Resolução de Problemas de Satisfação de Restrições e de Otimização em Sistemas de Programação por Restrições

Ana Paula Tomás

Departamento de Ciência de Computadores

Universidade do Porto

Abril 2018

Problema SEND+MORE=MONEY

Atribuir um dígito decimal distinto a cada letra de forma que a conta fique certa.

$$
\begin{array}{cccccc}
 & & S & E & N & D \\
+ & & M & O & R & E \\
\hline
 & M & O & N & E & Y
\end{array}
$$

Modelo matemático:

$$
\begin{aligned}
&(1000x_S + 100x_E + 10x_N + x_D) + (1000x_M + 100x_O + 10x_R + x_E) = \\
&= 10000x_M + 1000x_O + 100x_N + 10x_E + x_Y \\
&x_S \neq 0, x_M \neq 0 \\
&x_S, x_E, x_N, x_D, x_M, x_O, x_R, x_Y \in \{0,1,2,3,4,5,6,7,8,9\} \\
&x_S \neq x_E, x_S \neq x_N, x_S \neq x_D, x_S \neq x_M, x_S \neq x_O, x_S \neq x_R, x_S \neq x_Y, \\
&x_E \neq x_N, x_E \neq x_D, x_E \neq x_M, x_E \neq x_O, x_E \neq x_R, x_E \neq x_Y, \\
&x_N \neq x_D, x_N \neq x_M, x_N \neq x_O, x_N \neq x_R, x_N \neq x_Y, \\
&x_D \neq x_M, x_D \neq x_O, x_D \neq x_R, x_D \neq x_Y, \\
&x_M \neq x_O, x_M \neq x_R, x_M \neq x_Y, \\
&x_O \neq x_R, x_O \neq x_Y, \\
&x_R \neq x_Y
\end{aligned}
$$

Problema SEND+MORE=MONEY

Atribuir um dígito decimal distinto a cada letra de forma que a conta fique certa.

$$
\begin{array}{cccccc}
 & & S & E & N & D \\
+ & & M & O & R & E \\
\hline
 & M & O & N & E & Y
\end{array}
$$

Modelo matemático:

$$
\begin{aligned}
&(1000x_S + 100x_E + 10x_N + x_D) + (1000x_M + 100x_O + 10x_R + x_E) = \\
&= 10000x_M + 1000x_O + 100x_N + 10x_E + x_Y \\
&x_S \neq 0, x_M \neq 0 \\
&x_S, x_E, x_N, x_D, x_M, x_O, x_R, x_Y \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \\
&x_S \neq x_E, x_S \neq x_N, x_S \neq x_D, x_S \neq x_M, x_S \neq x_O, x_S \neq x_R, x_S \neq x_Y, \\
&x_E \neq x_N, x_E \neq x_D, x_E \neq x_M, x_E \neq x_O, x_E \neq x_R, x_E \neq x_Y, \\
&x_N \neq x_D, x_N \neq x_M, x_N \neq x_O, x_N \neq x_R, x_N \neq x_Y, \\
&x_D \neq x_M, x_D \neq x_O, x_D \neq x_R, x_D \neq x_Y, \\
&x_M \neq x_O, x_M \neq x_R, x_M \neq x_Y, \\
&x_O \neq x_R, x_O \neq x_Y, \\
&x_R \neq x_Y
\end{aligned}
$$

Problema SEND+MORE=MONEY

$$
\begin{array}{cccccc}
 & & S & E & N & D \\
+ & & M & O & R & E \\
\hline
 & M & O & N & E & Y
\end{array}
$$

Outro modelo para o problema (tradução do algoritmo de adição)

$$
\begin{array}{l}
x_D + x_E = x_Y + 10c_1 \\
c_1 + x_N + x_R = x_E + 10c_2 \\
c_2 + x_E + x_O = x_N + 10c_3 \\
c_3 + x_S + x_M = x_O + 10x_M \\
x_S \neq 0, x_M \neq 0 \\
x_S, x_E, x_N, x_D, x_M, x_O, x_R, x_Y \in \{0,1,2,3,4,5,6,7,8,9\} \\
c_1, c_2, c_3 \in \{0,1\} \\
x_S \neq x_E, x_S \neq x_N, x_S \neq x_D, x_S \neq x_M, x_S \neq x_O, x_S \neq x_R, x_S \neq x_Y, \\
x_E \neq x_N, x_E \neq x_D, x_E \neq x_M, x_E \neq x_O, x_E \neq x_R, x_E \neq x_Y, \\
x_N \neq x_D, \ldots, x_R \neq x_Y
\end{array}
$$

Programa (ECLiPSe) para SEND+MORE=MONEY

Utilização do sistema de CLP ECLiPSe (http://eclipseclp.org/)

```
:- use_module(library(ic)).

sendMoreMoney(T) :-
   T = [s-S,e-E,n-N,d-D,m-M,o-O,r-R,y-Y], L = [S,E,N,D,M,O,R,Y],
   L #:: 0..9,   [C_1,C_2,C_3] #:: 0..1,
   D+E #= Y+10*C_1,
   C_1+N+R #= E+10*C_2,
   C_2+E+O #= N+10*C_3,
   C_3+S+M #= O+10*M,
   S #\= 0, M #\= 0,
   S #\= E, S #\= N, S #\= D, S #\= M, S #\= O, S #\= R, S #\= Y,
   E #\= N, E #\= D, E #\= M, E #\= O, E #\= R, E #\= Y, N #\= D,
   N #\= M, N #\= O, N #\= R, N #\= Y,
   D #\= M, D #\= O, D #\= R, D #\= Y,
   M #\= O, M #\= R, M #\= Y, O #\= R, O #\= Y, R #\= Y,
   labeling(L).
```

Propagação de restrições não basta. **labeling**: procura valores para as variáveis.

