I. O. T1	MODELADO CON P. L.	HOJA N° 1
		FECHA 01/08/16

- Objetivo: $\min c^t x$ $\quad c, x \in \mathbb{R}^n, b \in \mathbb{R}^m$

  s.a. $Ax \leq b$ $\quad A \in M_{m,n}(\mathbb{R})$

* Ejemplos clásicos:
  - Producción: Willy (teoría)
  - Dieta: Ejemplo - menú cafetería

Precio fijo y clientela fija → Estrategia: minimizar costos, pero han de garantizar nutrición (y de ser posible un mínimo de sabor)

Restricciones
(H) hidratos de carbono > 90
(V) Vitaminas > 100
(P) Proteínas > 40
(S) Sabor > 5
(C) Colesterol < 50

En una ración

Alimentos:

	H	V	P	S	C	$
Papas	30	20	10	1	5	10
Carne	5	10	30	2	20	70
Salsa	10	15	0	4	25	15
Fruta	5	30	0	2	5	30

× Kg

Variables: x_i cantidad de alimento i

Objetivo: $\min \sum \$_i x_i$

Restricciones: → Cantidades mínimas (Dieta estandard)

→ No exceder Colesterol

→ Tamaño de la ración

Transporte: Minas carbon — Hornos
(i) (j)

Obj: min coste transporte

Rest: $\sum_j x_{ij} < \text{Produccion } i$

$\sum_i x_{ij} > \text{Demanda } j$

⊙ Que hay que tener claro:

- Que es constante y que puede variar
- Que tipo de variable tenemos: $\mathbb{Z}$, $\mathbb{R}$, binaria
- Que rango permitimos

Además (iremos viendo)

- Variable ppal o auxiliar
- Tamaño del problema
- Relaciones entre variables !!!

- Formulación no es única
- Debe ser capaz de dar una respuesta al problema
- Debe cumplir todas las restricciones

Con PL podemos abordar problemas aparentemente no lineales. Veamos unos ejemplos:

① Regresión lineal. Dada una nube de puntos $\{(x_i, y_i)\}_i$ elegir a y b tq la recta $ax+b$ minimiza la distancia a la nube según algún criterio

NOTA

I.O. T1 | Modelado con P.L. | HOJA Nº | FECHA

Típico: mínimos cuadrados: $\min \sum (ax_i + b - y_i)^2$

Otras opciones: • $\min_{a,b} \sum |ax_i + b - y_i|$

• $\min_{a,b} \{\max_i |ax_i + b - y_i|\}$

> Como lidiar con $|\cdot|$ en problemas de minim.

$|x| = x^+ + x^-$, s.a. $x = x^+ - x^-$

$\min \sum |x_i| \longrightarrow \min \sum z_i^+ + z_i^-$

¿Cómo cambia el tamaño del problema?

> ——— $|\cdot|$ ——— en problemas de max

$|x| = x^+ + x^-$ pero $x = x^+ - x^-$ ya no garantiza una de ellas es 0. Lo veremos en un par de días

> Como lidiar con min max

$\min \quad z$

s.a. $z \geq f_i$ (si f_i es un v.a. descomponemos como antes)

en este caso pasa a tener $\max |z|$, como luego minimizamos, $z_i = z_i^+ - z_i^-$ es suficiente

NOTA

② lidiar con fracciones en el objetivo :

LP estandar : max beneficios → max Ingresos - Coste
LPF max ratio benef/coste → max Ingr/Cost

$$\max \frac{c^t x + \alpha}{d^t x + \beta}$$

s.a

$$Ax \leq b$$

$$\boxed{\begin{array}{l} \text{o bien } \sum d^t x + \beta > 0 \quad \forall x \in RSF \\ \text{o bien } \sum d^t x + \beta < 0 \quad \forall x \in RSF \end{array}}$$

Como hacemos:

① $y = t \cdot x$, con $t = \dfrac{1}{d^t x + \beta}$ $\left(\Leftrightarrow t(d^t x + \beta) = 1 \right)$

$$\begin{array}{ll} \max & c^t y + \alpha t \\ \text{s.a} & Ay \leq bt \\ & d^t y + \beta t = 1 \\ & t > 0 \end{array} \longrightarrow x = \frac{1}{t} y$$

Obs si no hacemos el cambio $y = tx$ tenemos algo no lineal !!

