ON THE DIFFICULTY OF NONOGRAMS

K. Joost Batenburg[1] *Walter A. Kosters*[2]

1 Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
Mathematical Insitute, Leiden University, Leiden, The Netherlands
Vision Lab, University of Antwerp, Wilrijk, Belgium
2 Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands

ABSTRACT

Nonograms are a popular type of logic puzzles, where a pixel grid has to be filled with black and white pixels, based on a description that indicates the lengths of the consecutive black segments for each row and column. While the Nonograms that can be found in puzzle books can typically be solved by applying a series of highly local reasoning steps regarding single rows and columns, the general Nonogram problem is NP-hard. In this article, we explore the difficulty distributions for puzzles between these two extremes. After defining several difficulty measures and subclasses, we analyze the frequencies of various types of puzzles within the set of all possible Nonograms, using both exhaustive enumeration and sampling.

1. INTRODUCTION

Logic puzzles — which can be solved by applying logic reasoning — are very popular nowadays. By far the most prominent example is the Sudoku, which has not only drawn broad attention from the public, but has also attracted significant scientific interest (Ercsey-Ravasz and Toroczkai, 2012). Another popular type of logic puzzle (involving simple arithmetic) is the Nonogram, where a grid of black and white pixels has to be filled, based on a series of descriptions (Ishida, 1993): for every row and column, the lengths of the consecutive black segments are specified in order; see Figure 1 for an example. The resulting puzzle poses a combinatorial problem that combines elements of logical reasoning with integer calculations. It can be approached using methods from combinatorial optimization, logical reasoning or both, which makes Nonograms highly suitable for educational use in Computer Science (Salcedo-Sanz *et al.*, 2007b).

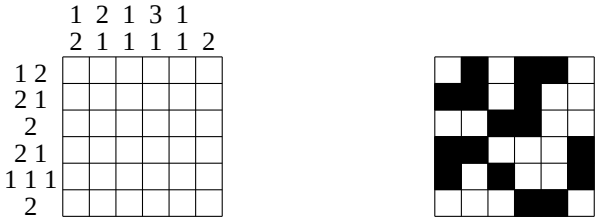

Figure 1: Relatively hard 6×6 Nonogram; left: puzzle; right: unique solution.

Several implementations of Nonogram solvers can be found on the Internet; see, e.g., (Wolter, 2012) for a list of solvers. Bosch proposed an Integer Linear Programming (ILP) formulation for the Nonogram problem in (Bosch, 2001). An evolutionary algorithm (EA) for solving Nonograms was described in (Ortiz-García *et al.*, 2008) and (Ortiz-García *et al.*, 2009), and a heuristic algorithm was proposed in (Salcedo-Sanz *et al.*, 2007a). The related problem of constructing Nonograms that are uniquely solvable is treated in (Ortiz-García *et al.*, 2007). As the Nonogram problem involves reasoning steps that link the values of the unknown cells, it can be approached using

[1]email: Joost.Batenburg@cwi.nl
[2]email: kosters@liacs.nl

models for reasoning about logical expressions, such as SAT-expressions. In (Batenburg and Kosters, 2009), a reasoning framework is proposed for solving Nonograms that uses a 2-SAT model for efficient computation of reasoning steps.

In (Ueda and Nagao, 1996), it was first proved that the general Nonogram problem is NP-hard. On the other side of the difficulty spectrum are the Nonograms that can be found in puzzle collections, which can usually be solved by hand, applying a sequence of elementary reasoning steps. This latter class of Nonograms is called the simple type in (Batenburg and Kosters, 2009). Such Nonograms can be solved without resorting to branching, yet there can still be a large variance in the number of steps required to find solutions. In (Batenburg *et al.*, 2009) a difficulty measure for this class, the so-called *simple* puzzles, is proposed and analyzed. In particular, a construction for a family of Nonograms that have asymptotically maximal difficulty, up to a constant factor, is provided. An overview of these results can be found in (Batenburg and Kosters, 2012).

Both Sudoku and Nonograms share the property that their instances can vary from very simple (i.e., easily solvable by hand) to highly complex (hard to solve by a computer program). For the Sudoku puzzles, it was recently shown that via an exact mapping of the set of puzzles into a deterministic, continuous-time dynamical system, their difficulty translates into transient chaotic behaviour of this system, allowing a Richter-like scale of puzzle difficulty (Ercsey-Ravasz and Toroczkai, 2012). For Nonograms, a difficulty model that may demonstrate similar behaviour is not yet available. At the same time, the results presented in (Batenburg *et al.*, 2009) hint at transient behaviour as well, showing abrupt transitions between rather simple solvable Nonograms and very hard Nonograms (often having many solutions) as the density of black pixels is gradually increased. Another reason for exploring difficulty measures is to obtain insight into the difficulty of Nonograms as observed by human puzzlers. Although the present paper does not deal with these issues, a successful link between the proposed concepts and the perceived difficulties would enable opportunities for automatic generation of Nonograms of varying difficulty. As an alternative route to defining a difficulty measure, we mention the use of the convergence rate of EAs that solve the Nonograms (see, e.g., (Ortiz-García *et al.*, 2009)).

In this article, we examine various difficulty measures for Nonograms, both for the simple type and for more complex puzzles. In particular, we analyze the distribution of small Nonograms over the difficulty levels. In Section 2 we define notation and concepts. Several difficulty classes and measures are introduced in Section 3. Section 4 has experimental results for Nonograms of small to medium size, and Section 5 concludes.

