Hashiwokakero

Logical encoding

March 15, 2024

1 Objective

As SAT-solvers typically only accept formulas in DIMACS format, our objective is to logically encode the game into Conjunctive Normal Form (CNF).

For a game field of size $M * N$ and every of its field indices (i, j), we instantiate a two-dimensional array **nodes** of type **Node** carrying the four variables

$$h^1_{i,j}, h^2_{i,j}, v^1_{i,j}, v^2_{i,j}$$

that represent the type of bridge present at index (i, j). In the following, we differentiate between so called non-island and island nodes.

2 Logical encoding

2.1 One Type

First, we want to enforce that for every non-island node there will only be one type of bridge set to true. This can be achieved by considering the conjunction of every possible pair of types and negating their simultaneous existence:

$$\bigwedge_{(t_1,t_2)\in Node} \neg(t_1 \wedge t_2)$$

2.2 Continuity

Now, we ensure that for every non-island node that has a bridge, the preceding and subsequent nodes in the direction of the bridge type also have the same type of bridge. Note that it is sufficient to check only the subsequent nodes:

$$(h^1_{i,j} \Rightarrow h^1_{i+1,j}) \wedge (h^2_{i,j} \Rightarrow h^2_{i+1,j}) \wedge (v^1_{i,j} \Rightarrow v^1_{i,j+1}) \wedge (v^2_{i,j} \Rightarrow v^2_{i,j+1})$$

2.3 Degree

For every island, the connecting bridges either extend horizontally or vertically. Therefore, a maximum of 4 variables can bet set to *true* and only eight of the immediately neighbouring 16 variables have to be considered (i.e. if $h^1_{i+1,j}$ is *true*, then that assignment obviously has no effect on the number of bridges connecting to our island at (i, j)).

This means that for any degree constraint n we can build a very intuitive DNF encoding over the eight relevant neighbouring variables. Consider for example the degree constraint two and its corresponding DNF:

$$\begin{aligned}
&(h^1_{i,j+1} \wedge h^1_{i,j-1} \wedge \neg h^2_{i,j+1} \wedge \neg h^2_{i,j-1} \wedge \neg v^1_{i+1,j} \wedge \neg v^2_{i+1,j} \wedge \neg v^1_{i-1,j} \wedge \neg v^2_{i-1,j}) \\
\vee\, &(h^1_{i,j+1} \wedge h^1_{i,j-1} \wedge \neg h^2_{i,j+1} \wedge \neg h^2_{i,j-1} \wedge \neg v^1_{i+1,j} \wedge \neg v^2_{i+1,j} \wedge \neg v^1_{i-1,j} \wedge \neg v^2_{i-1,j}) \\
\vee\, &...
\end{aligned}$$

Thus, we need to determine all valid assignments for a degree constraint n and express them in CNF instead of DNF. Rymanski relied on Tseitin transformation for the latter part which has linear time complexity. We propose a more efficient approach that does use any computational power on the transformation during the execution.

To find all valid assignments, that is all assigments without a crossing violation, we define the 1-indexed tuple of relevant variables

$$B = (h^1_{i,j+1}, h^2_{i,j+1}, h^1_{i,j-1}, h^2_{i,j-1}, v^1_{i+1,j}, v^2_{i+1,j}, v^1_{i-1,j}, v^2_{i-1,j})$$

and the operator $\mathit{type} : \{1, 2, ..., 8\} \mapsto \{0, 1\}$, $\mathit{type}(i) = 2 - (i \bmod 2)$ returning the number of bridges set by variable $B[i]$. Now, we can state the general DNF for a degree constraint n:

$$D(n) = \bigvee_{\substack{S\ \mathit{subtuple\ of}\ B,\\ \left(\sum_{s \in S} \mathit{type}(s)\right) = n}} \bigwedge_{b \in B} (b \in S)\ ?\ b : \neg b$$

We restrict the set of tuples S for D(n) by excluding invalid assignments.

In fact $D(n)$ is eqivalent to the conjuncted negations of $D(n')$ for $n' \in \{0, 1, ..., 7\} \backslash n'$, since we do not disregard any relevant valid assignments. We found a CNF encoding the degree constraint:

$$C(n) = \bigwedge_{n' \in \{0,1,...,7\} \backslash n} \neg D(n')$$

In the code, any assignment is represented by a list of numbers 1 to 8 (i.e. $h^1_{i,j+1} \vee h^1_{i,j-1} \vee \neg h^2_{i,j+1} \vee \neg h^2_{i,j-1} \vee \neg v^1_{i+1,j} \vee \neg v^2_{i+1,j} \vee \neg v^1_{i-1,j} \vee \neg v^2_{i-1,j} \equiv [1, 2, -3, -4, -5, -6, -7, -8]$). For every island node we define a dictionary mapping from (1,...,8) to B. This way, we never actually compute any transformations but only the target values for every degree contraint.

2.4 Connectivity