¿? Pq restricción $d^t x + \beta \geq 0$ necesaria ?
$\neq 0$
< 0

Labo: usario → invop - 01

NOTA clave → invop

~~TRUCOS DE PROG. LIN ENTERA~~ Clase 2

3> Eliminar productos

Caso 1: 2 variables binarias, x, y

$\Rightarrow z = x \cdot y$ con la misma tabla de verdad

x_1	y	xy
1	0	0
0	0	0
1	1	1
0	1	0

$\rightarrow$

$z \in \{0, 1\}$

$z \leq x_1$
$z \leq x_2$

si alguna es 0 $z = 0$

si $z = 1 \Rightarrow x_1 = x_2 = 1$

No suficiente:

$z \geq x_1 + x_2 - 1$

$\Rightarrow$ si $z = 0$ al menos una x_1 o $x_2 = 0$

si los dos son 1 $\Rightarrow z = 1$

Caso 2: x_1 binaria
$x_2 \in [0, u]$

$y = x_1 \cdot x_2$

x_1	x_2	y
0	$0 \leq w \leq u$	0
1	$0 \leq w \leq u$	w

$y \leq 0$
$y \leq w$
$y \geq w - u$
$y \geq 0$

$\Rightarrow y = 0$

$y \leq u$
$y \leq w$
$y \geq w$
$y \geq 0$

$\Rightarrow y = w$

$$\boxed{\begin{aligned} y &\leq u x_1 \\ y &\leq x_2 \\ y &\geq x_2 - u(1 - x_1) \\ y &\geq 0 \end{aligned}}$$

Caso 3: $x_1 \in [l_1, u_1]$, $x_2 \in [l_2, u_2]$

3.1 (Sencillo) Además se cumple:

- $l_1 \geq 0$ y $l_2 \geq 0$
- al menos una de las dos variables sólo aparece multiplicando la otra

[Aún más sencillo: si las dos aparecen únicamente como producto sustituimos y ya (no se da)]

$x_1 \cdot x_2 =: z$ con restricciones

$l_1 x_2 \leq z$

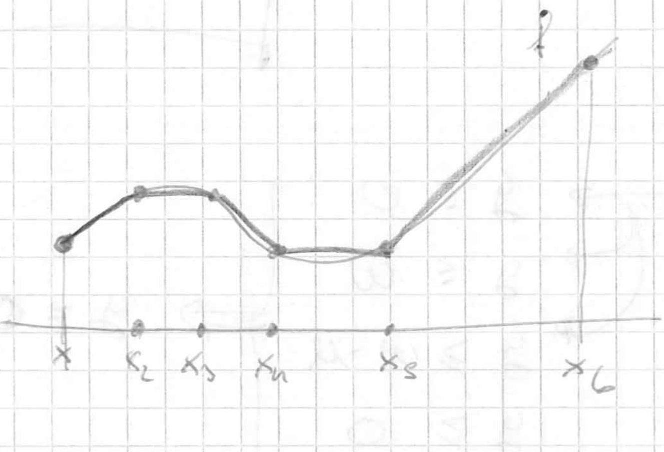
$u_1 x_2 \geq z$

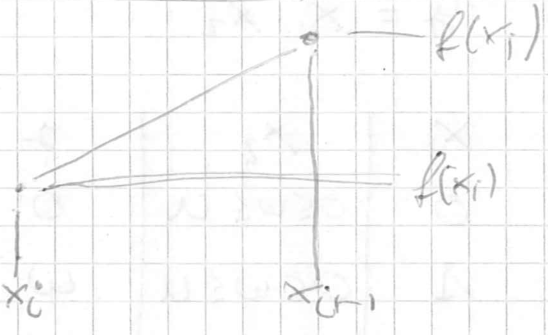
Ejercicio $u_1 \leq 0$, $u_2 \leq 0$

3.2 (General) $y_1 := \frac{1}{2}(x_1 + x_2)$, $y_2 = \frac{1}{2}(x_1 - x_2)$

$\Rightarrow xy = y_1^2 - y_2^2$ ← sigue sin ser lineal (−)
las variables están separadas (+)

4 Aproximación de funciones continuas





$x \in [x_i, x_{i+1}] \to x = \lambda x_i + (1-\lambda) x_{i+1}$ (comb. lin. convexa)

$= x_i + \mu (x_{i+1} - x_i)$ $\mu \in [0,1]$ ($\mu = 1 - \lambda$)

