

Técnicas, Entornos y Aplicaciones de Inteligencia Artificial

Práctica 3. Problemas de Satisfacción de Restricciones

Objetivo: Modelar y resolver problemas CSP, utilizando el entorno MiniZinc



MiniZinc:

- Entorno de desarrollo para la edición de modelos basados en Restricciones.
- Compilación del modelo en FlatZinc > Diversos resolvers (GECODE)
- Disponible (libre y código abierto) : <https://www.minizinc.org/>
Windows / MacOS / Linux <https://github.com/MiniZinc/MiniZincIDE/releases/>
- Ampla documentación: (Tutorial, Manual del Usuario, Manual de Referencia) ⇒ Handbook

Interfaz MiniZinc

The screenshot displays the MiniZinc IDE interface. At the top is a menu bar with File, Edit, MiniZinc, View, and Help. Below it is a toolbar with icons for New model, Open, Save, Copy, Cut, Paste, Undo, Redo, Shift left, Shift right, Run, Solver configuration (set to Geode 6.1.0 [built-in]), Show configuration editor, and Show project explorer. The main editor window shows a MiniZinc model named 'send-more-money.mzn' with the following code:

```
1 include "alldifferent.mzn";
2 var 1..9: S;
3 var 0..9: E;
4 var 0..9: N;
5 var 0..9: D;
6 var 1..9: M;
7 var 0..9: O;
8 var 0..9: R;
9 var 0..9: Y;
10 constraint 1000 * S + 100 * E + 10 * N + D
11 + 1000 * M + 100 * O + 10 * R + E
12 = 10000 * M + 1000 * O + 100 * N + 10 * E + Y;
13 constraint alldifferent([S,E,N,D,M,O,R,Y]);
14 solve satisfy;
15 output [ " ", show(S), show(E), show(N), show(D), "\n",
16 "+ ", show(M), show(O), show(R), show(E), "\n",
17 "= ", show(M), show(O), show(N), show(E), show(Y), "\n"];
```

Below the editor is an 'Output' window showing the execution results:

```
Compiling send-more-money.mzn
Running send-more-money.mzn
9567
+ 1085
= 10652
-----
Finished in 256msec
```

Annotations on the image identify key features:

- Ejecución**: Points to the Run button in the toolbar.
- Configuración Resolvedor**: Points to the Solver configuration dropdown and the Show configuration editor button.
- Edición del Modelo**: Points to the main code editor area.
- Salida Ejecución**: Points to the Output window.

Notas:

- Todas las líneas acaban con ;
- %: Símbolo de comentario (hasta final línea)
- Dependiente de mayúsculas/minúsculas (Identificadores y nombres de variables)
- Identificadores de variables empiezan por letra y pueden contener números, letras y _.
- Debe evitarse en lo posible en uso de variables reales (dominios continuos)
- La configuración de las ventanas es modificable.

Configuración Resolvedor (GECODE)

- (1) Resolvedor a utilizar : Gecode.
- (2) Tiempo máx. de resolución (0)
- (3) Nº Soluciones. Por defecto:
en optimización, soluciones intermedias,
en satisfabilidad, la 1ª solución
- (4) Nivel de preproceso:
básico (O1 – O3)
nodo consistencia (O4)
arco consistencia (O5)
- (5) Opciones a de salida.
Se recomienda 'Clear output before each run'.

The screenshot shows the 'Configuration' window for Gecode. It is divided into several sections: 'Solving', 'Compiler options', and 'Solver options'. Numbered callouts (1-5) point to specific settings: 1 points to the solver selection 'Gecode 6.1.0 [built-in]'; 2 points to the 'Time limit' set to 0 seconds; 3 points to the 'User-defined behavior' section, specifically the 'Stop after this many solutions' field set to 1; 4 points to the 'Compiler options' section, specifically the 'Compiler optimisation level' dropdown set to '-O1 (default)'; 5 points to the 'Solver options' section, specifically the 'Free search' checkbox.

Configuration

Gecode 6.1.0 [built-in]

Clone Reset to defaults ☒ default 1

Solving

Solver: Gecode 6.1.0

Time limit: 0 seconds (disabled) 2

☐ Default behavior

Optimization problems: print all intermediate solutions
Satisfaction problems: stop after first solution

☒ User-defined behavior 3

☐ Print intermediate solutions (for optimization problems)

Stop after this many solutions (0 means "all"): 1 (for satisfaction problems)

Compiler options

☐ Verbose compilation 4

☐ Output statistics for compilation

Compiler optimisation level: -O1 (default)

Additional data:

Additional compiler arguments:

Solver options

Number of threads: 1

☐ Random seed:

Additional solver command line arguments:

☐ Free search

☐ Verbose solving 5

☐ Output statistics for solving

Especificación Modelo CSP

% Esquema de un Modelo CSP en MiniZinc

```
include "alldifferent.mzn";      % Inclusión código restricciones especiales
include "datos1.dzn";           % Inclusión datos
```

```
int a;                          % Parámetros. Valor por asignación, fichero externo o interfaz.
var int: b;                     % Variables tipadas float|int|bool|string/enum
var 0..100: c;
```

```
constraint 250*b + 200*c <= 10*a; % Restricciones (aritmético-lógicas)
constraint .....
```

```
solve maximize 400*b + 450*c;    % Objetivo resolutor (solve satisfy; por defecto)
```

% Formato de salida (asignaciones a las variables)

```
Output ["Resultado b= ", show(b), "\n",
        "Resultado c = ", show(c)];
```

Especificación Modelo CSP

% Modelo ejemplo (Nº de pasteles de plátano y chocolate)

int b; % **Parámetro**. Numero de pasteles de plátano
var 0..100: c; % **Variable**. Numero de pasteles de chocolate

% gramos de harina
constraint 250*b + 200*c <= 4000;
% numero de platanos
constraint 2*b <= 6;
% gramos de azucar
constraint 75*b + 150*c <= 2000;
% gramos de mantequilla
constraint 100*b + 150*c <= 500;
% gramos de cacao
constraint 75*c <= 500;

% maximizar cantidad ponderada pasteles
solve maximize 400*b + 450*c;

output

["no. of bananas cakes = ", show(b), "\n",
"no. of chocolate cakes = ", show(c), "\n"];

Tipos: float/int/bool/string/enum

Restricciones (aritmético-lógicas)

Operad. relacionales: = (==), !=, >, <, <=, >=

Operad. aritméticos: +, -, *, /, div, mod, pow

Func. aritméticas: abs, sqrt, pow, ...

solve satisfy; %por defecto

solve maximize {arithmetic expression};

solve minimize {arithmetic expression};

output [{string|expr},];

expression: **show** (var)

"\n": Nueva línea

"\t": Tabulación

Parámetros y Fichero de datos

% Modelo con ficheros de datos

```
include "datos1.dzn";
```

%Parametros con valor adquirido por fichero

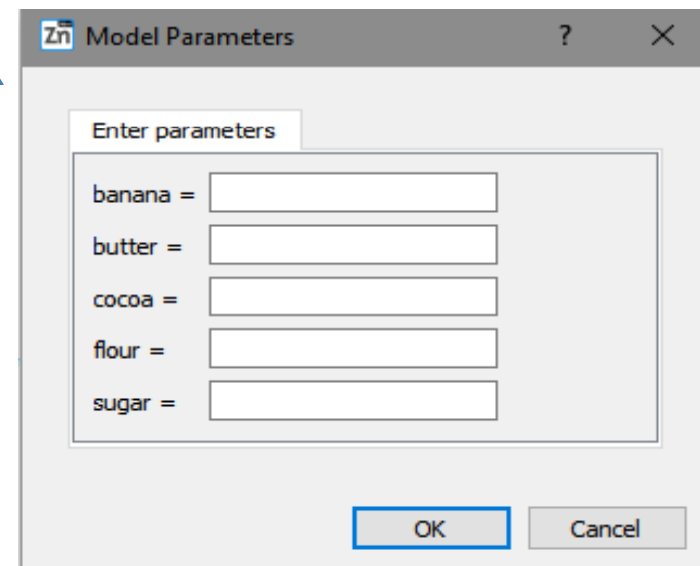
```
int: flour;    %no. grams of flour available  
int: banana;   %no. of bananas available  
int: sugar;    %no. grams of sugar available  
int: butter;   %no. grams of butter available  
int: cocoa;    %no. grams of cocoa available
```

%Parametros con Valor

```
int: flour = 4000;  
int: banana = 6;  
int: sugar = 2000;  
int: butter = 500;  
int: cocoa = 500;
```

%Fichero datos1 ("datos1.dzn")

```
flour = 4000;  
banana = 6;  
sugar = 2000;  
butter = 500;  
cocoa = 500;
```



VARIABLES: Conjuntos

set of int|float|bool : <var-name> = <expresion> | {<exp₁> <exp₂>, ...<exp_n>} ;

Ejemplos:

int: P; % P es un parámetro

set of int: Conjunto1 = 1 ..P; % Conjunto1: variable de 1..P enteros

set of float: Conjunto2 = {2.5, 6.5, 10.5}; % Variable conjunto de reales

Operaciones : Pertenencia (in, subset, superset),
 Union (union), Interseccion (inter),
 Diferencia (diff)
 Cardinalidad (card).

