
Universidad Complutense de Madrid
Departamento de Sistemas Informáticos y Programación

Parte 2:

Programación con restricciones

Fernando Sáenz Pérez
15/12/2004

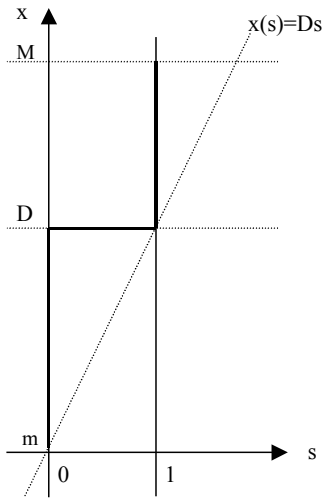
Contenido

1. Representación de funciones discretas en programación lineal mixta

Para modelar las funciones discretas, como la función escalón, con restricciones lineales se pueden aplicar dos enfoques. El primero es un enfoque topológico para determinar las regiones factibles en un plano continuo-discreto (ordenadas-coordenadas). El segundo es un enfoque lógico, más intuitivo, en el que se introducen variables binarias para modelar expresiones lógicas. Así, se hace uso de la programación lineal mixta para la expresión y resolución (eficaz) de las funciones discretas, evitando el uso de las conectivas lógicas de la programación con restricciones.
En lo que sigue, x es una variable real, s y s1 son variables binarias, y [m,M] es el dominio de x.
Si la variable binaria s se interpreta como una variable lógica, su complemento se expresa como 1-s. La disyunción exclusiva v1⊕v2 se expresa con la variable lógica auxiliar s como v1s+v2(1-s). Una inecuación "x operador (v1 si s) o (v2 si no s)" se expresa como "x operador v1s+v2(1-s)".

1.1. Función escalón 0-1

La función escalón se describe mediante las restricciones: Si x≤D y x≥m entonces s=0. Si x≤M y x≥D entonces s=1. Es decir: $x \le \begin{cases} D, para\ s=0 \\ M, para\ s=1 \end{cases}$ y $x \ge \begin{cases} m, para\ s=0 \\ D, para\ s=1 \end{cases}$, que se expresan en las dos restricciones lineales mixtas: $x(s) \le D(1-s) + Ms$ y $x(s) \ge m(1-s) + Ds$.

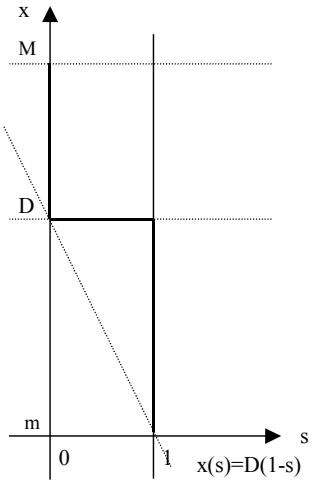


Modelo en OPL: esca01.prj y esca01.osc.

Nota: Para x=D, hay dos posibles valores de s porque las inecuaciones describen intervalos cerrados.

1.2. Función escalón 1-0

La función escalón se describe mediante las restricciones: Si x≤D y x≥m entonces s=1. Si x≤M y x≥D entonces s=0. Es decir: $x \le \begin{cases} M, para\ s=0 \\ D, para\ s=1 \end{cases}$ y $x \ge \begin{cases} D, para\ s=0 \\ m, para\ s=1 \end{cases}$, que se expresan en las dos restricciones lineales mixtas: $x(s) \le M(1-s) + Ds$ y $x(s) \ge D(1-s) + ms$.



Modelo en OPL: esca10.prj y esca10.osc.

Nota: Para x=D, hay dos posibles valores de s porque las inecuaciones describen intervalos cerrados.

Otra alternativa para modelar funciones discretas es mediante la composición de funciones discretas conocidas. Por ejemplo, la función escalón 1-0 ($f_{esc10}(x)$) se puede expresar en términos de la función escalón 0-1 ($f_{esc01}(x)$) como $f_{esc10}(x) = 1 - f_{esc01}(x)$.

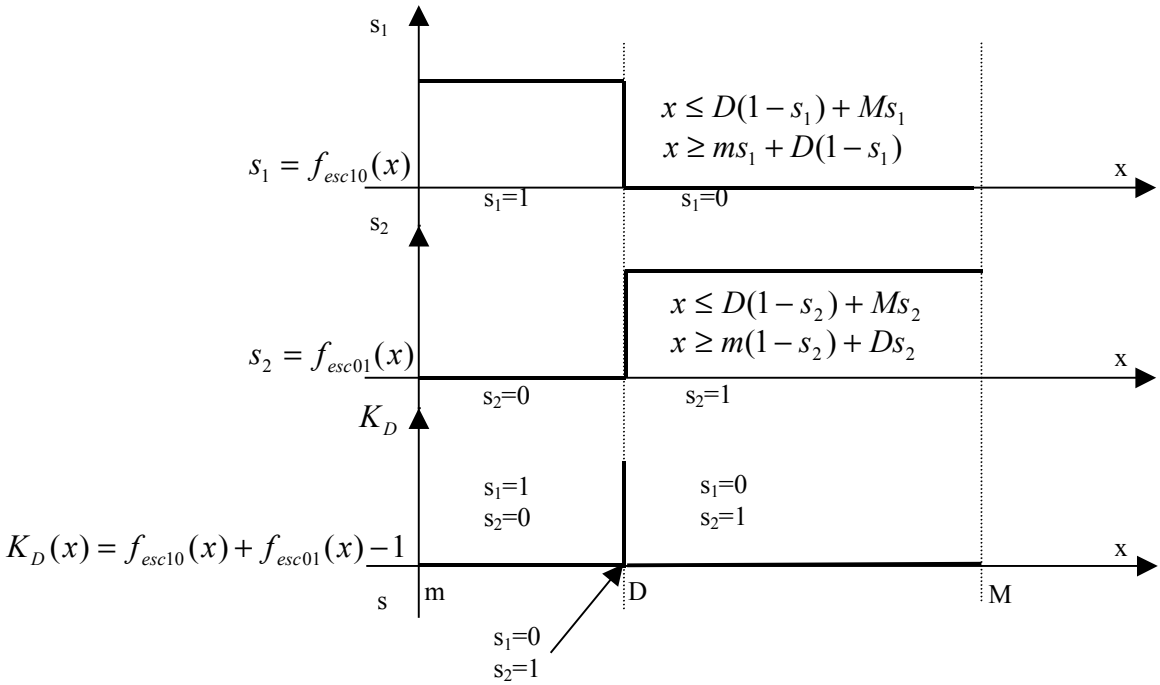
En 1), $f_{esc01}(x)$ está representada por s, por lo que $f_{esc10}(x) = 1 - s$. Para obtener las restricciones que la modelan (x(s)), basta con hacer el cambio de variable s por 1-s:

$x \le D(1-s) + Ms \Rightarrow x \le D(1-(1-s)) + M(1-s) = Ds + M(1-s)$ y

$x \ge m(1-s) + Ds \Rightarrow x \ge m(1-1+s) + D(1-s) = ms + D(1-s)$, por lo que se llega a las mismas restricciones.

1.3. Función delta de Kronecker

En el plano real, la delta de Kronecker se puede especificar por composición de las funciones escalón 01 y 10 como muestra la siguiente figura:



Así, las restricciones que describen a la función delta de Kronecker vendrían dadas por lo siguiente, si especificasen intervalos cerrados:

$x \ge ms_1 + D(1-s_1)$

$x \le Ds_1 + M(1-s_1)$

$x \le D(1-s_2) + Ms_2$

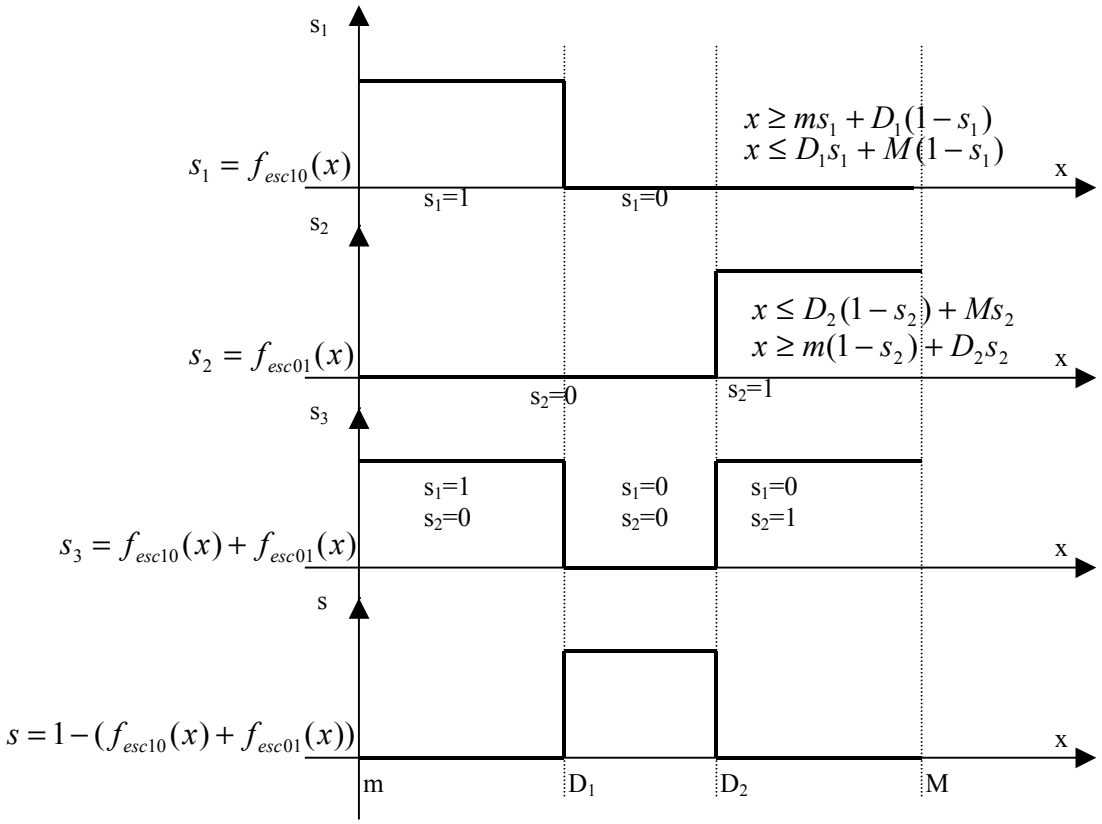
$x \ge m(1-s_2) + Ds_2$

$s = s_1 + s_2 - 1$

1.4. Función pulso cuadrado

Definimos la función pulso cuadrado como $pc_{D_1,D_2}(x) = \begin{cases} 1, x \in (D_1, D_2) \\ 0, e.o.c. \end{cases}$

Para obtener la función pulso cuadrado se aplicará también la composición de funciones discretas, como se observa en la siguiente figura:



Así, las restricciones que describen a la función pulso cuadrado vendrían dadas por:

$$x \geq ms_1 + D_1(1-s_1)$$
$$x \leq D_1 s_1 + M(1-s_1)$$
$$x \leq D_2(1-s_2) + Ms_2$$
$$x \geq m(1-s_2) + D_2 s_2$$
$$s = 1-(s_1 + s_2)$$

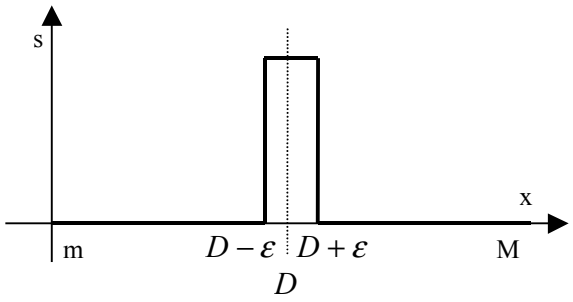
Sustituyendo s_2 de la última ecuación en las anteriores inecuaciones, se obtiene:

$$x \geq ms_1 + D_1(1-s_1)$$
$$x \leq D_1 s_1 + M(1-s_1)$$
$$x \leq D_2(s_1 + s) + M(1-s_1-s)$$
$$x \geq m(s_1 + s) + D_2(1-s_1-s)$$
$$s + s_1 \leq 1$$

Modelo en OPL: pulscuad.prj y pulscuad.osc.

1.5. Función Delta de Kronecker

La función Delta de Kronecker, que se ilustra en la siguiente figura se modelaría con las mismas restricciones anteriores con $D_1 = D - \varepsilon$ y $D_2 = D + \varepsilon$, siendo ε un número lo suficientemente pequeño.



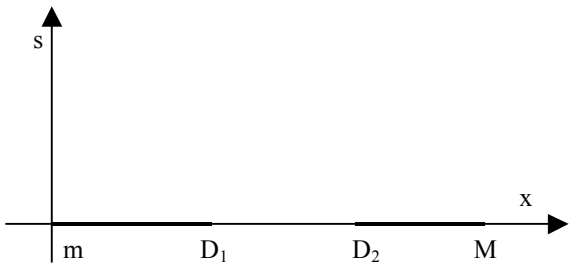
$$x \geq ms_1 + (D-\varepsilon)(1-s_1)$$
$$x \leq (D-\varepsilon)s_1 + M(1-s_1)$$
$$x \leq (D+\varepsilon)(s_1+s) + M(1-s_1-s)$$
$$x \geq m(s_1+s) + (D+\varepsilon)(1-s_1-s)$$
$$s + s_1 \leq 1$$

Modelo en OPL: deltaK.prj y deltaK.osc.

1.6. Función constante discontinua en un intervalo

Se representa en la siguiente figura, y se representa mediante la disyunción exclusiva $(x \geq m \wedge x \leq D_1) \oplus (x \geq D_2 \wedge x \leq M)$. Usando la variable binaria s que denota el argumento de la disyunción exclusiva que se cumple, se reescribe con restricciones de manera análoga al punto 1) como:

$$x \leq D_1 s + M(1-s)$$
$$x \geq ms + D_2(1-s)$$



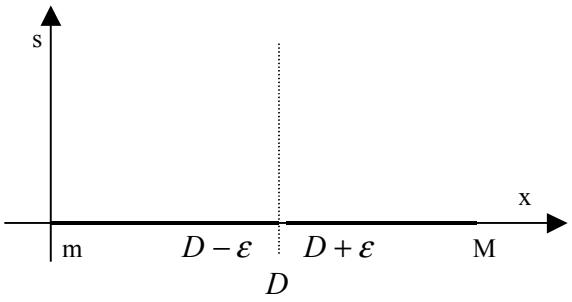
Modelo en OPL: discint.prj.

1.7. Función constante discontinua en un punto

Corresponde a la misma situación anterior con $D_1 = D - \varepsilon$ y $D_2 = D + \varepsilon$, siendo ε un número lo suficientemente pequeño.

