Solving Picross...

... using constraint programming

Benoît BOMPOL

17th of January, 2024

“Critical thinking is the key to success.”

Prof. Layton

Abstract

The objective of this project is to implement a solver for the Picross game, also known as nonogram. Picross is a visual logic puzzle whose goal is to unveil an image by shading specific cells within a grid, guided by numerical clues provided for each row and column. These numerical hints indicate the consecutive groups of filled cells in that particular row or column. The puzzle combines elements of reasoning, making it an intriguing problem for computational exploration.

Contents

. Introduction - Why bother solving Picross ?

Starting from what is considered a small grid (10 rows, 10 columns), there are 2^{100} possible configurations ($\sim 1.26 \times 10^{30}$). Solving such instances using brute force on a computer doing 10^9 operations per second would take 2.3 times the estimated age of the universe.

Furthermore, solving Picross is NP-complete (see The complexity of Klondike, Mahjong, nonograms and animal chess).

I. Description of an instance

I.1. Mathematical description

A Picross instance is defined as the following :

1. A grid of n rows and m columns.
2. For each row i, $\mathsf{R}[i] = \{b_1^{\mathsf{R}(i)}, \dots\}$ is a set of positive integers describing the consecutive groups of filled cells in the i-th row.
3. Likewise, for each column $\mathsf{C}[j] = \{b_1^{\mathsf{C}(j)}, \dots\}$ is a set of positive integers describing the consecutive groups of filled cells in the j-th column.

For instance $\mathsf{R}[0] = \{2, 4, 1\}$ would mean that in the top row, there are three blocs, the first one of size 2, the second one of size 4, the last one of size one. Each of these blocs is separated by (at least) one blank cell.

A **solution** of a Picross instance is for each cell, whether each cell has been shaded (or not), such that each group of filled cells is verified for each row and each column.

A proper instance of Picross admits a (unique) **solution**. The operation of **solving** a Picross, is to find on a grid the solution.

Our goal will be to find proper models to encapsulate as many human reasoning as possible, and (hopefully) to find reasoning that players haven't found until now.

I.2. `.px` file description

Each instance file will be described in a `.px` (litterally "p-cross") file. Each of these file share the same similar structure.

```
nb_rows,nb_cols
cst_row_0
...
cst_row_n_1
cst_col_0
...
cst_col_m_1
```

II. First approach : Enumerating each solution

II.1. Enumerate each solution in a line - `AllowedTuple`

We have a line of length n, with a vector of $k \geq 1$ consecutive filled cells $\{b^{(1)}, \dots, b^{(k)}\}$. In such a line, there will be $K = \sum_i b^{(i)}$ shaded cells in $n - (k-1) = n - k + 1$ remaining cells (once mandatory blanks are removed).

There are, at most $\binom{n-k+1}{K} \sim \mathcal{O}(n^K)$ solutions in a line (and even fewer in practice).

On a line, the following variable can be used : $\texttt{start}_i \in 0 \ldots n-1$, which corresponds to the start of the i-th bloc in the line, under the following constraints :

a. $\texttt{start}_i + (b^{(i)} + 1) \leq \texttt{start}_{i+1}$

b. $\texttt{start}_i + b^{(i)} \leq n$

The first constraint enforces that there is at least a blank cell between two consecutive blocs, and the second one enforces that everything *holds* in a line of length n.

This process of finding all valid solutions in one line will be called :
`AllowedTuple(int[] constraints, int size)` in the rest of this section.

II.2. Solving Picross (?)

Now that we know how to enumerate solutions for each row and each column, it it high time solving a complete grid.

We will consider only $n \times m$ boolean variables :

$$(grid_{i,j})_{i \in 0 \ldots (n-1), j \in 0 \ldots (m-1)} = 1 \iff \text{Cell at position } (i,j) \text{ has been shaded.}$$

Under the following constraints :

- $\forall i \in 0 \ldots (n-1), (\mathsf{grid}_{i,\bullet}) \in \texttt{AllowedTuple}(b^{\mathsf{R}(i)}, m)$
  - The i-th row respects its constraints.
- $\forall j \in 0 \ldots (m-1), (\mathsf{grid}_{\bullet,j}) \in \texttt{AllowedTuple}(b^{\mathsf{C}(j)}, n)$
  - The j-th row column its constraints.