Programa para resolução de SEND+MORE=MONEY

```
[eclipse 4]: compile('v0_send_more_money.ecl').
v0_send_more_money.ecl compiled 15160 bytes in 0.01 seconds

Yes (0.01s cpu)

[eclipse 5]: sendMoreMoney(T).

T = [s - 9, e - 5, n - 6, d - 7, m - 1, o - 0, r - 8, y - 2]
Yes (0.00s cpu, solution 1, maybe more) ? ;

No (0.00s cpu)
```

SEND+MORE=MONEY com restrições globais

```
:- use_module(library(ic)).
:- use_module(library(ic_global)).

sendMoreMoney(T) :-
  T = [s-S,e-E,n-N,d-D,m-M,o-O,r-R,y-Y],  L = [S,E,N,D,M,O,R,Y],
  L #:: 0..9,  [C_1,C_2,C_3] #:: 0..1,
  D+E #= Y+10*C_1,
  C_1+N+R #= E+10*C_2,
  C_2+E+O #= N+10*C_3,
  C_3+S+M #= O+10*M,
  S #\= 0, M #\= 0,
  ic_global:alldifferent(L),
  labeling(L).
```

ic_global:alldifferent(L) e **ic:alldifferent(L)** são propagadas de forma distinta. Para **ic**, a propagação é análoga a `#\=`, como em `v0_send_more_money.ecl`.

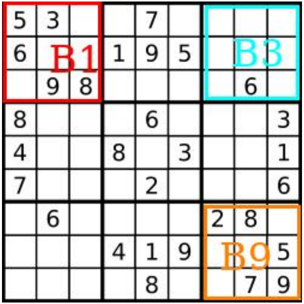
Sudoku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

É dada uma grelha 9×9, subdividida em 9 *regiões* quadradas (3×3). Algumas células têm dígitos inicialmente.

Objetivo: Preencher as restantes células com números de 1 a 9, de modo que em cada *linha*, *coluna* e *região* todos os números sejam distintos.

Sudoku



Variáveis de decisão:

X_{ij} é o valor que coloca na célula (i, j),

Restrições:

- $X_{ij} \in \{1, 2, \ldots, 9\}$, para todo (i, j).
- $X_{ij} = d_{ij}$, se d_{ij} for dado para (i, j).
- $alldifferent([X_{i1}, X_{i2}, \ldots, X_{i9}])$, para todo i.
- $alldifferent([X_{1j}, X_{2j}, \ldots, X_{9j}])$, para todo j.
- $alldifferent(B_k)$, para todo k.

com:

$B_1 = [x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}]$,

$B_2 = [x_{14}, x_{15}, x_{16}, x_{24}, x_{25}, x_{26}, x_{34}, x_{35}, x_{36}]$,

$\vdots$

,

$B_9 = [x_{77}, x_{78}, x_{79}, x_{87}, x_{88}, x_{89}, x_{97}, x_{98}, x_{99}]$.

Programa (ECLiPSe) – Sudoku

```
:- use_module(library(ic)).
:- use_module(library(ic_global)).

sudoku(G) :-  dados(G), flatten(G,Gf), Gf #:: 1..9,
   restrs_linhas(G),
   transposta(G,Gt), restrs_linhas(Gt),
   blocos(G,Bs), restrs_linhas(Bs),
   labeling(Gf).

dados([[5,3,_,_,7,_,_,_,_],
       [6,_,_,1,9,5,_,_,_],
       [_,9,8,_,_,_,_,6,_],
       [8,_,_,_,6,_,_,_,3],
       [4,_,_,8,_,3,_,_,1],
       [7,_,_,_,2,_,_,_,6],
       [_,6,_,_,_,_,2,8,_],
       [_,_,_,4,1,9,_,_,5],
       [_,_,_,_,8,_,_,7,9]]).
```

```
restrs_linhas([]).
restrs_linhas([L|RLinhas]) :- ic_global:alldifferent(L),
    restrs_linhas(RLinhas).

blocos([],[]).
blocos([L1,L2,L3|RLinhas],Blocos) :-
    blocos_(L1,L2,L3,Blocos,RBlocos),
    blocos(RLinhas,RBlocos).

blocos_([],[],[],RBsF,RBsF).
blocos_([A,B,C|RL1],[D,E,F|RL2],[G,H,I|RL3],[Bloco|RBs],RBsF) :-
    Bloco = [A,B,C,D,E,F,G,H,I],
    blocos_(RL1,RL2,RL3,RBs,RBsF).
```

Programa (ECLiPSe) – Sudoku

```
%--- Auxiliar -------------

transposta([],[]).
transposta([X],Sxs) :- transposta_(X,Sxs).
transposta([X,Y|R],Tf) :- transposta([Y|R],T),
   transposta__(X,T,Tf).

transposta_([],[]).
transposta_([X|R],[[X]|Rs]) :- transposta_(R,Rs).

transposta__([],[],[]).
transposta__([X|R],[Tx|Rt],[[X|Tx]|RRt]) :-
    transposta__(R,Rt,RRt).
```

Problema de Trocos - ToPAS 2012

Pretendemos formar uma quantia Q com no máximo N moedas. Existem n tipos de moedas, sendo os seus valores conhecidos. Existem pelo menos N moedas de cada tipo. Se não for possível formar Q, qual seria a quantia mais próxima de Q que se consegue formar (não inferior a Q)?
(Baseado no Problema F do ToPAS 2012)

Problema de Trocos - ToPAS 2012

Pretendemos formar uma quantia Q com no máximo N moedas. Existem n tipos de moedas, sendo os seus valores conhecidos. Existem pelo menos N moedas de cada tipo. Se não for possível formar Q, qual seria a quantia mais próxima de Q que se consegue formar (não inferior a Q)? Que moedas usa?

