BE Optimisation Combinatoire - Picross Problem

Shuang Hou, Ziqing Wu

October 25, 2020

Notations

- $m_i \in \{1, ..., \lceil n/2 \rceil\}$ represents the size of the list associated to row i;
- $p_i \in \{1, ..., \lceil n/2 \rceil\}$ represents the size of the list associated to column j.
- $lines_{i,k} \in \{0, ..., n-1\}$ is the k-th integer of the list associated to row i.
- $columns_{j,k} \in \{0, ..., n-1\}$ is the k-th integer of the list associated to column j.

1 Variables

Two ways are used to model the Picross problem:

- The first one is to represent the color of each cell in each line by a binary variable. We note it as $bl_{i,j}$ where row index is $i \in \{0, n-1\}$ and column index is $j \in \{0, n-1\}$. n is the grid dimension. The domain of this variable is $\{0, 1\}$ where 0 represents a blank cell and 1 represents a black cell.
- Similarly to the first one, one can represent the color of each cell in each column by a binary variable. We note it as $bc_{j,i}$ where row index is $i \in \{0, n-1\}$ and column index is $j \in \{0, n-1\}$. n is the grid dimension. The domain of this variable is $\{0, 1\}$ where 0 represents a blank cell and 1 represents a black cell.
- The second one is to represent the line index of the first cell for each block by an integer variable. We note it as $l_{i,k}$, where row index is $i \in \{0, n-1\}$ and column index is $k \in \{0, m_i - 1\}$. The domain of this variable is $\{0, n-1\}$ where n is the grid dimension.
- Similarly to the second one, one can represent the column index of the first cell for each block by an integer variable. We note it as $c_{j,k}$, where row index is $i \in \{0, n-1\}$ and column index is $k \in \{0, p_j - 1\}$. The domain of this variable is $\{0, n-1\}$ where n is the grid dimension.

In the code, four lists of variables are proposed, bl with dimension $n \times n$, bc with dimension $n \times n$, l with the same dimension as $lines$ and c with the same dimension as $columns$. The input $lines$ and $columns$ are proposed in the specific problem.

2 Constraints

Four constrains are added to solve this problem.

2.1 Consistency of each black cell

The binary variables are used to model this constraint. Considering that the color of each cell must be unique, thus the value of the line and column binary variables should be consistent, which is represented as follow:

- for each cell, $i \in \{0, n-1\}$ and $j \in \{0, n-1\}$

$$bl_{i,j} = bc_{j,i}$$

2.2 Two blocks do not overlap

To model this constraint, one would like to use integer variables l and c. Knowing that $lines_{i,k} \in \{0, ..., n\}$ is the k-th integer of the list associated to row i and $columns_{j,k} \in \{0, ..., n\}$ is the k-th integer of the list associated to column j, the constraint can be represented by:

- for each $i \in \{0, n-1\}$ and for each $k \in \{0, length(lines_i) - 1\}$

$$l_{i,k+1} - l_{i,k} > lines_{i,k}$$

- for each $j \in \{0, n-1\}$ and for each $k \in \{0, length(columns_j) - 1\}$

$$c_{j,k+1} - c_{j,k} > columns_{j,k}$$

2.3 Limit to the number of black cells on each line/column

To model this constraint, the choice of variable is naturally the binary ones *bl* and *bc*. The representation will be:

- for each line $i \in \{0, n-1\}$

$$\sum_{j=0}^{n-1} bl_{i,j} = \sum_{k=0}^{length(lines_i)-1} lines_{i,k}$$

- for each column $j \in \{0, n-1\}$

$$\sum_{i=0}^{n-1} bc_{j,i} = \sum_{k=0}^{length(columns_j)-1} columns_{j,k}$$

2.4 Links between binary variables and integer variables

To model this constraint, the integer variables are used as part of index of each cell, the number of cells occupied by each block in each line/column should be equal to the elements in lines.

- for each line $i \in \{0, n-1\}$ and for each $k \in \{0, length(lines_i) - 1\}$

$$\sum_{p=0}^{lines_{i,k}-1} bl_{i,(l_{i,k}+p)} = lines_{i,k}$$

- for each column $j \in \{0, n-1\}$ and for each $k \in \{0, length(columns_j) - 1\}$

$$\sum_{p=0}^{columns_{j,k}-1} bc_{j,(c_{j,k}+p)} = columns_{j,k}$$

3 Implementation and results

The implementation of the model is encapsulated in a function called $Picross(pattern)$, where *pattern* is a pattern that can be chosen from {'moon', 'star', 'cat', 'horse', 'duck', 'house'}. A *for loop* is used to draw all the patterns by calling the *Picross* function, the results all meet expectations, and the resolution time of each case is less than 1 second.