La función constante discontinua en un punto se representa con:

$$x \leq (D-\varepsilon)s + M(1-s)$$
$$x \geq ms + (D+\varepsilon)(1-s)$$

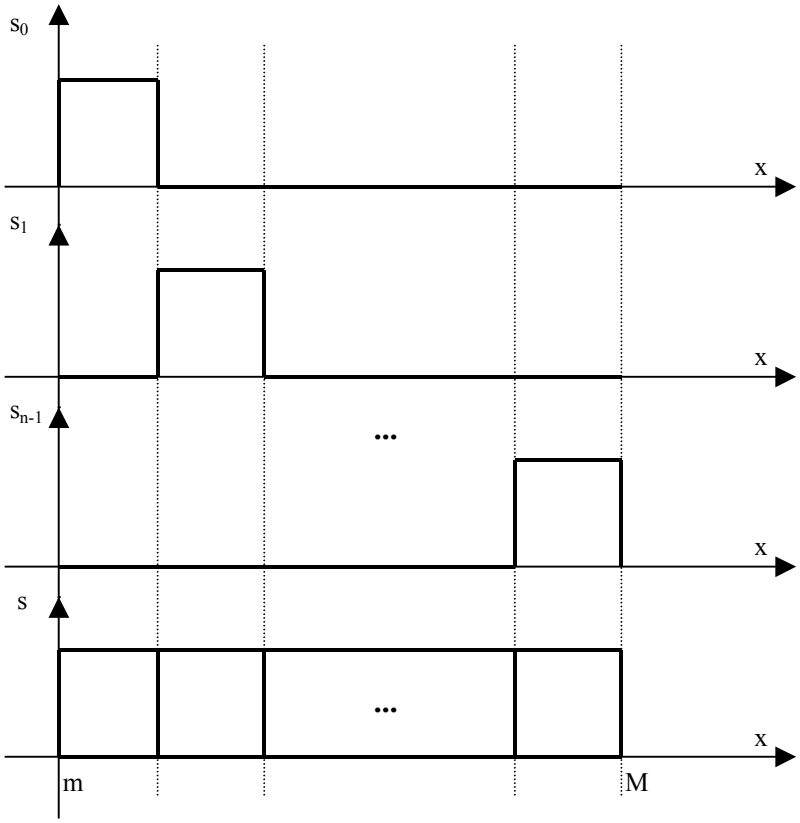


Modelo en OPL: discint.prj.

1.8. Recubrimiento de un segmento real con funciones pulso cuadrado

1.8.1. Intervalos cerrados

Dado un segmento real representado por una variable $x \in [m, M]$, se trata de recubrirlo completamente con pulsos cuadrados no solapados, como se muestra en la siguiente figura:



Asignando una variable binaria s_i a cada intervalo $[x_i, x_{i+1}]$, se impone que sólo una pueda valer 1 y que la variable independiente x tenga un valor en el intervalo $[x_i, x_{i+1}]$ cuando $s_i = 1$, que se modela con:

$$\sum_{i=0}^{n-1} s_i = 1$$

$$x \geq \sum_{i=0}^{n-1} x_i s_i$$

$$x \leq \sum_{i=0}^{n-1} x_{i+1} s_i$$

Aplicación: Gasoducto con presiones en un extremo p_A y en otro p_B sometido a un flujo x. La relación entre las tres variables se plantea por intervalos de confianza:

$$x \in [x_i, x_{i+1}] \Leftrightarrow p_A \in [mp_{Ai}, Mp_{Ai}] \wedge p_B \in [mp_{Bi}, Mp_{Bi}]$$

Esto se puede plantear de la siguiente forma alternativa:

$$x \in [x_i, x_{i+1}] \Leftrightarrow \begin{cases} mp_{Ai} \leq p_A \leq Mp_{Ai} \\ mp_{Bi} \leq p_B \leq Mp_{Bi} \end{cases}$$

Asignando una variable binaria s_i a cada intervalo $[x_i, x_{i+1}]$, se impone que sólo una pueda valer 1 y que la variable independiente x tenga un valor en el intervalo $[x_i, x_{i+1}]$ cuando $s_i = 1$, que se modela con:

$$\sum_{i=0}^{n-1} s_i = 1$$

$$x \geq \sum_{i=0}^{n-1} x_i s_i$$

$$x \leq \sum_{i=0}^{n-1} x_{i+1} s_i$$

Así, se pueden plantear las siguientes restricciones:

$$p_A \geq \sum_{i=0}^{n-1} s_i \cdot mp_{Ai}$$

$$p_A \leq \sum_{i=0}^{n-1} s_i \cdot Mp_{Ai}$$

$$p_B \geq \sum_{i=0}^{n-1} s_i \cdot mp_{Bi}$$

$$p_B \leq \sum_{i=0}^{n-1} s_i \cdot Mp_{Bi}$$

En total, dado n el número de intervalos, se necesitan 3 restricciones para modelar los pulsos cuadrados y 4 para acotar las presiones; n variables binarias para los pulsos cuadrados y tres para las variables reales. 7 restricciones, n variables binarias y 3 variables reales.

1.8.2. Intervalos semicerrados

Los intervalos son ahora de la forma $[x_i, x_{i+1})$. Es necesario añadir un desplazamiento ε para evitar los puntos comunes de los intervalos:

$$\sum_{i=0}^{n-1} s_i = 1$$

$$x \geq \sum_{i=0}^{n-1} x_i s_i$$

$$x \leq -\varepsilon + \sum_{i=0}^{n-1} x_{i+1} s_i$$

2. Representación de funciones no lineales en programación lineal mixta

2.1. Función lógica igualdad

La función lógica igualdad $s = (x = y)$, donde x e y son variables reales y s es binaria:

$$s = \begin{cases} 0, si\ x \neq y \\ 1, si\ x = y \end{cases}$$

Haciendo $d = x - y$, d es la función Delta de Kronecker. Por lo tanto:

$$d \geq ms_1 - \varepsilon(1 - s_1)$$
$$d \leq -\varepsilon s_1 + M(1 - s_1)$$
$$d \leq \varepsilon(s_1 + s) + M(1 - s_1 - s)$$
$$d \geq m(s_1 + s) + \varepsilon(1 - s_1 - s)$$

O, lo que es lo mismo, y sin introducir d:

$$x - y \geq ms_1 - \varepsilon(1 - s_1)$$
$$x - y \leq -\varepsilon s_1 + M(1 - s_1)$$
$$x - y \leq \varepsilon(s_1 + s) + M(1 - s_1 - s)$$
$$x - y \geq m(s_1 + s) + \varepsilon(1 - s_1 - s)$$

donde M es el máximo de la diferencia $x - y$, y m es el mínimo de la diferencia $x - y$. Si $x \in [m_x, M_x]$ e $y \in [m_y, M_y]$, entonces:

$$M = M_x - m_y$$
$$m = m_x - M_y$$

Modelo en OPL: igualdad.prj y igualdad.osc.

2.2. Función lógica desigualdad

La función lógica desigualdad $s = (x \neq y)$ se implementa como el complemento de la igualdad, es decir, haciendo el cambio de variable s por $1 - s$:

$$x - y \geq ms_1 - \varepsilon(1 - s_1)$$
$$x - y \leq -\varepsilon s_1 + M(1 - s_1)$$
$$x - y \leq \varepsilon(1 + s_1 - s) + M(s - s_1)$$
$$x - y \geq m(1 + s_1 - s) + \varepsilon(s - s_1)$$

(Con el mismo significado para los símbolos que en el caso anterior)

Modelo en OPL: desigual.prj y desigual.osc.

2.3. Equivalencia lógica

La equivalencia $L_1 \Leftrightarrow L_2$, siendo L_1 y L_2 variables lógicas, es el resultado lógico de $L_1 = L_2$, y se puede representar en programación lineal mixta con la inecuación $L_1 \leq L_2$, como se puede comprobar en la siguiente tabla de verdad:

L_1	L_2	$L_1 \Leftrightarrow L_2\ (L_1 \le L_2) \equiv \overline{L_1 \oplus L_1}$
0	0	1
0	1	0
1	0	0
1	1	1

2.4. Implicación lógica $(x = 0) \Rightarrow (y = c)$, x binaria, y real, c constante real

Se debe acotar y cuando $x = 0$. Por lo tanto:

$y \le cxM_x$

2.5. Implicación lógica

La implicación $L_1 \Rightarrow L_2$, siendo L_1 y L_2 variables lógicas, es el resultado lógico de $L_1 = L_2$, y se puede representar en programación lineal mixta con la inecuación $L_1 \le L_2$, como se puede comprobar en la siguiente tabla de verdad:

L_1	L_2	$L_1 \Rightarrow L_2\ (L_1 \le L_2) \equiv \overline{L_1 \oplus L_1}$
0	0	1
0	1	1
1	0	0
1	1	1

2.6. Producto de una variable real por una variable binaria

Se trata de representar $y = sx$, donde x e y son reales y s es binaria, $x, y \in [0,1],\ s \in \{0,1\}$

2.6.1. Primer planteamiento

El resultado se puede representar con las siguientes equivalencias lógicas:

$s = 0 \Leftrightarrow y = 0$

$s = 1 \Leftrightarrow y = x$

La condición lógica $s = 1$ se representa por s, y su complemento ($s = 0$) por $1 - s$. El valor 1 se interpreta como *true*, y el valor 0 como *false*.

Los antecedentes de las equivalencias se representan, respectivamente, por s y por $1 - s$, y los consecuentes, por $c_1 \equiv (y = 0)$ y $c_2 \equiv (y = x)$.

$c_1 = \begin{cases} 1, si\ y = 0 \\ 0, si\ y \neq 0 \end{cases}$, que corresponde a la función Delta de Kronecker.

$c_2 = \begin{cases} 1, si\ y = x \\ 0, si\ y \neq x \end{cases}$, que corresponde a la función lógica igualdad.

Representación en programación lineal mixta:

1) Equivalencia $s = 0 \Leftrightarrow y = 0$:

$1 - s = c_1$

2) Equivalencia $s = 1 \Leftrightarrow y = x$:

$s = c_2$

3) Delta de Kronecker $c_1 = \begin{cases} 1, si\ y = 0 \\ 0, si\ y \neq 0 \end{cases}$:

$$y \geq m_y s_1 - \varepsilon(1 - s_1)$$
$$y \leq -\varepsilon s_1 + M_y(1 - s_1)$$
$$y \leq \varepsilon(s_1 + c_1) + M_y(1 - s_1 - c_1)$$
$$y \geq m_y(s_1 + c_1) + \varepsilon(1 - s_1 - c_1)$$
$$s_1 + c_1 \leq 1$$

Donde m_y es el mínimo valor del dominio de y, y M_y es el máximo valor del dominio de y.

4) Función igualdad $c_2 = \begin{cases} 1, si\ y = x \\ 0, si\ y \neq x \end{cases}$

$$x - y \geq m s_2 - \varepsilon(1 - s_2)$$
$$x - y \leq -\varepsilon s_2 + M(1 - s_2)$$
$$x - y \leq \varepsilon(s_2 + c_2) + M(1 - s_2 - c_2)$$
$$x - y \geq m(s_2 + c_2) + \varepsilon(1 - s_2 - c_2)$$
$$s_2 + c_2 \leq 1$$

Donde M_{x-y} es el valor máximo que puede tomar la diferencia $x - y$, y m_{x-y} el valor mínimo, que se calculan como:

$$M_{x-y} = M_x - m_y$$
$$m_{x-y} = m_x - M_y$$

Sustituyendo c_1 de 1) y c_2 de 2) en 3) y 4), se obtiene:

$$y \geq m_y s_1 - \varepsilon(1 - s_1)$$
$$y \leq -\varepsilon s_1 + M_y(1 - s_1)$$
$$y \leq \varepsilon(s_1 + 1 - s) + M_y(s - s_1)$$
$$y \geq m_y(s_1 + 1 - s) + \varepsilon(s - s_1)$$
$$s_1 \leq s$$
$$x - y \geq m s_2 - \varepsilon(1 - s_2)$$
$$x - y \leq -\varepsilon s_2 + M(1 - s_2)$$
$$x - y \leq \varepsilon(s_2 + s) + M(1 - s_2 - s)$$
$$x - y \geq m(s_2 + s) + \varepsilon(1 - s_2 - s)$$
$$s_2 + s \leq 1$$

En resumen, se introducen tres nuevas variables binarias y diez restricciones. Además, el resultado no es exacto, tiene un error ε.

Modelo en OPL: prodrb.prj y prodrb.osc.

2.6.2. Segundo planteamiento

El producto

$$y = sx \quad (0)$$

donde x e y son reales y s es binaria se modela razonando en términos de las restricciones que deben aparecer para los distintos valores de las variables. La variable con menor cardinalidad en su dominio es s, por lo que se comienza planteando las restricciones que deben aparecer para sus dos posibles valores (0 y 1):

$$s = 0 \Rightarrow y = 0, \forall x \quad (1)$$

$$s = 1 \Rightarrow y = x \quad (2)$$

Para modelar (1) en función de s:

$$y \leq s \quad (3)$$

Por lo tanto: $y \leq 0 \Rightarrow y = 0$

que también satisface (2):

$$y \leq 1 \Rightarrow y = 0$$

Para modelar (2) en función de s no es suficiente una única restricción, puesto que se violaría (1)[1]. Así, la igualdad se modela a partir de:

$$y \leq x \quad (4)$$

$$y \geq x \quad (5)$$

La restricción (4) satisface (1):

$$y \leq 0$$

Sin embargo, la restricción (5) no satisface (1):

$$y \geq x$$

ya que el límite inferior de y no se debe acotar, y en esta restricción se expresa que coincide con el de x.

Por lo tanto, se reescribe para expresar una traslación en función de s de manera que el límite inferior del dominio de y no se modifique:

$$y \geq x + s - 1 \quad (6)$$

Así, cuando $s = 1$, la restricción queda como (5), y cuando $s = 0$, se satisface (1):

$$y \geq x - 1$$

ya que el mayor valor que puede tomar el lado derecho es 0 y, por lo tanto, no se acota y.

En resumen, $y = sx$ se modela con:

$y \leq s$	(3)
$y \leq x$	(4)
$y \geq x + s - 1$	(6)

Si $x, y \in [m, M]$, $s \in \{0,1\}$

$$y \leq m \cdot s \quad (3)$$

Demostración de la equivalencia semántica de {(0)} y {(3),(4),(6)}

1) Dominio de s: $s \in \{0,1\}$. Este caso está demostrado en el planteamiento.

2) Dominio de y: $y \in [0,1]$.

2.1) Para $y = 0 \Rightarrow s = 0 \vee x = 0$

$0 \leq s$ de (3) (7)

$0 \leq x$ de (4) (8)

$0 \geq x + s - 1$ de (6) (9)

[1] Nótese que para usar una restricción de igualdad que seleccione una variable del lado derecho, habría que usar una restricción no lineal, como la original $y = sx$.

