Nmbr9 as a Constraint Programming Challenge

Mikael Zayenz Lagerkvist

research@zayenz.se, https://zayenz.se

The Challenge

Nmbr9 is a popular board game, with a seemingly simple set of rules. Modelling Nmbr9 effectively is difficult, and finding the maximum possible score is an open hard challenge.

Rules

Nmbr9 is played using the polyominoes above representing 0-9, two copies per player. A common deck of 20 cards with the polyominoes are shuffled.

When a card is drawn, each player takes the corresponding part and places it in their own area. Placements are done in levels, where the first level is directly on the table.

- Placement must always be done fully supported. The bottom level is fully supported, and squares in layer n are supported by placements in respective squares in layer $n-1$.
- Each level must be fully 4-connected (i.e., not diagonally) after placement.
- A part must be on top of at least two different types of parts when not on bottom level.

The score is the sum of the value for parts times their level minus one. For example, an 8-part on level 3 is worth 16, while it is worth 0 on level 1 (the bottom level).

Variants

The standard game is intractable and just a single problem to solve. Variants of Nmbr9 are defined to get a larger sampling of easy to hard problems to solve.

A variant is denoted by $T{-}m{-}c{-}n$, with the following interpretations

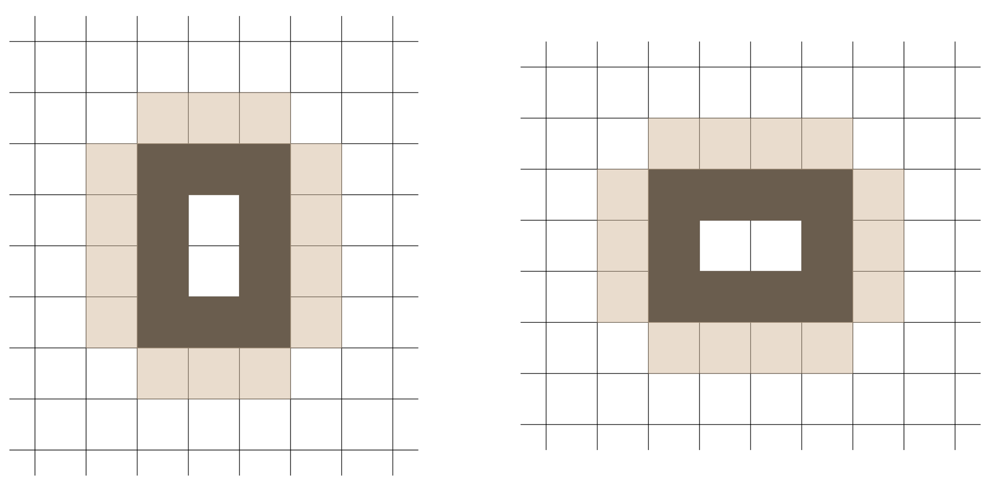
$T \in U, F, K$	Whether the draft is *unknown*, *free* to choose, or *known*. The first corresponds to a quantified problem, where the draft is $\forall$-quantified. The second models the question of the max possible score, while the last models the max possible score given a specific known shuffle.
m	The maximum value to use, from 0-9.
c	The number of available copies of each polyomino.
k	The number of cards in the deck to use, $k \leq (m+1) \cdot c$ must hold.

The standard game is $U{-}9{-}2{-}20$, where all available parts are used ($k = (m+1) \cdot c$).

Best known score for $F{-}9{-}2{-}20$

At Board Game Geek, a forum thread on solving the standard $F{-}9{-}2{-}20$ instance has a top score of 229 points using a total of 7 layers achieved by user Philippe Kuenzler.

Regular expression placement



The 0 part has two distinct rotations shown above, with the surrounding squares indicated in a lighter colour. With a grid of size s, empty squares, covered squares, and surrounding squares indicated by 0, 1, and 2 respectively, the following expression encodes reified placement on the grid in row-major order.

$$10^*(2^3 0^{s-4} 2 1^3 2 0^{s-5} (2101 2 0^{s-5})^2 2 1^3 2 0^{s-4} 2^3 \mid 2^4 0^{s-5} 2 1^4 2 0^{s-6} 2 1 0^2 1 2 0^{s-6} 2 1^4 2 0^{s-5} 2^4) 0^* \mid 00^*$$

