Constructing Simple Nonograms of Varying Difficulty

K. Joost Batenburg[*,1], Sjoerd Henstra[2], Walter A. Kosters[2], and Willem Jan Palenstijn[1]

[1]Vision Lab, Department of Physics, University of Antwerp, Belgium

[2]Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands

Abstract

Japanese puzzles, also known as *Nonograms*, are image reconstruction problems that can be solved by logic reasoning. Nonograms can have widely varying difficulty levels. Although the general Nonogram problem is NP-hard, the instances that occur in puzzle collections can usually be solved by hand.

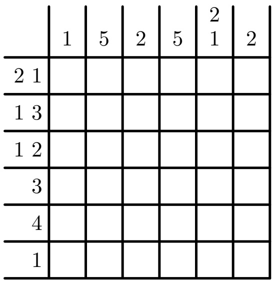
This paper focuses on a subclass of Nonograms that can be solved by a sequence of local reasoning steps. A *difficulty measure* is defined for this class, which corresponds to the number of steps required to reconstruct the image. In the first part of this paper, we investigate the difficulty distribution among this class, analyze the structure of Nonograms that have lowest difficulty, and give a construction for the asymptotically most difficult problems. The second part of the paper deals with the task of constructing Nonograms, based on a given gray level image. We propose an algorithm that generates a set of Nonograms of *varying difficulty* that all resemble the gray level input image. The effectiveness of the algorithm is demonstrated for several input images.

1 Introduction

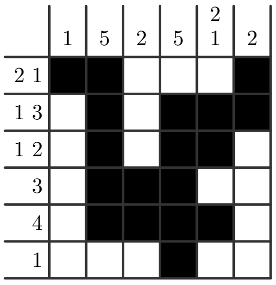
A *Nonogram*, also known as a *Japanese puzzle* in some countries, is a type of logic puzzle which can be considered as an image reconstruction problem. The goal is to find an image on a rectangular pixel grid that adheres to certain row and column constraints. Usually, the image is black-and-white, although Nonograms with more than two gray values exist as well. In addition to elementary logic, solving Nonograms requires some elementary integer calculations.

[*]Corresponding author, `joost.batenburg@ua.ac.be`

The combination of a logic problem with integer calculations results in a combinatorial problem that can be approached using methods from combinatorial optimization, logical reasoning or both, which makes Nonograms highly suitable for educational use in Computer Science [5].



(a) 6 × 6 Nonogram



(b) Solved Nonogram

Figure 1: A small Nonogram and its unique solution.

Fig. 1(a) shows an example of a Nonogram. Its solution is shown in Fig. 1(b). The description for each row and column indicates the order and length of consecutive unconnected black segments along those lines. For example, the description "2 1" in the first row indicates that from left to right, the row contains a black segment of length 2 followed by a single black square. The black segments are separated by one or more white pixels and there may be additional white pixels before the first segment, and after the last segment.

Several implementations of Nonogram solvers can be found on the Internet; see [6] for a list of solvers. In [1], an evolutionary algorithm is described for solving Nonograms. A heuristic algorithm for solving Nonograms is proposed in [4]. The related problem of *constructing* Nonograms that are uniquely solvable is discussed in [3]. In [2], a reasoning framework is proposed for solving Nonograms that uses a 2-SAT model for efficient computation of reasoning steps.

In [7], it was first proved that the general Nonogram problem is NP-hard. This also follows from the fact that Nonograms can be considered as a generalization of the reconstruction problem for hv-convex sets in discrete tomography, which is NP-hard [8]. 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 simple reasoning steps. In this paper, we focus on this latter class of Nonograms, referred to as the *simple type* in [2]. 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. We define a difficulty measure for this class and analyze several properties. In particular, we provide a construction for a family of Nonograms that have asymptotically maximal difficulty, up to a constant factor. As an application, we propose an algorithm for constructing Nonograms from the *simple* class of varying difficulty, based on given gray level images.

