Resolución de sudokus

Programación lineal entera

Investigación Operativa

Bilbao abril 2022

Índice

1. Sudoku.

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

Tabla 1: Datos del sudoku

La profesora María Merino muestra en www.divulgamat.net/ [1] (Sudokus y modelización) cómo resolver el problema con investigación operativa utilizando programación lineal entera. Aquí se muestra un resumen de ese documento que explica el método de resolución: las variables utilizadas y las ecuaciones necesarias.

Índices:

- i: subíndice correspondiente a fila, $i = 1, \dots, 9$.
- j: subíndice correspondiente a columna, $j = 1, \dots, 9$.
- k: subíndice correspondiente al valor dentro de la celda (i, j), $k = 1, \dots, 9$.

Variables:

- x_{ijk}: variable binaria para cada celda (i, j) tal que

$$x_{ijk} = \begin{cases} 1, & \text{si la solución es } k \\ 0, & \text{en caso contrario} \end{cases}$$

  O sea que cada celda contiene nueve variables, donde ocho de ellas serán nulas y únicamente una tendrá el valor 1 ¡Tenemos $9 \cdot 9 \cdot 9 = 729$ variables!

Esto a priori parece un *derroche*, pero al haber escogido variables binarias, resulta muy sencillo expresarlas mediante relaciones lineales:

- Regla I: rellenar con una única cifra del 1 al 9 cada una de las 81 celdas.

$$\sum_{k=1}^{9} x_{ijk} = 1, \quad \forall i, j = 1, \dots, 9. \tag{1}$$

- Regla II: no repetir ninguna cifra en una misma fila.

$$\sum_{j=1}^{9} x_{ijk} = 1, \quad \forall i, k = 1, \dots, 9. \tag{2}$$

- Regla III: no repetir ninguna cifra en una misma columna.

$$\sum_{i=1}^{9} x_{ijk} = 1, \quad \forall i, j = 1, \ldots, 9. \tag{3}$$

- Regla IV: no repetir ninguna cifra en una misma caja de dimensión 3×3.

$$\begin{aligned}
&\sum_{i=1}^{3}\sum_{j=1}^{3} x_{ijk} = 1, \quad k = 1, \ldots, 9, \qquad && \text{(9 restricciones)}\\
&\sum_{i=4}^{6}\sum_{j=1}^{3} x_{ijk} = 1, \quad k = 1, \ldots, 9, \qquad && \text{(9 restricciones)}\\
&\sum_{i=7}^{9}\sum_{j=1}^{3} x_{ijk} = 1, \quad k = 1, \ldots, 9, \qquad && \text{(9 restricciones)}\\
&\sum_{i=1}^{3}\sum_{j=4}^{6} x_{ijk} = 1, \quad k = 1, \ldots, 9, \qquad && \text{(9 restricciones)}\\
&\ldots\\
&\sum_{i=7}^{9}\sum_{j=7}^{9} x_{ijk} = 1, \quad k = 1, \ldots, 9, \qquad && \text{(9 restricciones)}
\end{aligned}$$

Cada región puede ser representada por la casilla superior izquierda. Con lo que esas 9 ecuaciones, que representan 81 restricciones, han sido encapsuladas en una sola para cualquier posible caja (i_0, j_0) y cifra k:

$$\sum_{i=i_0}^{i_0+2}\sum_{j=j_0}^{j_0+2} x_{ijk} \quad \forall i_0, j_0 = 1, 4, 7;\ k = 1, \ldots, 9. \tag{4}$$

Debemos añadir explícitamente al sistema de ecuaciones (1)-(4) el carácter binario de todas las variables:

$$x_{ijk} = \{0, 1\}, \quad \forall i, j, k = 1, \ldots, 9 \tag{5}$$

Para completar la modelización, debemos escribir el algoritmo que aporta la información inicial al modelo. Sea $D = (d_{ij})$ la matriz de entradas. Consideraremos $d_{ij} = 0$ si la casilla está vacía, y tomará su valor solución entre 1 y 9, en caso contrario. Por tanto:

$$\text{Si } d_{ij} = k \neq 0 \Rightarrow x_{ijk} = \begin{cases} 1, & \text{si } d_{ij} = k \\ 0, & \text{si } d_{ij} \neq k \end{cases} \tag{6}$$

O equivalentemente:

$$x_{ijk} = 1, \quad \forall d_{ij} = k > 0 \tag{7}$$

1.1. Explicación del programa

El programa ha sido realizado en lenguaje GLPK y ha sido resuelto con Gusek[1]. El código íntegro del programa se encuentra en el apéndice (A).

Se va a usar un conjunto de números N del 1 al 9 en una variable booleana $x_{i,j,k}$ que toma valores en $\{0, 1\}$, como se indica en la ecuación (5).

```
# Conjunto de datos
set N := 1..9 ;

# Variables de decision x_ijk
# Hay 9*9*9 = 729 variables binarias de decision
var x {(i,j,k) in {N cross N cross N}} binary ;
```

Así mismo se le dice al óptimizador que no hay nada que minimizar ni maximizar en este problema lineal.

```
minimize nothing : 0;
```