2. NOTATION AND CONCEPTS

We first define notation for a single line (i.e., row or column) of a Nonogram. After that, we combine these lines into rectangular puzzles. Let $\Sigma = \{\mathtt{0}, \mathtt{1}\}$, the alphabet of pixel values (more general alphabets are also allowed). We usually refer to $\mathtt{1}$ as *black* and $\mathtt{0}$ as *white*. While solving a Nonogram, the value of a pixel can also be *unknown*. Let $\Gamma = \Sigma \cup \{\mathtt{?}\} = \{\mathtt{0}, \mathtt{1}, \mathtt{?}\}$, where the symbol $\mathtt{?}$ refers to the unknown pixel value.

A *(general) description* d of length $k > 0$ is an ordered series $(d_1, d_2, \ldots, d_k)$ with $d_j = \sigma_j\{a_j, b_j\}$, where $\sigma_j \in \Sigma$ and $a_j, b_j \in \{0, 1, 2, \ldots\}$ with $a_j \leq b_j$ ($j = 1, 2, \ldots, k$). The curly braces are used here in order to stick to the conventions from regular expressions; so, in $\sigma_j\{a_j, b_j\}$ they do not refer to a set, but to an ordered pair. Any such d_j will correspond with between a_j and b_j characters σ_j, as defined below. Without loss of generality we will assume that consecutive characters σ_j differ, so $\sigma_j \neq \sigma_{j+1}$ for $j = 1, 2, \ldots, k-1$. We will sometimes write σ^* as a shortcut for $\sigma\{0, \infty\}$ (for $\sigma \in \Sigma$) and σ^+ as a shortcut for $\sigma\{1, \infty\}$, where ∞ is a suitably large number. We use σ^a as a shortcut for $\sigma\{a, a\}$ ($a \in \{0, 1, 2, \ldots\}$), and we sometimes omit parentheses and commas; also σ^0 is omitted. A finite string s over Σ *adheres* to a description d (as defined above) if $s = \sigma_1^{c_1}\sigma_2^{c_2}\ldots\sigma_k^{c_k}$, where $a_j \leq c_j \leq b_j$ for $j = 1, \ldots, k$. As an example, consider the following description:

$$d = (0\{0,\infty\}, 1\{a_1, a_1\}, 0\{1,\infty\}, 1\{a_2, a_2\}, 0\{1,\infty\}, \ldots, 1\{a_r, a_r\}, 0\{0,\infty\})$$

with $a_i > 0$ ($i = 1, 2, \ldots, r$). This is exactly what we consider to be a *Nonogram description* $a_1 a_2 \ldots a_r$ for a line (row or column), where we only mention the lengths of consecutive non-touching series of $\mathtt{1}$s. Note that it has length $2r + 1$ and can also be written as $\mathtt{0}^*\mathtt{1}^{a_1}\mathtt{0}^+\mathtt{1}^{a_2}\mathtt{0}^+\ldots\mathtt{1}^{a_r}\mathtt{0}^*$.

A string $s \in \Gamma^\ell$ ($\ell \geq 0$) can be *(fully) fixed* to a string $t \in \Sigma^\ell$ (referred to as a *fix*) if $s_j = t_j$ whenever $s_j \in \Sigma$ ($1 \leq j \leq \ell$). Loosely speaking, one should replace the ?s, or unknowns, with pixel values; we also say that we *fix* these string elements. If $s \in \Gamma^\ell$ can be fixed to a string in Σ^ℓ that adheres to a given description d, s is called *fixable*

with respect to d; in that case the Boolean function value $Fix\,(s,d)$ is defined to be `true`, and otherwise `false`. The formal operation SETTLE (s,d) constructs a (unique) string from a fixable string s and a description d in the following way: for all ? symbols in s such that all strings in Σ^ℓ that adhere to the description d have the same unique pixel value u, we fix this ? to u. In other words, all pixels that must have a certain value in order to adhere to the description, are set to that value; these are exactly the pixels that are the same in all fixes. This operation is also referred to as *settling*. As an example, for $s =$ `?1?1?0?????` (with $\ell = 11$) and Nonogram description $d = 3\ 2\ 1$ (so general description $\mathtt{0}^*\mathtt{1}^3\mathtt{0}^+\mathtt{1}^2\mathtt{0}^+\mathtt{1}^1\mathtt{0}^*$), we have SETTLE $(s,d) =$ `011100?1???`. In (Batenburg and Kosters, 2009) an efficient, polynomial-time algorithm is described for performing the SETTLE operation on a string, by using dynamic programming. The complexity of the computation of $Fix\,(s,d)$ is $O(k \cdot \ell^2)$. Note that we may assume that $k \leq \lceil \ell/2 \rceil$, otherwise there cannot be any fix.

An $m \times n$ *Nonogram puzzle description* D consists of $m > 0$ row Nonogram descriptions $r_1, r_2, \ldots, r_m$ and $n > 0$ column Nonogram descriptions $c_1, c_2, \ldots, c_n$. An *image* $P = (P_{ij}) \in \Sigma^{m \times n}$ *adheres* to the description if all lines adhere to their corresponding description. A *Nonogram* N is a pair (D, P), where D is a Nonogram puzzle description and P is an image in $\Gamma^{m \times n}$; its elements are referred to as *pixels*. We use the term *Nonogram* to refer both to the image and its description. Solving such a puzzle means finding an image $P' \in \Sigma^{m \times n}$ that adheres to D, and where every line in P is fixed to the corresponding line in P'. The image P can be viewed as a partial solution. Usually we will assume that all lines in P are still fixable with respect to their corresponding Nonogram descriptions, which means that the puzzle is *solvable*; otherwise it is *unsolvable* (in the next paragraph we mention situations where this occurs within the context of solvable puzzles). If there is precisely one solution, the Nonogram is called *uniquely solvable*; Nonograms in puzzle collections are usually of this class.

