,4} \\ &- 16z_{5,5} - 32z_{5,6} - 64z_{5,7} - z_{6,1} - 2z_{6,2} - 4z_{6,3} - 8z_{6,4} - 16z_{6,5} - 32z_{6,6} - z_{7,1} \\ &- 2z_{7,2} - 4z_{7,3} - 8z_{7,4} - 16z_{7,5} - 32z_{7,6} - 64z_{7,7} - z_{8,1} - 2z_{8,2} - 4z_{8,3} - 8z_{8,4} \\ &- 16z_{8,5} - 32z_{8,6} - 64z_{8,7} - z_{9,1} - 2z_{9,2} - 4z_{9,3} - 8z_{9,4} - 16z_{9,5} - 32z_{9,6} - 64z_{9,7}\end{aligned} \quad (36)$$

-Restricciones de capacidad de producción.

$$\frac{1}{85}z_{1,1} + \frac{2}{85}z_{1,2} + \frac{4}{85}z_{1,3} + \frac{8}{85}z_{1,4} + \frac{16}{85}z_{1,5} + \frac{32}{85}z_{1,6} + \frac{64}{85}z_{1,7} + \frac{1}{85}z_{2,1} + \frac{2}{85}z_{2,2} + \frac{4}{85}z_{2,3} + \frac{8}{85}z_{2,4} + \frac{16}{85}z_{2,5} + \frac{32}{85}z_{2,6} + \frac{64}{85}z_{2,7} + \frac{1}{70}z_{3,1} + \frac{2}{70}z_{3,2} + \frac{4}{70}z_{3,3} + \frac{8}{70}z_{3,4} + \frac{16}{70}z_{3,5} + \frac{32}{70}z_{3,6} + \frac{64}{70}z_{3,7} \leq 1 \quad (37)$$

$$\frac{1}{75}z_{4,1} + \frac{2}{75}z_{4,2} + \frac{4}{75}z_{4,3} + \frac{8}{75}z_{4,4} + \frac{16}{75}z_{4,5} + \frac{32}{75}z_{4,6} + \frac{64}{75}z_{4,7} + \frac{1}{65}z_{5,1} + \frac{2}{65}z_{5,2} + \frac{4}{65}z_{5,3} + \frac{8}{65}z_{5,4} + \frac{16}{65}z_{5,5} + \frac{32}{65}z_{5,6} + \frac{64}{65}z_{5,7} + \frac{1}{55}z_{6,1} + \frac{2}{55}z_{6,2} + \frac{4}{55}z_{6,3} + \frac{8}{55}z_{6,4} + \frac{16}{55}z_{6,5} + \frac{32}{55}z_{6,6} \leq 1 \quad (38)$$

$$\frac{1}{75}z_{7,1} + \frac{2}{75}z_{7,2} + \frac{4}{75}z_{7,3} + \frac{8}{75}z_{7,4} + \frac{16}{75}z_{7,5} + \frac{32}{75}z_{7,6} + \frac{64}{75}z_{7,7} + \frac{1}{70}z_{8,1} + \frac{2}{70}z_{8,2} + \frac{4}{70}z_{8,3} + \frac{8}{70}z_{8,4} + \frac{16}{70}z_{8,5} + \frac{32}{70}z_{8,6} + \frac{64}{70}z_{8,7} + \frac{1}{80}z_{9,1} + \frac{2}{80}z_{9,2} + \frac{4}{80}z_{9,3} + \frac{8}{80}z_{9,4} + \frac{16}{80}z_{9,5} + \frac{32}{80}z_{9,6} + \frac{64}{80}z_{9,7} \leq 1 \quad (39)$$

-Otras restricciones adicionales.