Para $s = 0$ (7) se satisface, (9) queda $0 \geq x - 1$ (el límite superior de x no se acota: $x \leq 1$), y (8) no acota el límite inferior de x.
Para $s = 1$ (7) se satisface, (9) queda $0 \geq x \Rightarrow x = 0$, y (8) se satisface.
2.2) Para $y = 1 \Rightarrow s = 1 \wedge x = 1$
$1 \leq s \Rightarrow s = 1$ de (3)
$1 \leq x \Rightarrow x = 1$ de (4)
$1 \geq 1 + 1 - 1$ de (6) se satisface
2.3) Para $y = a, a \in (0,1) \Rightarrow s = 1 \wedge x = a$
$a \leq s$ de (3) (10)
$a \leq x$ de (4) (11)
$a \geq x + s - 1$ de (6) (12)
Para $s = 0$, (3) no se satisface.
Para $s = 1$ (12) queda $a \geq x$, que con (11) se expresa $x = a$.
3) Dominio de x: $x \in [0,1]$.
3.1) Para $x = 0 \Rightarrow y = 0, \forall s$
$y \leq s$ de (3) (13)
$y \leq 0 \Rightarrow y = 0$ de (4)
$y \geq s - 1$ de (6) (14)
Para $s = 0$ se satisfacen (13) ($0 \leq 0$) y (14) ($0 \geq -1$).
Para $s = 1$ se satisfacen (13) ($0 \leq 1$) y (14) ($0 \geq 0$).
3.2) Para $x = 1 \Rightarrow y = s$
$y \leq s$ de (3) (15)
$y \leq 1$ de (4) (16)
$y \geq s$ de (6) (17)
De (15) y (17) se deduce $y = s$. (16) no modifica el dominio de y.
3.3) Para $x = a, a \in (0,1) \Rightarrow (y = 0 \wedge s = 0) \vee (y = a \wedge s = 1)$
$y \leq s$ de (3) (18)
$y \leq a$ de (4) (19)
$y \geq a + s - 1$ de (6) (20)
Para $s = 0$, de (18) se deduce $y = 0$, y se satisfacen (19) ($0 \leq a$) y (20) ($0 \geq a - 1 \equiv a \leq 1$).
Para $s = 1$, se satisface (18) ($y \leq 1$), y de (19) ($y \leq a$) y (20) ($y \geq a$) se deduce $y = a$.

Modelo en OPL: prodrb1.prj y prodrb1.osc.

2.7. Producto de variables binarias

Se trata de modelar el producto $a = s_0 \cdots s_{n-1}$, donde todas las variables implicadas son binarias. El producto binario se puede especificar a partir de su tabla como:

$$a = \begin{cases} 1, si \sum_{i=0}^{n-1} s_i = n \\ 0, e.o.c. \end{cases}$$

O, lo que es lo mismo, mediante implicaciones:

$$a = 1 \Rightarrow \sum_{i=0}^{n-1} s_i = n$$

$$a = 0 \Rightarrow \sum_{i=0}^{n-1} s_i < n$$

No es posible representar la desigualdad estricta con precisión en el eje real; sin embargo, este caso corresponde a un eje discreto, ya que s_i son binarias y su suma discreta. Por lo tanto, se puede reescribir lo anterior de la manera siguiente:

$$a = 1 \Rightarrow \sum_{i=0}^{n-1} s_i \geq n - 0{,}5 \quad (1)$$

$$a = 0 \Rightarrow \sum_{i=0}^{n-1} s_i \leq n - 0{,}5 \quad (2)$$

Donde se han sustituido la igualdad y la desigualdad estricta por desigualdades no estrictas (añadiendo los necesarios factores de desplazamiento $\pm 0{,}5$ para mantener ciertas las relaciones.
Esto se puede modelar como:

$$\sum_{i=0}^{n-1} s_i \geq (n - 0{,}5)a$$

$$\sum_{i=0}^{n-1} s_i \leq (n - 0{,}5)(1 - a) + na$$

Así, para $a = 1$: $\sum_{i=0}^{n-1} s_i \geq n - 0{,}5$ y $\sum_{i=0}^{n-1} s_i \leq n$ y, para $a = 0$: $\sum_{i=0}^{n-1} s_i \geq 0$ y $\sum_{i=0}^{n-1} s_i \leq n - 0{,}5$, por lo que se cumplen las relaciones (1) y (2).

Modelo en OPL: prodbin.prj y prodbin.osc.

Aplicación:
Con el producto de variables binarias se modela la conjunción n-aria.

2.8. Producto de una variable binaria por una variable real o entera

Se trata de representar $y = sx$, donde x e y son reales o enteras y s es binaria.

2.8.1. Primer caso. Variable real positiva

$x \in [m, \ldots, M]$, $m \geq 0$, $dom(x) \subseteq dom(y)$, $s \in \{0,1\}$
Razonando como en 2.6.2:
$y \leq sM$
$y \leq x$
$y \geq x + (s - 1)M$
Así, si $s = 0$:
$y \leq 0$, $y \leq x$ se satisface, $y \geq x - M$ siempre será menor o igual que 0, por lo que $y = 0$.
Si $s = 1$:
$y \leq M$, que sólo acota por arriba a y, $y \leq x$ y $y \geq x$, por lo que $y = x$.

Modelo en OPL: prodrpb.mod.

2.8.2. Segundo caso. Variable real negativa

$x \in [m,\ldots,M]$, $m, M \leq 0$, $dom(x) \subseteq dom(y)$, $s \in \{0,1\}$

Razonando como en 2.6.2:

$y \geq sM$

$y \geq x$

$y \leq x + (s-1)M$

Así, si $s = 0$:

$y \geq 0$, $y \geq x$ se satisface con $y = 0$, $y \leq x - M$, $x - M$ siempre será mayor o igual que 0, por lo que $y = 0$.

Si $s = 1$:

$y \geq M$, que sólo acota por abajo a y, $y \leq x$ y $y \geq x$, por lo que $y = x$.

Modelo en OPL: prodrnb.mod.

2.8.3. Tercer caso. Variable real con dominio de número positivos y negativos

$x \in [-m,\ldots,M]$, $m \geq 0, M \geq 0$, $dom(x) \subseteq dom(y)$, $s \in \{0,1\}$

La variable real se descompone en una parte positiva y otra negativa. Después se aplica el razonamiento anterior a ambas.

$x = xp - xn$

$y = s \cdot xp - s \cdot xn = psxp - psxn$, donde $psxn$ y $psxn$ son dos nuevas variables que representan los productos $s \cdot xp$ y $s \cdot xn$ respectivamente.

$psxp \leq sM$

$psxp \leq xp$

$psxp \geq xp + (s-1)M$

$psxn \leq sm$

$psxn \leq xn$

$psxn \geq xn + (s-1)m$

Número de variables binarias añadidas: 1
Número de variables reales añadidas: 2+2.
Número de restricciones: 1+7.

Modelo en OPL: prodrb.prj.

2.9. Producto de variables enteras

Se trata de representar $z = x \cdot y$, donde x, y y z son enteras.

2.9.1. Primer planteamiento

El producto $z = x \cdot y$ puede expresarse como:

$$z = \sum_{i \in \{mx,\ldots,Mx\}} \sum_{j \in \{my,\ldots,My\}} \delta_x(i) \cdot \delta_y(j) \cdot i \cdot j$$

$$x = \sum_{i \in \{mx,\ldots,Mx\}} \delta_x(i) \cdot i$$

$$y = \sum_{j \in \{my,\ldots,My\}} \delta_y(j) \cdot j$$

$$\sum_{i\in\{mx,..,Mx\}} \delta_x(i) = 1$$
$$\sum_{j\in\{my,..,My\}} \delta_y(j) = 1$$

Siendo δ_x y δ_y variables binarias.

Los productos $\delta_{xy}(i,j) = \delta_x(i) \cdot \delta_y(j)$ se expresan con las siguientes restricciones (véase el apartado 2.7):

$$\delta_x(i) + \delta_y(j) \geq 1{,}5 \cdot \delta_{xy}(i,j)$$
$$\delta_x(i) + \delta_y(j) \leq 1{,}5 \cdot (1 - \delta_{xy}(i,j)) + 2 \cdot \delta_{xy}(i,j)$$

2.9.2. Segundo planteamiento

El producto $z = x \cdot y$ puede expresarse como:

$$z = \sum_{i\in\{mx,..,Mx\}} \delta_x(i) \cdot i \cdot y \qquad (1)$$
$$x = \sum_{i\in\{mx,..,Mx\}} \delta_x(i) \cdot i$$
$$\sum_{i\in\{mx,..,Mx\}} \delta_x(i) = 1$$

Siendo δ_x variables binarias.

Los productos $p(i) = \delta_x(i) \cdot y$ se expresan con las restricciones indicadas en los apartados 2.8.1 al 2.8.3, según sea el dominio de y.

Alternativamente, el producto $z = x \cdot y$ puede expresarse como:

$$z = \sum_{j\in\{my,..,My\}} \delta_y(j) \cdot j \cdot x \qquad (2)$$
$$y = \sum_{j\in\{my,..,My\}} \delta_y(j) \cdot j$$
$$\sum_{j\in\{my,..,My\}} \delta_y(j) = 1$$

Siendo δ_y variables binarias.

Los productos $p(j) = \delta_y(j) \cdot x$ se expresan con las restricciones indicadas en los apartados 2.8.1 al 2.8.3, según sea el dominio de x.

La elección de una alternativa u otra puede estar guiada por los siguientes criterios:

- (1) es mejor que (2) cuando el dominio de x incluye positivos y negativos y el dominio de y no, y viceversa.
- Si los dominios tanto de x como de y incluyen positivos y negativos, se elegirá (1) si la cardinalidad del dominio de x es menor que la del dominio de y (menor número de restricciones para modelar el producto de una variable entera por otra binaria), y viceversa.

Modelos en OPL:

- Entero positivo por entero positivo: prodepep.prj
- Entero positivo por entero: prodepe.prj
- Entero por entero: prodee.prj

2.10. Aproximación de funciones por tramos

2.10.1. Aproximación de funciones bidimensionales por tramos constantes

Dada una función $f(x)$, como la representada en la Figura 1-(a), $\tilde{f}(x)$ la aproxima por tramos como se representa en la Figura 1-(b).

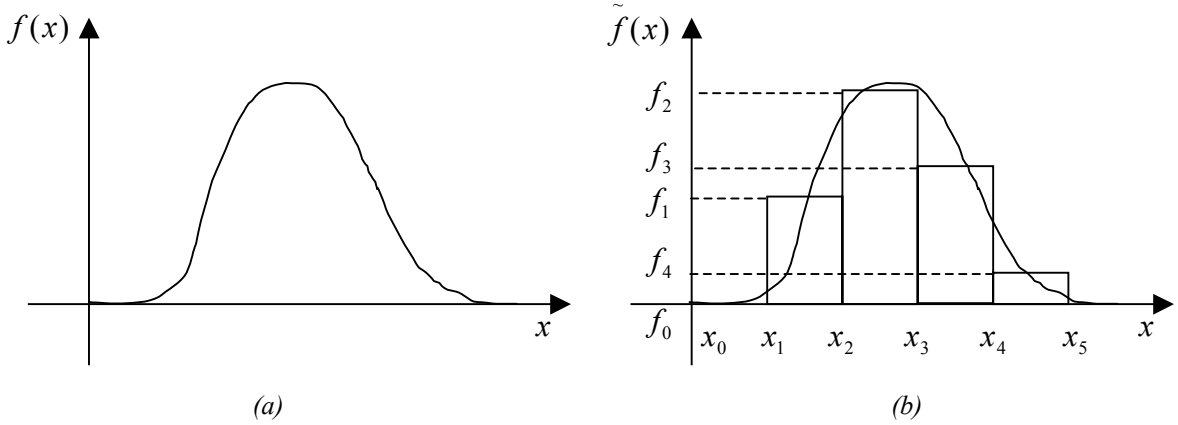


Figura 1. Representación de funciones por tramos constantes

Es decir, dada una función a discretizar $f(x)$, x_i los valores de x que definen los intervalos de discretización y f_i los valores constantes de $f(x)$ en el intervalo $[x_i, x_{i+1}]$:

$$\tilde{f}(x) = \begin{cases} f_0, si\ x \in [x_0, x_1) \\ f_1, si\ x \in [x_1, x_2) \\ \vdots \\ f_{n-1}, si\ x \in [x_{n-1}, x_n) \end{cases}$$

Primer planteamiento

Asignando una variable binaria s_i a cada intervalo $[x_i, x_{i+1}]$, se impone que sólo una pueda valer 1 y que la variable independiente x tenga un valor en el intervalo $[x_i, x_{i+1}]$ cuando $s_i = 1$. Así, el valor de $\tilde{f}(x)$ se calcula como:

$$\tilde{f}(x) = \sum_{i=0}^{n-1} s_i f_i$$

El resto de restricciones queda:

$$\sum_{i=0}^{n-1} s_i = 1$$

$$x \geq \sum_{i=0}^{n-1} x_i s_i$$

$$x \leq \sum_{i=0}^{n-1} x_{i+1} s_i$$

Es decir, se impone que para un $s_i = 1$: $x \geq x_i$ y $x \leq x_{i+1}$, que es lo mismo que $x \in [x_i, x_{i+1}]$. Por tanto: $\tilde{f}(x) = f_i$.

Nota: En los puntos x_i, la función $\tilde{f}(x)$ puede tomar tanto el valor f_i como el valor f_{i-1}.

Modelo en OPL: ftramoc.prj y ftramoc.osc.

Aplicación: Compresor con presión de entrada p_A y de salida p_B, definido por el flujo x como se muestra en la tabla:

x	p_A	p_B
$[0,75]$	0.0000	0.9980
$[75,150]$	0.0577	0.9985
$[150,225]$	0.2310	0.9999
$[225,300]$	0.5212	1.0024
$[300,375]$	0.9302	1.0059
$[375, M]$	1.0104	1.4609

Tanto p_A como p_B se pueden modelar por tramos constantes exactamente (por su propia definición $p_i(x) = \tilde{p}_i(x)$). Dado p_A modelado como se indica arriba, p_B basta con expresarlo con: $p_B(x) = \sum_{i=0}^{n-1} s_i p_i$, siendo s_i las mismas que las utilizadas en el modelo de p_A. El número de restricciones es 3 (para s_i)+2 (para p_A y p_B) = 5, y el de variables 5 (para s_i) + 3 (originales)=8.