In most puzzles P fully consists of `?`s. However, analogous to Sudoku, it is possible to have extra information in the form of clues. A *clue* is a pixel that is fixed to either `0` or `1`, consistent with the final solution. The more clues are given, the easier the puzzle will be. But it is more subtle than this: apart from a clue being incorrect, meaning that the Nonogram has no solution at all, it can also be that adding one or more clues turns a non-uniquely solvable puzzle into a uniquely solvable one, or more generally decreases the number of solutions. And clues often influence the difficulty. In a similar vein, the shape of the puzzle grid can also be other than rectangular. In fact, one can imagine directed graphs where the nodes represent the pixels, and where descriptions infer restrictions on the pixels in paths in the graph. It is also possible to provide clues during the puzzle solving process: so-called *online clues*; in this case a person that got stuck during solving, can be helped through hints.

3. THE DIFFICULTY OF NONOGRAMS

We will distinguish between simple and non-simple Nonograms, the simple ones only using knowledge regarding single lines. We recall some necessary material from (Batenburg *et al.*, 2009) in the next subsection.

3.1 Simple Nonograms

Most Nonograms that appear in puzzle collections can be solved by applying a series of local reasoning steps, each involving just a single row or column. Recall that the SETTLE operation as defined in Section 2 fills in all unknown elements of a row or column that are uniquely defined by the combination of the line description and the set of known elements on that line. As the SETTLE operation considers one line at a time, it can be performed in parallel for all rows, or all columns respectively. The operation H-SWEEP (N) applies the SETTLE operation to all rows of the Nonogram N: a horizontal sweep; and the operation V-SWEEP (N) applies the SETTLE operation to all columns of the Nonogram N: a vertical sweep. The Nonogram that is returned by these operations has fewer unknowns than the input Nonogram, unless no entries could be deduced by using only information from a single direction (horizontal or vertical). Note that H-SWEEP (H-SWEEP (N)) = H-SWEEP (N) and V-SWEEP (V-SWEEP (N)) = V-SWEEP (N).

A Nonogram N is called *simple* if it can be (uniquely) solved by applying an alternating sequence of H-SWEEP and V-SWEEP operations. Equivalently, one can say that a *simple* Nonogram can be solved by applying a sequence of SETTLE operations, each involving just a single line.

The total number of sweep-operations that must be performed to solve a simple $m \times n$ Nonogram can be used as a difficulty measure for simple Nonograms, as proposed in (Batenburg *et al.*, 2009), using the algorithm SIMPLE-

SOLVER: it starts with a H-SWEEP operation and then alternates between V-SWEEP and H-SWEEP operations, until no further unknown entries are fixed by both V-SWEEP and H-SWEEP. The number of sweep-operations that is needed to completely solve the Nonogram is then called the *difficulty*; its value is between 1 and $mn + 1$. By definition, the SIMPLESOLVER algorithm always terminates if the input Nonogram N is simple. The algorithm can also be used for non-simple Nonograms, in which case some, but not all, unknown entries may be filled.

Note that SIMPLESOLVER could start with a V-SWEEP operation instead of a H-SWEEP operation, which results in a difficulty that differs from the one defined above by at most 1. One could also take the average of these two numbers, but we will use the first definition. An alternative approach would be to compute the minimum number of SETTLE operations needed to solve a given Nonogram. However, this would be very hard to compute due to extensive backtracking, especially when considering all puzzles of a given size.

In (Batenburg *et al.*, 2009) a construction is given for an $m \times n$ Nonogram of the simple type, that has a very high difficulty. Indeed, it is shown that the proposed Nonogram (with $m = 8r + 2$ for some integer $r \geq 1$ and $n \geq 14$ even) has difficulty $(m + 2)(2n - 15)/4 + 10 \approx mn/2$. Solving the Nonogram requires the consecutive traversal of the so-called 5-strips, ending in a single 2-strip. These 5-strips are largely solved at a speed of only one pixel per sweep, yielding a high difficulty. In the first steps of the solution process the 3-strips including the so-called split rows are fixed, serving as a kind of separator between the different 5-strips and the final 2-strip. Figure 2 illustrates the construction for $m = n = 18$. Full details can be found in (Batenburg *et al.*, 2009).