Detailed model

Variables

- R_p is the regular expression for placing part p reified.
- $P = \{p_1, \ldots, p_n\}$ ($n = (m+1) \cdot c$) is the set of parts, $val(p) \in P \to 0..m$ their values
- $D = \langle d_1, \ldots, d_k \rangle \in P$ represent the deck of polyominoes.
- $O = \langle o_1, \ldots, o_n \rangle \in \{1..n\}$ the order of the polyominoes as they occur in the deck. Values above k all represent parts not in the deck..
- G_l is the grid for level l for all parts on that level (size $s \times s$, domain $0..n$, border is 0)
- G_{lp} the grid for a part p on level l (size $s \times s$, domain $0..2$).
- Variables G^1_{lp} and G^2_{lp} are Boolean matrices indicating where G_{lp} is 1 and 2 respectively.
- Boolean variables L_{pl} indicating if p is placed on level l
- $L_p \in 0..l_\top$ is the level of p (if any, 0 if not)
- Y_p are Boolean variables representing that p is placed, with N_p the inverse.
- $B(p, p')$ are Boolean variables indicating if p is before p' in D.

Constraints

Constraints are defined with implicit looping: p, p' ranges over P with $p \neq p'$, l over $1..l_\top$, l' over $2..l_\top$, v over $0..m$, and i, j over $1..s$.

Operators $\dot\vee$ and $\dot\wedge$ are used for point-wise logical operations. The notation $[M]$ is used to indicate true iff any element in the matrix M is true.

$\texttt{global\_cardinality}(D, \langle Y_1, \ldots, Y_n \rangle)$	(Deck is a shuffle of cards)
$\texttt{regular}(R_p, L_{pl} G_{lp})$	(Placement of parts)
$\texttt{inverse}(O, \langle D_1, \ldots, D_k, E_{k+1}, \ldots, E_n \rangle), E \in P$	(Order of parts in deck)
$B(p, p') \leftrightarrow O(p) < O(p'),\ O(p) \leq k \leftrightarrow Y_p$	(Order channeling)
$\texttt{int\_to\_bool}(L_p, \langle N_p, L_{p1}, \ldots, L_{pl_\top} \rangle),\quad Y_p = 1 - N_p$	(Level channeling)
$G_{lp}(i,j) = 1 \leftrightarrow G^1_{lp}(i,j),\quad G_{lp}(i,j) = 2 \leftrightarrow G^2_{lp}(i,j)$	(Aspect channeling)
$G_l(i,j) = p \leftrightarrow G_{lp}(i,j) = 1$	(Grid channel)
$\left(L_{pl} \wedge \exists_{p'' \mid B(p'',p)} L_{p''l}\right) \to \left[G^2_{lp} \dot\wedge \left(\dot\vee_{p' \mid B(p',p)} G^1_{lp'}\right)\right]$	(4-connected levels)
$G^1_{l'p}(i,j) \to \vee_{p' \mid B(p',p)} G^1_{l'-1p'}(i,j)$	(Placement supported underneath)
$L_{pl'} \to \left(2 \leq \sum_{p' \mid B(p',p)} [G^1_{lp} \dot\wedge G^1_{l-1p'}]\right)$	(On top of at least 2)
$S = \sum_{p \in P} (L_p - Y_p) \cdot val(p)$	(Score summation)

Implied and symmetry breaking constraints

Let $area(p)$ be the area of part p, variables $a = \langle a_1, \ldots, a_{l_\top} \rangle \in 0..s^2$ be the area of a layer, and variables V of shape $s \times s$ be the values of the bottom layer.

$a_l = \sum_{p \in P} area(p) \cdot L_{lp},\quad a_{l'-1} \geq a_{l'}$	(Area)
$\forall^k_{c=1} L_{D_c} \leq \lceil i/2 \rceil$	(Tile layer count)
$\texttt{precede}(D, \langle t \mid \forall t \in P \text{ where } v = val(t) \rangle)$	(Value symmetry)
$V(i,j) = val(G_1(i,j)),\quad \forall_{\alpha \in \{90,180,270\}} V \geq rot_\alpha(V)$	(Rotation symmetry)

Implementation

Implementation done using Gecode 6.2.0 and C++17.

Heuristic choices on deck first, then levels of parts, and finally placement. Placement uses spiralling static order from center to keep placements centered. Full problem is intractable with 2.5M propagators and 5 seconds to set up root space. $F{-}6{-}2{-}5$ with 3 levels and grid size 8 takes almost 3 minutes and 140k failures to solve for optimal score 15.

https://github.com/zayenz/cp-2019-nmbr9/