Nurikabe

Johan Groenen*

September 17, 2008

Abstract

Nurikabe, a pencil-and-paper, binary determination puzzle from the creators of Sudoku, is on the verge of breakthrough in Europe as the successor of the Sudoku puzzle. Played on an $m \times n$ grid, the goal is to determine for every cell whether it should be colored white or black, much like Sudoku and Minesweeper. A number in a cell on the puzzle grid indicates the size of the island the cell containing the number belongs to. These white islands are separated by a black wall, which is called *Nurikabe*. Since it is NP-complete, it can be difficult to solve a given puzzle. Compared to Sudoku, relatively little effort has been made in finding algorithms for solving these puzzles. In this paper we will describe methods for automated solving and generation of Nurikabe Puzzles.

1 Introduction

Nurikabe is a pencil-and-paper, binary determination puzzle, first published in March 1991 [Nikoli, 1991]. The puzzle consists of a rectangular grid of cells, some of which contain a number describing the size of the islands they belong to. The goal of the game is to identify all islands and the stream separating them (the *Nurikabe*). Examples of other binary determination puzzles are Sudoku and Minesweeper. These puzzles are studied by computer scientists in the field of *Artificial Intelligence* [Russell and Norvig, 2003].

*Leiden University, LIACS, jgroenen@liacs.nl

In Japanese folklore, a Nurikabe is an invisible wall that blocks roads and upon which delays in foot travel are blamed [Wikipedia, 2008]. An other name for the game is "Islands in the Stream", which to our opinion is a better metaphor. We will use this metaphor throughout this paper.

Nurikabe is one of many "culture independent" puzzles in the library of logical puzzle publisher Nikoli, and might well be the successor of the popular Sudoku puzzle, as it has the key ingredient for a good logical puzzle: it is easy to understand, but it can be hard to solve.

This Bachelor Thesis is the result of the research done by Johan Groenen as part of the Bachelor Computer Science at the Leiden Institute of Advanced Computer Science, under supervision of Walter Kosters[1].

2 Related Research

Brandon McPhail [McPhail, 2003] has proven that Nurikabe is NP-complete, by showing that logical expressions can be translated into a Nurikabe by constructing Nurikabe `NOT` and `OR` ports. This suggests that solving a Nurikabe Puzzle becomes exponentially harder when the puzzle becomes larger, and there is no simple way of solving it.

Caroline Terzer [Terzer, 2007] defines a way to translate Nurikabe Puzzles into propositional logic statements, which can then be solved/evaluated by a "state of the art SAT-solver".

In his Bachelor Thesis, Ramon van Dam [van Dam, 2008] introduces a systematic way to generate *Slitherlink* puzzles using only 3 kinds of mutations, that can be used to iteratively alter a simple puzzle into a larger one.

Tim van Meurs is currently preparing a Bachelor Thesis on another puzzle from Nikoli called *Takegaki* [van Meurs, 2008].

3 Notations and Concepts

In this section we will define the *Nurikabe Grid*, after which we will use this concept to introduce the *Nurikabe Puzzle* and its Solutions.

[1]Leiden Institute of Advanced Computer Science, Leiden University, kosters@liacs.nl

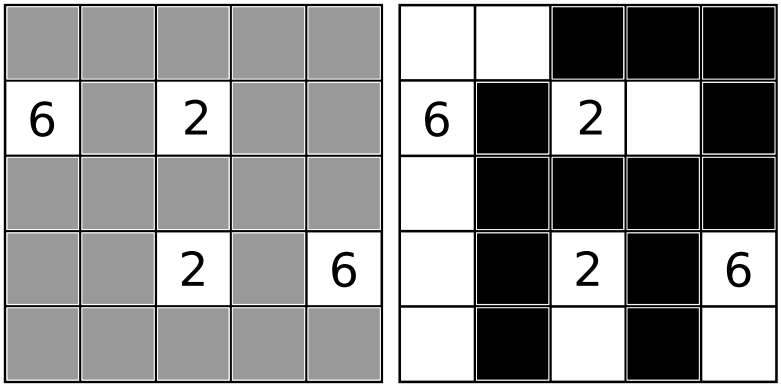

Figure 1: A Nurikabe Puzzle and its Full Solution of it.

3.1 Graph Representation of Nurikabe Grid

An $m \times n$ *Nurikabe Grid* is defined as an undirected graph $G = (V, E)$ with

$$V = \{(i, j) | i \in \{1, \ldots, m\}, j \in \{1, \ldots, n\}\}$$
$$E = \{(a, b) | a, b \in V, adj(a, b)\}$$

Here V is the set of all *cells* of the grid, and $adj(a, b)$ is the adjacency relation on these cells (see Figure 2). Two cells $a = (i, j)$ and $b = (k, \ell)$ are defined to be *adjacent*, indicated with $adj(a, b)$, if

$$|i - k| + |j - \ell| = 1$$

A 2×2 *square* is defined as a set of four cells such that every cell in it is adjacent to exactly two other cells in the set.