The important part of this model is the fact that a single variable $\mathsf{grid}_{i,j}$ will act on both the i-th row and the j-th column, which will act on propagation.

It *kind of* works... However, this model does not scale, and not only will fail to solve the Picross, but will also fail to load on big enough instances, and encapsulates little to no reasoning.

III. Solving Picross (!)

We now have a problem that is twofold :

1. Create a model that encapsulates human reasoning
2. Create a model that accepts bigger instances

To do so, we will use a redundant model, in the sense that we will add variables that have a different *meaning* than ($\mathsf{grid}_{i,j}$), but that will allow to implement reasoning.
We keep $\mathsf{grid}_{i,j}$, and we introduce :

- $x_{i,k} \in \{0, \dots, m-1\}$ is the position at which the k-th bloc in the i-th row starts.
- Similarly, $y_{j,\ell} \in \{0, \dots, n-1\}$ is the position at which the ℓ-th bloc in the j-th column starts.

It is indeed redundant :

- If all ($\mathsf{grid}_{i,j}$) are known, $x_{i,k}$ can be deduced (same occurs for $y_{j,\ell}$).
- If all $x_{i,k}$ are known, ($\mathsf{grid}_{i,j}$) can be deduced (thus $y_{j,\ell}$ can be deduced as well)
- If all $y_{j,\ell}$ are known, ($\mathsf{grid}_{i,j}$) can be deduced (thus $x_{i,k}$ can be deduced as well)

Using the work done on solving one line, we also get the following constraints :

- $x_{i,k} + b_k^{\mathsf{R}(i)} + 1 \leq x_{i,k+1}$
- $y_{j,\ell} + b_\ell^{\mathsf{C}(j)} + 1 \leq y_{j,\ell+1}$

III.1. Introducing a new kind of constraint

This constraint will highlight the link between such variables. If the cell at position (i, j) is shaded, then there is a bloc $b_k^{\mathsf{R}(i)}$ (exactly one, in fact) such that $x_{i,k} \leq j$ **and** $x_{i,k} \geq j - b_k^{\mathsf{R}(i)} + 1$. That is, there is a bloc, which has begun before j, but not after j, shifted from the size of the bloc. We get the similar formulas for the columns.
Conversely, assume that there is no bloc such that it begins before j and ends after j, plus the bloc size. Then, the cell at position j is not shaded.
This leaves an equivalent proposition :

$$\mathsf{grid}_{i,j} = 1 \iff \bigvee_{k \in 1 \dots |\mathsf{R}(i)|} (x_{i,k} \leq j) \land (x_{i,k} \geq j - b_k^{\mathsf{R}(i)} + 1)$$

Likewise :

$$\mathsf{grid}_{i,j} = 1 \iff \bigvee_{\ell \in 1 \dots |\mathsf{C}(j)|} (x_{j,\ell} \leq i) \land (y_{j,\ell} \geq i - b_\ell^{\mathsf{C}(j)} + 1)$$

III.2. Stacking cells

A bloc k in a row cannot start before every other $k' < k$ blocs have begun (and have ended). This implies additional inequalities :

- **Leftmost Stack (or Top Stack)**
  - $x_{i,k} \geq \left(\sum_{k'<k} b_{k'}^{\mathsf{R}(i)}\right) + k$

  - $y_{j,\ell} \geq \left(\sum_{\ell'<\ell} b_{\ell'}^{\mathsf{C}(j)}\right) + \ell$

- **Rightmost Stack (or Bottom Stack)**
  - $x_{i,k} \leq m - (|\mathsf{R}(i)| - k - 1) - \sum_{k' \geq k} b_{k'}^{\mathsf{R}(i)}$
  - $y_{j,\ell} \leq n - (|\mathsf{C}(j)| - \ell - 1) - \sum_{\ell' \geq \ell} b_{\ell'}^{\mathsf{C}(j)}$

III.3. Enforcing blank cells

We will use once more reification in order to enforce that if a bloc starts at position $j > 0$, then there is an empty cell at position $j - 1$:

$$\forall j > 0, \forall i \in 0 \ldots (n-1), \forall k \in 1 \ldots |\mathsf{R}(i)|, x_{i,k} = j \implies \mathsf{grid}_{i,j-1} = 0$$