Segundo planteamiento

El planteamiento de la resolución se hace componiendo funciones escalón 01 [Sae00b], de la siguiente forma:

$\tilde{f}(x) = f_0 f^{01}{}_{x_0}(x) + (f_1 f^{01}{}_{x_1}(x) - f_0 f^{01}{}_{x_1}(x)) + \cdots + (f_{n-1} f^{01}{}_{x_{n-1}}(x) - f_{n-2} f^{01}{}_{x_n}(x))$ dado que, para un x determinado en el intervalo $[x_i, x_{i+1}]$, $f^{01}{}_{x_i}(x) = 1, i \in \{0,\ldots,i\}$ y, por tanto, hay que eliminar de $\tilde{f}(x)$ las contribuciones de los términos en los que aparecen las $f^{01}{}_{x_i}(x)$ anteriores a x. Resumiendo:

$\tilde{f}(x) = \sum_{i=0}^{n-1} (f_i f^{01}{}_{x_i}(x) - \sum_{j=0}^{i-1} f_j f^{01}{}_{x_i}(x))$, con las restricciones de las funciones escalón 01 expresadas como:

$$\left.\begin{array}{l} x \leq x_i(1 - f^{01}{}_{x_i}(x)) + M f^{01}{}_{x_i}(x) \\ x \geq m(1 - f^{01}{}_{x_i}(x)) + x_i f^{01}{}_{x_i}(x) \end{array}\right\} \forall i \in \{0,\ldots,n-1\}$$

Hay, por tanto, $2n+1$ restricciones y n variables binarias (sin contar la que podría ser necesaria para $\tilde{f}(x)$).

Notas:

- Las funciones escalón $f^{01}{}_{x_i}(x)$ están multivaloradas en x_i, por lo que en x_i el resultado de $\tilde{f}(x)$ puede ser tanto el valor del tramo anterior como el del posterior.

- $f_0 > m \Rightarrow \tilde{f}(x) = 0, x \in [m, f_0]$
- $\tilde{f}(x) = f_{n-1}, x \in [x_n, M]$

Modelo en OPL: ftramoc1.prj y ftramoc1.osc.

Tercer planteamiento

El planteamiento de la resolución se hace componiendo funciones pulso cuadrado [Sae00b], de la siguiente forma, donde $f(x)$ es la función a discretizar, x_i son los valores de x que definen los intervalos de discretización, f_i son los valores constantes de $f(x)$ en el intervalo $[x_i, x_{i+1}]$:

$$\tilde{f}(x) = \begin{cases} f_0, si\ x \in [x_0, x_1) \\ f_1, si\ x \in [x_1, x_2) \\ \vdots \\ f_{n-1}, si\ x \in [x_{n-1}, x_n) \end{cases}$$

$\tilde{f}(x)$ se podría especificar como:

$\tilde{f}(x) = f_0 pc_{x_0,x_1}(x) + f_1 pc_{x_1,x_2}(x) + \cdots + f_{n-1} pc_{x_{n-1},x_n}(x)$, con las restricciones de las funciones pulso cuadrado definidas por:

$$\left.\begin{array}{l} x \geq m s_i + x_i(1 - s_i) \\ x \leq x_i s_i + M(1 - s_i) \\ x \leq x_{i+1}(s_i + pc_{x_i,x_{i+1}}(x)) + M(1 - s_i - pc_{x_i,x_{i+1}}(x)) \\ x \geq m(s_i + pc_{x_i,x_{i+1}}(x)) + x_{i+1}(1 - s_i - pc_{x_i,x_{i+1}}(x)) \\ pc_{x_i,x_{i+1}}(x) + s_i \leq 1 \end{array}\right\} \forall i \in \{0, \ldots, n-1\}$$

donde

Nota: En los puntos $x_i, i \in \{1, \ldots, n-1\}$, $\tilde{f}(x_i)$ puede tomar los valores 0, f_i o $f_i + f_{i+1}$, en lugar de $\tilde{f}(x_i) = f_i$, dado que $pc_{x_i,x_{i+1}}(x)$ y $pc_{x_{i+1},x_{i+2}}(x)$ pueden tomar cualquier combinación de los valores 0 o 1 en x_i (Véase [Sae00]).

2.10.2. Aproximación de funciones bidimensionales por tramos lineales

Dada una función $f(x)$, como la representada en la Figura 2-(a), $\tilde{f}(x)$ la aproxima por tramos como se representa en la Figura 2-(b).

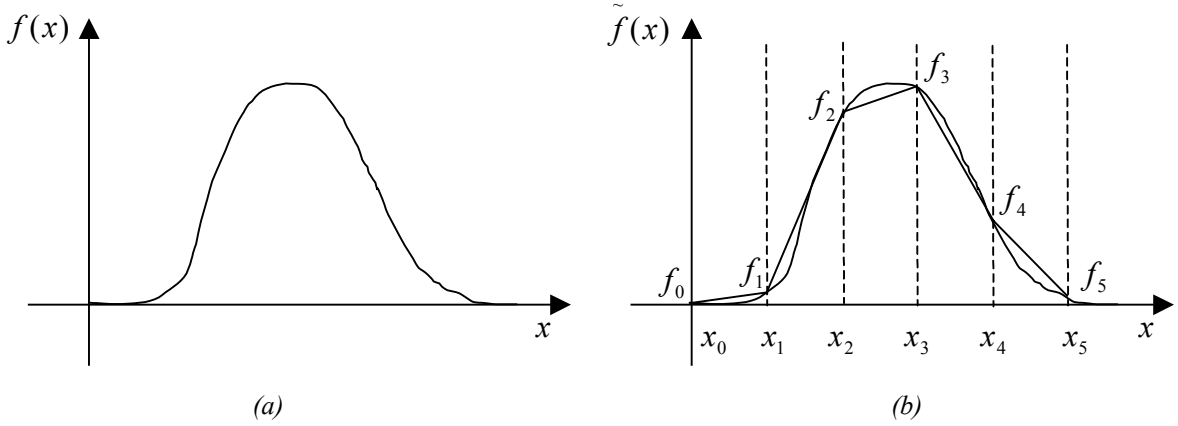


Figura 2. Representación de funciones por tramos lineales

Es decir, dada una función a discretizar $f(x)$, x_i los valores de x que definen los intervalos de discretización y f_i los valores de $\tilde{f}(x_i)$, $\tilde{f}(x)$ se define como:

$$\tilde{f}(x) = \begin{cases} a_0 + m_0 x, si\ x \in [x_0, x_1] \\ a_1 + m_1 x, si\ x \in [x_1, x_2] \\ \vdots \\ a_{n-1} + m_{n-1} x, si\ x \in [x_{n-1}, x_n] \end{cases}$$

donde $a_i + m_i x$ es la recta que une los puntos x_i y x_{i+1}. Por tanto:

$$m_i = \frac{f_{i+1} - f_i}{x_{i+1} - x_i}$$

$$a_i = f_i - m_i x_i$$

Asignando una variable binaria s_i a cada intervalo $[x_i, x_{i+1}]$, se impone que sólo una pueda valer 1 y que la variable independiente x tenga un valor en el intervalo $[x_i, x_{i+1}]$ cuando $s_i = 1$, que se modela con:

$$\sum_{i=0}^{n-1} s_i = 1$$

$$x \geq \sum_{i=0}^{n-1} x_i s_i$$

$$x \leq \sum_{i=0}^{n-1} x_{i+1} s_i$$

Así, el valor de $\tilde{f}(x)$ se calcula como:

$$\tilde{f}(x) = \sum_{i=0}^{n-1} (a_i + m_i x) s_i$$

Esta expresión involucra un término no lineal ($m_i x s_i$) para el que se aplica la solución del apartado 2.6. Para ello es necesario un cambio de variable para expresar el producto $x s_i$ como el producto de una

variable real en el intervalo $[0,1]$ por una variable binaria 0-1. Si $x \in [m, M]$, entonces hacemos el cambio de variable $x = x_{01}(M-m)+m$ y $\tilde{f}$ queda expresada por:

$$\tilde{f}(x_{01}) = \sum_{i=0}^{n-1}(s_i(a_i + m_i m) + k_i s_i x_{01}) = \sum_{i=0}^{n-1}(s_i(a_i + m_i m) + k_i y_i),$$

siendo $k_i = m_i(M-m)$ $y_i = s_i x_{01}$, por lo que desaparece el término no lineal y se sustituye por las inecuaciones:

$$y_i \le s_i$$
$$y_i \le x_{01}$$
$$y_i \ge x_{01} + s_i - 1$$

En resumen, $\tilde{f}(x)$ se modela con:

Restricciones:

$$x = x_{01}(M-m)+m$$
$$\tilde{f}(x_{01}) = \sum_{i=0}^{n-1}(s_i(a_i + m_i m) + k_i y_i)$$
$$y_i \le s_i$$
$$y_i \le x_{01}$$
$$y_i \ge x_{01} + s_i - 1$$
$$\sum_{i=0}^{n-1} s_i = 1$$
$$x \ge \sum_{i=0}^{n-1} x_i s_i$$
$$x \le \sum_{i=0}^{n-1} x_{i+1} s_i$$

Datos:

$$m_i = \frac{f_{i+1} - f_i}{x_{i+1} - x_i}$$
$$a_i = f_i - m_i x_i$$
$$k_i = m_i(M-m)$$

Son necesarias ocho restricciones $n-1$ (por s_i) + $n-1$ (por y_i) +1 (por x_{01}) $= 2n-1$ variables.

Modelo en OPL: ftramo.prj y ftramo.osc.

Aplicación 1: Gasoducto con presiones en un extremo p_A y en otro p_B, definidas por la ecuación lineal:

$$p_A(p_B, x) = m_i(x)p_B + a_i(x) \quad (1)$$

estando los coeficientes m_i y a_i definidos en función del flujo x como se muestra en la siguiente tabla:

i	x	m_i	a_i	$p_A = m_i p_B + a_i$
0	$[0,75]$	0.9980	0.0000	$p_A = 0.9980 p_B + 0.0000$

1	$[75,150]$	0.9985	0.0577	$p_A = 0.9985 p_B + 0.0577$
2	$[150,225]$	0.9999	0.2310	$p_A = 0.9999 p_B + 0.2310$
3	$[225,300]$	1.0024	0.5212	$p_A = 1.0024 p_B + 0.5212$
4	$[300,375]$	1.0059	0.9302	$p_A = 1.0059 p_B + 0.9302$
5	$[375, M]$	1.0104	1.4609	$p_A = 1.0104 p_B + 1.4609$

Cuando el flujo es negativo, los intervalos de flujo son simétricos con respecto a 0 en la recta real, y los papeles de p_A y de p_B se intercambian, por lo que los coeficientes m_i y a_i se calculan como:

$p_B = \dfrac{p_A(p_B, x) - a_i(x)}{m_i(x)}$, despejando p_B de (1) y, así:

$$p_B(p_A, x) = m'_i(x) p_A + a'_i(x)$$

$$m'_i(x) = \frac{1}{m_i(x)},\ a'_i(x) = -\frac{a_i(x)}{m_i(x)} \qquad (2)$$

La función $p_A(p_B, x)$ se puede representar como:

$$p_A(p_B, x) = \sum_{i=0}^{n-1} (a_i(x) + m_i(x) \cdot p_B) \cdot s_i(x) \qquad (3)$$

En este problema hay tres variables involucradas, pero se puede aplicar la solución de la aproximación de funciones por tramos haciendo el cambio de variable p_B por x en las funciones a_i, m_i y s_i, con lo que nos queda la ecuación anterior (3). Los coeficientes a_i y m_i no se calculan como se indica en la sección de datos anterior, sino que se obtienen de la tabla de arriba y de las expresiones de los coeficientes (2) para los flujos negativos. Ahora, la función por tramos será en general discontinua (a diferencia de la suposición que se hacía en este apartado para el cálculo de coeficientes), como muestra la Figura 3-a:

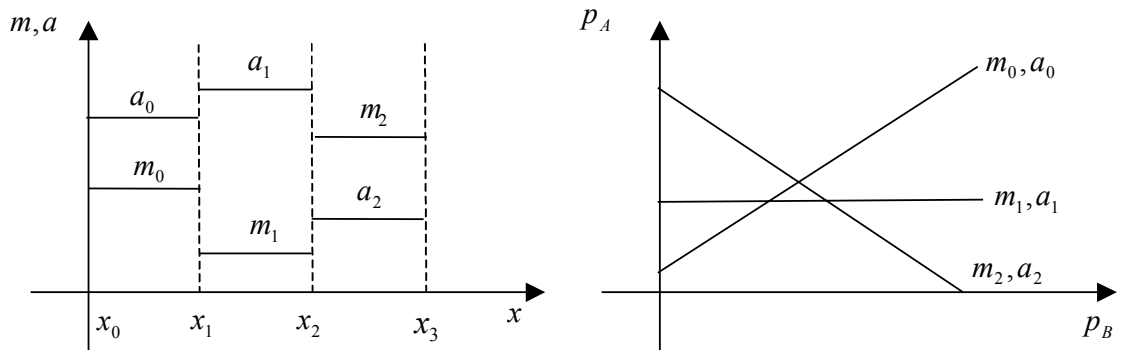


Figura 3 Relación de presiones en los extremos de un gasoducto

Modelo en OPL: gaso.prj y gaso.osc.

Aplicación 2: Gasoducto con presiones en un extremo p_1 y en otro p_2, definidas por la ecuación lineal:

$$p_1(p_2, f) = a_i(f) \cdot p_2 + b_i(f) + c_i(f) \cdot f \qquad (1)$$

estando los coeficientes a_i, b_i y c_i definidos en función del flujo f como se muestra en la siguiente tabla:

i	f	a_i	b_i	c_i	$p_1 = a_i p_2 + b_i + c_i f$
0	$[0,75]$	0.9980	0.0000	1	$p_1 = 0.9980 p_2 + 0.0000 + f$
1	$[75,150]$	0.9985	0.0577	1	$p_1 = 0.9985 p_2 + 0.0577 + f$
2	$[150,225]$	0.9999	0.2310	1	$p_1 = 0.9999 p_2 + 0.2310 + f$
3	$[225,300]$	1.0024	0.5212	1	$p_1 = 1.0024 p_2 + 0.5212 + f$
4	$[300,375]$	1.0059	0.9302	1	$p_1 = 1.0059 p_2 + 0.9302 + f$
5	$[375, M]$	1.0104	1.4609	1	$p_1 = 1.0104 p_2 + 1.4609 + f$

Los intervalos de flujo se definen explícitamente.

La función $p_1(p_2, f)$ se puede representar como:

$$p_1(p_2, f) = \sum_{i=0}^{n-1} (a_i(f) \cdot p_2 + b_i(f) + c_i(f) \cdot f) \cdot s_i(f) \quad (3)$$

En este problema se puede aplicar también la solución de la aproximación de funciones por tramos constantes modificando $\tilde{f}(x)$ como se muestra a continuación:

$$\tilde{f}(x, y) = \sum_{i=0}^{n-1} (b_i + c_i x + a_i y) s_i = \sum_{i=0}^{n-1} b_i s_i + c_i x s_i + a_i y s_i$$

Los coeficientes a_i, b_i y c_i se obtienen de la tabla anterior. La función por tramos será en general discontinua. A diferencia de la aplicación anterior, en este caso se definen planos por cada tramo en lugar de rectas.