This paper is structured as follows. In Section 2, notation is introduced to describe the objects of this paper and their properties. Both the *simple* class and the difficulty measure are defined. In Section 3, further motivation is provided for studying this particular difficulty measure, and its distribution is analyzed for small Nonograms. Section 4 considers the question what the maximum difficulty can be, as a function of Nonogram size. First, we derive properties of single lines in Nonograms, illustrating the range between simple Nonograms of lowest difficulty and Nonograms that are not even of simple type. Next, a construction is given that obtains asymptotically maximal difficulty for Nonograms of arbitrarily large size. The remainder of the paper deals with an application of this difficulty concept: constructing Nonograms of varying difficulty that resemble a gray level input image. An algorithm is proposed for this task, followed by a series of computational experiments. Section 6 concludes this paper.

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 into rectangular puzzles. Let $\Sigma = \{0, 1\}$, the alphabet of pixel values. We also refer to 1 as *black* and 0 as *white*. While solving a Nonogram, the value of a pixel can also be *unknown*. Let $\Gamma = \{0, 1, ?\}$, where the symbol '?' refers to the unknown pixel value.

A *description* d of length $k \geq 0$ is a (possibly empty) ordered series of positive integers $d_1 d_2 \dots d_k$. A finite string s over Σ *adheres* to such a description d if s satisfies the regular expression $0^* 1^{d_1} 0^+ 1^{d_2} 0^+ \dots 1^{d_k} 0^*$. A string $s \in \Gamma^\ell$ ($\ell \geq 0$) can be *fixed* to a string $t \in \Sigma^\ell$ if $s_j = t_j$ whenever $s_j \in \Sigma$ ($1 \leq j \leq \ell$). A description d is called *ℓ-consistent* if $\sum_{i=1}^{k} d_i + n - 1 \leq \ell$. Given a string $s \in \Gamma^\ell$ and a description d, we define:

$$\begin{aligned} S(s) &= \{ t \in \Sigma^\ell \mid s \text{ can be fixed to } t \}, \\ A_\ell(d) &= \{ t \in \Sigma^\ell \mid t \text{ adheres to } d \}, \\ F(s, d) &= S(s) \cap A_\ell(d). \end{aligned}$$

The operation *Settle* (s, d) constructs a string t from a string s over Γ and an ℓ-consistent description d by replacing all '?' symbols in s for which all strings in $F(s, d)$ have a unique value in Σ by this value. In other words, all pixels that must have a certain value in order to adhere to the description, are set to that value. In [2], an efficient, polynomial-time algorithm is described for performing the *Settle* operation on a string, by using dynamic programming.

An $m \times n$ *Nonogram description* D consists of $m > 0$ row descriptions $r_1, r_2, \dots, r_m$ and $n > 0$ column descriptions $c_1, c_2, \dots, c_n$. An *image* $P = (P_{ij}) \in \Gamma^{m \times n}$ *adheres* to the description if P only contains values from Σ and all rows and columns adhere to their corresponding description. A *Nonogram* N consists of a pair (D, P), where D is a Nonogram description and P is a (partially filled) image.

A Nonogram description is called *simple* if it can be reconstructed by applying a sequence of *Settle* operations, each time using only information from a single row or column. In other words, it is never necessary to consider information from several rows and columns simultaneously. Nearly all Nonograms that appear in puzzle collections satisfy this property. From this point on, we focus exclusively on the class of *simple* Nonograms. Note that for Nonograms of the simple type, there is a bijective map between the set of images and their descriptions. Therefore, we sometimes use the term *Nonogram* to refer to either the image, or its description. Even though the order of applying the *Settle* operations does not affect whether or not a solution can be found, the required number of operations depends heavily on the order in which rows and columns are selected.

We define the following operations:

- The operation *h-sweep* (N) applies the *Settle* operation to all rows of the Nonogram N: a horizontal sweep.
- The operation *v-sweep* (N) applies the *Settle* operation to all columns of the Nonogram N: a vertical sweep.

Both operations return the "updated" Nonogram, usually having fewer unknowns.