$$z_{1,1} + 2z_{1,2} + 4z_{1,3} + 8z_{1,4} + 16z_{1,5} + 32z_{1,6} + 64z_{1,7} + z_{4,1} + 2z_{4,2} + 4z_{4,3} + 8z_{4,4} + 16z_{4,5} + 32z_{4,6} + 64z_{4,7} + z_{7,1} + 2z_{7,2} + 4z_{7,3} + 8z_{7,4} + 16z_{7,5} + 32z_{7,6} + 64z_{7,7} - z_{2,1} - 2z_{2,2} - 4z_{2,3} - 8z_{2,4} - 16z_{2,5} - 32z_{2,6} - 64z_{2,7} - z_{5,1} - 2z_{5,2} - 4z_{5,3} - 8z_{5,4} - 16z_{5,5} - 32z_{5,6} - 64z_{5,7} - z_{8,1} - 2z_{8,2} - 4z_{8,3} - 8z_{8,4} - 16z_{8,5} - 32z_{8,6} - 64z_{8,7} = 0 \quad (40)$$

$$z_{1,1} + 2z_{1,2} + 4z_{1,3} + 8z_{1,4} + 16z_{1,5} + 32z_{1,6} + 64z_{1,7} + z_{4,1} + 2z_{4,2} + 4z_{4,3} + 8z_{4,4} + 16z_{4,5} + 32z_{4,6} + 64z_{4,7} + z_{7,1} + 2z_{7,2} + 4z_{7,3} + 8z_{7,4} + 16z_{7,5} + 32z_{7,6} + 64z_{7,7} - 0.5 \cdot z_{3,1} - 1z_{3,2} - 2z_{3,3} - 4z_{3,4} - 8z_{3,5} - 16z_{3,6} - 32z_{3,7} - 0.5z_{6,1} - 1z_{6,2} - 2z_{6,3} - 4z_{6,4} - 8z_{6,5} - 16z_{6,6} - 0.5z_{9,1} - 1z_{9,2} - 2z_{9,3} - 4z_{9,4} - 8z_{9,5} - 16z_{9,6} - 32z_{9,7} = 0 \quad (41)$$

-Restricciones de no negatividad:

$$-z_{1,1} - 2z_{1,2} - 4z_{1,3} - 8z_{1,4} - 16z_{1,5} - 32z_{1,6} - 64z_{1,7} \leq 0 \quad (42)$$
$$-z_{2,1} - 2z_{2,2} - 4z_{2,3} - 8z_{2,4} - 16z_{2,5} - 32z_{2,6} - 64z_{2,7} \leq 0 \quad (43)$$
$$-z_{3,1} - 2z_{3,2} - 4z_{3,3} - 8z_{3,4} - 16z_{3,5} - 32z_{3,6} - 64z_{3,7} \leq 0 \quad (44)$$
$$-z_{4,1} - 2z_{4,2} - 4z_{4,3} - 8z_{4,4} - 16z_{4,5} - 32z_{4,6} - 64z_{4,7} \leq 0 \quad (45)$$
$$-z_{5,1} - 2z_{5,2} - 4z_{5,3} - 8z_{5,4} - 16z_{5,5} - 32z_{5,6} - 64z_{5,7} \leq 0 \quad (46)$$
$$-z_{6,1} - 2z_{6,2} - 4z_{6,3} - 8z_{6,4} - 16z_{6,5} - 32z_{6,6} \leq 0 \quad (47)$$
$$-z_{7,1} - 2z_{7,2} - 4z_{7,3} - 8z_{7,4} - 16z_{7,5} - 32z_{7,6} - 64z_{7,7} \leq 0 \quad (48)$$
$$-z_{8,1} - 2z_{8,2} - 4z_{8,3} - 8z_{8,4} - 16z_{8,5} - 32z_{8,6} - 64z_{8,7} \leq 0 \quad (49)$$
$$-z_{9,1} - 2z_{9,2} - 4z_{9,3} - 8z_{9,4} - 16z_{9,5} - 32z_{9,6} - 64z_{9,7} \leq 0 \quad (50)$$

Declaración del modelo en MATLAB para bintprog

Se presenta en la tabla 3.