Se ha añadido una nueva variable (y) que representará a la presión en el punto 2 (p_2), $\tilde{f}$ representa a la presión en el punto 1 (p_1), y el flujo está representado por x.

Esta expresión involucra dos términos no lineales ($c_i x s_i$ y $c_i y s_i$) para los que se aplica la solución del apartado 2.6. Para ello es necesario un cambio de variable para expresar los productos $x s_i$ e $y s_i$ como el producto de una variable real en el intervalo $[0,1]$ por una variable binaria 0-1. Si $x \in [x_m, x_M]$ e $y \in [y_m, y_M]$, entonces hacemos los cambios de variable $x = x_{01}(x_M - x_m) + x_m$ y $y = y_{01}(y_M - y_m) + y_m$, y $\tilde{f}$ queda expresada por:

$$x = x_{01}(x_M - x_m) + x_m$$
$$y = y_{01}(y_M - y_m) + y_m$$
$$\tilde{f} = \sum_{i=0}^{n-1} (s_i(b_i + c_i x_m + a_i y_m) + k_i u_i + l_i v_i)$$
$$u_i \le s_i$$
$$u_i \le x_{01}$$
$$u_i \ge x_{01} + s_i - 1$$
$$v_i \le s_i$$
$$v_i \le y_{01}$$
$$v_i \ge y_{01} + s_i - 1$$

$$\sum_{i=0}^{n-1} s_i = 1$$
$$x \geq \sum_{i=0}^{n-1} x_i s_i$$
$$x \leq \sum_{i=0}^{n-1} x_{i+1} s_i$$

con $u_i = x_{01} s_i$ y $v_i = y_{01} s_i$.

$x \in [x_m, x_M]$ (variable limitada explícitamente en las restricciones anteriores)

$y \in [y_m, y_M]$ (variable no limitada explícitamente en las restricciones anteriores)

Por tanto, se pueden añadir las siguientes restricciones para la variable y:

$$y \geq y_m$$
$$y \leq y_M$$

Volviendo a la nomenclatura del problema:

$\tilde{f} = p_1$
$x = f$
$y = p_2$
$f \in [f_m, f_M]$
$p_2 \in [p_{2m}, p_{2M}]$

Modelo en OPL: gaso2.prj y gaso2.osc.

2.10.3. Aproximación de funciones tridimensionales por planos constantes

Dada una función $f(x,y)$, $\tilde{f}(x,y)$ la aproxima por planos (rectangulares) constantes discretizando los ejes x e y.

Es decir, dada una función a discretizar $f(x,y)$, x_i e y_i los valores de x e y que definen los rectángulos en el plano xy de discretización y f_{ij} los valores constantes de $f(x,y)$ en el rectángulo definido por $[x_i, x_{i+1}]$ y $[y_j, y_{j+1}]$:

$$\tilde{f}(x,y) = \begin{cases} f_{00}, si\ x \in [x_0, x_1), y \in [y_0, y_1) \\ f_{01}, si\ x \in [x_0, x_1), y \in [y_1, y_2) \\ \vdots \\ f_{n-1,m-1}, si\ x \in [x_{n-1}, x_n), y \in [y_{m-1}, y_m) \end{cases}$$

Asignando una variable binaria δ_{x_i} a cada intervalo $[x_i, x_{i+1}]$, y otra δ_{y_j} a cada intervalo $[y_j, y_{j+1}]$, se impone que sólo una de las δ_{x_i} pueda valer 1 y sólo una de las δ_{x_i} pueda valer 1. Se impone también que la variable independiente x tenga un valor en el intervalo $[x_i, x_{i+1}]$ cuando $\delta_{x_i} = 1$, y que la

variable independiente y tenga un valor en el intervalo $\lfloor y_j, y_{j+1} \rfloor$ cuando $\delta_{y_j} = 1$. Así, el valor de $\tilde{f}(x,y)$ se calcula como:

$$\tilde{f}(x,y) = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \delta_{x_i} \cdot \delta_{y_j} \cdot f_{ij}$$

El resto de restricciones queda:

$$\sum_{i=0}^{n-1} \delta_{x_i} = 1$$

$$\sum_{j=0}^{m-1} \delta_{y_j} = 1$$

$$x \geq \sum_{i=0}^{n-1} x_i \delta_{x_i}$$

$$x \leq \sum_{i=0}^{n-1} x_{i+1} \delta_{x_i}$$

$$y \geq \sum_{i=0}^{m-1} y_i \delta_{y_i}$$

$$y \leq \sum_{i=0}^{m-1} y_{i+1} \delta_{y_i}$$

Con el producto $\delta_{xy_{ij}} = \delta_{x_i} \cdot \delta_{y_j}$ expresado con las siguientes restricciones:

$$\delta_{x_i} + \delta_{y_j} \geq 1{,}5 \cdot \delta_{xy_{ij}}$$

$$\delta_{x_i} + \delta_{y_j} \leq 1{,}5 \cdot (1 - \delta_{xy_{ij}}) + 2 \cdot \delta_{xy_{ij}}$$

Es decir, se impone que para un par $(\delta_{x_i}, \delta_{y_j}) = (1,1)$: $x \geq x_i$, $x \leq x_{i+1}$, $y \geq y_j$, $y \leq y_{j+1}$, que es lo mismo que $x \in [x_i, x_{i+1}]$, $y \in \lfloor y_j, y_{j+1} \rfloor$. Por tanto: $\tilde{f}(x,y) = f_{ij}$.

Nota: En los puntos (x_i, y_j), la función $\tilde{f}(x,y)$ puede tomar los valores f_{ij}, $f_{i-1,j}$, $f_{i,j-1}$ o $f_{i-1,j-1}$. Para evitarlo, se puede tomar el intervalo cerrado con:

$$x \leq \sum_{i=0}^{n-1} x_{i+1} \delta_{x_i} - \varepsilon$$

$$y \leq \sum_{i=0}^{m-1} y_{i+1} \delta_{y_i} - \varepsilon$$

Siendo ε un valor suficientemente pequeño.

Modelo en OPL: ftramoc.prj y ftramoc.osc.

Aplicación: Compresor.

2.11. Módulo

$y = |x|,\ x \in [-m,\ldots,M],\ m \geq 0, M \geq 0,\ dom(y) \supseteq \{0..max(m,M)\}$

2.11.1. Primer planteamiento

La función $y = |x|$ se puede expresar con las siguientes restricciones:

$y = x \cdot s - x \cdot (1-s) = 2 \cdot s \cdot x - x$ (1)

$y \geq 0$ (2)

(1) involucra un producto, que se puede reescribir como:

$y = 2 \cdot psx - x$, donde psx es una nueva variable que representa el producto $x \cdot s$

$x = xp - xn$

$psx = s \cdot xp - s \cdot xn = psxp - psxn$, donde $psxn$ y $psxn$ son dos nuevas variables que representan los productos $s \cdot xp$ y $s \cdot xn$ respectivamente.

$psxp \leq s \cdot M$

$psxp \leq xp$

$psxp \geq xp + (s-1) \cdot M$

$psxn \leq s \cdot m$

$psxn \leq xn$

$psxn \geq xn + (s-1) \cdot m$

(2) se puede eliminar si el dominio de y está restringido a números positivos.

Modelo en OPL: modulo.prj y modulo.osc.

2.11.2. Segundo planteamiento

La función $y = |x|$ se puede expresar descomponiendo x en su parte positiva xp y su parte negativa xn:

$x = xp - xn$

Por lo que $y = xp + xn$, ya que, a lo sumo, xp o xn será distinto de cero.

$x = xp - xn$

$xp \leq s \cdot M$

$xn \leq (1-s) \cdot M$

Modelo en OPL: modulo2.prj y modulo2.osc.

2.12. Signo

$s = \text{sgn}(x),\ x \in [-m,\ldots,M],\ m \geq 0, M \geq 0,\ s \in \{0,1\}$

La función $s = \text{sgn}(x)$ se puede expresar aplicando un razonamiento similar a la función módulo:

$x = xp - xn$

$xp \leq s \cdot M$

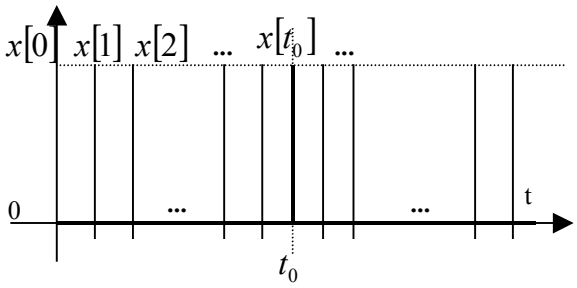
$xn \leq (1-s) \cdot M$

Modelo en OPL: signo.prj y signo.osc.

3. Representación de funciones temporales discretas en programación lineal mixta

3.1. Función intervalo temporal

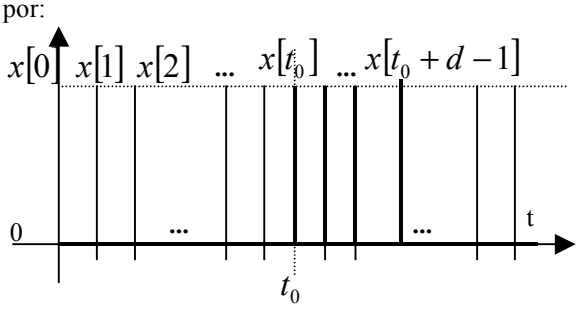
Dado un instante inicial de tiempo (t) en un plano discreto y un número de unidades de tiempo para definir la anchura del intervalo (d), se trata de representar el intervalo de tiempo [t,t+d].
Se elige como representación de la función delta de Kronecker en el plano discreto un vector $\bar{x} =< x[0]\ldots x[n-1] >$ de variables binarias cuyo único 1 representa el instante inicial de tiempo.



Donde $x[i] = \begin{cases} 1, si\ i = t_0 \\ 0, e.o.c. \end{cases}$, y, por tanto: $\sum_{i=0}^{n-1} x[i] = 1$.
Se trata de generar un vector definido por:

$$v[i] = \begin{cases} 1, \forall i \in [t_0, t_0 + d - 1] \\ 0, e.o.c. \end{cases}$$

representado gráficamente por:



3.1.1. Primer planteamiento

Función intervalo temporal para n<<

$\bar{x}$ se puede interpretar como una codificación binaria de un número decimal al que se le pueden añadir las potencias de 2 adyacentes que sean necesarias para cubrir la anchura *d*. Así, se calcula un nuevo vector binario cuyo valor decimal es $2^0 + \cdots + 2^{d-1} = 2^d - 1$ [2] veces el valor decimal de $\bar{x}$.
a) Si el vector $\bar{v}$ tiene $n + d - 1$ componentes para alojar el intervalo de anchura d para cualquier $t_0 \in \{0, \ldots, n-1\}$, se debe cumplir:

[2] Suma de la serie geométrica: $s = \frac{a_1(r^n - 1)}{r - 1}$.

$$(2^d - 1)\sum_{i=0}^{n-1} x[i] \cdot 2^i = \sum_{i=0}^{n+d-2} v[i] \cdot 2^i$$

Modelo en OPL: intervaloa.mod.

b) Si el vector $\overline{v}$ tiene n componentes, asumiendo que se cumple $t_0 + d - 1 \leq n - 1$ (el intervalo de anchura d se puede alojar en el intervalo $[0,\ldots,n-1]$), se debe cumplir:

$$(2^d - 1)\sum_{i=0}^{n-1} x[i] \cdot 2^i = \sum_{i=0}^{n-1} v[i] \cdot 2^i$$

$$\sum_{i=0}^{n-d} x[i] = 1$$

Modelo en OPL: intervalob.mod.

Función intervalo temporal para n>>

a) El vector $\overline{v}$ tiene $n + d - 1$ componentes para alojar el intervalo de anchura d para cualquier $t_0 \in \{0,\ldots,n-1\}$.

Cuando n es lo suficientemente grande se producirá desbordamiento de la representación de los números calculados con la expresión del apartado anterior. Para resolver el problema se divide el cómputo por tramos. El caso básico es dividirlo en dos tramos, de manera que si anteriormente:

$$x = \sum_{i=0}^{n-1} x[i] \cdot 2^i$$

$$v = \sum_{i=0}^{n+d-2} v[i] \cdot 2^i$$

$$(2^d - 1) \cdot x = v$$

Ahora se divide:

$$x_1 = \sum_{i=0}^{s-1} x[i] \cdot 2^i$$

$$x_2 = \sum_{i=0}^{n-s-1} x[i+s] \cdot 2^i$$

Se añaden los vectores $\overline{v_1}$ y $\overline{v_2}$:

$$v_1 = \sum_{i=0}^{s+d-2} v_1[i] \cdot 2^i$$

$$v_2 = \sum_{i=0}^{n-s+d-2} v_2[i] \cdot 2^{i-s}$$

De manera que se debe cumplir:

$$(2^d - 1) \cdot x_1 = v_1$$

$$(2^d - 1) \cdot x_2 = v_2$$

Notas:

1. Las dos ecuaciones anteriores son equivalentes a $(2^d - 1) \cdot x = v$, dado que v_1 y v_2 nunca son simultáneamente distintas de cero.
2. No es necesario añadir los vectores $\overline{x_1}$ y $\overline{x_2}$ porque son segmentos disjuntos de $\overline{x}$.