- Dados:

  Q, N, n, e os valores das moedas $v_1 \geq v_2 \geq \ldots \geq v_n$

- Variáveis:

  x_i indica quantas moedas do tipo i usa, para $1 \leq i \leq n$

  y indica o excesso do valor formado relativamente a Q

- Restrições:

  $\sum_{i=1}^{n} x_i \leq N$ (não usa mais do que N moedas)
  $\sum_{i=1}^{n} v_i x_i = Q + y$ (a quantia formada é $Q + y$)
  $0 \leq x_i \leq N$, para todo i
  $0 \leq y \leq v_1 N - Q$

- Objetivo: minimizar y

Problema de Trocos - ToPAS 2012

```
:- lib(ic).
:- lib(branch_and_bound).

moedas(Quantia,Nmax,Valores,Y) :-
   length(Valores,Nv), length(X,Nv),
   X #::0..Nmax,
   Valores = [Vmax|_], Ymax is Nmax*Vmax-Quantia,
   Y #::0..Ymax,
   X*Valores #= Quantia+Y,
   sum(X) #=< Nmax,
   minimize(labeling([Y|X]),Y),
   escrever_sol(X,Y,Quantia).

escrever_sol(X,Y,Quantia) :-  escr_lista(X),
   Quantiaf is Quantia+Y, nl, write(quantia = Quantiaf).

escr_lista([]).
escr_lista([X|R]) :-  write(X), nl, escr_lista(R).
```

Exemplo – Ases de Santa Cruz

O clube “Ases de Santa Cruz” organizou o seu campeonato anual de patinagem artística. Para a fase final apuraram-se três concorrentes (Ângela, Nisa, Cláudia) que tiveram de disputar então uma série de provas. Em cada uma destas provas, a primeira classificada obtinha 6 pontos, a segunda 3 e a terceira 1, não havendo empates. O campeonato foi muito renhido. Veja-se, por exemplo, que nas duas primeiras provas a Ângela ficou em segundo lugar, a Nisa ganhou uma e a Cláudia ganhou outra. A classificação final foi a seguinte:

1. Nisa — 41 pontos, 2. Cláudia — 40 pontos e 3. Ângela — 39 pontos.

Quais foram as classificações da campeã Nisa nas várias provas? E as das outras concorrentes? (em J. Público, 10.1.99)

Exemplo – Ases de Santa Cruz (cont)

Dados

n_c	número de concorrentes (igual ao número de lugares)
b_j	total de pontos da concorrente j no campeonato
p_k	pontos atribuídos pelo lugar k
N	número de provas, $N = (\sum_j b_j)/(\sum_k p_k)$

Variáveis de Decisão

x_{ij} pontos da concorrente j na prova i, com $j = 1, \ldots, n_c$, $i = 1, \ldots, N$

(Caso particular $n_c = 3$: 1-Nisa, 2-Claúdia, 3-Ângela)

Exemplo – Ases de Santa Cruz (cont)

Restrições para um modelo genérico

- $x_{ij} \in \{p_k \mid k = 1, \ldots, n_c\}$
- Total de pontos obtidos pela concorrente j é b_j

$$\sum_{i=1}^{N} x_{ij} = b_j, \quad j = 1, \ldots, n_c$$

- Não há empates

$$x_{ij} \neq x_{ij'} \text{ se } j \neq j', \text{ com } 1 \leq j < j' \leq n_c, 1 \leq i \leq N$$

- Eliminar simetria (sem os dados das duas primeiras provas)

$$(x_{i\,1}, \ldots, x_{i\,n_c}) \preceq_{LEX} (x_{i+1\,1}, \ldots, x_{i+1\,n_c}), \text{ para } 1 \leq i < N$$

(sem estas restrições, qualquer permutação das linhas da matriz é uma solução)

Exemplo – Ases de Santa Cruz (cont)

Versão J. Público: Ângela (39), Nisa (41), Cláudia (40) N: 120/10 = 12

- $x_{ij} \in \{1, 3, 6\}$
- Total de pontos obtidos pela concorrente j é b_j, sendo $\mathbf{b} = [39, 41, 40]$

$$\sum_{i=1}^{12} x_{ij} = b_j, \quad \text{para } j = 1, 2, 3$$

- Não há empates

$$x_{ij} \neq x_{ij'} \text{ se } j \neq j', \text{ com } 1 \leq j < j' \leq 3, \; 1 \leq i \leq 12$$

- Pontos nas duas primeiras provas:

$$x_{11} = x_{21} = 3, \; x_{12} + x_{22} = 7, \text{ e } x_{13} + x_{23} = 7$$

- Eliminar simetria (compatível com dados das duas primeiras provas):

$$(x_{1\,1}, x_{1\,2}, x_{1\,3}) \preceq_{LEX} (x_{2\,1}, x_{2\,2}, x_{2\,3})$$

$$(x_{i\,1}, x_{i\,2}, x_{i\,3}) \preceq_{LEX} (x_{i+1\,1}, x_{i+1\,2}, x_{i+1\,3}), \text{ para } 3 \leq i < 12$$

Programa (ECLiPSe) para "Ases de Santa Cruz"

```
:-lib(ic).
:-lib(ic_global).

asesStaCruz :- pontos(Xt),
  write_sol(['Angela','Nisa','Claudia'],Xt).

pontos(Xt) :-
   matriz(12,3,X),
   flatten(X,Xf),
   Xf #:: [1,3,6],
   sem_empates(X), X=[X1,X2|Xnxt], ic_global:lex_le(X1,X2),
   elimina_simetria(Xnxt),
   transposta(X,Xt), Xt = [PtAngela,PtNisa,PtClaudia],
   PtAngela = [3,3|_],PtNisa = [N1,N2|_], N1+N2 #= 7,
   PtClaudia = [C1,C2|_], C1+C2 #= 7,
   total_pontos(Xt,[39,41,40]),
   length(Xf,N), write(N),nl, labeling(Xf).
```

Programa (ECLiPSe) para “Ases de Santa Cruz”

```
%--------  Outras restricoes ---------------------------
sem_empates([]).
sem_empates([X|R]) :- ic_global:alldifferent(X),
  sem_empates(R).

elimina_simetria([]).
elimina_simetria([_]).
elimina_simetria([X,Y|R]) :- ic_global:lex_le(X,Y),
  elimina_simetria([Y|R]).

total_pontos([],[]).
total_pontos([X|R],[Tx|Tr]) :- sum(X) #= Tx,
  total_pontos(R,Tr).
```

