Generating a "fair" Settlers of Catan game board layout using Constraint Programming

Frej Knutar

27th November 2017

1 Settlers of Catan

1.1 Game introduction

Settlers of Catan[**?**] is a multi-player board game that is played on a board consisting of 19 hexagon terrain pieces. Each terrain piece has one of 6 terrain *piece types* and on top of each an *allotment number* is placed, except if the terrain type is the desert terrain type, then the terrain piece has no allotment number placed on top of it. The allotment numbers represent what result a pair of six-sided dies needs to show in order for that field to yield a resource corresponding to it. The terrain piece types are fields, forest, pasture, mountains, hills, and desert, yielding the resources grain, wood, wool, ore, brick, and nothing respectively. During the game the players can place *settlements*. Placing settlements is one of the main ways of the players to advance the game in their favor. The settlements are placed on *junctions*. Junctions are where two or three terrain pieces meet at the edges of those terrain piece hexagons. Given that that the allotment number of a terrain piece is shown on the dies, then each player that has one or more settlements on edge on the terrain piece receives one resource of the appropriate for each such settlement.

1.2 Game board fairness

The placement of the allotment numbers and the terrain pieces greatly dictate what junctions and terrain types that are considered *profitable* and *unprofitable*. It is considered profitable to place settlements in junctions with high *scoring* allotment values. The score of an allotment number is how likely it is that two six-sided dies shows the allotment number.

1.2.1 Junction fairness

The score of a junction the board can be described as the sum over all the scores of the allotment numbers in that junction. The overall fairness of the allotment numbers of the junctions can then be said to be the difference between a highest scoring junction compared to a lowest scoring junction, where only junctions with three terrain pieces are compared. A small difference indicates that the allotment numbers have a good distribution over the junctions of the board, while a big difference indicates that they have not.

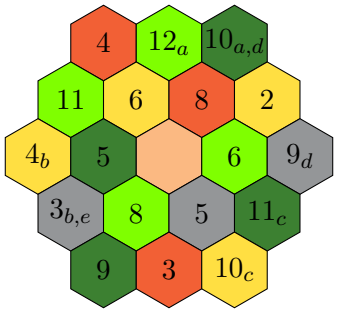

Figure 1: An example of a Settlers of Catan board setup, and a solution to the COP. The gold yellow, dark green, light green, dark gray, orange, and light brown terrain pieces are fields, forest, pasture, mountains, hills and desert respectively. The subscript notation of the board pieces with characters a, b, c, d, and e indicates that those pieces are resource harbours and that pieces of piece types fields, forest, pasture, mountains, and hills cannot be placed on those respectively.

1.2.2 Terrain type fairness

The allotment score of a terrain type t can be described as the mean of the of the allotment number scores of all the terrain pieces of type t. Similarly to when scoring junctions, the overall fairness of the allotment numbers of the terrain types can be said to be the difference between a highest mean of the terrain types compared to a lowest mean of the terrain types. A small difference indicates that the allotment numbers have a good distribution over the terrain types, while a big difference indicates that they have not.

1.2.3 Harbours

Some junctions on the board are considered to be *harbours*. Some of these harbours have a resource type associated with it, and terrain pieces with the corresponding resource type may not be placed so the harbour is a junction of that tile.

1.3 Game board layout requests

Some requests of players have helped stating constraints that either make the game more balanced or make the game more fun for those players. One such constraint is that each terrain piece may not have any neighbouring terrain pieces with the same type as itself. Another request was that 3 the 6 most centric allotment numbers must be either 8 or 6.

2 Model

This section will give a constraint optimisation model (COP) of the Settlers of Catan game board fairness problem. In this problem the desert terrain piece is always placed in the centre of the board; on row 3 and column 3. The desert terrain piece is considered to be not active.

2.1 Data and Derived Constants

- *rows* designates the ordered array of row indices $[1, 2, \ldots, 5]$.
- *columns* designates the ordered array of column indices $[1, 2, \ldots, 5]$.

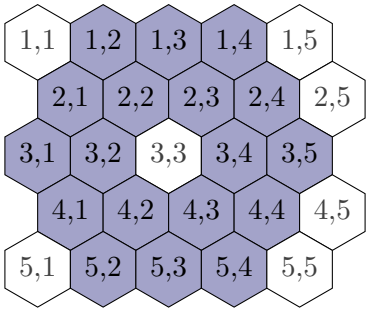

Figure 2: The hexagon tiles. The blue tiles are the active tiles of the board, while the white are the non-active tiles. The pair of numbers on each tile represents that tiles row and column position respectively. In this model the desert tile is considered to not be active.