Por lo tanto:

$$(2^d - 1)\sum_{i=0}^{s-1} x[i] \cdot 2^i = \sum_{i=0}^{s+d-2} v_1[i] \cdot 2^i$$

$$(2^d - 1)\sum_{i=0}^{n-s-1} x[i+s] \cdot 2^i = \sum_{i=0}^{n-s+d-2} v_2[i] \cdot 2^i$$

$$v[i] = \begin{cases} v_1[i], i \in \{0,\ldots,s-1\} \\ v_1[i] + v_2[i-s], i \in \{s,\ldots,s+d-2\} \\ v_2[i-s], i \in \{s+d-1,\ldots,n+d-2\} \end{cases}$$

Que es lo mismo que decir:

$$\overline{v} = < v_2[n-s+d-2],\ldots,v_2[d-1], v_2[d-2] + v_1[s+d-2],\ldots,v_2[0] + v_1[s], v_1[s-1]\ldots,v_1[0] >$$

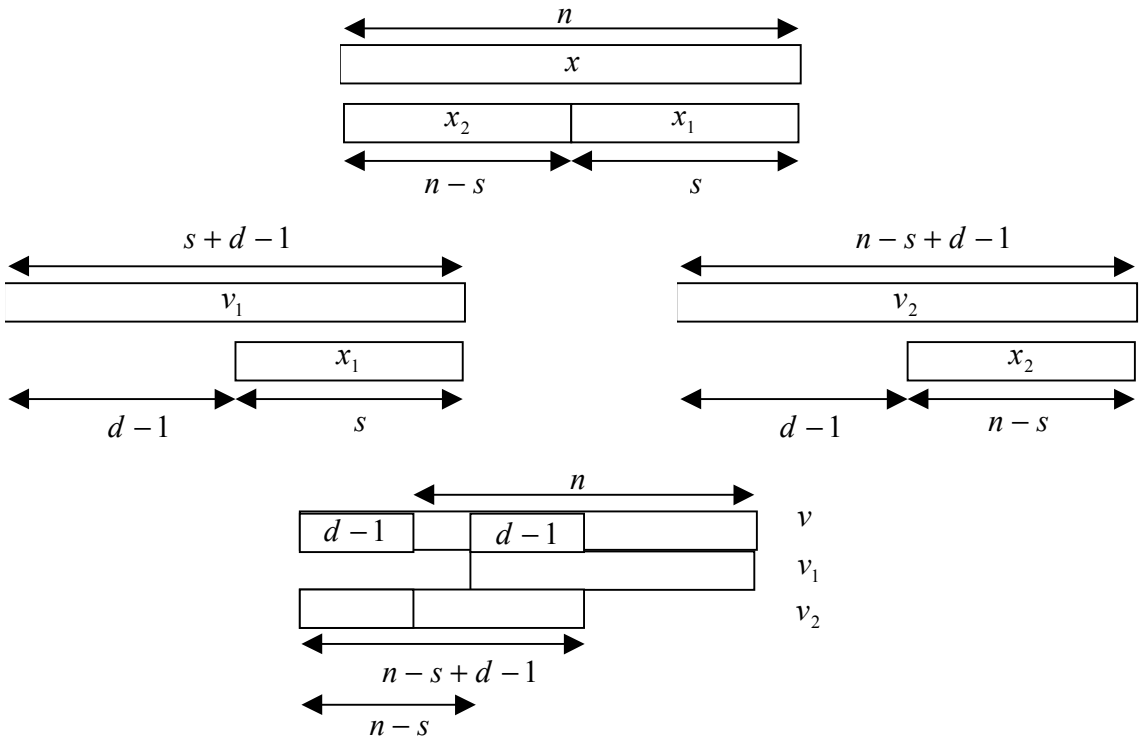
O también:

$$v[i] = v_1[i], i \in \{0,\ldots,s-1\}$$

$$v[i+s] = v_1[i+s] + v_2[i], i \in \{0,\ldots,d-2\}$$

$$v[i+s+d-1] = v_2[i+d-1], i \in \{0,\ldots,n-s-1\}$$

Las siguientes figuras ilustran los vectores usados.



Modelo en OPL: intervalo2a.mod.

b) El vector $\overline{v}$ tiene n componentes, asumiendo que se cumple $t_0 + d - 1 \leq n - 1$ (el intervalo de anchura d se puede alojar en el intervalo $[0,\ldots,n-1]$).

Como antes:

$$x = \sum_{i=0}^{n-1} x[i] \cdot 2^i$$

$$v = \sum_{i=0}^{n-1} v[i] \cdot 2^i$$

$$(2^d - 1) \cdot x = v$$

Ahora se divide:

$$x_1 = \sum_{i=0}^{s-1} x[i] \cdot 2^i$$

$$x_2 = \sum_{i=0}^{n-s-1} x[i+s] \cdot 2^i$$

Se añaden los vectores $\overline{v_1}$ y $\overline{v_2}$ ($\overline{v_1}$ no puede ser mayor que $\overline{x}$):

$$v_1 = \sum_{i=0}^{min(s+d-2,n-1)} v_1[i] \cdot 2^i$$

$$v_2 = \sum_{i=0}^{n-s-1} v_2[i] \cdot 2^{i-s}$$

De manera que se debe cumplir:

$$(2^d - 1) \cdot x_1 = v_1$$

$$(2^d - 1) \cdot x_2 = v_2$$

Notas:

1. Las dos ecuaciones anteriores son equivalentes a $(2^d - 1) \cdot x = v$, dado que v_1 y v_2 nunca son simultáneamente distintas de cero.

2. No es necesario añadir los vectores $\overline{x_1}$ y $\overline{x_2}$ porque son segmentos disjuntos de $\overline{x}$.

Por lo tanto:

$$(2^d - 1) \sum_{i=0}^{s-1} x[i] \cdot 2^i = \sum_{i=0}^{min(s+d-2,n-1)} v_1[i] \cdot 2^i$$

$$(2^d - 1) \sum_{i=0}^{n-s-1} x[i+s] \cdot 2^i = \sum_{i=0}^{n-s-1} v_2[i] \cdot 2^i$$

$$v[i] = \begin{cases} v_1[i], i \in \{0,\ldots,s-1\} \\ v_1[i] + v_2[i-s], i \in \{s,\ldots,min(s+d-2,n-1)\} \\ v_2[i-s], i \in \{min(s+d-1,n-1),\ldots,n+d-2\} \end{cases}$$

Que es lo mismo que decir:

$$\overline{v} = < v_2[n-s-1],\ldots,v_2[d-1], v_2[d-2] + v_1[min(s+d-2,n-1)],\ldots,v_2[0] + v_1[s], v_1[s-1]\ldots,v_1[0] >$$

O también:

$$v[i] = v_1[i], i \in \{0,\ldots,s-1\}$$

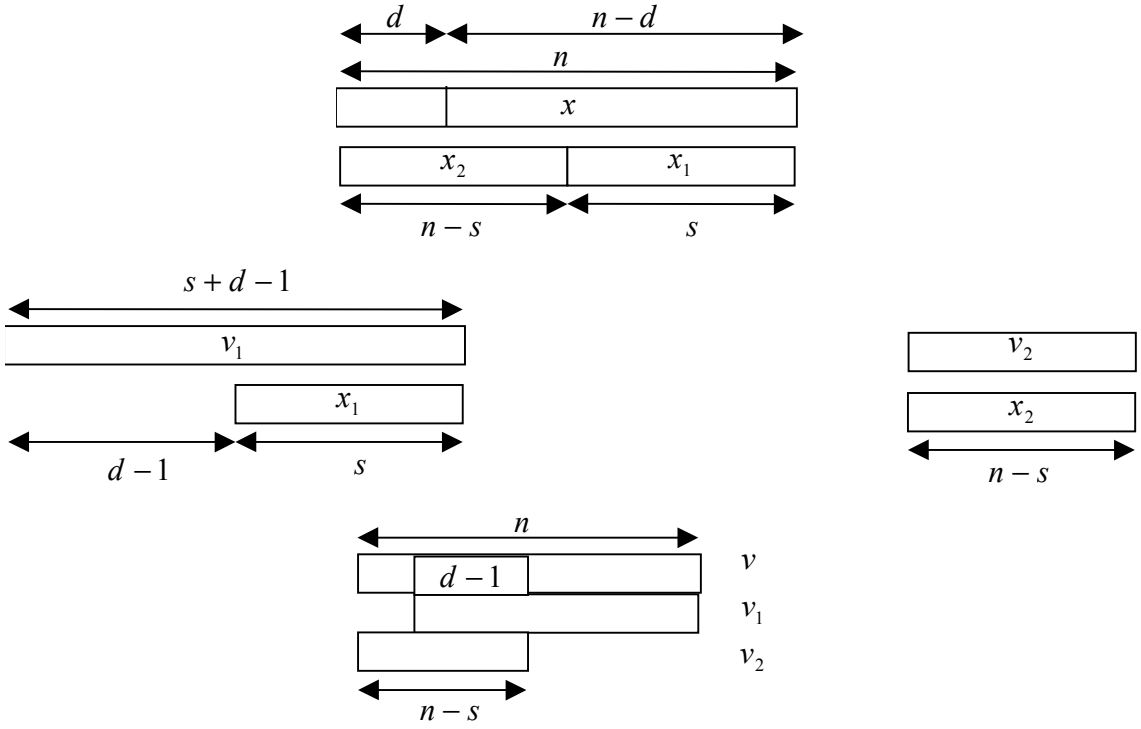
$$v[i+s] = v_1[i+s] + v_2[i], i \in \{0,\ldots,min(s+d-2,n-1)-s\}$$

$$v[i+s+d-1] = v_2[i+d-1], i \in \{0,\ldots,n-s-d\}$$

Dado que el intervalo de anchura d se debe alojar en $\overline{x}$:

$$\sum_{i=0}^{n-d} x[i] = 1$$

$x[i]=0, \forall i \in \{n-d+1,\ldots,n-1\}$ (Véase el apartado Relaciones de igualdad y desigualdad)
Las siguientes figuras ilustran los vectores usados.



Modelo en OPL: intervalo2b.mod.

3.1.2. Segundo planteamiento

a) Vector con índice de componentes $0,\ldots,n-1$.
Es necesario imponer que, dado un $x[i]=1$ entonces $v[i+j]=1, \forall j \in \{0,\ldots,d-1\}$. Es decir, la equivalencia:
$x[i]=1 \Leftrightarrow v[i+j]=1, \forall j \in \{0,\ldots,d-1\}, \forall i \in \{0,\ldots,n-d\}$
El valor de los d componentes de v a partir de $v[i]$ dependen del valor de $x[i]$ y, por tanto, esto se puede expresar de la siguiente forma:
$v[i+j] \geq x[i], \forall j \in \{0,\ldots,d-1\}, \forall i \in \{0,\ldots,n-d\}$
$\sum_{i=0}^{n-d} x[i] = 1$
$x[i]=0, \forall i \in \{n-d+1,\ldots,n-1\}$
$\sum_{i=0}^{n-1} v[i] = d$
ya que para un $x[i]=1$ se está imponiendo $v[i+j] \geq 1$, es decir, $v[i+j]=1$, al ser v un vector binario, y para $x[i]=0$ no se impone nada sobre v ($v[i+j] \geq 0$).

Modelo en OPL: intervaloc.mod, intervaloc.osc

b) Vector con índice de componentes $1,\ldots,n$.
Es necesario imponer que, dado un $x[i]=1$ entonces $v[i+j]=1, \forall j \in \{0,\ldots,d-1\}$. Es decir, la equivalencia:

$$x[i] = 1 \Leftrightarrow v[i+j] = 1, \forall j \in \{0,\ldots,d-1\}, \forall i \in \{1,\ldots,n-d+1\}$$

El valor de los d componentes de v a partir de $v[i]$ dependen del valor de $x[i]$ y, por tanto, esto se puede expresar de la siguiente forma:

$$v[i+j] \geq x[i], \forall j \in \{0,\ldots,d-1\}, \forall i \in \{1,\ldots,n-d+1\}$$

$$\sum_{i=1}^{n-d+1} x[i] = 1$$

$$x[i] = 0, \forall i \in \{n-d+2,\ldots,n\}$$

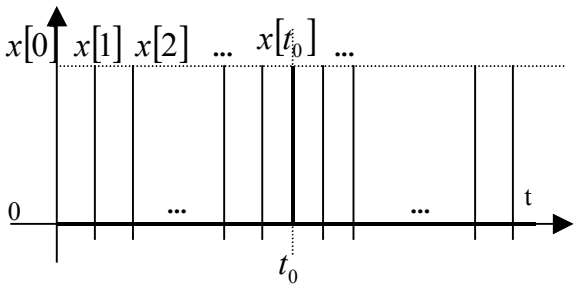
$$\sum_{i=1}^{n} v[i] = d$$

ya que para un $x[i] = 1$ se está imponiendo $v[i+j] \geq 1$, es decir, $v[i+j] = 1$, al ser v un vector binario, y para $x[i] = 0$ no se impone nada sobre v ($v[i+j] \geq 0$).

Modelo en OPL: intervaloc1.mod, intervaloc1.osc

3.2. Relaciones de igualdad y desigualdad

Para modelar $t_0 \nabla p$, donde $\nabla \in \{=, \leq, \geq\}$ y p es un valor determinado y t_0 es el que aparece en la siguiente figura:



se puede usar la expresión:

$$\sum_{i=0}^{n-1} x[i] \cdot 2^i \nabla 2^p$$

dado:

$$\sum_{i=0}^{n-1} x[i] = 1$$

Modelo en OPL: relacionesa.mod.

Se puede expresar alternativamente (más eficaz en su ejecución):

a) Igualdad.

$$x[p] = 1$$

$$x[i] = 0, \forall i \in \{0,\ldots,p-1,p+1,\ldots,n-1\}$$

Modelo en OPL: igualdadb.mod.

b) Desigualdad $t_0 \leq p$

$$\sum_{i=0}^{p} x[i] = 1$$

$$x[i] = 0, \forall i \in \{p+1,\ldots,n-1\}$$

Modelo en OPL: menorigualb.osc.

c) Desigualdad $t_0 \geq p$

$$\sum_{i=p}^{n-1} x[i] = 1$$

$$x[i] = 0, \forall i \in \{0, \ldots, p-1\}$$

Modelo en OPL: mayorigualb.osc.

3.2.1. Relaciones de igualdad y desigualdad para n>>

a) Igualdad
Dados:

$$x_1 = \sum_{i=0}^{s-1} x_1[i]$$

$$x_2 = \sum_{i=0}^{n-s-1} x_2[i]$$

$$x = \sum_{i=0}^{n-1} x[i]$$

$$x = x_1 + x_2$$

La igualdad $t_0 = p$ se puede expresar mediante:

$$\sum_{i=0}^{s-1} x_1[i] \cdot 2^i = (p < s) \cdot 2^{p \bmod s} \quad (1)$$

$$\sum_{i=0}^{n-s-1} x_2[i] \cdot 2^i = (p \geq s) \cdot 2^{(p-s)\cdot(p \geq s)} \quad (2)$$

$$\sum_{i=0}^{s-1} x_1[i] + \sum_{i=0}^{n-s-1} x_2[i] = 1 \quad (3)$$

$$\overline{x} = < x_2[n-s-1], \ldots, x_2[0], x_1[s-1] \ldots, x_1[0] > \quad (4)$$

En (1) se expresa la igualdad para cuando el único uno del vector $\overline{x}$ se encuentra en el tramo de menor peso (cuando $p < s$). El exponente $p \bmod s$ se usa en lugar de p para que la potencia de 2 no se salga de la representación. $p < s$ es uno cuando se cumple la relación lógica y cero en caso contrario. En (2) se expresa la igualdad para cuando el único uno del vector x se encuentra en el tramo de mayor peso (cuando $p \geq s$). El exponente $(p-s) \cdot (p \geq s)$ se usa en lugar de $p - s$ para evitar las potencias de 2 negativas. (3) expresa que sólo hay un uno en el vector resultado de la concatenación de $\overline{x}_1$ y $\overline{x}_2$. Finalmente, (4) expresa esta concatenación en el vector resultado $\overline{x}$.