3.2 Nurikabe Puzzle

An $m \times n$ *Nurikabe Puzzle* P is a function that maps the set of vertices V of a Nurikabe Grid G to the *Nurikabe Puzzle Space* $\{e\} \cup \mathbb{Z}^+$:

$$P : V \to \{e\} \cup \mathbb{Z}^+$$

A cell $a \in V$ is called a *pivot* if $P(a) \in \mathbb{Z}^+$. If $P(a) = e$, the cell is said to be *empty*. The *radius* of a pivot is the area formed by all cells that are part of a possible island containing this pivot. The left grid in Figure 1 is a graphical

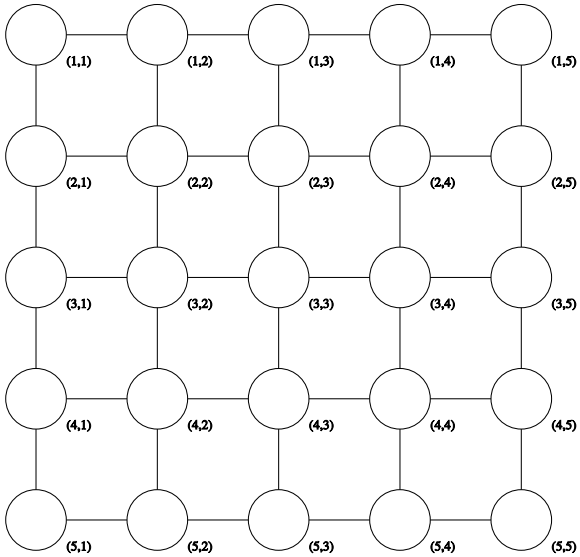

Figure 2: A 5×5 Nurikabe Grid.

representation of a Nurikabe Puzzle, where empty cells are colored grey, and pivots are colored white and contain their number. If coloring is unavailable, as is the case in text input files, empty cells are indicated with 0, as can be seen in Figure 3, which represents the same puzzle as Figure 1. (The two fives at the top represent the dimensions of the Nurikabe Puzzle.)

```
5 5
0 0 0 0 0
6 0 2 0 0
0 0 0 0 0
0 0 2 0 2
0 0 0 0 0
```

Figure 3: A Nurikabe Puzzle text input file.

3.3 Partial Solution

A *Potential Solution*, S_{Pot}, is a function that maps V to the *Partial Solution* space $\{e, w, b\} \cup \mathbb{Z}^+$:

$$S_{Pot} : V \to \{e, w, b\} \cup \mathbb{Z}^+$$

A cell $a \in V$ for which $S_{Pot}(a) = e$ is called *empty*, represented by a grey cell. If $S_{Pot}(a) = w$, the cell is called *white*, represented by a white cell, and *black* if $S_{Pot}(a) = b$, represented by a black cell. If S_{Pot} maps a to $\mathbb{Z}^+$, it is a *pivot*. Both white cells and pivots are called *positive*.

If a 2×2 square consists of only black cells, it is called a *pool*. Looking at the black cells only, a connected component is called a *stream*. Looking at the possitive cells, a connected component is called an *island*. A cell is called *covered* if it is in the radius of one or more pivots, and *uncovered* if not.

An island is called *fixed* if no cell of the island is adjacent to an empty cell. If an island is not fixed, it is called *open*. A fixed island is called *complete* if its size in cells is equal to the value of its pivot, and *incomplete* if smaller.

The *neighborhood* of an island / stream is the set of empty cells that are adjacent to one or more cells in the island / stream.

A *Partial Solution* of a Nurikabe Puzzle P, S_{Part}, is a Potential Solution in which

- For every cell $a \in V$ holds that $S_{Part}(a) \in \mathbb{Z}^+$ if and only if $P(a) \in \mathbb{Z}^+$, and in this case $S_{Part}(a) = P(a)$.
- Every open island contains at most one pivot and if it contains a pivot, then it consists of at most the number of positive cells indicated by the value of this pivot.
- Every fixed island contains exactly one pivot and it consists of exactly the number of positive cells indicated by the value of this pivot.
- The total number of positive cells is smaller than or equal to the sum of all pivot values.
- There are no pools.

3.4 Full Solution

A *Full Solution* (or simply *Solution*) of a Nurikabe Puzzle P, S_{Full} (or simply S) is a partial solution in which

- There are no empty cells.

- There is only one stream (the *Nurikabe*).

We define $Sol(P) = \{S_1, S_2, \ldots, S_k\}$, the set of all full solutions of a Nurikabe Puzzle P. A Nurikabe Puzzle is called *solvable* if it has at least one full solution ($|Sol(P)| \geq 1$). If not, it is called *unsolvable*. A Nurikabe Puzzle P is called *good* or *correct* if it has exactly one full solution ($|Sol(P)| = 1$).