Likewise :

$$\forall i > 0, \forall j \in 0 \ldots (m-1), \forall \ell \in 1 \ldots |\mathsf{C}(j)|, y_{j,\ell} = i \implies \mathsf{grid}_{i-1,j} = 0$$

III.3.1 . Chanelling : Enforcing a blank cell after a bloc

In a similar way, if we know at which position a bloc starts, we can deduce immediately at which position the associated blank cell will be :

$$\forall i \in 0 \ldots (n-1), \forall k \in 1 \ldots |\mathsf{R}(i)|, \forall j \geq 0 \text{ such that } j + b_k^{\mathsf{R}(i)} < m, x_{i,k} = j \implies \mathsf{grid}_{i,j+\mathsf{b}_k^{(i)}} = 0$$

Likewise,

$$\forall j \in 0 \ldots (m-1), \forall \ell \in 1 \ldots |\mathsf{C}(j)|, \forall i \geq 0 \text{ such that } i + b_\ell^{\mathsf{C}(j)} < n, y_{j,\ell} = i \implies \mathsf{grid}_{i+b_\ell^{\mathsf{C}(j)},j} = 0$$

IV. Final Model (?)

This section has, for main goal, to summarise every constraint, in order to write the full model :

Data

- $n \in \mathbb{N}^\star$: Number of rows
- $m \in \mathbb{N}^\star$: Number of columns
- $(b^{\mathsf{R}(i)})_{i \in 0 \ldots (n-1)}$: Set of contiguous blocs in each row.
- $(b^{\mathsf{C}(j)})_{j \in 0 \ldots (m-1)}$: Set of contiguous blocs in each column.

Variables

- $(\mathsf{grid}_{i,j})_{i \in 0 \ldots (n-1), j \in 0 \ldots (m-1)} \in \mathbb{B}$: $\mathsf{grid}_{i,j} = 1 \iff$ cell at position (i,j) is shaded.
- $(x_{i,k})_{i \in 0 \ldots (n-1), k \in 1 \ldots |\mathsf{R}(i)|} \in 0 \ldots (m-1)$: Starting position of each bloc in each row.
- $(y_{j,\ell})_{j \in 0 \ldots (m-1), \ell \in 1 \ldots |\mathsf{C}(j)|} \in 0 \ldots (n-1)$: Starting position of each bloc in each column.

Under constraints

$$\forall i \in 0\dots(n-1), \sum_{j=0}^{m-1} \mathsf{grid}_{i,j} = \sum_{k=1}^{|\mathsf{R}(i)|} b_k^{\mathsf{R}(i)} \quad (1a)$$

$$\forall j \in 0\dots(m-1), \sum_{i=0}^{n-1} \mathsf{grid}_{i,j} = \sum_{\ell=1}^{|\mathsf{C}(j)|} b_\ell^{\mathsf{C}(j)} \quad (1b)$$

$$\forall i \in 0\dots(n-1), \forall k < |\mathsf{R}(i)|, x_{i,k} + b_k^{\mathsf{R}(i)} + 1 \leq x_{i,k+1} \quad (1c)$$

$$\forall j \in 0\dots(m-1), \forall \ell < |\mathsf{C}(j)|, y_{j,\ell} + b_\ell^{\mathsf{C}(j)} + 1 \leq y_{j,\ell+1} \quad (1d)$$

$$\mathsf{grid}_{i,j} = 1 \iff \bigvee_{k \in 1\dots|\mathsf{R}(i)|} (x_{i,k} \leq j) \wedge (x_{i,k} \geq j - b_k^{\mathsf{R}(i)} + 1) \quad (1e)$$

$$\mathsf{grid}_{i,j} = 1 \iff \bigvee_{\ell \in 1\dots|\mathsf{C}(j)|} (y_{j,\ell} \leq i) \wedge (y_{j,\ell} \geq i - b_\ell^{\mathsf{C}(j)} + 1) \quad (1f)$$

$$x_{i,k} \geq \left(\sum_{k'<k} b_{k'}^{\mathsf{R}(i)} \right) + k \quad (1g)$$

$$y_{j,\ell} \geq \left(\sum_{\ell'<\ell} b_{\ell'}^{\mathsf{C}(j)} \right) + \ell \quad (1h)$$

$$x_{i,k} \leq m - (|\mathsf{R}(i)| - k - 1) - \sum_{k' \geq k} b_{k'}^{\mathsf{R}(i)} \quad (1i)$$