Modelo en OPL: igualdad2.mod.

b) Desigualdad $\leq$
La desigualdad $t_0 \leq p$ se puede expresar mediante:

$$\sum_{i=0}^{s-1} x_1[i] \cdot 2^i \leq (p < s) \cdot 2^{p \bmod s} + (p \geq s) \cdot 2^{s-1} \quad (1)$$

$$\sum_{i=0}^{n-s-1} x_2[i]\cdot 2^i \le (p \ge s)\cdot 2^{(p-s)\cdot(p\ge s)} \quad (2)$$

$$\sum_{i=0}^{s-1} x_1[i] + \sum_{i=0}^{n-s-1} x_2[i] = 1 \quad (3)$$

$$\overline{x} = < x_2[n-s-1],\ldots,x_2[0],x_1[s-1]\ldots,x_1[0] > \quad (4)$$

En este caso, se ha modificado (1) para expresar que, cuando $p \ge s$, t_0 puede ser uno de los índices de x_1, no sólo de x_2. Así, los casos posibles son:

i) $p < s$

$$\sum_{i=0}^{s-1} x_1[i]\cdot 2^i \le 2^{p \bmod s}$$

$$\sum_{i=0}^{n-s-1} x_2[i]\cdot 2^i \le 0$$

ii) $p \ge s$

$$\sum_{i=0}^{s-1} x_1[i]\cdot 2^i \le 2^{s-1}$$

$$\sum_{i=0}^{n-s-1} x_2[i]\cdot 2^i \le 2^{p-s}$$

Por lo tanto, queda demostrado.

c) Desigualdad $\ge$

Abordar la resolución de este problema desde la misma perspectiva anterior es más complicado que usar el dual de la resolución del caso anterior. La idea es cambiar los pesos de las posiciones de los vectores más significativas por las menos significativas. Así, el peso de la componente $x[n-1]$ pasa a ser 0, mientras que el de la componente $x[0]$ pasa a ser $n-1$. Por lo tanto hay que cambiar en las restricciones del caso anterior:

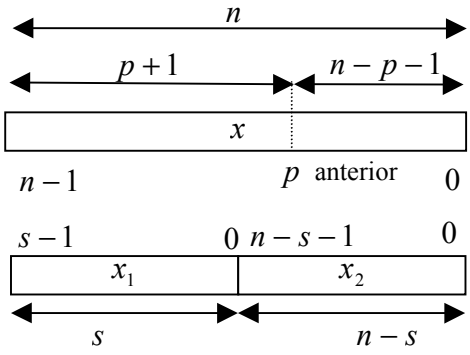
i) Los papeles de x_1 y de x_2.

ii) Invertir los índices de x_1, x_2 y de x.

iii) Cambiar p por $n-p-1$.

iii) Cambiar s por $n-s$.

La siguiente figura ilustra los índices en este caso:



Con los cambios indicados, y simplificando, se llega a:

$$\sum_{i=0}^{n-s-1} x_2[n-s-1-i]\cdot 2^i \le (p+1>s)\cdot 2^{(n-p-1) \bmod (n-s)} + (p+1\le s)\cdot 2^{n-s-1} \quad (1)$$

$$\sum_{i=0}^{s-1} x_1[s-1-i]\cdot 2^i \le (p+1\le s)\cdot 2^{(s-p-1)\cdot(p+1\le s)} \quad (2)$$

$$\sum_{i=0}^{s-1} x_1[i] + \sum_{i=0}^{n-s-1} x_2[i] = 1 \quad (3)$$

$$\bar{x} =< x_2[n-s-1],\ldots,x_2[0],x_1[s-1]\ldots,x_1[0] > \quad (4)$$

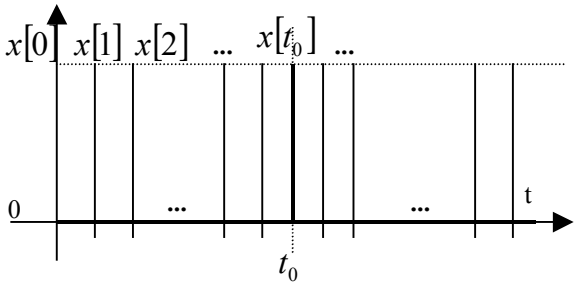
Nota: (3) y (4) permanecen inalteradas.

3.3. Patrones

Se pueden generar patrones binarios como generalización del apartado 1.

3.4. Desplazamiento

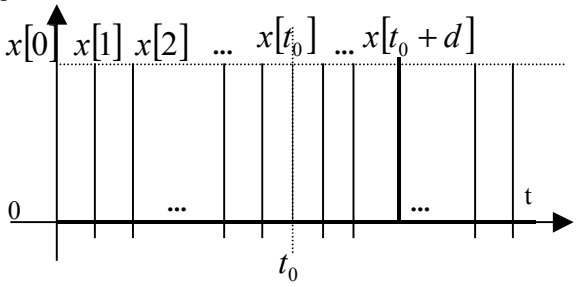
Dado un instante inicial de tiempo (t) en un plano discreto y un número de unidades de tiempo para definir el desplazamiento (d , positivo o negativo), se trata de representar el instante de tiempo $t + d$.
Se elige como representación de la función delta de Kronecker en el plano discreto un vector $\bar{x} =< x[0]\ldots x[n-1] >$ de variables binarias cuyo único 1 representa el instante inicial de tiempo.



Donde $x[i] = \begin{cases} 1, si\ i = t_0 \\ 0, e.o.c. \end{cases}$, y, por tanto: $\sum_{i=0}^{n-1} x[i] = 1$.
Se trata de generar un vector definido por:

$$v[i] = \begin{cases} 1, si\ i = t_0 + d \\ 0, e.o.c. \end{cases}$$

representado gráficamente por:



3.4.1. Primer planteamiento

Desplazamiento para n<<

$\bar{x}$ se puede interpretar como una codificación binaria de un número decimal x que se transforme en otro número v separado de x por d potencias de 2. Así, se calcula un nuevo vector binario cuyo valor decimal es 2^d veces el valor decimal de $\bar{x}$.
a) Si el vector $\bar{v}$ tiene $n + d$ componentes se debe cumplir:

$$2^d \sum_{i=0}^{n-1} x[i] \cdot 2^i = \sum_{i=0}^{n+d-1} v[i] \cdot 2^i$$

Modelo en OPL: desplaza11a.mod.

b) Si el vector $\overline{v}$ tiene n componentes, asumiendo que se cumple $t_0 + d \leq n-1$ (el desplazamiento d se puede alojar en el intervalo $[0,\ldots,n-1]$), se debe cumplir:

$$2^d \sum_{i=0}^{n-1} x[i] \cdot 2^i = \sum_{i=0}^{n-1} v[i] \cdot 2^i$$

Para asegurar que $t_0 + d \leq n-1$:

$$\sum_{i=0}^{n-d-1} x[i] = 1$$

$$x[i] = 0, \forall i \in \{n-d,\ldots,n-1\}$$

Desplazamiento para n>>

a) El vector $\overline{v}$ tiene $n+d$ componentes.
Como antes:

$$x = \sum_{i=0}^{n-1} x[i] \cdot 2^i$$

$$v = \sum_{i=0}^{n+d-1} v[i] \cdot 2^i$$

$$2^d \cdot x = v$$

Ahora se divide:

$$x_1 = \sum_{i=0}^{s-1} x[i] \cdot 2^i$$

$$x_2 = \sum_{i=0}^{n-s-1} x[i+s] \cdot 2^i$$

Se añaden los vectores $\overline{v_1}$ y $\overline{v_2}$:

$$v_1 = \sum_{i=0}^{s+d-1} v_1[i] \cdot 2^i$$

$$v_2 = \sum_{i=0}^{n-s+d-1} v_2[i] \cdot 2^{i-s}$$

De manera que se debe cumplir:

$$2^d \cdot x_1 = v_1$$

$$2^d \cdot x_2 = v_2$$

Notas:

1. Las dos ecuaciones anteriores son equivalentes a $2^d \cdot x = v$, dado que v_1 y v_2 nunca son simultáneamente distintas de cero.

2. No es necesario añadir los vectores $\overline{x_1}$ y $\overline{x_2}$ porque son segmentos disjuntos de $\overline{x}$.

Por lo tanto:

$$2^d \sum_{i=0}^{s-1} x[i] \cdot 2^i = \sum_{i=0}^{s+d-1} v_1[i] \cdot 2^i$$

$$2^d \sum_{i=0}^{n-s-1} x[i+s] \cdot 2^i = \sum_{i=0}^{n-s+d-1} v_2[i] \cdot 2^i$$

$$v[i] = \begin{cases} v_1[i], i \in \{0,\ldots,s-1\} \\ v_1[i] + v_2[i-s], i \in \{s,\ldots,s+d-1\} \\ v_2[i-s], i \in \{s+d,\ldots,n+d-1\} \end{cases}$$

Que es lo mismo que decir:

$$\bar{v} = < v_2[n-s+d-1],\ldots,v_2[d],v_2[d-1]+v_1[s+d-1],\ldots,v_2[0]+v_1[s],v_1[s-1]\ldots,v_1[0] >$$

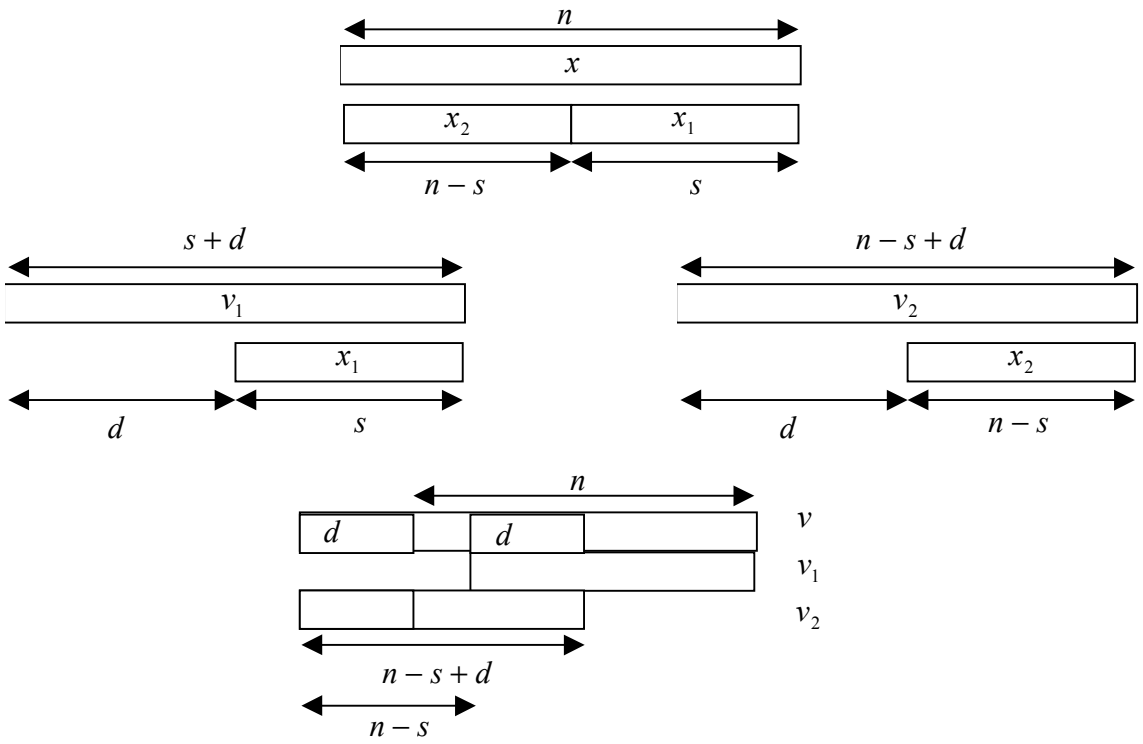
O también:

$$v[i] = v_1[i], i \in \{0,\ldots,s-1\}$$

$$v[i+s] = v_1[i+s] + v_2[i], i \in \{0,\ldots,d-1\}$$

$$v[i+s+d] = v_2[i+d], i \in \{0,\ldots,n-s-1\}$$

Las siguientes figuras ilustran los vectores usados.



Modelo en OPL: desplaza12a.mod.

b) El vector $\bar{v}$ tiene n componentes, asumiendo que se cumple $t_0 + d \le n-1$.

Como antes:

$$x = \sum_{i=0}^{n-1} x[i] \cdot 2^i$$

$$v = \sum_{i=0}^{n-1} v[i] \cdot 2^i$$

$$(2^d - 1) \cdot x = v$$

Ahora se divide:

$$x_1 = \sum_{i=0}^{s-1} x[i] \cdot 2^i$$

$$x_2 = \sum_{i=0}^{n-s-1} x[i+s] \cdot 2^i$$

Se añaden los vectores $\overline{v_1}$ y $\overline{v_2}$ ($\overline{v_1}$ no puede ser mayor que $\overline{x}$):

$$v_1 = \sum_{i=0}^{min(s+d-1,n-1)} v_1[i] \cdot 2^i$$

$$v_2 = \sum_{i=0}^{n-s-1} v_2[i] \cdot 2^{i-s}$$

De manera que se debe cumplir:

$$2^d \cdot x_1 = v_1$$

$$2^d \cdot x_2 = v_2$$

Notas:

1. Las dos ecuaciones anteriores son equivalentes a $2^d \cdot x = v$, dado que v_1 y v_2 nunca son simultáneamente distintas de cero.

2. No es necesario añadir los vectores $\overline{x_1}$ y $\overline{x_2}$ porque son segmentos disjuntos de $\overline{x}$.