3.5 Added Restrictions

In this paper we will only look at puzzles in which every pivot has a value smaller than 10, because this makes input easier, the search space smaller and will leave the puzzle intact. Also, it looks nicer.

4 Solution Methods

Different strategies can be used to solve a given Nurikabe Puzzle. For example, *Logical Rules* can be used. The only problem is that it might be hard or even not possible to find a finite set of rules that will solve any given correct $m \times n$ Nurikabe Puzzle. It is also hard to find a finite set of rules for given m and n, without knowing all correct puzzles and their solutions. Since we want to solve all solvable puzzles and check for unique solvability, we will look at *Brute Force* algorithms, extended with some *backtracking* and optimizations.

4.1 Permutate Positive Cells

Full enumeration of positive cells on the grid gives a very large search space of $2^{m \times n}$ possibilities, which is highly inefficient. Since the exact number of positive cells of the Solution is always the sum N of all p pivot values, and thus is known in advance, permutations of exactly this number of positive cells on a grid give a much smaller search space, of size

$$\binom{m \times n}{N} = \frac{(m \times n)!}{(m \times n - N)! \times N!}$$

Also, since the pivots themselves are positive, they can be fixed at their original position, and the search space becomes even smaller:

$$\binom{m \times n - p}{N - p} = \frac{(m \times n - p)!}{(m \times n - p - (N - p))! \times (N - p)!}$$
$$= \frac{(m \times n - p)!}{(m \times n - N)! \times (N - p)!}$$

Still, if we would like to find a solution to for example the 10×10 puzzle in Figure 4 (44 white cells, 15 pivots), the number of possibilities to try would be

$$\binom{10 \times 10 - 15}{44 - 15} = \frac{(10 \times 10 - 15)!}{(10 \times 10 - 44)! \times (44 - 15)!}$$
$$= \frac{85!}{56! \times 29!}$$
$$= 4.48 \times 10^{22}$$

If we would test one configuration per clockcycle, then it would take about $140,000$ years on a 10 GHz processor to test all configurations. This is much too long, of course.

							5		2
3									
	4			2					
						3			
	4				4				
									3
	3			3					
		1			1		3		3

Figure 4: A 10×10 correct Nurikabe Puzzle.

4.2 Recursively Expand Neighborhoods

If we only examine all possible island shapes by iteratively adding cells from an islands neighborhood to the island until it is full, it is possible to reduce the search space even more. This is because of the following observations:

1. White cells are distributed only over positions in which they belong to exactly one island.

2. Islands are always of the right size.

3. Since islands may be unable to expand because of other islands or grid boundaries, or because the stream is divided by such an expansion, it is possible to skip large parts of the configuration space (*"back tracking"*).

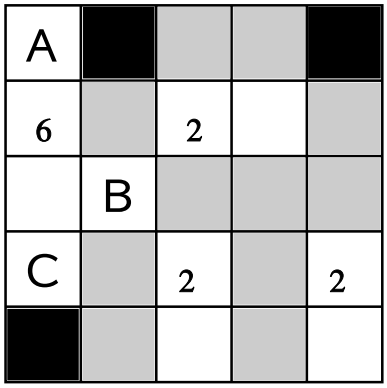

Figure 5: White cells are assigned. Cells A, B and C represent the neighborhood of "6", to which this island can be expanded.

The algorithm expands the islands recursively, every step adding a cell from the island's neighborhood to it, then updating the neighborhood (see Figure 5). If the island is full, we expand the next island. If there is no next island, then we check if there are no pools and only one stream, and if so, we found a solution.

It is possible to make sure that every configuration is only generated once, by not allowing expansion to cells that have been tried earlier (since every configuration containing that cell has been tried before). As this is a brute-force algorithm, it is possible to find all solutions to a given Nurikabe Puzzle, thus checking whether it is (uniquely) solvable.

An upper limit for the number of possible configurations can then be calculated by multiplying the maximum number of configurations (see Figure 7 and Figure 8) of an island of certain size. Since most pivots will not be free in a uniquely solvable puzzle, but rather be constrained by other islands, the upper limit is very loose. For example, the puzzle in Figure 4 has an

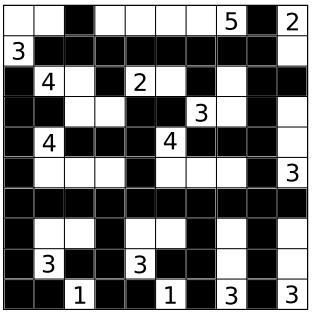

Figure 6: Solution of Nurikabe Puzzle from Figure 4.

upper limit of about 1.4×10^{18}, which would still take about 4 years if no backtracking took place, but in reality it takes about 10 seconds to find its solution (see Figure 6).