Programa (ECLiPSe) para “Ases de Santa Cruz”

```
%-----   auxiliares ------------------
matriz(0,_,[]) :- !.
matriz(M,N,[Vs|LVs]) :- length(Vs,N),
   Mm is M-1, matriz(Mm,N,LVs).

write_sol([],[]).
write_sol([Nome|RNomes],[Pt|RPs]) :-
   write(Nome), nl, write(Pt),nl, nl,
   write_sol(RNomes,RPs).

%-- transposta/2
% predicado definido como no problema "Sudoku"
```

Equipa de Natação (4x100 metros estilos)

Um clube de natação tem vinte nadadores, cada um deles capaz de nadar os quatro estilos – costas, bruços, mariposa e crawl. O treinador conhece o tempo médio de cada nadador em cada um dos estilos para provas de 100 metros. Pretende formar uma equipa para os 4×100 metros estilos. Quais escolher?

Variáveis de decisão: x_{ij} indica se o nadador i nada o estilo j ou não.

Dados: t_{ij} é o tempo médio do nadador i para o estilo j, m o número de nadadores e n o número de estilos.

Modelo de Programação Inteira com variáveis booleanas:

$$\text{minimizar} \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij} x_{ij} \quad \text{sujeito a}$$
$$\begin{cases} \sum_{i=1}^{m} x_{ij} = 1 \quad \text{para } j = 1, 2, \ldots, n & \text{(o estilo } j \text{ é atribuído a um nadador)} \\ \sum_{j=1}^{n} x_{ij} \leq 1, \quad \text{para } i = 1, 2, \ldots, m & \text{(o nadador } i \text{ nada no máximo um estilo)} \\ x_{ij} \in \{0, 1\}, \quad \text{para todo } (i, j) \end{cases}$$

Equipa de Natação (4x100 metros estilos)

Um clube de natação tem vinte nadadores, cada um deles capaz de nadar os quatro estilos – costas, bruços, mariposa e crawl. O treinador conhece o tempo médio de cada nadador em cada um dos estilos para provas de 100 metros. Pretende formar uma equipa para os 4×100 metros estilos. Quais escolher?

Variáveis de decisão: x_{ij} indica se o nadador i nada o estilo j ou não.

Dados: t_{ij} é o tempo médio do nadador i para o estilo j, m o número de nadadores e n o número de estilos.

Modelo de Programação Inteira com variáveis booleanas:

$$\text{minimizar} \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij} x_{ij} \quad \text{sujeito a}$$

$$\begin{cases} \sum_{i=1}^{m} x_{ij} = 1 \quad \text{para } j = 1, 2, \ldots, n & \text{(o estilo } j \text{ é atribuído a um nadador)} \\ \sum_{j=1}^{n} x_{ij} \leq 1, \quad \text{para } i = 1, 2, \ldots, m & \text{(o nadador } i \text{ nada no máximo um estilo)} \\ x_{ij} \in \{0, 1\}, \quad \text{para todo } (i, j) \end{cases}$$

Equipa de Natação (4x100 metros estilos)

Um clube de natação tem vinte nadadores, cada um deles capaz de nadar os quatro estilos – costas, bruços, mariposa e crawl. O treinador conhece o tempo médio de cada nadador em cada um dos estilos para provas de 100 metros. Pretende formar uma equipa para os 4×100 metros estilos. Quais escolher?

Variáveis de decisão: x_{ij} indica se o nadador i nada o estilo j ou não.

Dados: t_{ij} é o tempo médio do nadador i para o estilo j, m o número de nadadores e n o número de estilos.

Modelo de Programação Inteira com variáveis booleanas:

$$\text{minimizar} \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij} x_{ij} \quad \text{sujeito a}$$
$$\begin{cases} \sum_{i=1}^{m} x_{ij} = 1 \quad \text{para } j = 1, 2, \ldots, n & \text{(o estilo } j \text{ é atribuído a um nadador)} \\ \sum_{j=1}^{n} x_{ij} \leq 1, \quad \text{para } i = 1, 2, \ldots, m & \text{(o nadador } i \text{ nada no máximo um estilo)} \\ x_{ij} \in \{0, 1\}, \quad \text{para todo } (i, j) \end{cases}$$

Equipa de Natação – Modelo de CP

Um clube de natação tem vinte nadadores, cada um deles capaz de nadar os quatro estilos – costas, bruços, mariposa e crawl. O treinador conhece o tempo médio de cada nadador em cada um dos estilos para provas de 100 metros. Pretende formar uma equipa para os 4×100 metros estilos. Quais escolher?

Modelo de Programação por Restrições:

y_j identifica o nadador que nadará o estilo j

minimizar $\sum_{j=1}^{n} t_{y_j j}$
sujeito a
$y_j \in \{1, 2, \ldots, m\}$, para todo j
$y_j \neq y_{j'}$, para todo (j, j') com $j \neq j'$

Notar que $t_{y_j j}$ tem **uma variável como índice** (não é uma constante).

Equipa de Natação – Modelo de CP

Um clube de natação tem vinte nadadores, cada um deles capaz de nadar os quatro estilos – costas, bruços, mariposa e crawl. O treinador conhece o tempo médio de cada nadador em cada um dos estilos para provas de 100 metros. Pretende formar uma equipa para os 4×100 metros estilos. Quais escolher?

Modelo de Programação por Restrições:

y_j identifica o nadador que nadará o estilo j

minimizar $\sum_{j=1}^{n} t_{y_j j}$
sujeito a
$y_j \in \{1, 2, \ldots, m\}$, para todo j
$y_j \neq y_{j'}$, para todo (j, j') com $j \neq j'$

Notar que $t_{y_j j}$ tem **uma variável como índice** (não é uma constante).

Equipa de Natação – Modelo de CP

Um clube de natação tem vinte nadadores, cada um deles capaz de nadar os quatro estilos – costas, bruços, mariposa e crawl. O treinador conhece o tempo médio de cada nadador em cada um dos estilos para provas de 100 metros. Pretende formar uma equipa para os 4×100 metros estilos. Quais escolher?