Tabla 3. Modelo en MATLAB para bintprog

```
%Función objetivo
f=[1 2 4 8 16 32 64 1 2 4 8 16 32 64 1 2 4 8 16 32 64 1 2 4 8 16 32
64 1 2 4 8 16 32 1 2 4 8 16 32 64 1 2 4 8 16 32 64]'*-1;
%Restricciones de desigualdad
A=zeros(12,62);
b=[1 1 1 0 0 0 0 0 0 0 0 0]';
A(1,:)=[1/85 2/85 4/85 8/85 16/85 32/85 64/85 1/85 2/85 4/85 8/85 16/85 32/85 64/85
1/70 2/70 4/70 8/70 16/70 32/70 64/70 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
A(2,:)=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1/75 2/75 4/75 8/75 16/75 32/75
64/75 1/65 2/65 4/65 8/65 16/65 32/65 64/65 1/55 2/55 4/55 8/55 16/55 32/55 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
A(3,:)=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1/75 2/75 4/75 8/75 16/75 32/75 64/75 1/70 2/70 4/70 8/70 16/70 32/70 64/70
1/80 2/80 4/80 8/80 16/80 32/80 64/80];
A(4,[1:7])=[1 2 4 8 16 32 64]*-1;A(5,[8:14])=[1 2 4 8 16 32 64]*-1;
A(6,[15:21])=[1 2 4 8 16 32 64]*-1;A(7,[22:28])=[1 2 4 8 16 32 64]*-1;
A(8,[29:35])=[1 2 4 8 16 32 64]*-1;A(9,[36:41])=[1 2 4 8 16 32]*-1;
A(10,[42:48])=[1 2 4 8 16 32 64]*-1;A(11,[49:55])=[1 2 4 8 16 32 64]*-1;
A(12,[56:62])=[1 2 4 8 16 32 64]*-1;
%Restricciones de igualdad
Aeq=zeros(2,62);
Aeq(1,:)=[1 2 4 8 16 32 64 -1 -2 -4 -8 -16 -32 -64 0 0 0 0 0 0 0 1 2 4 8 16 32 64 -1
-2 -4 -8 -16 -32 -64 0 0 0 0 0 0 1 2 4 8 16 32 64 -1 -2 -4 -8 -16 -32 -64 0 0 0 0 0
0 0];
Aeq(2,:)=[1 2 4 8 16 32 64 0 0 0 0 0 0 0 -.5 -1 -2 -4 -8 -16 -32 1 2 4 8 16 32 64 0
0 0 0 0 0 0 -.5 -1 -2 -4 -8 -16 1 2 4 8 16 32 64 0 0 0 0 0 0 0 -.5 -1 -2 -4 -8 -16
-32];
beq=[0 0]';
[x,fval,exit,out]=bintprog(f,A,b,Aeq,beq,[],optimset('Display','iter','TolFun',0));
X=zeros(9,1);
X(1,1)=x(1,1)+2*x(2,1)+4*x(3,1)+8*x(4,1)+16*x(5,1)+32*x(6,1)+64*x(7,1);
X(2,1)=x(8,1)+2*x(9,1)+4*x(10,1)+8*x(11,1)+16*x(12,1)+32*x(13,1)+64*x(14,1);
X(3,1)=x(15,1)+2*x(16,1)+4*x(17,1)+8*x(18,1)+16*x(19,1)+32*x(20,1)+64*x(21,1);
X(4,1)=x(22,1)+2*x(23,1)+4*x(24,1)+8*x(25,1)+16*x(26,1)+32*x(27,1)+64*x(28,1);
X(5,1)=x(29,1)+2*x(30,1)+4*x(31,1)+8*x(32,1)+16*x(33,1)+32*x(34,1)+64*x(35,1);
X(6,1)=x(36,1)+2*x(37,1)+4*x(38,1)+8*x(39,1)+16*x(40,1)+32*x(41,1);
X(7,1)=x(42,1)+2*x(43,1)+4*x(44,1)+8*x(45,1)+16*x(46,1)+32*x(47,1)+64*x(48,1);
X(8,1)=x(49,1)+2*x(50,1)+4*x(51,1)+8*x(52,1)+16*x(53,1)+32*x(54,1)+64*x(55,1);
X(9,1)=x(56,1)+2*x(57,1)+4*x(58,1)+8*x(59,1)+16*x(60,1)+32*x(61,1)+64*x(62,1);
```