Variable: Vectores

array [$\langle \text{index-1} \rangle$, $\langle \text{index-2} \rangle$,, $\langle \text{index-n} \rangle$] **of var** int|float|string|bool : $\langle \text{var-name} \rangle$;

Ejemplos:

int: N; %N es un parámetro

int: k=10; %k es un parámetro con valor indicado

array [1..N, 1..N] of var int: celda1; % celda1:array bi-dimensional de enteros

array [1..k] of var 1..100: celda2; % Array uni-dimensional, con valores 1..100

array [1..10, 1..5, 1..15] of var bool: celda3; % 3-dimensional de booleanos

Los vectores se pueden inicializarse :

celda1 = [| 3, 5, | 6, 7,]; % celda1: array bi-dimensional de N x N elementos

celda2 = [3, 5, 6, 7, 76, 66]; % celda2: array unidimensional de 10 elementos

O adquirir sus valores de ficheros de datos externos.

Ver diversos ejemplos en boletín sobre la definición, operativa e impresión de vectores!

Restricciones Especiales

Diversos ejemplos en boletín y en documentación!

OR:	<code>constraint s1 + d1 <= s2 ∨ s2 + d2 <= s1;</code>
AND:	<code>constraint s1 + d1 <= s2 ∧ s2 + d2 <= s1;</code>
Condicional:	<code>constraint if a > b then c > 10 else c < 10 endif; constraint if b > c then d > 10 endif;</code> <code>constraint if (s1 + d1 <= s2 ∧ s2 + d2 <= s1)</code> <code> then (s1 + d1 >= s3 ∨ s2 + d2 >= s4) else c < 10 endif;</code>
Implicación:	<code>constraint s1 + d1 <= s2 -> s2 + d2 <= s1; % Si</code> <code>constraint s1 + d1 <= s2 <- s2 + d2 <= s1; % Solo si</code> <code>constraint s1 + d1 <= s2 <-> s2 + d2 <= s1; % Si y solo si</code>
Negación:	<code>constraint not (s1 + d1 <= s2 ∧ s2 + d2 <= s1);</code>
forall:	<code>constraint forall (i,j in 1..3 where i < j) (a[i] != a[j]); % a[1] != a[2] ∧ a[1] != a[3] ∧ a[2] != a[3];</code>
exists:	<code>constraint exists (i,j in 1..3 where i < j) (a[i] != a[j]); % a[1] != a[2] ∨ a[1] != a[3] ∨ a[2] != a[3]</code>
alldifferent:	<code>include "alldifferent.mzn"; % requiere incluir restricción global</code> <code>constraint alldifferent ([S,E,N,D,M,O,R,Y]);</code> <code>constraint alldifferent (Q); % Los valores de las celdas del vector Q son todos diferentes</code> <code>constraint alldifferent (j in 1..N) (Q[j]); % Solo los primeros N valores son diferentes</code>

Hay varios problemas resueltos en el boletín!!

Running sudoku.mzn

sudoku:

2 7 5 1 4 3 8 6 9

1 3 6 7 9 8 2 4 5

8 4 9 5 6 2 7 1 3

7 1 2 8 3 5 4 9 6

4 6 3 2 1 9 5 7 8

5 9 8 4 7 6 1 3 2

6 5 4 3 2 1 9 8 7

3 2 1 9 8 7 6 5 4

9 8 7 6 5 4 3 2 1

%Modelo de un Sudoku N x N

par int: S; %Parámetro pedido en la ejecución del modelo.

int: N = S*S; %parametro, para usarlo como índice de la matriz

array [1..N, 1..N] of var 1..N: celda; % Sudoku, celda[i, j]

include "alldifferent.mzn";

% Todas las celdas en una fila son diferentes.

constraint forall (i in 1..N) (alldifferent (j in 1..N) (celda[i,j]));

% Todas las celdas en una columna son diferentes.

constraint forall(j in 1..N) (alldifferent (i in 1..N) (celda[i,j]));

% Todas las celdas en una submatriz son diferentes.

constraint forall (i,j in 1..S)

(alldifferent (p,q in 1..S) (celda[S*(i-1)+p, S*(j-1)+q]));

solve satisfy; %solo requerimos satisfabilidad

output ["sudoku:\n"] ++ [show(celda[i,j]) ++ % Blancos separadores de submatrices

if j = N then if i mod S = 0 /\ i < N then "\n\n" else "\n" endif else

if j mod S = 0 then " " else " " endifendif | i,j in 1..N];

Práctica 3: CSP

Tarea:

- Realizar el ejercicio propuesto (se necesitará para el día de la evaluación, en el que se planteará ampliaciones o modificaciones)

Calendario:

Sem	<u>LABORATORIO</u>	Evaluación
10-XI	CSP-MiniZinc	
24-XI	CSP-MiniZinc	
1-XII		P3: <i>Eval:</i> CSP-MiniZinc

Practica CSP (15%) P3