Modelo de Programação por Restrições:

y_j identifica o nadador que nadará o estilo j

minimizar $\sum_{j=1}^{n} t_{y_j j}$

sujeito a

$y_j \in \{1, 2, \ldots, m\}$, para todo j

$y_j \neq y_{j'}$, para todo (j, j') com $j \neq j'$

Notar que $t_{y_j j}$ tem **uma variável como índice** (não é uma constante).

Programa (ECLiPSe) – Equipa de Natação

```
:- use_module(library(ic)).
:- use_module(library(ic_global)).
:- use_module(library(branch_and_bound)).

:- compile('dados.ecl').     % define nadadores/1 e tempos/1

quatroEstilos :- nadadores(M), tempos(T),
   quatroEstilos(M,T,TempoEquipa,Y), escrResultado(Y,TempoEquipa).

quatroEstilos(M,T,TempoEquipa,Y) :-
   length(Y,4), Y #:: 1..M,
   length(TemposY,4), transposta(T,Tt),
   ligar_tempos(Tt,Y,TemposY),
   sum(TemposY) #= TempoEquipa,
   ic_global:alldifferent(Y),
%  minimize(labeling(Y),TempoEquipa)                     % 1
   minimize(labeling([TempoEquipa|Y]),TempoEquipa).     % 2
```

Programa (ECLiPSe) – Equipa de Natação

Utilização da restrição element/2

```
ligar_tempos([],[],[]).
ligar_tempos([TpAtletasJ|RTpAtletas],[Yj|RY],[TempoYj|RTempY]) :-
    element(Yj,TpAtletasJ,TempoYj),
    ligar_tempos(RTpAtletas,RY,RTempY).


escrResultado(Y,TempoEquipa) :-
   write(Y), nl, write('Tempo Optimo':TempoEquipa),
   nl.

% predicado  transposta/2 como em "Sudoku"
```

Programa (ECLiPSe) – Equipa de Natação

Resultados para a Estratégia 1: `minimize(labeling(Y),TempoEquipa)`

```
[eclipse 2]: quatroEstilos.
lists.eco  loaded in 0.00 seconds
Found a solution with cost 41
Found a solution with cost 40
Found a solution with cost 37
Found a solution with cost 35
Found a solution with cost 32
Found a solution with cost 31
Found a solution with cost 30
Found a solution with cost 29
Found a solution with cost 28
Found a solution with cost 27
Found a solution with cost 24
Found a solution with cost 20
Found a solution with cost 18
Found a solution with cost 17
Found a solution with cost 12
Found a solution with cost 10
Found a solution with cost 9
Found no solution with cost 8.0 .. 8.0
[3, 6, 8, 4]
Tempo Optimo : 9
```

Programa (ECLiPSe) – Equipa de Natação

Resultados para a Estratégia 2:

```
minimize(labeling([TempoEquipa|Y]),TempoEquipa)

[eclipse 4]: quatroEstilos.
Found a solution with cost 9
Found no solution with cost 8.0 .. 8.0
[3, 6, 8, 4]
Tempo Optimo : 9
```

labeling atribui à primeira variável da lista o valor mínimo do seu domínio tenta verificar se é possível completar a solução para as restantes variáveis.

Problema de Corte de Andares

Uma empresa muda de instalações passando a ocupar quatro andares que terão de ser divididos. Os possíveis modos de divisão (a – j), os respectivos custos, e as salas necessárias constam da tabela seguinte. Pretende-se uma solução de custo mínimo.

Necessárias	Sala	a	b	c	d	e	f	g	h	i	j
1	G. Dir.	1	1	1	1	1					
6	Gab.	2			1	4	3	2	2		
2	S. Máq.		2		1				1		2
2	Ar. Gr.			1				1		2	
2	Ar. Peq.	1		1			1		1	1	1
3	Hall			1	1		1	1			1
	Custo	10	20	12	11	15	12	10	12	15	14

Corte de Andares – Três modelos matemáticos

Dados

$\mathcal{A}$	–	número de andares ($\mathcal{A} = 4$)
$\mathcal{C}$	–	número de cortes ($\mathcal{C} = 10$)
$\mathcal{S}$	–	número de tipos de salas ($\mathcal{S} = 6$)
c_j	–	custo duma divisão do tipo j
a_{ij}	–	número de salas do tipo i obtidas na divisão j
b_i	–	número de salas do tipo i necessárias, $i \in \{1, \ldots, \mathcal{S}\}$

Três possibilidades para escolha das variáveis de decisão

$w_{kj} \in \{0, 1\}$	–	se aplica corte j no andar k, com $j \in \{1, \ldots, \mathcal{C}\}$ e $k \in \{1, \ldots, \mathcal{A}\}$
$x_j \in \{0, 1, 2, 3, 4\}$	–	número de vezes que se usa divisão $j \in \{1, \ldots, \mathcal{C}\}$
$y_k \in \{1, 2, 3, \ldots, \mathcal{C}\}$	–	tipo de corte para andar k, com $k \in \{1, \ldots, \mathcal{A}\}$

Corte de Andares – Três modelos matemáticos

Estimativas de dimensão do espaço de procura por força-bruta

- $\boxed{\text{Modelo I} \quad (w_{kj} \in \{0,1\})}$

  $|\mathcal{A} \times \mathcal{C}| = 40$ variáveis binárias ...... $2^{40} = 1\ 099\ 511\ 627\ 776$