$\Rightarrow f(x) \sim \lambda f(x_i) + (1-\lambda) f(x_{i+1})$

$(f(x_i) + \mu(f(x_{i+1}) - f(x_i)))$

NOTA

$$f(x) \approx \tilde{f}(x) = \lambda_0 f(x_0) + \_\_ + \lambda_N f(x_N)$$

x_i Constante!
$f(x_i)$ Constante!
λ_i variables

Restricciones:

- $\lambda_0 x_0 + \_\_ + \lambda_N x_N = X$
- $\lambda_0 + \_\_ + \lambda_N = 1$
- Como mucho dos λ's consecutivos pueden ser diferentes de cero

③ S.O.S (Special Ordered Sets)

SOS 1: De un conjunto de variables de decisión solo una pue ser cierta:

$$y_i \in \{0,1\} \rightarrow \sum y_i \leq 1$$

más general: $x_i \in [0, u_i]$ tq $\sum a_i x_i$

y solo una de ellas puede ser diferente de cero: (x_i)

añadimos restricciones: $x_i \leq u_i y_i$

$\sum y_i = 1$

y_i binaria

Ejemplo: Elegir ubicación para producción

SOS 2: Como mucho 2 diferentes de cero (ref: piecewise linear)

Como SOS 1 con restricciones

$$x_i \leq u_i y_i$$

$$\sum y_i \leq 2$$

$$y_i + y_j \leq 1 \quad \forall i \in [0, N-2], j \geq i+2$$

6) Restricción de intervalo

Obs: Surgió la necesidad de hablar del problema de la dieta, como medio de cumplir la condición "sabe bien"

$$\min \sum c_j x_j$$

s.a.

$$l_i \leq \sum_j a_{ij} x_j \leq u_i \quad \forall i$$

$$x_j \geq 0 \quad \forall j$$

Solución evidente: $\sum a_{ij} x_j \leq u_j$

$\sum a_{ij} x_j \geq l_j$

(−) Se duplican las restricciones. Si hay algún cambio deben modificarse las dos restricciones

Alternativa: $z_i + \sum a_{ij} x_j = u_i$

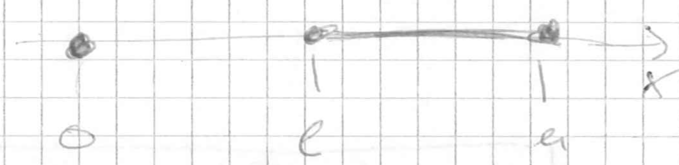
introducimos variables auxiliares z_i que hagan cumplir la igualdad. Necesitamos

$$0 \leq z_i \leq u_i - l_i$$

clase 8

7) Variable que toma valores discontinuos

$$x = \begin{cases} 0 \\ \omega \in [l, u] \end{cases}$$



[Ejemplo: Un proveedor no acepta pedidos de menos de l

introducimos: $y = \begin{cases} 0 & x = 0 \\ 1 & x \in [l, u] \end{cases}$

con restricciones: $x \leq u y$

$x \geq l y$

[Otro ejemplo: Costes fijos

NOTA

$$\min \; C(x)$$
$$\text{sa} \quad a_i x + Aw \leq b$$
$$x, w \geq 0$$

Con $C(x) = \begin{cases} 0 & x = 0 \\ k + cx & x > 0 \end{cases}$

↳ Cambiamos $C(x) \rightsquigarrow C^*(y, x) = ky + cx$

$$\min \; ky + cx$$
$$\text{sa} \quad a_i x + Aw \leq b$$
$$x \leq My$$
$$x, w \geq 0$$
$$y \text{ binaria}$$

$\left(y = \begin{cases} 0 & x = 0 \\ 1 & x > 0 \end{cases} \right.$

No necesitamos condición $y = 0$ cuando $x = 0$ pq es de minimizar pero se puede añadir $y \leq x$

en el caso anterior sería $x \geq 0 \cdot y \rightsquigarrow x \geq 0$ no dice nada de y