Now the difficulty of a Nonogram of the simple type is determined by starting with an image N for which all pixel values are unknown, and running the algorithm in Fig. 2.

```
Difficulty (N) :
    diff ← 0;
    while N is not solved do
        if diff is even then N ← h-sweep (N);
        else N ← v-sweep (N);  fi
        diff ← diff + 1;
    od
    return diff;
```

Figure 2: Algorithm that solves a *simple* Nonogram and determines its difficulty.

The algorithm starts with a horizontal sweep, intertwines horizontal and vertical sweeps, and counts the total number of sweeps until the Nonogram is solved. It is clear that any $m \times n$ Nonogram of simple type has difficulty at most equal to $mn + 1$, since every sweep (except perhaps the first one) must at least fix one unknown pixel. In the next section, we motivate the choice for this particular difficulty measure.

3 A few remarks on difficulty

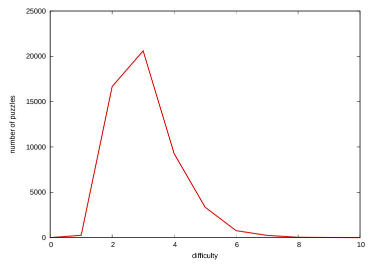
We remark that our definition of "difficulty" is rather subjective. Quantifying the amount of work required to solve a particular Nonogram is not straightforward, as it depends on the particular solution strategy employed. In Section 5, we will consider the task of constructing Nonograms of *varying difficulty* that resemble a gray level input image. As these Nonograms are intended to be solved by human puzzlers, it is important that the difficulty measure corresponds to the amount of work required by a puzzler to solve the Nonogram. We observed that while solving Nonograms, people rarely combine information from several rows and columns simultaneously, which motivates studying the simple class. A major advantage of the proposed measure is that it does not depend on the *order* in which individual rows and columns are considered. The only degree of freedom in this strategy, is whether one starts with the rows or columns. This choice can make a difference of at most 1 in the resulting difficulty. An interesting property of Nonograms is that small local changes in the image can have a profound impact on the solution process for the corresponding Nonogram. The difficulty can vary wildly, by changing just a single pixel.

The proposed difficulty measure can be computed efficiently, by using the *Settle* algorithm from [2]. This allows for enumeration of a large set of Nonograms, to perform a statistical analysis of the difficulty distribution. Fig. 3(a-c) show the difficulty histogram for all simple square Nonograms of size $4{\times}4$ up to $6{\times}6$, obtained by a complete enumeration. Note that all these Nonograms have a unique solution. It can be observed that a large fraction of all simple $n{\times}n$ Nonograms ($4 \leq n \leq 6$) has low difficulty (close to n), while high difficulty Nonograms (difficulty close to $\frac{1}{2}n^2$) occur rarely. For $n = 6$, out of 2^{36} images, 70.76 % yields a Nonogram of the simple type. Fig. 3(d) shows the same histogram, but using a logarithmic scale. It can be observed that the average difficulty is 4.51, whereas the difficulty can be as large as 26.

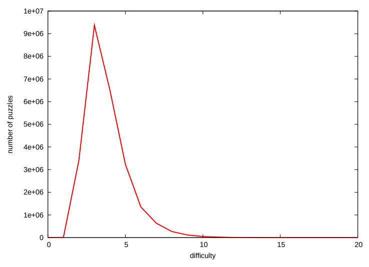
Similar trends can be observed for larger Nonograms, but an exhaustive search is no longer possible in that case. An interesting question is how the maximum possible difficulty varies with Nonogram size. A Nonogram of high difficulty should satisfy two properties:

- In each consecutive *h-sweep* and *v-sweep* only a few new pixels should be determined. Ideally, this number of newly discovered pixel values should be bound by a constant.
- In each consecutive *h-sweep* and *v-sweep* the value of at least one new pixel should be determined, as otherwise the Nonogram is not of the *simple* type.