$$y_{j,\ell} \leq n - (|\mathsf{C}(j)| - \ell - 1) - \sum_{\ell' \geq \ell} b_{\ell'}^{\mathsf{C}(j)} \quad (1j)$$

$$\forall j > 0, \forall i \in 0\dots(n-1), \forall k \in 1\dots|\mathsf{R}(i)|, x_{i,k} = j \implies \mathsf{grid}_{i,j-1} = 0 \quad (1k)$$

$$\forall i > 0, \forall j \in 0\dots(m-1), \forall \ell \in 1\dots|\mathsf{C}(j)|, y_{j,\ell} = i \implies \mathsf{grid}_{i-1,j} = 0 \quad (1l)$$

$$\forall i \in 0\dots(n-1), \forall k \in 1\dots|\mathsf{R}(i)|, \forall j \geq 0 \text{ s.t. } j + b_k^{\mathsf{R}(i)} < m, x_{i,k} = j \implies \mathsf{grid}_{i,j+\mathsf{b}_k^{(i)}} = 0 \quad (1m)$$

$$\forall j \in 0\dots(m-1), \forall \ell \in 1\dots|\mathsf{C}(j)|, \forall i \geq 0 \text{ s.t. } i + b_\ell^{\mathsf{C}(j)} < n, y_{j,\ell} = i \implies \mathsf{grid}_{i+b_\ell^{\mathsf{C}(j)},j} = 0 \quad (1n)$$

Constraints $(1a)$ and $(1b)$ ensure that the correct number of cells is shaded on each line and column, preventing duplicates. Constraints $(1c)$ and $(1d)$ guarantee the proper ordering of the starting positions of each block. Constraints $(1e)$ and $(1f)$ establish links between redundant variables. Constraints $(1g)$ refer to stacking blocks on the leftmost part of a row, while $(1i)$ pertain to stacking on the rightmost part. Constraints $(1h)$ and $(1j)$ serve the same purpose, addressing stacking on the upper and lower sections of the grid, respectively. Finally, constraints $(1k)$ through $(1n)$ mandate the presence of blank cells.

IV.1. Benchmarking

The goal of this benchmark is multiplefold :

1. Checking which `.px` files will work (or not on given models)
2. Seeing whether or not there is a time improvement.
3. On instances that could not be solved, the score is determined by the fraction of fixed cells.

The following models are :

M1 : $(1a)$ up to $(1f)$

M2 : $(1a)$ up to $(1j)$

M3 : $(1a)$ up to $(1n)$

The following criteria will be noted :

- Ratio of undetermined cells after propagation
- Number of fails done in the branching

	(1a)...(1f)	(1a)...(1j)	(1a)...(1n)
Bird	0	0	0
Clock	0	0	0
Godzilla	73.36	0	0
Hare	98	51.8	0
Kabuki	96.50	35.5	33.00
Knife	97.77	76.44	76.44
Layton	66.66	0	0
Batman	66.21	65.75	60.52
Panda	98.25	87.25	87.25
Pikachu	93.75	0	0
Phantom	57.79	37.72	37.45
QR Code	97.38	62.07	28.42
Ratz	99.11	80.89	80.89
Spade	0	0	0
Tarantino	98.28	14.40	14.4
Tardis	0	0	0
Zen	99.01	0	0

Figure 1: Ratio (% of the whole grid) of unfixed vertices after propagation

	(1a)...(1f)	(1a)...(1j)	(1a)...(1n)
Bird	0	0	0
Clock	0	0	0
Godzilla	13	0	0
Hare	✗	8	0
Kabuki	1325	740	748
Knife	136	45	25
Layton	24	0	0
Batman	526	27	40
Panda	1060	53	16
Phantom	✗	✗	✗
Pikachu	20	0	0
QR Code	5644	642	116
Ratz	535	43	4
Spade	0	0	0
Tarantino	1568	191	203
Tardis	0	0	0
Zen	299	0	0

Figure 2: Number of fails during the branching scheme.
✗ represents instances in which the solved was stopped (by me)

V. Concluding remarks

There are a few remarks that might - given more time - find an hopeful answer :

1. Why does the number of fails increases from time to time ?
2. Which heuristics might help reducing the number of fails ?
3. Will we, one day, get `kabuki.px` fully solved ?

   YES

4. How could we filter more ?