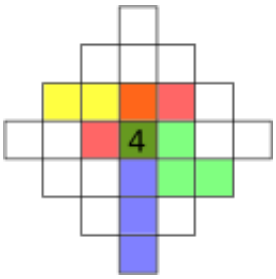

Figure 7: Possible configurations of an island of size 4 are all positive connected components of 4 cells that contain the pivot.

5 Generating Puzzles

To guarantee a uniform random pick from the collection of all $m \times n$ Nurikabe Puzzles, one can imagine generating all possible puzzles one way or the other, checking which of them are correct and saving these in a database. Since then the number of good puzzles would be known, a uniform random pick can be made.

There are at least two ways of doing this. The first method is to generate all possible solutions (which, for example, for 5×5 puzzles could be represented by all 25-bit strings), throw away all solutions containing pools

size	free	wall	corner
1	1	1	1
2	4	3	2
3	18	10	5
4	76	33	13
5	315	112	37
6	1296	391	112
7	5320	1394	353
8	21800	5094	1146
9	89190	18485	3811

Figure 8: The number of possible configurations for given island size.

or having multiple (or zero) streams, then create all corresponding puzzles by placing numbers in every way possible, sort the resulting set of puzzles and select all puzzles in this set that appear only once (meaning they have only one solution).

If the goal is to generate all puzzles, then this method is nice: it is fast, because it does not need to solve all puzzles. If the goal is to generate puzzles, then this method will become very bad, since it has to generate and sort all puzzles before being able to select the correct ones. This is not a problem for small m and n, but when the size increases, both time and space needed to generate all correct puzzles become infeasibly large, and this method will be practically useless.

The second method is to generate all puzzles and check them for correctness.

The number of grids to generate for the 5×5 grid with maximum pivot value 9 would be 10^{25}, since each of the 25 cells is assigned a number out of $[0, 9]$. But since no two pivots may be neighbors and the number of white cells is bound to a maximum, it is possible to skip a lot of configurations.

For example, it is possible to calculate the minimal number of black cells needed to separate all islands. An underestimate of the number of black cells needed to separate an island, I, from its neighboring islands is given by

$$min(min(m, n), \lfloor \sqrt{|I|} \rfloor)$$

To calculate the minimum number of black cells needed in a grid to separate all islands, ignore the largest island and for all others add the minimum cal-

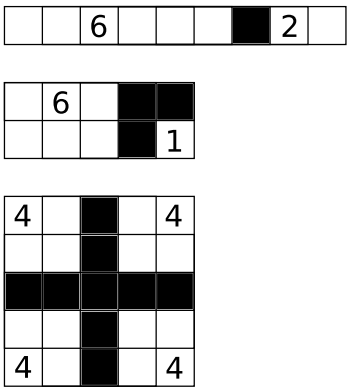

Figure 9: Grids showing black cells needed to separate islands.

culated with the value above. Although this might not seem very promising, it decreases the number of puzzles that have to be created significantly, since the maximum number of white cells is $m \times n$ minus the number of black cells, and the sum of all pivots may not exceed the maximum number of white cells.

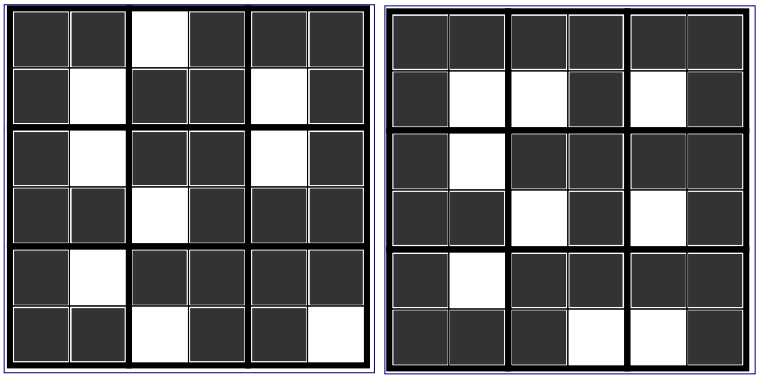

Figure 10: An "even" grid can be divided into 2×2 squares. As can be seen in the left grid, this does not guarantee absense of pools or of multiple streams.

Note that at least 25% of the cells need to be white in the case m and n are even[2]. This is because in this case the grid can be divided into 2×2 squares, and in every square there will have to be at least 1 white cell for it to

[2]When m or n is uneven, this percentage may vary.

not be a pool, as can be seen in Figure 10. Note that this type of placement of white cells does not guarantee absense of pools, nor does it eliminate the possibility of multiple streams.

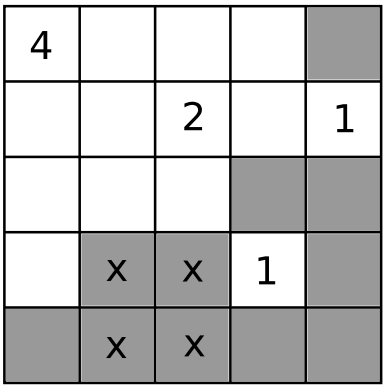

Figure 11: A 2×2 area is not covered, which will always lead to a pool.