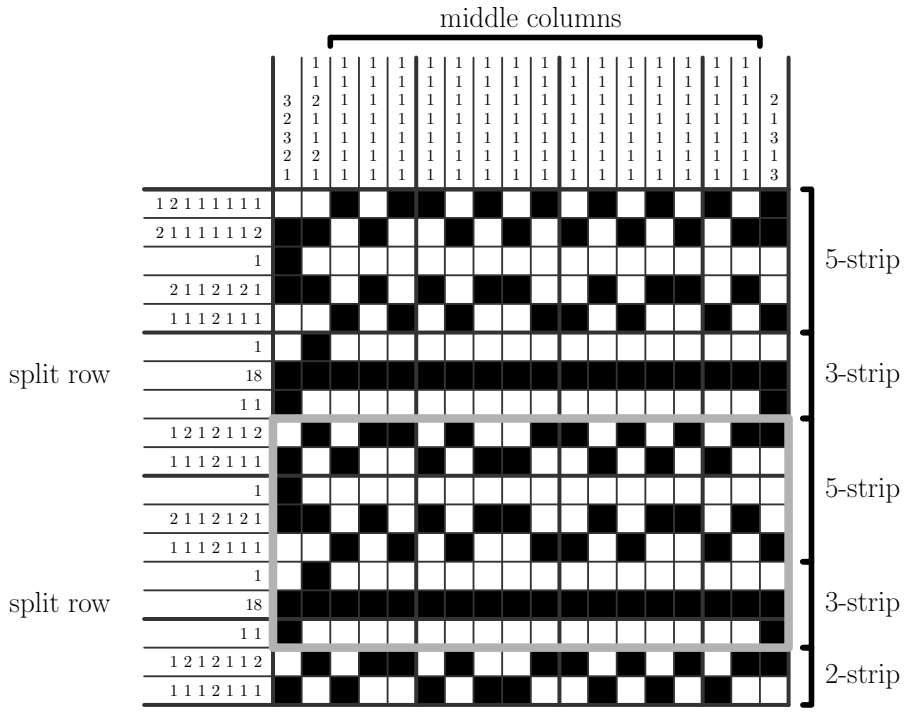

Figure 2: Overview of the construction of an 18×18 Nonogram with difficulty 115. The construction can be extended in the vertical direction by inserting consecutive copies of the marked block (rows 9–16). Extension in the horizontal direction is relatively straightforward by adding more "middle columns".

3.2 Non-simple Nonograms

Having defined a difficulty measure for simple Nonograms, we now proceed to more complicated ones. If a Nonogram is non-simple, SIMPLESOLVER will leave a partially filled solution. We now introduce the (p, q)-SOLVER for integer p and q, with $1 \leq p \leq m$ and $1 \leq q \leq n$. In every step of the solving process we take p rows and q columns. For all $p \cdot q$ intersections we consider every possibility of the unknown pixels; any such combination of pixels is referred to as a configuration. For all $p + q$ lines involved, we apply SETTLE, and keep track of those configurations that can be extended in every line; we then can fix the unknown intersection pixels that are the same in all of them. If at least one pixel can be fixed, we call this a *successful* (p, q)*-intersection*. After

that we can freely apply SIMPLESOLVER. The order in which these operations are done, is of no importance for the final result: this will be the (partially) solved Nonogram where no more progress can be made.

If either p or q is equal to 1, (p, q)-SOLVER is nothing more than SIMPLESOLVER. We observe that (p, q)-SOLVER "contains" all (p', q')-SOLVERs with $p' \leq p$ and $q' \leq q$. Therefore, the simplest (p, q)-SOLVER that is stronger than SIMPLESOLVER is the $(2, 2)$-SOLVER, also referred to as the FOURSOLVER: it considers four pixels at a time, that are in rectangular position and unknown so far. A Nonogram that can be solved by (p, q)-SOLVER, but not by any (p', q')-SOLVER with $p' \leq p$, $q' \leq q$ and $(p', q') \neq (p, q)$, is called (p, q)-*hard*. This leads to a partial ordering of the solvers, where SIMPLESOLVER is the least element, having FOURSOLVER immediately above it. Allowing p or q to be 0, yields only horizontal and vertical sweeps: the $(1, 0)$-SOLVER, or equivalently any $(p, 0)$-SOLVER with $p \geq 1$, would in fact just use single horizontal sweeps.

The time complexity of the (p, q)-SOLVER can be large, but the operation in total takes polynomial time (if p and q are fixed). If all the pixels of a configuration are unknown yet, there are $2^{p \cdot q}$ of these, but in general there may be less. However, there are $\binom{m}{p} \times \binom{n}{q}$ sets of lines to consider, giving an enormous number of possibilities. If $p = m$ and $q = n$, this method in fact just tries all possibilities of the full Nonogram, and it will solve the Nonogram in one successful (m, n)-intersection — provided it is uniquely solvable. If SIMPLESOLVER could not fix a single pixel, this just tries all $2^{m \cdot n}$ possibilities, showing that the method in this case could be considered infeasible.

For (p, q)-hard puzzles it is possible to distinguish a difficulty level analogous to the difficulty measure for simple Nonograms. It seems natural to define the (p, q)-difficulty of a Nonogram as the minimum number of successful (p, q)-intersections needed to solve the puzzle. Notice that such a difficulty computation requires quite a lot of backtracking, but a definition analogous to the one for the difficulty of simple puzzles seems impossible. Again, if the solver cannot fully solve the puzzle, the difficulty is defined as ∞. Clearly, all Nonograms that are not uniquely solvable have (p, q)-difficulty ∞. According to this definition, a simple Nonogram has (p, q)-difficulty 0. And any uniquely solvable non-simple Nonogram has (m, n)-difficulty 1.

We now focus on the situation with $p = q = 2$. First we mention that, when considering two unknowns in a line for which SETTLE cannot fix any more pixels, the following seven possibilities can arise:

- All four combinations `0–0`, `0–1`, `1–0` and `1–1` can occur; e.g., consider the two outmost pixels in a string `?????` with description 1 1.
- Only the combination `1–0` is forbidden; e.g., consider the two leftmost pixels in a string `??1??` with description 3.
- Only the combination `0–1` is forbidden; e.g., consider the two rightmost pixels in a string `??1??` with description 3.
- Only the combination `0–0` is forbidden; e.g., consider the two outmost pixels in a string `????` with description 1 1.
- Only the combination `1–1` is forbidden; e.g., consider the two outmost pixels in a string `????` with description 2.
- Only the combinations `0–0` and `1–1` are forbidden, which means that the two pixels must be different; e.g., consider the two pixels in a string `??` with description 1.
- Only the combinations `1–0` and `0–1` are forbidden, which means that the two pixels must be the same; e.g., consider the two leftmost pixels in a string `??0??` with description 2.