Por lo tanto:

$$2^d \sum_{i=0}^{s-1} x[i] \cdot 2^i = \sum_{i=0}^{min(s+d-1,n-1)} v_1[i] \cdot 2^i$$

$$2^d \sum_{i=0}^{n-s-1} x[i+s] \cdot 2^i = \sum_{i=0}^{n-s-1} v_2[i] \cdot 2^i$$

$$v[i] = \begin{cases} v_1[i], i \in \{0,\ldots,s-1\} \\ v_1[i] + v_2[i-s], i \in \{s,\ldots,min(s+d-1,n-1)\} \\ v_2[i-s], i \in \{min(s+d,n),\ldots,n-1\} \end{cases}$$

Que es lo mismo que decir:

$$\overline{v} = < v_2[n-s-1],\ldots,v_2[d-1], v_2[d-2] + v_1[min(s+d-1,n-1)],\ldots,v_2[0] + v_1[s], v_1[s-1],\ldots,v_1[0] >$$

O también:

$$v[i] = v_1[i], i \in \{0,\ldots,s-1\}$$

$$v[i+s] = v_1[i+s] + v_2[i], i \in \{0,\ldots,min(s+d-1,n-1)-s\}$$

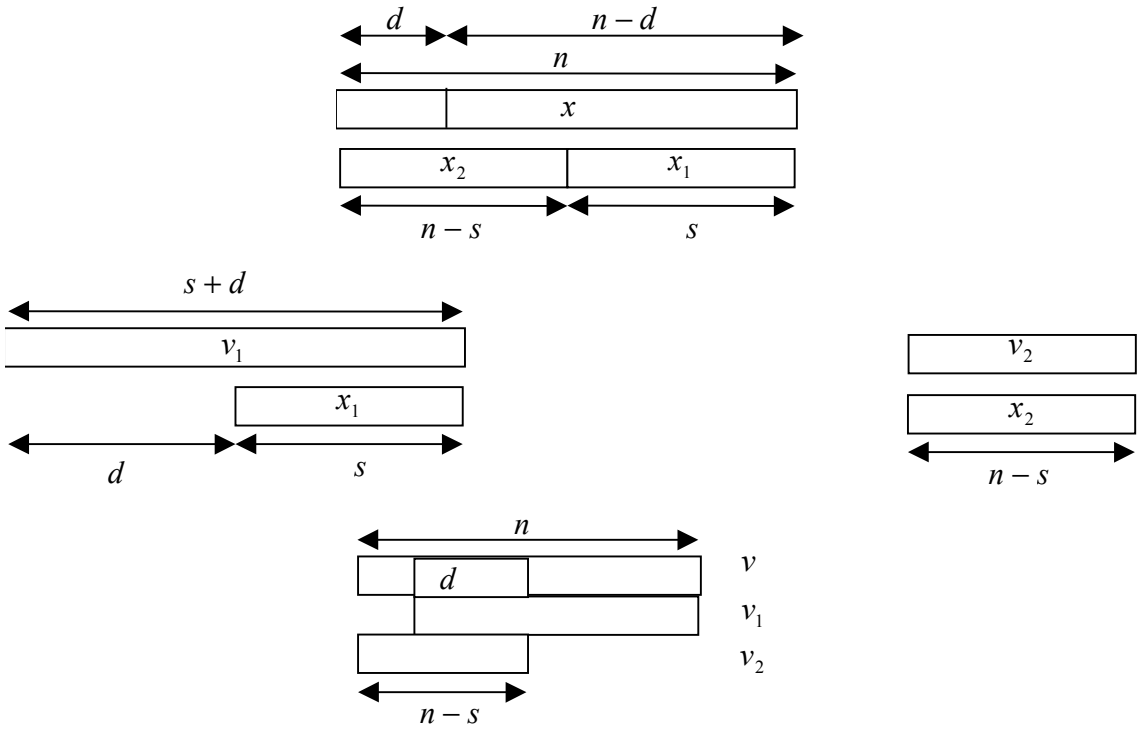
$$v[i+s+d] = v_2[i+d], i \in \{0,\ldots,n-s-d-1\}$$

Dado que $t_0 + d \leq n-1$:

$$\sum_{i=0}^{n-d-1} x[i] = 1$$

$x[i] = 0, \forall i \in \{n-d,\ldots,n-1\}$ (Véase el apartado Relaciones de igualdad y desigualdad)

Las siguientes figuras ilustran los vectores usados.



Modelo en OPL: desplaza12b.mod.

3.4.2. Segundo planteamiento

a) Vector con índice de componentes $0,\ldots,n-1$.

Es necesario imponer que, dado un $x[i]=1$ entonces $v[i+d]=1$. Es decir, la equivalencia:

$x[i]=1 \Leftrightarrow v[i+d]=1, \forall i \in \{0,\ldots,n-d-1\}$

El valor del componente $i+d$ de v depende del valor de $x[i]$ y, por tanto, se puede expresar de la siguiente forma:

$v[i+d] \geq x[i], \forall i \in \{0,\ldots,n-d-1\}$

$\sum_{i=0}^{n-1-d} x[i] = 1$

$x[i]=0, \forall i \in \{n-d,\ldots,n-1\}$

$\sum_{i=0}^{n-1-d} v[i+d] = 1$

$v[i]=0, \forall i \in \{0,\ldots,d-1\}$

ya que para un $x[i]=1$ se está imponiendo $v[i+d] \geq 1$, es decir, $v[i+d]=1$, al ser v un vector binario, y para $x[i]=0$ no se impone nada sobre v ($v[i+d] \geq 0$).

Modelo en OPL: desplaza2a.mod, desplaza2a.osc

b) Vector con índice de componentes $1,\ldots,n$.

Es necesario imponer que, dado un $x[i]=1$ entonces $v[i+d]=1$. Es decir, la equivalencia:

$x[i] = 1 \Leftrightarrow v[i+d] = 1, \forall i \in \{1,\ldots,n-d\}$

El valor del componente $i+d$ de v depende del valor de $x[i]$ y, por tanto, se puede expresar de la siguiente forma:

$v[i+d] \geq x[i], \forall i \in \{1,\ldots,n-d\}$

$\sum_{i=1}^{n-d} x[i] = 1$

$x[i] = 0, \forall i \in \{n-d+1,\ldots,n\}$

$\sum_{i=1}^{n-d} v[i+d] = 1$

$v[i] = 0, \forall i \in \{1,\ldots,d\}$

ya que para un $x[i] = 1$ se está imponiendo $v[i+d] \geq 1$, es decir, $v[i+d] = 1$, al ser v un vector binario, y para $x[i] = 0$ no se impone nada sobre v ($v[i+d] \geq 0$).

Modelo en OPL: desplaza2b.mod, desplaza2b.osc

4. Representación de funciones temporales discretas en programación con restricciones

4.1. Función intervalo temporal

4.1.1. Primera alternativa

Con orden superior:

a) Vector con índice de componentes $0,\ldots,n-1$.

b) Vector con índice de componentes $1,\ldots,n$.

```
int+ top=31;
range Indx 1..top;
int+ d=4;
var int+ x[Indx] in 0..1;
var int+ p in 1..top-d+1;
solve {
  sum(i in Indx) x[i] = d;
  sum(i in Indx) (i*(x[i]=1)) = (p+p+d-1)*d/2;
  //La suma de la serie aritmética es s=(a1+an)*n/2
};
```

Modelo en OPL: intervalo3.mod, intervalo3.osc.

4.1.2. Segunda alternativa

Con implicaciones lógicas.

a) Vector con índice de componentes $0,\ldots,n-1$.

b) Vector con índice de componentes $1,\ldots,n$.

```
var int+ x[1..n] in 0..1;
var int+ v[1..n] in 0..1;
solve {
  sum(i in 1..n-d+1) x[i] = 1;
  forall(i in 0..d-2) x[n-i]=0;
  forall(i in 1..n-d+1)
    forall(j in 0..d-1)
      (x[i]=1) => (v[i+j]=1);
```

Modelo en OPL: intervalo4.mod, intervalo4.osc.

4.2. Función desplazamiento

Con implicaciones lógicas.

a) Vector con índice de componentes $0,\ldots,n-1$.

b) Vector con índice de componentes $1,\ldots,n$.

```
var int+ x[1..n] in 0..1;
var int+ v[1..n] in 0..1;
solve {
  sum(i in 1..n-d) x[i] = 1;
  forall(i in 0..d-1) x[n-i]=0;
  forall(i in 1..n-d)
      (x[i]=1) => (v[i+d]=1);

};
```

Modelo en OPL: desplaza4b.mod, desplaza4b.osc.

5. Implementación de restricciones globales mediante restricciones locales

5.1. alldifferent

```
//
// Modelo de la restricción global alldifferent con restricciones
locales
//
// Representación de datos: array de datos.
//
int n=3;

range dominio 1..n;

var dominio array[dominio];

solve
{
   forall(i,j in dominio : i < j)
     array[i] <> array[j];
};
```

5.2. circuit

La restricción circuit calcula un camino hamiltoniano en un grafo. Asume un grafo totalmente conectado. Se estudian dos representaciones:
2.1) Array de nodos del grafo. Cada elemento del array representa un nodo del grafo, i.e., circuito[*i*]=*nodo* (*nodo* es el i-ésimo nodo visitado). Por lo tanto, cualquier permutación es un camino hamiltoniano diferente. El índice del array identifica el orden en que se visitan los nodos del grafo.

```
//
// Modelo de la restricción global circuit con restricciones locales
//
// Representación de datos: array de nodos.
// Interpretación: el valor de cada elemento del array es el orden en
el
// que se visita el nodo al que representa.
// Cualquier permutación es un circuito hamiltoniano.
//
int n=3;

range dominio 1..n;

var dominio circuito[dominio];

solve
{
   alldifferent(circuito);
};
```

2.2) Array de nodos destino. Cada elemento del array representa el destino del nodo que está representado en el índice del array, i.e., destino[*nodo_destino*] = *nodo_origen*. Esta representación corresponde a la implementación en OPL de la restricción circuit. No toda permutación es válida, porque pueden aparecer subcircuitos.
Ej:
destino[1]=2;
destino[2]=1;

destino[3]=4;
destino[4]=3;
Para implementarlo se utiliza la representación del caso anterior y se hace corresponder con la nueva representación.
circuito[1]=$nodo_1$;
circuito[n]=$nodo_n$;
Como el nodo circuito[i] precede al nodo circuito[i+1], se debe cumplir:
destino[circuito[i]] = circuito[i+1], i=1..n-1
destino[circuito[n]]=circuito[1], (i=n)
El problema que aparece son las simetrías, puesto que aparecen permutaciones equivalentes con la representación anterior. Se pueden eliminar forzando el nodo de partida a un valor concreto. En la representación anterior aparecen necesariamente más soluciones puesto que no está especificado el nodo de origen. En la representación actual no hay noción explícita en la representación del primer nodo que se visita y, por lo tanto, aparecen menos soluciones.

```
//
// Modelo de la restricción global circuit con restricciones locales
//
// Representación de datos: array de nodos destino.
// Interpretación: el índice de cada elemento del array es el nodo
origen
// y su valor es el nodo destino. circuito[origen]=destino
// No es válida cualquier permutación, porque puede contener
subcircuitos.
// Se hace un cambio de representación: el array de nodos destino se
hace
// corresponder con un array de nodos, como en circuit1.mod
// Problema: aparecen simetrías
//

int n=3;

range dominio 1..n;

var dominio destino[dominio];

var dominio circuito[dominio];

solve
{
   forall(i in dominio : i < n)
     destino[circuito[i]] = circuito[i+1];
   destino[circuito[n]] = circuito[1];
//   circuito[1]=1;
 alldifferent(circuito);
};
```

6. Implementación de restricciones globales con programación lineal mixta

6.1. alldifferent

Se debe implementar:

$x_i \neq x_j, i \neq j, i, j \in \{1,..,n\}, x_i \in \{1,..,n\}$

La desigualdad se representa con $d = (x_i - x_j) \neq 0$, siendo *d* una función constante discontinua en 0, y se implementa con $|d| \leq \Delta$:

$d \leq -\Delta s + M(1-s)$

$d \geq ms + \Delta(1-s)$

Si es necesario implementar la igualdad, se hace el cambio de variable *s* por1-*s* resultando:

$d \leq Ms - \Delta(1-s)$

$d \geq \Delta s + m(1-s)$

Modelo en OPL (alldiffi.prj):

```
int n=...;
float+ delta=...;
var int x[1..n] in 1..n;
var int s[1..n,1..n] in 0..1;
var float d[1..n,1..n];
solve
{
forall(i,j in 1..n: i<j)
  {
     d[i,j]=x[i]-x[j];
     d[i,j]<=-delta*s[i,j]+(n-1)*(1-s[i,j]);
     d[i,j]>=-(n-1)*s[i,j]+delta*(1-s[i,j]);
  };
};
```

6.2. circuit

La variable binaria d[i,j] = 1 representa que hay una conexión del nodo i al nodo j, si es 0 que no hay.
Con:

```
forall(j in 1..n)
     sum(i in 1..n: i<>j) d[i,j]=1;
```

se asegura de que salga un tramo de cada ciudad.
Con:

```
forall(i in 1..n)
     sum(j in 1..n: i<>j) d[i,j]=1;
```

se asegura de que llegue un tramo de cada ciudad.
Con:

```
forall(i,j in 2..n:i<>j)
     y[i]-y[j]+(n-1)*d[i,j]<=n-2;
```

se asegura de que no haya ningún subcircuito.
El modelo en OPL:

```
int n=...;
var int d[1..n,1..n] in 0..1;
var float y[2..n];
solve
{
   forall(j in 1..n)
     sum(i in 1..n: i<>j) d[i,j]=1;
```

```
   forall(i in 1..n)
     sum(j in 1..n: i<>j) d[i,j]=1;
   forall(i,j in 2..n:i<>j)
     y[i]-y[j]+(n-1)*d[i,j]<=n-2;
};
```

7. Comparación entre programación lineal mixta y programación con restricciones

7.1.1. alldifferent

Búsqueda de la primera solución. Los resultados se expresan en segundos.
Restricción global en Solver: alldiffg.prj
Restricciones locales en Solver: alldiffl.prj
Restricciones lineales mixtas: alldiffi.prj
Resultados:

n	**alldiffg** (onValue)	**alldiffl**	**alldiffi**
10			0.08
11			0.20
12			0.34
13			16.86
14			0.48
15			>775
16			>725
50			96.83
500		2.13	
1000		12.34	
1100		16.43	
1200		26.70	
1300		44.29	
1400		45.50	
1500		55.48	
1600		77.62	
2000	3.16		
3000	9.57		
4000	27.66		
5000	57.55		
6000	96.53		
7000	175.68		
8000	325.70		

7.1.2. circuit

Restricción global en Solver: circuitg.prj
Restricción global en Solver con alldifferent: circuit2.prj
Restricciones locales en Solver: circuitl.prj
Restricciones lineales mixtas: circuiti.prj
Resultados:

n	**circuitg**	**circuit2 (alldifferent)**	**circuitl**	**circuiti**
10	0.05	0.03		1.34
15				0.96
20				0.24
30				0.92
40				44.19
50	0.00	0.07		60.25
100	0.01	0.55		
500	0.33	58.21		
600		102.13		
1000	1.10	478.14		

2000	7.00			
3000	23.30			
5000	153.78			
6000	316.51			

8. Ejemplos de programación

8.1. Números primos

Dada $x \in \{1,..,n\}$, una posible expresión que expresa el conjunto de números primos es:

$$\forall_{i \in \{2,..,n\}} (x > i) \Rightarrow ((x \bmod i) <> 0)$$

Modelo en OPL: primos.mod.

8.2. Reparto de pesos

Dadas n variables reales $x_0, \ldots, x_{n-1}$ y sus porcentajes de participación $p_0, \ldots, p_{n-1}$ en una cantidad común, se trata de expresar dicho acoplo. Es decir, se debe satisfacer:

$$x = \sum_{i=0}^{n-1} x_i p_i \text{ , donde } \sum_{i=0}^{n-1} p_i = 1$$

Modelo en OPL: pesos.mod.