Solución mediante bintprog

$XA_{A\text{-}10} = 0$	$XB_{A\text{-}10} = 57$	$XC_{A\text{-}10} = 0$
$XA_{A\text{-}12} = 51$	$XB_{A\text{-}12} = 6$	$XC_{A\text{-}12} = 0$
$XA_{A\text{-}15} = 28$	$XB_{A\text{-}15} = 6$	$XC_{A\text{-}15} = 80$

Valor de la función objetivo (fval)=228
Razón de parada (exit)=se excedió el número de ramificaciones por defecto de *bintprog*

Declaración del modelo en MATLAB para nbintprog

Se presenta en la tabla 4.

Tabla 4. Modelo en MATLAB para nbintprog

```
%Función objetivo
f=-ones(9,1);
%Restricciones de desigualdad
A=zeros(3,9);A(1,1:3)=[1/85 1/85 1/70];
A(2,4:6)=[1/75 1/65 1/55];A(3,7:9)=[1/75 1/70 1/80];
b=ones(3,1);
%Restricciones de igualdad
Aeq=[1 -1 0 1 -1 0 1 -1 0; 2 0 -1 2 0 -1 2 0 -1];
beq=zeros(2,1);
%Frontera inferior
LB=zeros(9,1);
options_int=optimset('Display','iter','TolFun',0);
[x,Fo,exit,out]=nbintprog(f,A,b,Aeq,beq,LB,[],[],options_int)
```

Solución mediante nbintprog

$XA_{A\text{-}10} = 2$ $\quad XB_{A\text{-}10} = 55$ $\quad XC_{A\text{-}10} = 0$

$XA_{A\text{-}12} = 52$ $\quad XB_{A\text{-}12} = 3$ $\quad XC_{A\text{-}12} = 2$

$XA_{A\text{-}15} = 25$ $\quad XB_{A\text{-}15} = 12$ $\quad XC_{A\text{-}15} = 77$

Valor de la función objetivo (fval)=228
Razón de parada (exit)=convergencia a la solución

Comparación de los resultados

Como se observa *bintprog* se detiene producto que excede el número de ramificaciones por defecto aproximadamente a los 28 minutos como muestra la tabla 5; aquí cabe la posibilidad de modificar el valor de MaxNodes, pero no es el objetivo. Pese a que el programa se detiene, ofrece la mejor solución que presenta hasta el momento, con el inconveniente que dicha solución posee una TolFun=0,78%. Esta tolerancia significa que, a los 28 minutos de cómputo, existen nodos vivos y por tanto la posibilidad de encontrar mejores soluciones que la mostrada. Como se muestra en la tabla 5, cuando *bintprog* se detiene aún presenta 23165 nodos vivos. Esta significativa desaceleración de la convergencia a la solución se debe al aumento del número de variables.

¿Por qué el incremento de variables desacelera significativamente la convergencia a la solución?

En la primera ramificación del ejemplo 3, el nodo 2 tiene la solución [x=0,67; y=1], sin embargo al ramificarlo, el Nodo 2-1 tiene como variables [x=0; y=1,5], véase que al hacer entera (x), la variable (y) que era entera producto de la primera ramificación del modelo vuelve a tomar un valor fraccionario. Por tanto, una variable entera dentro de la solución puede dejar de serlo ante la ramificación de otras, por lo que el programa va restringiendo el dominio de cada variable fraccionaria hasta que encuentre una combinación de variables enteras o se cierre el dominio y no se encuentre solución, entonces salta a otro nodo vivo. Si el número de variables es grande, mientras se hace entera una, son más las que pueden dejar de serlo. De modo que para que *bintprog* logre que las 62 variables binarias sean enteras tiene que realizar muchas ramificaciones. Por dicha razón *bintprog* no encuentra su primera solución hasta la ramificación 12789. Además, un incremento de ramificaciones significa generar mayor cantidad de nodos vivos que posteriormente puede ser obligatorio ramificar. De modo que cuando *bintprog* encontró su primera solución ya contaba con 6596 nodos vivos, esto se puede corroborar en la tabla de resultados iterativos de *bintprog* cuando se corre el modelo. Por esta razón el hecho de

que la cantidad de variables binarias de *bintprog* dependa de la capacidad máxima de las variables no binarias puede resultar insatisfactorio.