Se escriben las restricciones relativas a cualquier sudoku:

```
# Regla I: rellenar con cifras del 1 al 9 cada una de las 81
    celdas
subject to all_filled {i in N, j in N}:
sum {k in N} x[i,j,k] = 1;

# Regla II: no repetir ninguna cifra en una misma fila
subject to Rows {i in N, k in N}:
sum {j in N} x[i,j,k] = 1;

# Regla III: no repetir ninguna cifra en una misma columna
subject to Columns {j in N, k in N}:
sum {i in N} x[i,j,k] = 1;

# Bloques 3 x 3
set BLOCKS := {1, 4, 7} ;

# Regla IV: no repetir ninguna cifra en una misma caja
subject to Squares {k in N, i0 in BLOCKS, j0 in BLOCKS}:
sum {i in (i0..(i0+2))} sum {j in (j0..(j0+2))} x[i,j,k] = 1;
```

A continuación se muestran los datos iniciales de nuestro sudoku (Tabla 1) en lenguaje GLPK:

```
# DATA: INFORMACION INICIAL DEL SUDOKU ———
# using the . character instead of a value in the matrix makes
    that particular value to be defaulted.
param DATA :     1 2 3   4 5 6   7 8 9 :=
```

[1]Gusek es un programa gratuito que resuelve problema de programación lineal para Windows y puede descargarse desde `http://gusek.sourceforge.net/gusek.html`

```
1    8 . .   6 . .   9 . 5
2    . . .   . . .   . . .
3    . . .   . 2 .   3 1 .

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

7    . . 2   7 9 .   1 . .
8    5 . .   . 8 .   . 3 6
9    . . 3   . . .   . . . ;
```

Se asignan los datos conocidos a nuestras variables usando la ecuación (7):

```
# Asignar los datos inicialmente conocidos del sudoku
subject to known {i in N, j in N : DATA[i,j] > 0}:
x[i,j,DATA[i,j]] = 1;
```

Se escribe un pequeño programa para mostrar la solución:

```
# Mostrar todas las filas sin separacion por  bloques
for {i in N}
{
    # Mostar los columnas de cada fila
    printf{j in N} "%1d ", sum{k in N} k * x[i,j,k];
    printf "\n";
}
```

La tabla 2 muestra la **solución**.

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

Tabla 2: Solución del sudoku

A. Programa sudoku

El programa del sudoku ha sido realizado en lenguaje GLPK. A continuación se muestra la versión íntegra que se puede resolver con Gusek. Este programa se puede descargar directamente de GitHub en el siguiente link: Sudoku - GLPK (`https://github.com/ImJaviPerez/sudoku/blob/main/sudoku.mod`).

```
# GLPK Math program
#
# Conjunto de datos
set N := 1..9 ;

# Variables de decision x_ijk
# Hay 9*9*9 = 729 variables binarias de decision
var x {(i,j,k) in {N cross N cross N}} binary ;

# DATA[i,j] = informcion inicial del sudoku
param DATA {i in N, j in N}, default 0;

# En este problema no hay nada que minimizar ni maximizar
minimize nothing : 0;

# Regla I: rellenar con cifras del 1 al 9 cada una de las 81
    celdas
subject to all_filled {i in N, j in N}:
sum {k in N} x[i,j,k] = 1;

# Regla II: no repetir ninguna cifra en una misma fila
subject to Rows {i in N, k in N}:
sum {j in N} x[i,j,k] = 1;

# Regla III: no repetir ninguna cifra en una misma columna
subject to Columns {j in N, k in N}:
sum {i in N} x[i,j,k] = 1;

# Bloques 3 x 3
set BLOCKS := {1, 4, 7} ;

# Regla IV: no repetir ninguna cifra en una misma caja
subject to Squares {k in N, i0 in BLOCKS, j0 in BLOCKS}:
sum {i in (i0..(i0+2))} sum {j in (j0..(j0+2))} x[i,j,k] = 1;

# Asignar los datos inicialmente conocidos del sudoku
subject to known {i in N, j in N : DATA[i,j] > 0}:
x[i,j,DATA[i,j]] = 1;

# RESOLUCION
solve;

# MOSTRAR EL RESULTADO --------------------------
printf "\n";
# Mostrar las filas y las columnas separando por bloques
for {bf in 0..2}
{
    for {i in 1..3}
    {
```

```
        # Mostar los columnas de una fila separadas por bloques
        for {bc in 0..2}
        {
            printf{j in 1..3} "%1d ", sum{k in N} k * x[3*bf+i
               ,3*bc+j,k];
            printf " ";
        }
        printf "\n";
    }
    printf "\n";
}

printf "\n";
# Mostrar todas las filas sin separacion por  bloques
for {i in N}
{
    # Mostar los columnas de cada fila
    printf{j in N} "%1d ", sum{k in N} k * x[i,j,k];
    printf "\n";
}

# DATOS USADOS EN ESTE PROBLEMA ---------------------
data;
# DATA: INFORMACION INICIAL DEL SUDOKU --------
# using the . character instead of a value in the matrix makes
   that particular value to be defaulted.
param DATA :      1 2 3   4 5 6   7 8 9 :=
               1  8 . .   6 . .   9 . 5
               2  . . .   . . .   . . .
               3  . . .   . 2 .   3 1 .

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

               7  . . 2   7 9 .   1 . .
               8  5 . .   . 8 .   . 3 6
               9  . . 3   . . .   . . . ;

end;
```

B. Créditos y fuentes.

- Este documento ha sido escrito en LaTeXcon el editor gratuito TeXstudio.
- El código ha sido resuelto usando el *solver* gratuito Gusek.

Referencias

[1] María Merino. Sudokus y modelización, 2010.