- $\boxed{\text{Modelo II} \quad (x_j \in \{0,1,2,\ldots,\mathcal{A}\})}$

  $|\mathcal{C}| = 10$ variáveis inteiras ...... $5^{10} = 9\ 765\ 625$

- $\boxed{\text{Modelo III} \quad (y_k \in \{1,2,\ldots,\mathcal{C}\})}$

  $|\mathcal{A}| = 4$ variáveis inteiras ...... $10^4 = 10\ 000$

Corte de Andares – Modelo II e III

Prog. Inteira

minimizar $\sum_{j=1}^{\mathcal{C}} c_j x_j$

$$\sum_{j=1}^{\mathcal{C}} a_{ij} x_j = b_i, \; i = 1, \ldots, \mathcal{S}$$

$$\sum_{j=1}^{\mathcal{C}} x_j = \mathcal{A}$$

$$x_j \in \mathbb{Z}_0^+, j = 1, \ldots, \mathcal{C}$$

Prog. por Restrições

minimizar $\sum_{k=1}^{\mathcal{A}} c_{y_k}$

$$\sum_{k=1}^{\mathcal{A}} a_{iy_k} = b_i, \; i = 1, \ldots, \mathcal{S}$$

$$y_k \in \{1, \ldots, \mathcal{C}\}, k = 1, \ldots, \mathcal{A}$$

$$y_k \leq y_{k+1}, \; 1 \leq k < \mathcal{A}$$

(evitar simetrias)

Programa (ECLiPSe) - Corte de Andares - II (IP)

```
%-- Modelo II (Prog. Inteira)
:- lib(ic).
:- lib(branch_and_bound).

% chamada -- solve(dataAndares).

solve(Ex) :-  open(Ex,read,StrEx), read(StrEx,A), read(StrEx,B),
  read(StrEx,C), read(StrEx,Andares), close(StrEx),
  length(C,NVars), length(Xs,NVars),
  Xs #:: 0..Andares, sum(Xs) #= Andares,
  restrs(A,Xs,B),
  C*Xs #= Z,
  minimize(labeling(Xs),Z),
  nl, write('optimal solution' ), nl, write_sol(Z,Xs).

restrs([],_,[]).
restrs([Ai|A],X,[AiX|AX]) :- Ai*X #= AiX, restrs(A,X,AX).
```

Programa (ECLiPSe) - Corte de Andares - II (IP)

```
write_sol(Z,X) :- write('Custo: '), write(Z), nl,
  write_sol_(X,1), nl.

write_sol_([],_).
write_sol_([0|Xs],N) :- !, Nn is N+1, write_sol_(Xs,Nn).
write_sol_([X|Xs],N) :- write(' tipo '), write(N),
  write(': '), write(X), nl,  Nn is N+1, write_sol_(Xs,Nn).
```

Programa (ECLiPSe) - Corte de Andares - III (CP)

```
%-- Modelo III (Prog. Restricoes)
:- lib(ic).
:- lib(branch_and_bound).

% chamada -- solve(dataAndares).

solve(Ex) :- open(Ex,read,StrEx), read(StrEx,A), read(StrEx,B),
    read(StrEx,C), read(StrEx,Andares), close(StrEx),
    length(C,NCortes),  length(B,NSalas),
    length(CYks,Andares), sum(CYks) #= Z ,
    length(Yks,Andares), Yks #:: 1..NCortes,
    restrs_simetria(Yks),
    liga_custo(Yks,CYks,C),
    matriz(NSalas,Andares,AYks),
    restr_aij(A,B,AYks,Yks),
    minimize(labeling(Yks),Z),
    write_sol(Z,Yks).
```

Programa (ECLiPSe) - Corte de Andares - III (CP)

```
matriz(0,_,[]) :- !.
matriz(M,N,[Vs|LVs]) :- length(Vs,N),
  Mm is M-1, matriz(Mm,N,LVs).

restrs_simetria([_]).
restrs_simetria([Yk,Yk1|Ys]) :- Yk #=< Yk1,
  restrs_simetria([Yk1|Ys]).

% ligar Cyk ao valor do custo de Yk
liga_custo([],_,_).
liga_custo([Yk|Ys],[Ck|CYks],Cs) :- element(Yk,Cs,Ck),
  liga_custo(Ys,CYks,Cs).
```

Programa (ECLiPSe) - Corte de Andares - III (CP)

```
% restricoes do problema
restr_aij([],[],_,_).
restr_aij([Ai|A],[Bi|B],[AYki|AYks],Yks) :-
  sum(AYki) #= Bi, liga_aij_(Yks,AYki,Ai),
  restr_aij(A,B,AYks,Yks).

% ligar Aiyk a Yk
liga_aij_([],[],_).
liga_aij_([Yk|Yks],[AiYk|AYki],Ai) :- element(Yk,Ai,AiYk),
  liga_aij_(Yks,AYki,Ai).

% escrever solucao
write_sol(Z,X) :- write('Custo: '), write(Z), nl,
     write_list(X), nl.

write_list([]).
write_list([X|Xs]) :- write(X), put(32), write_list(Xs).
```

Dados (ECLiPSe) - Corte de Andares

Ficheiro de dados:

```
[[1,0,1,1,0,0,0,0,0,1],
 [2,2,1,0,3,2,0,4,0,0],
 [0,0,1,0,0,1,2,0,0,2],
 [0,1,0,1,0,0,0,0,2,0],
 [1,0,0,1,1,1,1,0,1,0],
 [0,1,1,1,1,0,1,0,0,0]].

[1,6,2,2,2,3].
[10, 10, 11, 12, 12, 12, 14, 15, 15, 20].
4.
```

Problema de Corte 1D

Uma fábrica de papel produz o seu papel em rolos padrão com largura de 20 pés e corta esses rolos noutras larguras menores, e possivelmente diferentes, para satisfazer as encomendas dos seus clientes. Neste processo resulta geralmente o desperdício de alguma porção de cada rolo. A fábrica recebeu as encomendas a seguir indicadas e naturalmente deseja cortar os rolos padrão de modo a minimizar o desperdício total. Quaisquer sobras são consideradas desperdício. Que tipos de corte pode fazer sentido efetuar? Quantas vezes deve aplicar cada tipo de corte para satisfazer a encomenda?

largura (pés)	9	7	5.5
número de rolos	31	150	65

Problema de Corte 1D

Uma fábrica de papel produz o seu papel em rolos padrão com comprimento de 20 pés e corta esses rolos noutros de dimensão menores, e possivelmente diferentes, para satisfazer as encomendas dos seus clientes. Neste processo resulta geralmente o desperdício de alguma porção de cada rolo. A fábrica recebeu as encomendas a seguir indicadas e naturalmente deseja cortar os rolos padrão de modo a minimizar o desperdício total. Quaisquer sobras são consideradas desperdício. Que tipos de corte pode fazer sentido efetuar? Quantas vezes deve aplicar cada tipo de corte para satisfazer a encomenda?

comprimento (pés)	9	7	5.5
número de rolos	31	150	65

Tipos de corte a considerar:

Tipo	**9**	**7**	**5.5**	Sobra
1	2	0	0	2
2	1	1	0	4
3	1	0	2	0
4	0	2	1	0.5
5	0	1	2	2
6	0	0	3	3.5

Supomos que já os calculámos.