One last optimization that can be used is to check in advance if all pools can be eliminated. This can be done using the radius of every pivot. A cell is called *reachable* from some pivot if there exists a path from the pivot to the cell which has a length smaller than the pivot value. The part of the grid that consists of cells that are reachable from one or more pivots, is called *covered*. If there exist 2×2 squares in uncovered area of the grid, these will always lead to pools, and thus the puzzle is unsolvable and does not need further evaluation, as can be seen in Figure 11.

6 Results

6.1 Generating Nurikabe Puzzles

We generated all correct Nurikabe Puzzles for $m \times n \leq 25$, with the exeption of $1 \times n > 20$, with the value of each pivot in the range of $[0, 9]$. This is only possible for very small m and n, since the time needed to generate all puzzles and check for unique solvability increases exponentially, as can be seen in Figure 12.

From Figure 13 it seems that the number of puzzles grows exponentially with the grid size. Because of the restriction that pivot values may not be larger than 9, the $1 \times n$ grid has a limited number of possible correct puzzles.

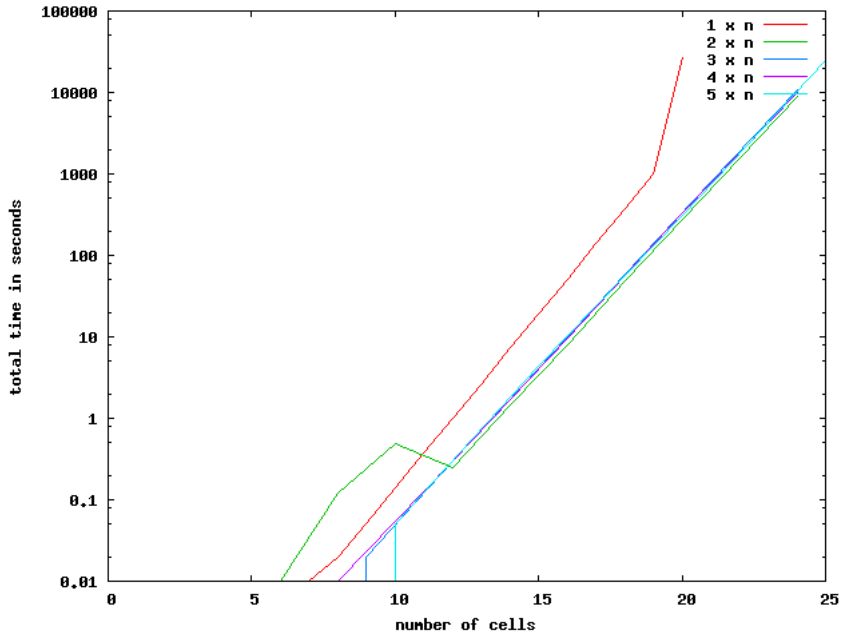

Figure 12: Times to generate all puzzles.

(Because multiple streams are forbidden, all black cells have to be in the center.)

Also, as can be seen in Figure 14, the relative presence of correct Nurikabe Puzzles in the searchspace containing all possible grids where every node has a value in $[0, 9]$ decreases exponentially.

In Figure 15 we see that the total time to calculate all puzzles grows linearly with the total number of puzzles.

6.2 Statistical Analysis

The total number of correct 5×5 puzzles with a maximum pivot value of 9 is 935763. Since we generated all of them, it is possible to give some statistical information.

The number of puzzles containing only pivots of value 1 is 303. The number of puzzles containing no pivots of value 1 is 128407.

The number of pivots in a 5×5 Nurikabe Puzzle is 4.72 ± 0.58. In Figure 16 the chances of finding a pivot at a certain position are given, as well as the average value per cell, which is 2.48 ± 1.62.