Nbintprog no se encuentra sometido a dicho fenómeno por lo que converge más rápido a la solución. En el caso del modelo de aspersores no solo converge a la solución 14 veces más rápido que *bintprog* (tabla 5), sino que el resultado presenta una calidad superior (*TolFun*=0), lo que significa que se ramificaron todos los nodos vivos y no se encontró mejor solución entera. Para hacer una comparación con soluciones de igual calidad se puede decir que *nbintprog* alcanza una *TolFun* ligeramente menor que 0,78% en la iteración número 23 para un tiempo de un minuto y medio, queda por parte del lector comprobarlo.

Tabla 5. Proceso de búsqueda en ambas funciones

FUNCIÓN	ITERACIONES	NODOS RAMIFICADOS	NODOS NO RAMIFICADOS	TolFun EXISTENTE	TIEMPO
BINTPROG	178302	62000	23165	0,78%	28 min y 40 seg
NBINTPROG	115	3175	0	0%	2 min 30 seg

CONCLUSIONES

Se presenta la función *nbintprog* para MATLAB, capaz de resolver modelos de programación lineal entera pura (PLEP) utilizando como base matemática el método de ramificación y acotación (branch and bound) empleando la estrategia BIFO (best input first output). Dicha herramienta, a diferencia de la función *bintprog* (intrínseca de MATLAB), no requiere de una adecuación de su formulación estándar a un modelo binario para resolver problemas no binarios de programación lineal entera.

Se realiza una comparación integral basada en el empleo de *bintprog* y *nbintprog* para resolver un problema de ingeniería. De esta comparación se puede concluir que se evidencian los inconvenientes que trae consigo el hecho de que la cantidad de variables binarias que se requieren para la adecuación de un modelo no binario dependa de la capacidad máxima de las variables no binarias. Esto hace que la solución brindada por la función *bintprog* tarde 14 veces más que la brindada por *nbintprog*. Además con la solución de *nbintprog* se ramifican todos los nodos vivos y no se encuentra mejor solución entera, mientras que *bintprog* brinda un resultado cuando excede el número máximo prefijado de ramificaciones por defecto.

REFERENCIAS

Cabrera E. (2012). "Programación lineal.", Conferencia de planeamiento y operación de los recursos hidráulicos, CIH, Facultad de Ingeniería Civil, Cujae, La Habana.

Lahoz P. (2009). "Programación lineal entera", Investigacion operativa, extraído de: http://ocw.unizar.es/modelos-de-investigacion-operativa/OCWProgEntera.pdf en enero 2013.

Troncoso A. (2005). "Técnicas avanzadas de predicción y optimización aplicadas a sistemas de potencia", Tesis de doctorado, Dpto. de lenguajes y sistemas, Univ. de Sevilla, España.

Marrero N. (1985). "Técnicas de optimización aplicadas a la ingeniería hidráulica", Ed. ISPJAE, Facultad de Ingeniería Civil, ISPJAE, La Habana.

MathWorks (2012). "Solve binary integer programming problems", Optimization Toolbox For Use with MATLAB®, MATLAB, USA.

Society I. C. (2010). "The Nature of Mathematical Programming". Mathematical Programming Glossary, extraído de: http://www.glossary.computing.society.informs.org en enero 2013.

Wolsey L. and Nemhauser G. (1988). "Integer and Combinatorial Optimization.", Ed. John Wiley& Sons, New York.