- Dados: q_i números de rolos do tipo i necessários; a_{ij} quantos rolos do tipo i produz por corte j.
- Variáveis: x_j quantas vezes usa o corte j, com $j = 1, \ldots, 6$
- Restrições: $\sum_{j=1}^{6} a_{ij} x_j \geq q_i$, com $i = 1, 2, 3$
- Objetivo: $\min \sum_{j=1}^{6} x_j$

Problema de Corte 1D

Uma fábrica de papel produz o seu papel em rolos padrão com comprimento de 20 pés e corta esses rolos noutros de dimensão menores, e possivelmente diferentes, para satisfazer as encomendas dos seus clientes. Neste processo resulta geralmente o desperdício de alguma porção de cada rolo. A fábrica recebeu as encomendas a seguir indicadas e naturalmente deseja cortar os rolos padrão de modo a minimizar o desperdício total. Quaisquer sobras são consideradas desperdício. Que tipos de corte pode fazer sentido efetuar? Quantas vezes deve aplicar cada tipo de corte para satisfazer a encomenda?

comprimento (pés)	9	7	5.5
número de rolos	31	150	65

Tipos de corte a considerar:

Tipo	**9**	**7**	**5.5**	Sobra
1	2	0	0	2
2	1	1	0	4
3	1	0	2	0
4	0	2	1	0.5
5	0	1	2	2
6	0	0	3	3.5

Supomos que já os calculámos.

- Dados: q_i números de rolos do tipo i necessários; a_{ij} quantos rolos do tipo i produz por corte j.
- Variáveis: x_j quantas vezes usa o corte j, com $j = 1, \ldots, 6$
- Restrições: $\sum_{j=1}^{6} a_{ij} x_j \geq q_i$, com $i = 1, 2, 3$
- Objetivo: $\min \sum_{j=1}^{6} x_j$

Problema de Corte 1D

Uma fábrica de papel produz o seu papel em rolos padrão com comprimento de 20 pés e corta esses rolos noutros de dimensão menores, e possivelmente diferentes, para satisfazer as encomendas dos seus clientes. Neste processo resulta geralmente o desperdício de alguma porção de cada rolo. A fábrica recebeu as encomendas a seguir indicadas e naturalmente deseja cortar os rolos padrão de modo a minimizar o desperdício total. Quaisquer sobras são consideradas desperdício. Que tipos de corte pode fazer sentido efetuar? Quantas vezes deve aplicar cada tipo de corte para satisfazer a encomenda?

comprimento (pés)	9	7	5.5
número de rolos	31	150	65

Tipos de corte a considerar:

Tipo	**9**	**7**	**5.5**	Sobra
1	2	0	0	2
2	1	1	0	4
3	1	0	2	0
4	0	2	1	0.5
5	0	1	2	2
6	0	0	3	3.5

Supomos que já os calculámos.

- Dados: q_i números de rolos do tipo i necessários; a_{ij} quantos rolos do tipo i produz por corte j.
- Variáveis: x_j quantas vezes usa o corte j, com $j = 1, \ldots, 6$
- Restrições: $\sum_{j=1}^{6} a_{ij} x_j \geq q_i$, com $i = 1, 2, 3$
- Objetivo: $\min \sum_{j=1}^{6} x_j$

Problema de Corte 1D

Uma fábrica de papel produz o seu papel em rolos padrão com comprimento de 20 pés e corta esses rolos noutros de dimensão menores, e possivelmente diferentes, para satisfazer as encomendas dos seus clientes. Neste processo resulta geralmente o desperdício de alguma porção de cada rolo. A fábrica recebeu as encomendas a seguir indicadas e naturalmente deseja cortar os rolos padrão de modo a minimizar o desperdício total. Quaisquer sobras são consideradas desperdício. Que tipos de corte pode fazer sentido efetuar? Quantas vezes deve aplicar cada tipo de corte para satisfazer a encomenda?

comprimento (pés)	9	7	5.5
número de rolos	31	150	65

Tipos de corte a considerar:

Tipo	**9**	**7**	**5.5**	Sobra
1	2	0	0	2
2	1	1	0	4
3	1	0	2	0
4	0	2	1	0.5
5	0	1	2	2
6	0	0	3	3.5

Supomos que já os calculámos.

- Dados: q_i números de rolos do tipo i necessários; a_{ij} quantos rolos do tipo i produz por corte j.
- Variáveis: x_j quantas vezes usa o corte j, com $j = 1, \ldots, 6$
- Restrições: $\sum_{j=1}^{6} a_{ij} x_j \geq q_i$, com $i = 1, 2, 3$
- Objetivo: $\min \sum_{j=1}^{6} x_j$

Problema de Corte 1D

Uma fábrica de papel produz o seu papel em rolos padrão com comprimento de 20 pés e corta esses rolos noutros de dimensão menores, e possivelmente diferentes, para satisfazer as encomendas dos seus clientes. Neste processo resulta geralmente o desperdício de alguma porção de cada rolo. A fábrica recebeu as encomendas a seguir indicadas e naturalmente deseja cortar os rolos padrão de modo a minimizar o desperdício total. Quaisquer sobras são consideradas desperdício. Que tipos de corte pode fazer sentido efetuar? Quantas vezes deve aplicar cada tipo de corte para satisfazer a encomenda?

comprimento (pés)	9	7	5.5
número de rolos	31	150	65

Tipos de corte a considerar:

Tipo	**9**	**7**	**5.5**	Sobra
1	2	0	0	2
2	1	1	0	4
3	1	0	2	0
4	0	2	1	0.5
5	0	1	2	2
6	0	0	3	3.5

Supomos que já os calculámos.