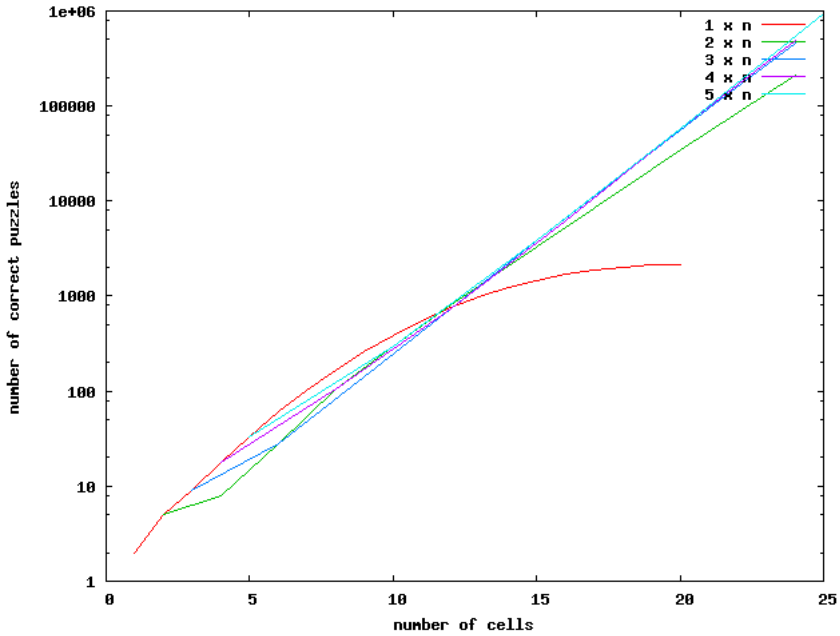

Figure 13: Number of puzzles.

Over all, the percentage black cells is 53.1 ± 7.8, which correctly correspond to 4.72 pivots $\times$ 2.48 white cells per island, out of 25 (which is 46.9%) white cells.

Of all puzzles that have a solution, nearly half is uniquely solvable (45.4%). 18.4% has exactly two solutions and 10.2% has three solutions. There are 10 puzzles with 212 solutions, which is the highest number of solutions. One of those puzzles can be found in Figure 17.

7 Scalability

There are some special Nurikabe Puzzles, which are in a sense "scalable", that is, for given n, there are instant uniquely solvable Nurikabe Puzzles. Some of them are discussed in this section.

The first of these are regular patterns that can be seen in Figure 18 and 19. These patterns can be repeated and will always give a uniquely solvable (though of course not very amusing) Nurikabe Puzzle.

It is possible to rotate the 2×2 squares, and when neighbouring pivots are

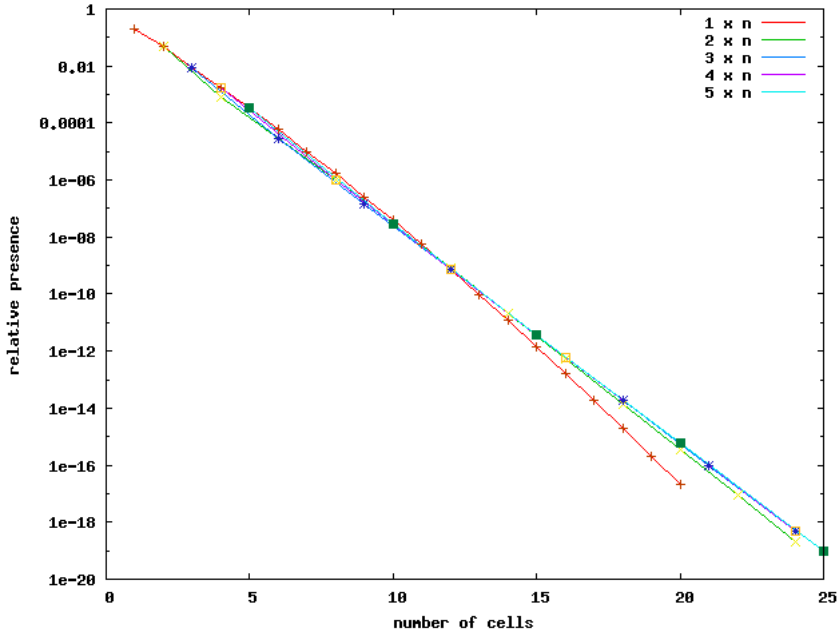

Figure 14: Relative presence of puzzles.

merged into one, then this will lead to a good Nurikabe Puzzle. We expect that carefully connecting some islands, adding white cells and merging them, constantly checking the (unique) solvability to generate might be a nice way to generate new good Nurikabe Puzzles.

Also, there are some examples of Nurikabe Puzzles with a high percentage of white cells that are interesting. For example an $n \times n$ Nurikabe Puzzle with odd n with in every corner a pivot of value $\lfloor n/2 \rfloor^2$. This leaves $2n-1$ black cells to separate the white islands, which have to be in a cross formation at the center of the puzzle, connecting the center cell of the top row in a straight line with the center cell of the bottom row, and the center cell of the left colum with the center cell of the right colum in a straight line too. This Nurikabe Puzzle is uniquely solvable.

This can be informally proved by noticing that the minimal number of cells needed to separate the left two pivots from the right two pivots is $n+k$, the Manhattan distance between C and D, where n is the grid size, and k is the column distance between C and D (see Figure 20). To separate the top pivots from the bottom pivots, we need at least $n+l$, the Manhattan distance between A and B, where l is the row distance between A and B.