Indeed, forbidding any other group of combinations would have allowed SETTLE to fix a pixel. When we consider $p = 2$ rows and $q = 2$ columns, we only need to examine those situations where these lines intersect in four `?`s. We can encounter all the seven cases mentioned above along the four lines. Clearly, if for one or more of the lines all combinations are allowed, we cannot draw any conclusion. However, if for all four lines one or more combinations are forbidden, we might conclude that one or more of the four pixels involved can be fixed. Indeed, it is easy to construct a lookup table for the 6^4 = 1,296 possible situations, exactly 512 of them leading to one, two, three or even four pixels fixed; in any of these situations at least 12 from the $2^{2 \cdot 2}$ = 16 configurations are forbidden. Note that in Nonograms that have one or more solutions 32 of the situations cannot occur in practice; for instance, if for three of the four lines involved the two pixels at the line must be the same, but the fourth

line requires them to be different, no solution would be possible. Any examination of two rows and two columns requires 16 SETTLE operations and a single lookup, allowing for polynomial time evaluation.

The concept of analyzing the relations between pairs of pixels on a single line is in fact a special case of the approach described in (Batenburg and Kosters, 2009), which builds a set of 2-SAT clauses that express the relations that must hold between pixel values. There, instead of considering two specific pixels (on the intersection points of the two horizontal and two vertical lines), all such relations between two pixels within each horizontal and each vertical line are collected and assembled into a larger 2-SAT problem. The resulting 2-SAT problem involves pixels from all rows and columns. Although that model is very suitable for *solving* Nonograms, the present concept of choosing specific rows and columns (in this case: 2 rows and 2 columns) is more suitable for defining a difficulty measure. Each pair of pixel values within one of the four lines that is not allowed forms a 2-SAT clause. The invariant pixels in the solution set of the 2-SAT problem that results from combining these clauses can be fixed, whereas the pixels that can be either 0 or 1 cannot be inferred based on the $(2, 2)$-intersection. Determining the possible subsets of pixel values that can occur for the intersection points is equivalent to restricting the collection of 2-SAT clauses in the algorithm from (Batenburg and Kosters, 2009) to the intersection points of two rows and two columns.

As the proposed difficulty measure can be computed effectively for small Nonograms (and sometimes for larger Nonograms, as long as the difficulty does not become too high), it is well suited for creating a series of benchmark Nonograms that can be used for algorithm evaluation. In such an evaluation, the algorithm from (Batenburg and Kosters, 2009) would prove to be at least $(2, 2)$-*capable*, meaning that all $(2, 2)$-hard instances can be solved. This concept can be used to rank algorithms by the highest difficulty for which all benchmark instances can be solved successfully.

4. NONOGRAMS OF SMALL TO MEDIUM SIZE

We analyze small Nonograms using exhaustive enumeration, and examine medium size Nonograms by sampling.

4.1 Small Nonograms

In this subsection we first consider the set of all $2^{5\cdot 5} = 2^{25}$ = 33,554,432 5×5 black and white images. For every image we compute its line descriptions, and we examine the difficulty of the corresponding Nonogram.

Data are presented in Figure 4. Note that there are symmetries involved, using the dihedral group D_4: most puzzles occur in groups of 8. The first column contains the difficulty level, while the second column has the number of simple puzzles of this difficulty. The hardest puzzles of the simple type have difficulty 17; there are 4 of them. Figure 3, first panel, has a representative of its symmetry equivalence class (the other 4 in its symmetry class have difficulty 16). The third column has the number of puzzles that can be solved by FOURSOLVER, indexed by the number of sweeps that were used by SIMPLESOLVER until no further progress was made (including the last two that did not fix any more pixels). An example is shown in Figure 3, second panel; for this puzzle 14 sweeps are used, before a single successful $(2, 2)$-intersection is needed. Out of the 317,944 puzzles of this class, 224,751 could be solved by involving one successful $(2, 2)$-intersection; these puzzles have $(2, 2)$-difficulty 1. Figure 3, third panel, shows a puzzle with higher $(2, 2)$-difficulty. The fourth column shows the number of puzzles for which FOURSOLVER could make some progress, but was not able to fully solve them. The fifth (and last) column has the number of puzzles for which FOURSOLVER could not fix any pixel at all. In particular, there are 124,189 puzzles where neither any line SETTLE could fix a pixel nor any successful $(2, 2)$-intersection could be found.

From the puzzles where FOURSOLVER made some progress but was not able to fully solve them, only 8,400 are in fact uniquely solvable (an example is provided in Figure 3, fourth panel); 1,192,994 puzzles can be solved by SIMPLESOLVER after adding one well-chosen clue (see the next paragraph). And for those puzzles where FOURSOLVER could not make any progress at all only 6,720 are in fact uniquely solvable (an example is given in Figure 3, fifth panel); 6,124,630 puzzles can be solved by SIMPLESOLVER after adding one well-chosen clue. We conclude that out of the 333,064 uniquely solvable non-simple Nonograms, 95.46% is (2,2)-hard; the total number of uniquely solvable 5×5 Nonograms is 25,309,575, with only 0.06% being not simple or (2,2)-hard..