8 Condiciones de "O bien ___ O bien ___"

$$\min \; c^t x$$
$$\text{sa} \quad \sum a_{1j} x_j \leq b_1$$
$$\sum a_{2j} x_j \leq b_2$$

donde al menos una de las dos restricciones es necesaria

Esto se puede dar en un proceso de producción donde se pueden elegir diferentes estrategias (o ubicaciones) con restricciones diferentes

$$\sum a_{1j} x_j \leq b_1 + M_1 y$$

$$\sum a_{2j} x_j \leq b_2 + M(1-y)$$

y binaria

(9) Restricciones condicionales

Queremos ~~d~~expresar: "Si $A \Rightarrow B$"

la implicación "$A \Rightarrow B$" es equivalente a "$(\neg A) \vee B$" ($A \Rightarrow B \rightsquigarrow A \wedge (\neg B)$ falso
$\rightsquigarrow \neg(A \wedge (\neg B))$ cierto
$\rightsquigarrow \neg A \vee B$ cierto)

$$\text{Si } \left(\sum a_{1j} x_j \leq b_1\right) \Rightarrow \left(\sum a_{2j} x_x \leq b_2\right)$$

Se convierte en

$$\sum a_{1j} x_j \geq b_1 + \varepsilon - Ly$$
$$\sum a_{2j} x_j \leq b_2 + M(1-y)$$

y binaria

ε se introduce para evitar $\sum a_{1j} x_j \geq b_1$ que sería la negación de "A"

A	B	$A \Rightarrow B$	$\neg A \vee B$
1	1	1	1
1	0	0	0
0	1	1	1
0	0	1	1

NOTA

Corte de Stock:

$$\min \sum_p x_p$$

$$\text{s.a.} \quad \sum_p a_{fp} x_p \geq d_f$$

Índices: $p \to$ patrones
$f \to$ finales

Parámetros: $d_f \to$ Cantidad requerida del tamaño final f
$a_{fp} \to$ Cantidad de finales f en el patrón p

Variables: $x_p \to$ cantidad de rollos originales cortados según p

Se complica cuando crece el número de patrones
↳ Soluciones (requieren posproceso)

Modificación 1:

Heurística LILE (Largest In Least Empty)

→ Redondear sol fraccionaria hacia abajo (Trunca)

→ Determinar los finales que faltan y ordenarlos mayor a menor

Loop:

→ Colocar el mayor final pendiente en el original menos vacío que pueda contenerlo

→ Si no es posible, añadir un original

→ Continuar hasta que no queden finales ptes

Ejemplo:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
450	2	1	1	1	1	1	1	1	1											
360		1	1							2	2	2	1	1	1	1	1	1	1	1
310				1	1								2	1	1	1				
140		1		1		3	2	1		2	1			2	1		4	3	2	1

	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
450																	
360	1																
310		3	2	2	2	1	1	1	1	1							
140			2	1		4	3	2	1		7	6	5	4	3	2	1

37 patrones para cortar rollos de 1000 en finales de 450, 360, 310, 140.

Sol: 452'25

LILE → 453

Delayed cutting pattern generation

→ Inicializar Submodelo (elegir subconjunto de Patrones)

→ Mientras se mejore:

→ Resolver Problema C.S. para el submodelo

→ Resolver el problema de generar un patron

→ Si el nuevo patron mejora la sol → añadirlo a la lista de patrones

¿Como generar Patrones?

Un patron es un vector $(P_{f_1}, \ldots, P_{f_N})$ ← Cantidad de finales f_i

$$P_f \geq 0$$
$$\sum_f w_f P_f \leq W$$
$$1 - \sum_f \lambda_f P_f < 0$$ ← Tendrá sentido más adelante

$$\lambda = C_B B^{-1}$$

$$\max \sum_f \lambda_f P_f$$
s.a.
$$P_f \geq 0$$
$$\sum w_f P_f \leq M$$

En el posproceso, si $\lambda_j > 1$ añadimos patrón si no no

Ampliaciones: Varios tamaños de original

Ejemplo de uso de 7: Heladeras Ampliación - HAY QUE COMPLETAR UN % DE UNA HELADERA EL DÍA QUE SE EMPIEZA

$$\max \sum_k g_k x_k$$

s.a.