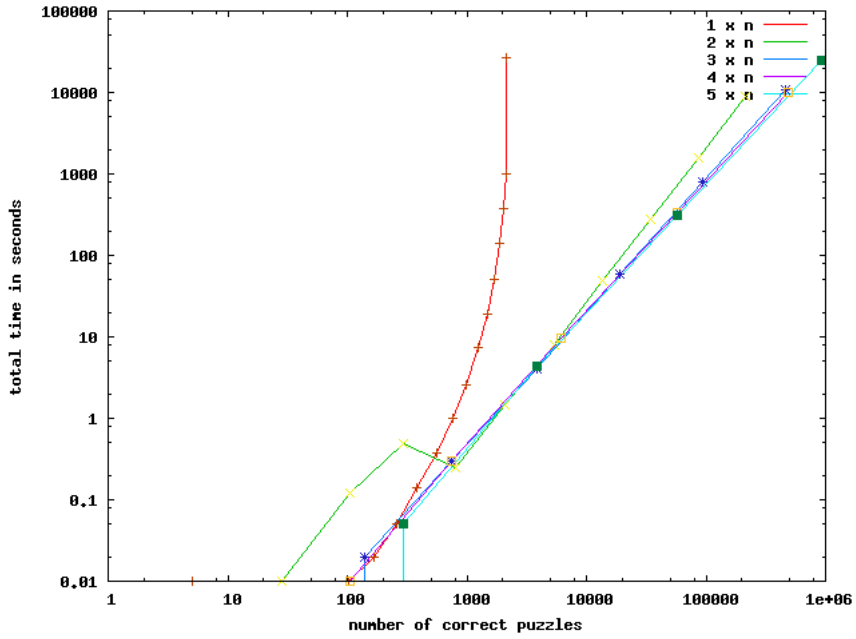

Figure 15: Total number of puzzles vs. total time to generate them all.

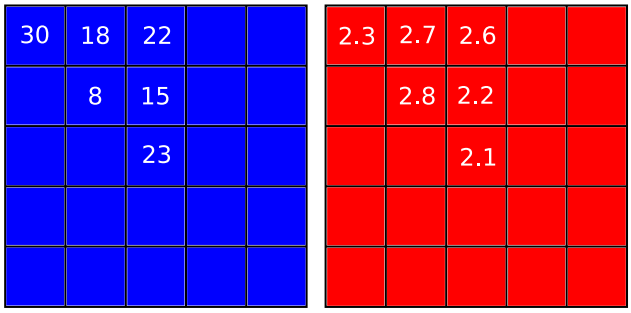

Figure 16: On the left the chance of finding a pivot at a certain position in the grid; On the right the average value of the pivot at a certain position.

The number of black cells needed to separate all pivots is the sum of these two, minus the overlap between them, $2n + k + l - Overlap$. Since we have $2n - 1$ black cells, $k + l - Overlap = -1$, thus $k + l = Overlap - 1$.

This means that A and C need to be connected rectangularly, as well as B and D, and the connecting stream must have the minimal number of cells to connect $X1$ and $X2$ (see Figure 21). Now if A would be above the center

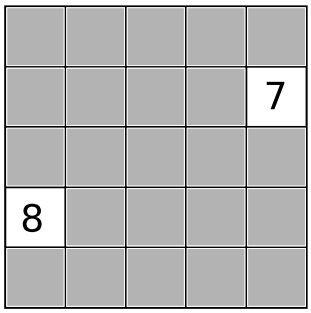

Figure 17: This puzzle has 212 solutions.

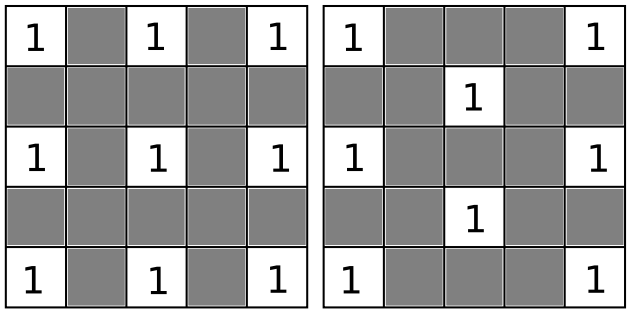

Figure 18: Two regular patterns leading to a good Nurikabe Puzzles. Notice that these patterns work for both even and odd grid sizes.