- Dados: q_i números de rolos do tipo i necessários; a_{ij} quantos rolos do tipo i produz por corte j.
- Variáveis: x_j quantas vezes usa o corte j, com $j = 1, \ldots, 6$
- Restrições: $\sum_{j=1}^{6} a_{ij} x_j \geq q_i$, com $i = 1, 2, 3$
- Objetivo: $\min \sum_{j=1}^{6} x_j$

Problema de Corte 1D - Determinar os tipos de corte

- Dados
  - l_i comprimento de um rolo do tipo i, necessário para a encomenda
  - L comprimento do rolo original
  - $Min = \min\{l_i \mid ; 1 \leq i \leq m\}$
- Variáveis de decisão

  $y_i \in \mathbb{Z}_0^+$ número de rolos do tipo i que se obtém num corte
- Restrições

$$\sum_{i=1}^{m} l_i y_i \leq L$$

comprimento total do corte não pode exceder o comprimento do rolo original

$$L - \sum_{i=1}^{m} l_i y_i < Min$$

garante que o que sobra é inferior à dimensão do menor rolo da encomenda

Problema de Cobertura "Postos de Vigia"

A Rede Nacional de Postos de Vigia constitui um dos principais mecanismos de detecção e localização inicial de incêndios florestais, sendo assegurada por vigilantes. É dada prioridade à vigilância de regiões que apresentam maior risco de incêndio. Suponha que $\mathcal{R}$ designa o conjunto dessas regiões, $\mathcal{P}$ o de postos de vigia, e $\mathcal{G}_r$ o conjunto dos postos de vigia dos quais a região r é visível, para $r \in \mathcal{R}$. Minimizar o número total de postos de vigia ativos e garantir que cada região fica sob vigilância.

- *Minimum set cover* (cobertura mínima de conjuntos): dado A e uma família $\mathcal{F}$ de subconjuntos de A, determinar $\mathcal{C} \subseteq \mathcal{F}$ tal que $|\mathcal{C}|$ seja mínimo e $A = \cup_{S \in \mathcal{C}} S$. Problema **NP-hard**.

- Variáveis de decisão: $x_p \in \{0, 1\}$ diz se coloca um guarda no posto p ou não.
- Modelo matemático:

minimizar $\sum_{p \in \mathcal{P}} x_p$ sujeito a

$$\begin{cases} \sum_{p \in \mathcal{G}_r} x_p \geq 1, & \text{para todo } r \in \mathcal{R} \\ x_p \in \{0, 1\}, & \text{para } p \in \mathcal{P} \end{cases}$$

(pelo menos um guarda cobrirá a região r)

Problema de Cobertura "Postos de Vigia"

A Rede Nacional de Postos de Vigia constitui um dos principais mecanismos de detecção e localização inicial de incêndios florestais, sendo assegurada por vigilantes. É dada prioridade à vigilância de regiões que apresentam maior risco de incêndio. Suponha que $\mathcal{R}$ designa o conjunto dessas regiões, $\mathcal{P}$ o de postos de vigia, e $\mathcal{G}_r$ o conjunto dos postos de vigia dos quais a região r é visível, para $r \in \mathcal{R}$. Minimizar o número total de postos de vigia ativos e garantir que cada região fica sob vigilância.

- *Minimum set cover* (cobertura mínima de conjuntos): dado A e uma família $\mathcal{F}$ de subconjuntos de A, determinar $\mathcal{C} \subseteq \mathcal{F}$ tal que $|\mathcal{C}|$ seja mínimo e $A = \cup_{S \in \mathcal{C}} S$. Problema **NP-hard**.

- Variáveis de decisão: $x_p \in \{0, 1\}$ diz se coloca um guarda no posto p ou não.
- Modelo matemático:

minimizar $\sum_{p \in \mathcal{P}} x_p$ sujeito a

$$\begin{cases} \sum_{p \in \mathcal{G}_r} x_p \geq 1, & \text{para todo } r \in \mathcal{R} \\ x_p \in \{0,1\}, & \text{para } p \in \mathcal{P} \end{cases}$$

(pelo menos um guarda cobrirá a região r)

Problema de Cobertura "Postos de Vigia"

A Rede Nacional de Postos de Vigia constitui um dos principais mecanismos de detecção e localização inicial de incêndios florestais, sendo assegurada por vigilantes. É dada prioridade à vigilância de regiões que apresentam maior risco de incêndio. Suponha que $\mathcal{R}$ designa o conjunto dessas regiões, $\mathcal{P}$ o de postos de vigia, e $\mathcal{G}_r$ o conjunto dos postos de vigia dos quais a região r é visível, para $r \in \mathcal{R}$. Minimizar o número total de postos de vigia ativos e garantir que cada região fica sob vigilância.

- *Minimum set cover* (cobertura mínima de conjuntos): dado A e uma família $\mathcal{F}$ de subconjuntos de A, determinar $\mathcal{C} \subseteq \mathcal{F}$ tal que $|\mathcal{C}|$ seja mínimo e $A = \cup_{S \in \mathcal{C}} S$. Problema **NP-hard**.

- Variáveis de decisão: $x_p \in \{0, 1\}$ diz se coloca um guarda no posto p ou não.
- Modelo matemático:

$$\text{minimizar } \sum_{p \in \mathcal{P}} x_p \text{ sujeito a}$$
$$\begin{cases} \sum_{p \in \mathcal{G}_r} x_p \geq 1, & \text{para todo } r \in \mathcal{R} \\ x_p \in \{0, 1\}, & \text{para } p \in \mathcal{P} \end{cases}$$

(pelo menos um guarda cobrirá a região r)