- *pieceTypes* designates the ordered array of the different board piece types. The different board pieces are fields, forest, pasture, mountain, hills, and desert, corresponding to the integers 1, 2, 3, 4, 5, and 6 respectively. Hence $pieceTypes = [1, 2, 3, 4, 5, 6]$
- *pieceCounts* designates the number of instances of each terrain piece type. There exist $pieceCounts[f]$ piece types of terrain piece type $f \in pieceTypes$. Here $pieceCounts = [4, 4, 4, 3, 3, 1]$.
- *allotmentNumberValues* designates the values that the allotment numbers can take, here the ordered array $[2, \ldots, 6, 8, \ldots, 12]$. Note that 7 is **not** an allotment number.
- *allotmentNumberScores* designates the scores of the allotment numbers, here the ordered array $[\frac{1}{36}, \frac{2}{36} \ldots, \frac{5}{36}, \frac{5}{36}, \frac{4}{36} \ldots, \frac{1}{36}]$,
  hence $|allotmentNumberValues| = |allotmentNumberScores| = 10$.
- *allotmentNumberCounts* designates the number of instances for each allotment number. There are $allotmentNumberCounts[i]$ allotment numbers for allotment number value $a = allotmentNumberValues[i]$, hence $|allotmentNumberValues| = |allotmentNumberCounts| = 10$. Here $allotmentNumberCounts = [1, 2, 2, 2, 2, 2, 2, 2, 2, 1]$.
- *activePieces* designates the two dimensional indicating if game board pieces are considered active or not. Given $a = activePieces[r][c], r \in rows, c \in columns$, then a game board piece must be placed on the board on row r and column c if $a = 1$, otherwise $a = 0$ and a terrain piece cannot be placed on row r column c. Hence $activePieces[r, c] \in \{0, 1\}$. The active pieces can be seen in 2.
- *junctions* designates the ordered array holding the junctions on the board where three active terrain pieces intersect. Each junction holds a set of three board positions $\{\langle r_1, c_1\rangle, \langle r_2, c_2\rangle, \langle r_3, c_3\rangle\}$, where $r_i \in rows, c_i \in columns, i \in \{1, 2, 3\}$.
- *harbours* designates the terrain piece locations that are considered resource harbours on the board. There are five harbours, one for each resource type, and each occupy two adjacent tiles.

2.1.1 Decision Variables

- *pieces* designates the ordered array of terrain piece types of the board. If the piece in row $r \in rows$ and column $c \in columns$ is active, $activePieces[r, c] = 1$, then $pieces[r, c] \in$

pieceTypes, else the piece is not active, $activePieces[r, c] = 0$, and $pieces[r, c] = 0$

- *allotmentNumbers* designates the allotment numbers of the terrain pieces. If the piece in row $r \in rows$ and column $c \in columns$ is active, $activePieces[r, c] = 1$, then $allotmentNumbers[r, c] \in allotmentNumberValues$, else the piece is not active, $activePieces[r, c] = 0$, and $allotmentNumbers[r, c] = 0$

- *allotmentScores* designates the ordered array of allotment number scores of the terrain pieces. If the piece in row $r \in rows$ and column $c \in columns$ is active, $activePieces[r, c] = 1$, then $allotmentScores[r, c] \in allotmentNumberScores$, else the piece is not active, $activePieces[r, c] = 0$, and $allotmentScores[r, c] = 0$

- *junctionScores* designates the ordered array of sums over the allotment number scores in the junctions where three active pieces meet. The $junctionScores[i]$ is the sum over the allotment numbers scores of the junction $junctions[i]$, hence $|junctionScores| = |junctions|$.

- *PieceTypeScoreMean* designates the ordered array of mean allotment number scores of the terrain piece types. The terrain piece type $1 \leq t \leq 5$ has mean score $PieceTypeScoreMean[t]$, since the desert type of 6 is ignored.