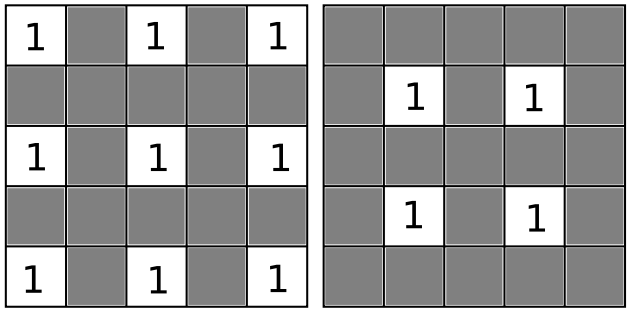

Figure 19: The same regular pattern shifted to minimize the number of white cells.

of the column, then C would have to be to the right of the center of the row, since the number of cells is fixed. Then B would have to be below the middle of the column and D would have to be left of C. But D cannot be left of C,

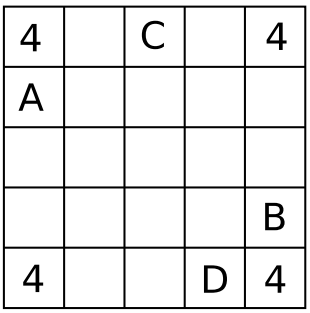

Figure 20: There must be at least one black cell on every side of the square (for example A, B, C and D), because otherwise the two pivots on that side would be connected by white cells.

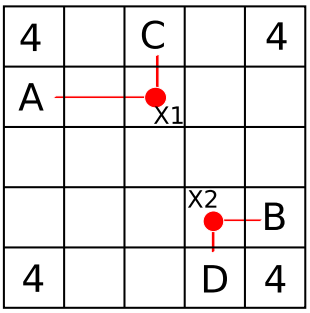

Figure 21: X1 and X2 need to be connected with the minimal number of black cells.

because then the number of black cells would not be minimal.

Since the puzzle is symmetrical, the same goes for A below the center of the column, and also for B, C, and D. They all need to be in the center of their row/column, and thus this gives a unique solution, as can be seen in Figure 22.

Another example is an $n \times n$ Nurikabe Puzzle where there are two pivots that add up to $n^2 - n$. This leaves n black cells to separate the white islands, which is just enough to draw one straight line from the bottom to the top or from the left to the right. If the pivots are placed in a way that disables one of the two lines, and $\lceil 2 \times \sqrt{p_{min}} + 1 \rceil > n$, where p_{min} is the smallest pivot, the puzzle is uniquely solvable. An example of this can be found in Figure 23.

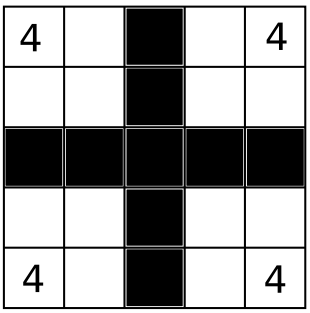

Figure 22: The unique solution.

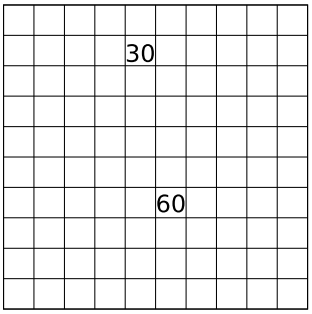

Figure 23: The solution has a straight black row.

8 Conclusions and Future Research

The number of correct Nurikabe Puzzles seems to grow exponentially with the number of cells in the grid. Also, the time to generate all these puzzles grows exponentially, making it unfeasible to generate all puzzles for large grids. This seems to indicate that perhaps other methods should be used, like evolutionary algorithms. On the other hand, the relative presence of correct Nurikabe Puzzles in the searchspace decreases exponentially, making it very hard to find such a puzzle.

One of the main targets for the future would be generating larger puzzles, in a randomly fashion. Perhaps using evolutionary algorithms, since random initialisation followed by total enumeration will be less and less effective when the searchspace gets more and more sparce, as is the case for Nurikabe Puzzles of increasing size. For this evolutionary algorithm, correct initial grid and mutation and selection operators need to be found. Selection could (or maybe should) be multi-objective.

Another way to generate these puzzles would be to expand the Nurikabe from one or more seeds, iteratively increasing the stream size(s) (and connecting them). Ramon van Dam has done something simular in his Bachelor Thesis on Slitherlink [van Dam, 2008].

Another thing is to solve (large) Nurikabe Puzzles using logic. Perhaps the full set of 5×5 puzzles should be mined for logical rules.

References

[McPhail, 2003] McPhail, B. (2003). The Complexity of Puzzles: NP-Completeness Results for Nurikabe and Minesweeper. Reed College. Undergraduate Thesis.

[Nikoli, 1991] Nikoli (1991). Puzzle Communication Nikoli, 33rd issue. Nikoli.

[Russell and Norvig, 2003] Russell, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition.

[Terzer, 2007] Terzer, C. (2007). Nurikabe as SAT Problem. University of Innsbruck. Bachelor Thesis.

[van Dam, 2008] van Dam, R. (2008). Slitherlink. Universiteit Leiden. Bachelor Thesis.

[van Meurs, 2008] van Meurs, T. (2008). Takegaki. Universiteit Leiden. Bachelor Thesis.

[Wikipedia, 2008] Wikipedia (2008). Page on Nurikabe. `http://en.wikipedia.org/wiki/Nurikabe`. [accessed: 2008.06.09].