Clues that make a puzzle uniquely solvable, or make them easier, can be generated by just trying all pixels that are unknown so far. Fixing such a pixel to the value it should have in the solution, can either allow a particular

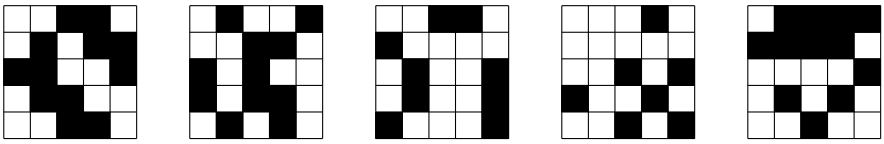

Figure 3: Hardest simple 5×5 Nonogram (first panel), $(2,2)$-hard 5×5 Nonogram of $(2,2)$-difficulty 1 (second panel), $(2,2)$-hard 5×5 Nonogram of higher $(2,2)$-difficulty (third panel), uniquely solvable Nonogram where FOURSOLVER could make some progress (fourth panel; note the 2-fold symmetry), and uniquely solvable Nonogram where FOURSOLVER could not make any progress (fifth panel). The corresponding line descriptions can be easily derived.

difficulty	simple Nonograms	(2,2)-hard Nonograms	some progress by FOURSOLVER	no progress by FOURSOLVER
1	7,776	0	0	0
2	3,409,924	5,752	72,052	124,189
3	9,367,027	56,158	299,778	576,293
4	6,514,096	156,236	701,152	1,846,395
5	3,232,762	50,796	233,590	1,621,838
6	1,350,200	31,844	145,070	1,277,062
7	633,400	9,654	42,232	590,436
8	267,632	5,396	18,660	378,354
9	115,178	1,208	5,004	153,364
10	47,584	580	2,644	93,320
11	19,972	180	888	39,904
12	7,132	120	404	24,512
13	2,864	16	0	6,892
14	728	4	0	4,664
15	216	0	0	744
16	16	0	0	496
17	4	0	0	24
18	0	0	0	16
19	0	0	0	0
total	24,976,511	317,944	1,521,474	6,738,503
	74.44%	0.94%	4.53%	20.08%

Figure 4: Statistics for the 5×5 images and the corresponding Nonograms.

solver to solve the puzzle, make it easier or have no effect (but perhaps one arrives at more fixed pixels when solving); this also depends on the point in time that the clue is provided. It seems hard to devise a method to find such clues without trying, though simple heuristics could be conceived. The clues mentioned in the previous paragraph were indeed found by just trying all possibilities.

For the set of all $2^{6 \cdot 6} = 2^{36} = 68{,}719{,}476{,}736$ 6×6 black and white images, results are presented in Figure 5. From the puzzles where FOURSOLVER made some progress but was not able to fully solve them, 57,158,639 are in fact uniquely solvable. And for those puzzles where FOURSOLVER could not make any progress at all 12,106,334 are in fact uniquely solvable. We conclude that out of the 1,119,951,439 uniquely solvable non-simple Nonograms, 93.82% is (2,2)-hard; the total number of uniquely solvable 6×6 Nonograms is 49,745,060,370, with only 0.14% being not simple or (2,2)-hard.

4.2 Medium Size Nonograms

The numbers thus obtained suggest that most uniquely solvable puzzles are simple, while there are also quite many (2,2)-hard ones. There are just relatively few remaining uniquely solvable puzzles. In order to assess this issue, we now explore larger puzzles. However, due the enormous number of images, we have to use sampling.

In order to better understand larger puzzles, we first perform sampling for small images. Figure 6 shows results

difficulty	simple Nonograms	(2,2)-hard Nonograms	some progress by FOURSOLVER	no progress by FOURSOLVER
1	531,441	0	0	0
2	1,892,287,503	168,790,528	806,262,697	110,274,370
3	12,508,732,323	97,411,401	381,662,356	237,190,334
4	14,306,915,587	242,390,921	988,352,708	1,927,512,716
5	9,524,244,629	201,850,110	892,360,826	3,271,038,528
6	5,110,680,772	155,867,778	694,418,286	3,243,954,226
7	2,645,426,982	84,706,478	382,676,366	2,125,778,718
8	1,295,645,826	51,037,482	221,491,162	1,550,293,616
9	666,279,114	24,030,088	102,398,912	844,903,646
10	336,135,394	13,572,368	56,564,246	560,868,894
11	173,613,082	5,915,660	23,220,212	277,760,640
12	85,805,862	2,979,724	11,498,196	169,399,812
13	42,890,084	1,206,236	4,131,016	77,735,516
14	19,823,016	583,232	1,930,888	44,335,244
15	9,241,252	211,856	601,564	18,482,368
16	3,992,532	86,032	232,984	9,667,988
17	1,750,180	29,776	71,404	3,742,560
18	683,456	11,536	25,600	1,796,320
19	275,508	3,612	6,604	628,632
20	100,160	1,156	1,716	270,200
21	36,692	352	432	85,932
22	12,204	88	128	36,172
23	3,760	36	48	11,136
24	1,176	12	12	4,248
25	332	4	12	820
26	64	0	0	256
27	0	0	0	56
28	0	0	0	16
29	0	0	0	0
total	48,625,108,931	1,050,686,466	4,567,908,375	14,475,772,964
	70.76%	1.53%	6.65%	21.07%

Figure 5: Statistics for the 6×6 images and the corresponding Nonograms.

for 6×6 images, with percentage of black pixels ranging from 0 to 100. For every integer percentage 1,000 random images were generated, and the corresponding puzzles were examined. In the figure, the numbers of simple and — cumulative — (2,2)-hard Nonograms are plotted, as is the number of puzzles where some (often just a little) progress could be made by FOURSOLVER. When interpreting such a graph, one should keep in mind that the actual number of possible Nonograms having a given percentage of black pixels varies enormously as a function of that percentage, following a binomial distribution centered at 50% (see the right panel from Figure 6). The relative number of Nonograms for which less than 40% or more than 60% is black is negligible, even though the absolute number of such puzzles can still be huge.