2.2 Problem Constraints.

- *Harbour constraint* require that a terrain piece with the same type as a harbour is not placed on the same board position as the harbour:

$$\begin{gathered}\textbf{for all } r \textbf{ in } rows, c \textbf{ in } columns, \textbf{ where } harbours[r, c] \neq 0 : \\ pieces[r, c] \neq harbours[r, c]\end{gathered} \tag{1}$$

- *Allotment score constraint* require the values in each column of:

$$\begin{gathered}\textbf{for all } r \textbf{ in } rows, c \textbf{ in } columns \textbf{ where } activePieces[r, c] = 1 : \\ allotmentScores[r, c] = allotmentNumberScores\Big[allotmentNumbers\,[r, c]\,\Big]\end{gathered} \tag{2}$$

- *Piece type counts* require that the number of instances of each piece type is equal to the number of terrain pieces of that type:

$$\textbf{for all } t \textbf{ in } pieceTypes : \textsc{Count}(pieces, t) = pieceCounts[t] \tag{3}$$

- *Unique piece type junctions* require that the piece types of all pieces in a junction are pairwise different:

$$\textbf{for all } j \textbf{ in } junctions : \textsc{Distinct}\Big(\,\{pieces[r, c] \mid \langle r, c\rangle \in j\}\,\Big) \tag{4}$$

- *Unique allotment number junctions* require that the allotment numbers of all pieces in a junction are pairwise different:

$$\textbf{for all } j \textbf{ in } junctions : \textsc{Distinct}\Big(\,\{allotmentNumbers[r, c] \mid \langle r, c\rangle \in j\}\,\Big) \tag{5}$$

- *Junction allotment scores* sets the scores of the allotment numbers of the junctions:

$$\begin{gathered}\textbf{for all } 1 \leq i \leq |junctions| : \\ junctionScores[i] = \textsc{Sum}\Big(\,\{allotmentScores[r, c] \mid \langle r, c\rangle \in junctions[i]\}\,\Big)\end{gathered} \tag{6}$$

- *Piece type score means* sets the score of the means of the terrain piece types:

$$\begin{gathered}\textbf{for all } 1 \leq t \leq 5: \\ PieceTypeScoreMean[t] = \\ \frac{\textsc{Sum}\Big(\{allotmentScores[r,c] \mid r \in rows, c \in columns, pieces[r,c] = t)\}\Big)}{pieceCounts[t]}\end{gathered} \tag{7}$$

None of the problem constraints is enforced automatically by the choice of decision variables.

2.2.1 Objective Function

This problem is a constraint satisfaction optimization problem, so it requires an objective function

Objective function The objective function is a minimisation function that gives the overall fairness of the board s:

$$s = \frac{\max(junctionScores) - \min(junctionScores)}{3} + \begin{pmatrix}\max(PieceTypeScoreMean) \\ - \\ \min(PieceTypeScoreMean)\end{pmatrix} \tag{8}$$

2.2.2 Redundant Decision Variables

My model could be said has redundant decision variables, since *allotmentNumbers* and *allotmentScores* both represent the allotment numbers of the board. However, the scores are the probabilities of the allotment numbers where two different allotment numbers have the same score.

2.2.3 Symmetry-Breaking Constraints

Since the harbours might break the symmetries of how the terrain pieces are placed, no symmetry-breaking constraints are posted over the terrain piece types. However, there exist symmetries over the allotment scores since an allotment score can give two different allotment numbers. This symmetry is broken by stating that the top left active terrain piece must have an allotment number less than 7:

Symmetry Breaking this constraint ensures symmetry-breaking over the allotment scores, here $allotmentScores[r,c]$ is the top-left most active terrain piece, $activePieces[r,c] = 1$:

$$allotmentScores[r,c] \leq 7 \tag{9}$$

2.3 Implementation

A *MiniZinc* [**?**] implementation of the described model is posted in the repository as file `catan.mzn`.

2.3.1 Compilation and Running Instructions

Open and run the program in the MiniZinc solver. For now the model only solves the problem for the active fields setup described in **??**

2.4 Experiments

My experiments were run under Linux 4.13.5-1-ARCH (64 bit) on an Intel Core i5 2557M of 1.7 GHz with an 3 MB L2 cache and a 4 GB RAM.

2.4.1 Result

The most fair board that was found is shown in 1.3. This board layout was found after 1.45 seconds. The program was left running for over 10 hours, and no better board layout was found.