$$x_k h_k \le \sum_{ij} z_{ijk} \quad \forall k$$

$$\sum_k z_{ijk} = 8\, y_{ij} \quad \forall ij$$

$$\sum_{jk} z_{ijk} = 40 \quad \forall i$$

$$\left\{ \begin{array}{l} \sum_i (z_{i1k} - h_k \tilde{x}_{1k}) \le h_k \tilde{y}_{1k} \\ \sum_i (z_{i1k} - h_k \tilde{x}_{1k}) \ge c_1 h_k \tilde{y}_{1k} \end{array} \right. \rightarrow \text{caso inicial}$$

$$\left\{ \begin{array}{l} \sum_i z_{ijk} + u_{j-1,k} - h_k \tilde{x}_{jk} + u_{jk} = h_k \tilde{y}_{jk} \\ u_{jk} \le h_k \tilde{y}_{jk} \end{array} \right.$$

x_k: entero, cantidad heladeras tipo k, ≥ 0

$\tilde{x}_{jk}$: entero, cantidad heladeras tipo k el día j, ≥ 0

z_{ijk}: entero, horas dedicadas por el trabajador i al modelo k el día j, ≥ 0

y_{ij}: binaria, el trabajador i trabaja el día j

$\tilde{y}_{jk}$: binaria, el día k se ~~empieza/produce~~ dedican horas al modelo k pero queda una unidad incompleta.

u_{jk}: entero, horas faltantes para terminar unidad del modelo k el día j ≥ 0

Ampliación 2: dos empleados no pueden trabajar en el mismo modelo a la vez

NOTA

MIX DE PRODUCCIÓN:

25 escuelas, 30 puentes, 15 puentes

↓ 3 tipos ↓ 2 tipos ↓ 2 tipos

una escuela, puente o industria, de alguna de los tipos

↓

r_{ij} = Cantidad del recurso i necesaria para el proyecto j

Usar los recursos disponibles de modo la proporción de proyectos llevados a cabo se parezca lo más posible a la dada:

$$\max z$$

$$\text{s.a.} \quad z \leq \frac{p_1 + p_2 + p_3}{25}$$

$$z = \min\left\{ \frac{a}{25}, \frac{a_2}{30}, \frac{a_3}{15} \right\}$$

$$z \leq \frac{p_4 + p_5}{30}$$

$$z \leq \frac{p_6 + p_7}{15}$$

$$\sum_i r_{ij} p_i \leq R_j$$

p_i Cantidad de proyectos del tipo i llevados a cabo

Referencia: Kantrowich "Métodos matemáticos para la organización y planificación de la producción"

I.O. T1 | COMPLEJIDAD | HOJA Nº | FECHA

P ← Solución eficiente ("polinomial")

NP ← Non deterministic Polynomial

"Existe un verificador V para el problema"

→ Dada una instancia I con respuesta "SI"

∃ un certificado (witness) W t.q. $V(I,W)$ devuelve SI en tiempo polynomial

Si la respuesta es "no" $V(I,W)$ devuelve "no" en tiempo polynomial $\forall W$

V puede devolver "no" para W incorrecto

Problema de Satisfacción Booleana

DECIDIR SI SE PUEDE ASIGNAR VALORES A LOS LITERALES DE FORMA QUE LA FORMULA SEA VERDADERA

SAT satisfabilidad

→ literal: $x_i, \neg x_i$

→ Clausula (disjuntiva) $x_1 \vee x_3 \vee \neg x_5$

→ FORMA normal Conjuntiva $(x_1 \vee x_3) \wedge (x_3 \vee \neg x_5 \vee x_4)$

Ej 1 $(x_1 \vee x_2 \vee \neg x_4) \wedge (x_2 \vee \neg x_3) \wedge x_5$ es SAT si $x_i = 1$

Ej 2: $(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$

no es SAT

Def: 3SAT: todas las clausulas con 3 literales

Prop: 3SAT es NP-Completo ($3SAT \leq_p SAT \leq 3SAT$)

Dem. 1) $C = x_i$ $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3) \wedge$

$\Rightarrow C = \cancel{x_i \vee x_i \vee \neg x_i}$ usar 2) 2 veces $(x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3)$

2) $C = x_1 \vee x_2 \Rightarrow C = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3)$

NOTA

3) $x_1 \vee x_2 \vee x_3$

4) Partimos la clausula:

Σ, Υ dos proposiciones

$\Sigma \vee \Upsilon = (\Sigma \vee z) \wedge (\Upsilon \vee \neg z)$

$\Rightarrow C = (x_1 \vee \ldots \vee x_k)$

$= (x_1 \vee x_2 \vee z_1) \wedge \underbrace{(x_3 \vee \ldots \vee x_k \vee \neg z_1)}$

Un literal menos
seguimos partiendolo

Resumen de Zimpl/scip:

Idea gral: Escribimos código ZIMPL en editor de texto → Ejecutamos desde scip.