Due to the small image size for 6×6 puzzles, specific behaviour can be observed in the graph in Figure 6. If there are no black pixels, or just one, all puzzles are of the simple type. For the case of two black pixels (5.5% black), two cases can arise. Either the black pixels are in the same line horizontally or vertically (with probability approximately 28%), in which case the puzzle is always simple, or they are in different rows and columns. In the latter case, we see the occurrence of a so-called *switching component*, a subset of the pixels where interchanging the zeros and ones results in a different Nonogram with an identical description (see Figure 7). In this case, there is no unique solution and FOURSOLVER can also not make any progress. As the percentage of black pixels increases towards 50%, the number of possible patterns for each line increases as well. For a low percentage of black pixels (around 20%), the SIMPLESOLVER algorithm is only rarely capable of solving the puzzle, while the increased power of using the information from 2 rows and 2 columns simultaneously provides substantially more information about the solution. As the percentage of black pixels increases further, SIMPLESOLVER becomes

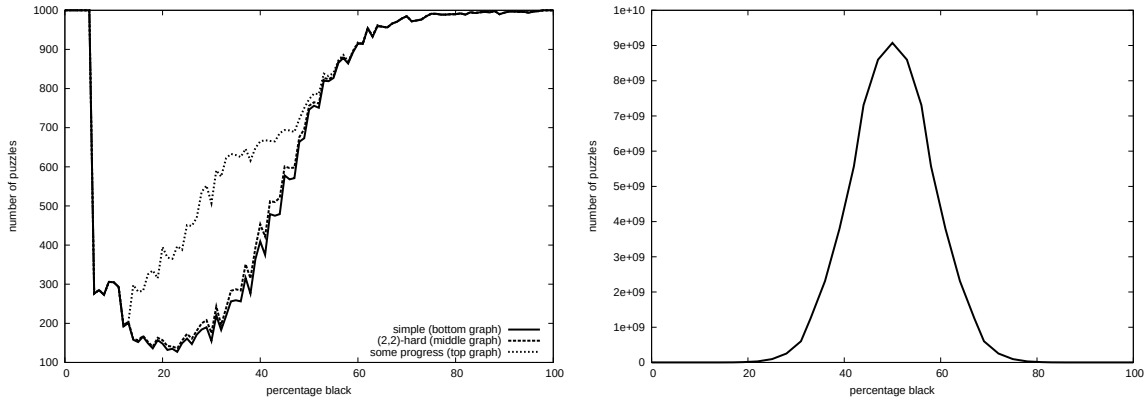

Figure 6: Left: results for random 6×6 images/puzzles with varying percentage of black pixels; range 0–100; right: binomial distribution.

increasingly more powerful, up to the point where each line has a large fraction of black pixels and solving the Nonogram is either straightforward, or there is no unique solution. In these cases, FOURSOLVER can hardly make a difference. Solvability seems to have a complicated, but apparent, correlation with the percentage of black pixels of the images.

Figure 7: Switching component in its simplest form: 2×2 Nonogram with two solutions.

The behaviour for larger instances (10×10 and 30×30) is shown in Figure 8. In both situations, it can be observed that up to a certain percentage of black pixels (around 30% for 10×10, around 50% for 30×30), less than 2% of the Nonograms are either simple or $(2, 2)$-hard. The specific percentage at which a significant fraction of the Nonograms starts being simple tends to rise as the size of the puzzle increases. Going up from this percentage, there is a region where a substantial fraction of $(2, 2)$-hard puzzles can be found. We recall that due to the shape of the binomial distribution, this region contains a large part of the complete set of possible Nonograms. Further increasing the percentage of black pixels then enters a region where the puzzles are typically either simple or more difficult than $(2, 2)$-hard. With evidence being based on simulations, we speculate that the non-simple puzzles in this region do not have a unique solution, and can therefore also not be solved by FOURSOLVER.

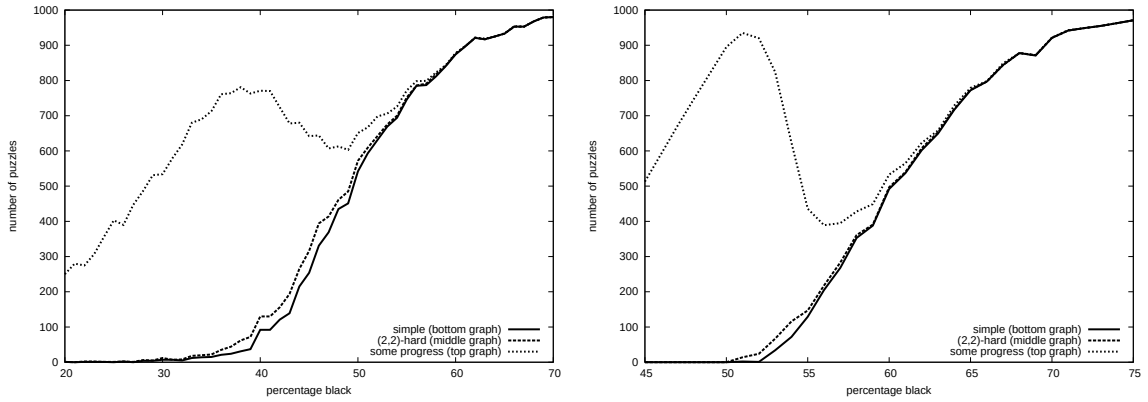

Figure 8: Results for random images/puzzles with varying percentage of black pixels; left: 10×10, range 20–70; right: 30×30, range 45–75.

Another interesting pattern in Figure 8 is revealed by the curve which tracks the number of Nonograms for which some progress could be made by the FOURSOLVER method. After increasing initially the curve shows a distinctive peak, followed by a region in which it decreases. Indeed, when the percentage of black pixels increases, the FOURSOLVER method can recover the value of more pixels, as there is less freedom to shift the segments on

each line. The ability to deduce the value of these pixels for the SIMPLESOLVER method lags behind, as it is less powerful. From some point on, the SIMPLESOLVER is also capable of making progress, after which the added benefit of using the FOURSOLVER drops, explaining the peak.

In the Nonogram solving tournament held during TAAI2011 (Sun *et al.*, 2012) several carefully designed 25×25 puzzles were provided by the competitors to test the strengths and weaknesses of the contributions by the other contestants. Without exception these Nonograms are harder than $(2, 2)$-hard, pushing the programs to their limits.

5. CONCLUSIONS AND FURTHER RESEARCH

In this paper, we have proposed, analyzed and discussed difficulty measures for Nonograms. The difficulty of Nonograms follows a complex behaviour, ranging from simple Nonograms that can be found in puzzle books to highly difficult ones that can only be solved by an exhaustive search, handling the underlying NP-hard combinatorial problem. For simple Nonograms a difficulty measure was defined in (Batenburg *et al.*, 2009). We proposed a new general difficulty measure, which ranks the set of all uniquely solvable (in general non-simple) Nonograms by the number of rows and columns that must be involved in a simultaneous logical step to solve the complete puzzle. The set of simple Nonograms corresponds to the "easiest" instances according to this difficulty measure, using only logical steps within a single line. For Nonograms of sizes up to 30×30, we performed computational experiments to explore the space of all Nonograms, focusing on those that are either simple or are $(2, 2)$-hard. In that case we only use information regarding two rows and two columns simultaneously. The observed patterns match well with intuitive explanations of Nonogram characteristics, but also provide more quantitative insights into the transition of the difficulty of Nonograms from very simple ones to highly difficult instances.

Our future work will focus on a further stratification of the set of non-simple Nonograms, exploring the difficulty levels that go beyond the $(2, 2)$-hard Nonograms. The distribution of Nonograms over each of the possible difficulty levels is still to be revealed, e.g., for what values of p and q do (p, q)-hard puzzles exist? For those Nonograms that are not uniquely solvable, we want to classify the basic structures of non-uniqueness in a similar hierarchical ordering as the proposed difficulty measure.

Acknowledgement

KJB was financially supported by NWO (Netherlands Organisation for Scientific Research, research programme 639.072.005).

6. REFERENCES

Batenburg, K. J., Henstra, S., Kosters, W. A., and Palenstijn, W. J. (2009). Constructing simple Nonograms of varying difficulty. *Pure Mathematics and Applications*, Vol. 20, pp. 1–15.

Batenburg, K. J. and Kosters, W. A. (2009). Solving Nonograms by combining relaxations. *Pattern Recognition*, Vol. 42, pp. 1672–1683.

Batenburg, K. J. and Kosters, W. A. (2012). Nonograms. *Newsletter of the Dutch Organization for Theoretical Computer Science (NVTI)*, Vol. 16, pp. 49–62.

Bosch, R. A. (2001). Painting by numbers. *OPTIMA*, Vol. 65, pp. 16–17.

Ercsey-Ravasz, M. and Toroczkai, Z. (2012). The chaos within Sudoku. *Scientific Reports*, Vol. 2: 725, pp. 1–8.

Ishida, N. (1993). *Sunday Telegraph Book of Nonograms.* Pan Publishers.

Ortiz-García, E. G., Salcedo-Sanz, S., Leiva-Murillo, J. M., Pérez-Bellido, Á. M., and Portilla-Figueras, J. A. (2007). Automated generation and visualization of picture-logic puzzles. *Computers and Graphics*, Vol. 31, pp. 750–760.

Ortiz-García, E. G., Salcedo-Sanz, S., Pérez-Bellido, Á. M., Portilla-Figueras, J. A., and Yao, X. (2008). Solving very difficult Japanese puzzles with a hybrid evolutionary-logic algorithm. *Proceedings of the 7th Simulated Evolution and Learning Conference, LNCS 5361*, pp. 360–369.

Ortiz-García, E. G., Salcedo-Sanz, S., Pérez-Bellido, Á. M., Portilla-Figueras, J. A., and Yao, X. (2009). Improving the performance of evolutionary algorithms in grid-based puzzles resolution. *Evolutionary Intelligence*, Vol. 2, pp. 169–181.

Salcedo-Sanz, S., Ortiz-García, E. G., Pérez-Bellido, Á. M., Portilla-Figueras, J. A., and Yao, X. (2007a). Solving Japanese puzzles with heuristics. *Proceedings IEEE Symposium on Computational Intelligence and Games (CIG)*, pp. 224–231.

Salcedo-Sanz, S., Portilla-Figueras, J. A., Ortiz-García, E. G., Pérez-Bellido, Á. M., and Yao, X. (2007b). Teaching advanced features of evolutionary algorithms using Japanese puzzles. *IEEE Transactions on Education*, Vol. 50, pp. 151–156.

Sun, D.-J., Wu, K.-C., Wu, I.-C., Yen, S.-J., and Kao, K.-Y. (2012). Nonogram tournaments in TAAI 2011. *ICGA Journal*, Vol. 35(2).

Ueda, N. and Nagao, T. (1996). NP-completeness results for Nonogram via parsimonious reductions. Technical Report TR96–008, Department of Computer Science, Tokyo Institute of Technology.

Wolter, J. (2012). Website Survey of Paint-by-number puzzle solvers. http://webpbn.com/survey/ [accessed 14.12.2012].