FORMATO CODIGO ZIMPL.

6 tipos de instrucciones / definiciones

→ ~~conjuntos~~ define conjunto de índices
→ Parámetros:
→ Variables: Auxiliares y de decisión
→ Restricciones
→ Función objetivo
→ Def. funciones

Generalidades:

* Comentarios: "#"
* Instrucciones terminan con ";"
* include "nombre fichero" (con comillas)
* Expresiones matemáticas: tablas 2, 3 p7
* Cadenas: delimitadas con comillas " "
  Concatenación con "+" ("cadena1" + "cadena2" = "cadena1cadena2")
* boolean: <, <=, ==, !=, >=, >, and, or, xor, not

Conjuntos:

Set A := {e1, e2, — en}

→ Todos los elementos del mismo tip
→ elementos pueden ser tuplas

→ tuplas pueden contener numeros y ~~strings~~ cadenas

→ Ver tabla 4. Particularmente interesantes:

$A * B$ → prod cartesiano

$A \setminus B$: $A - B$ → Diferencia

{m .. n by s} → {m, m+|s|, m+2|s|, ..., n}

{m to n by s} → Como .., pero con signo. ↑ o el mas cercano por defecto

→ Conjuntos condicionales:

Set A := { el in S with boolean }

→ Conjuntos indexados:

Set $A[I]$:= <1> A_1, <2>, A_2, <3> A_3 ;

(A conjunto de conjuntos A_1, A_2, A_3)

Set A[<i> in I] := {3 * i};

:= if "boolean" then {exp1}

else {exp 2}

Parametros:

Set A := {}

param q := 5

param b[A] := <a_1> b_1, <a_3> b_3 default b_d ;

param c[<i> in "set" with "boolean"] := e(i);

param h[I * J] := | j_1, j_2, j_3 |

| i_1 | x, x, x |

| i_3 | x, x, x | default h_d

g[I * I * I] := | i_2, i_4, i_5 |

| i_1, i_1 | |

| i_1, i_2 | |

| i_2, i_3 | |

h[I * J] := <i_1, j_1> $h_{i_1 j_1}$, —

~~...~~ CONJUNTOS Y PARAMETROS DESDE FICHERO

```
set P := { read "foo.txt" as "<1s, 5n, 2n>" skip 1
                                           use 2 }

param c[I] := read "c.txt" as "<1n> n"   comment #
```

Variables:

Por defecto: continuas $\in [0, \infty)$

```
var nombre tipo  >= l  <= u ;
```

tipo	$\geq l$	$\leq u$
binary	número	número
integer	-infinity	
real		

$\underbrace{\text{tipo} \geq l \leq u}_{\text{opcional}}$

```
var x[I]
```

Objetivo:

```
minimize
maximize   nombre: expresión ;

minimize coste: sum <i> in I : c[i] * x[i];
```

Restricciones:

```
subto nombre: expresión ≥≤ expresión
              expresión ≥≤ expresión ≥≤ expresión

subto r1: forall <a> in A : sum <a,b,c> in C : z[a,b,c] = 5
```

→ Con condiciones:

* En expresiones:

```
subto c2: sum <i> in I :
      if (i mod 2 == 0) then 3 * x[i]
      else -2 * y[i] end ≤ 3;
```

en la restricción misma.

```
subto c1: forall <i> in I do
    if (i mod 2 == 0) then 3 * x[i] >= 4
                      else -2 * y[i] <= 3 end;
```

Se pueden definir SOS

```
sos nombre: [tipo] [prior] expresión;

sos s1: type1 100*x[1] + ... + 300*x[5]
    s2: type2 sum <i> in I: a[i] * x[i]
```

Definición de funciones: no la vemos

Lo ejecutamos en SCIP: - read modelo.zpl
- optimize
- display solution / write solution sol.txt

PROBLEMA 8 REINAS: Como colocar 8 reinas en un tablero sin que se amenacen unas a otras

Obj: max $\sum x_{ij}$

sets: $I = \{1..8\}$, $d = \{-7..7\}$

$T := I \times I$

var $X[T]$ binaria

$\forall i \in I \quad \sum_j x_{ij} \leq 1$

$j \in I \quad \sum_i x_{ij} \leq 1$

$\forall k \in d \quad \sum_{i-j=k} x_{ij} \leq 1$

$\forall k \in d \quad \sum_{i+j=k+9} x_{ij} \leq 1$