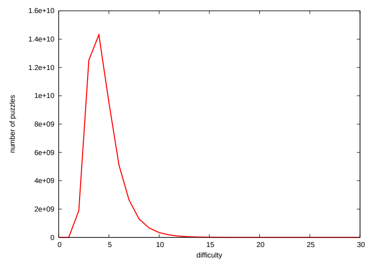
For a Nonogram of size $n{\times}n$, $n^2 + 1$ is clearly an upper bound on its difficulty. However, it is not clear at all that the maximum difficulty that can be reached increases linearly with the number of pixels. In the next section, we will show how to construct arbitrarily large Nonograms for which the asymptotic difficulty is $\frac{1}{2}n^2$, which demonstrates that the upper bound can be attained up to a constant factor.



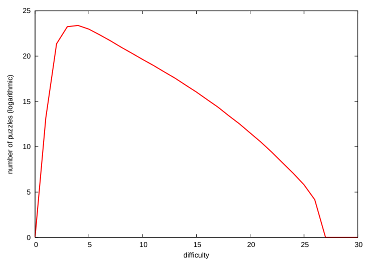
(a) Size 4×4



(b) Size 5×5



(c) Size 6×6



(d) Idem as (c), logarithmic

Figure 3: Number of Nonograms of a given size as a function of difficulty level. In (d), the vertical axis has logarithmic scaling.

4 Constructing difficult Nonograms of simple type

The *Settle* operation plays a crucial role in the construction of simple Nonograms, and in the construction of *simple Nonograms of high difficulty* in particular. In each sweep, there must be at least one line for which a new entry can be fixed. On the other hand, to construct a difficult Nonogram, it is important to avoid lines that can be fixed entirely in a single application of the *Settle* operation. In this section we will first consider scenarios where the *Settle* operation can either infer no information at all for a given line, or where the entire line can be fixed in a single step. This immediately provides a necessary condition for a Nonogram to be simple, and a characterization of the simple Nonograms of *lowest difficulty* (difficulty 1). We then proceed with a construction of the asymptotically *most difficult* Nonograms of simple type in Section 4.2.

4.1 Elementary cases of Settle operations

Simple Nonograms can be solved by performing a sequence of *Settle* operations, using only the information from a single line at a time. We first characterize those situations where the *Settle* operation immediately results in a unique solution:

Lemma 4.1. *Let $d = d_1 d_2 \dots d_k$ be an ℓ-consistent description. Then we have Settle $(?^\ell, d) \in \Sigma^\ell$ if and only if $\sum_{i=1}^{k} d_i + k - 1 = \ell$.*

Proof. By assumption, $\sum_{i=1}^{n} d_i + k - 1 \leq \ell$, as d is ℓ-consistent. Suppose that $\sum_{i=1}^{n} d_i + k - 1 = \ell$. We show that there is a unique string in Σ^ℓ that adheres to the description. Indeed, $1^{d_1}01^{d_2}0 \dots 1^{d_n} \in F(?^\ell, d)$ and has length ℓ. Furthermore, it contains a minimal number of 0 symbols, and therefore any other string that adheres to d must have length greater than ℓ, showing that $F(?^\ell, d)$ consists of a single element.

Conversely, if $\sum_{i=1}^{n} d_i + k - 1 < \ell$, then also $01^{d_1}01^{d_2}0 \dots 1^{d_n} \in F(?^\ell, d)$, so there are at least two different strings that adhere to the description. Therefore, the *Settle* operation cannot fix all entries. □

Next, we characterize those situations where the *Settle* operation cannot infer any information:

Lemma 4.2. *Let $d = d_1 d_2 \dots d_k$ be an ℓ-consistent description. Then we have Settle $(?^\ell, d) = ?^\ell$ if and only if $\sum_{i=1}^{k} d_i + k - 1 \leq \ell - \max_{1 \leq i \leq k} d_i$.*

Proof. We will first show that if $\sum_{i=1}^{k} d_i + k - 1 \leq \ell - \max_{1 \leq i \leq k} d_i$, none of the entries can be fixed by the *Settle* operation. Put $s := 1^{d_1}01^{d_2}0 \dots 1^{d_n}0^t$, where $t = \ell - \sum_{i=1}^{k} d_i - k + 1 \geq \max_{1 \leq i \leq k} d_i$. Then $s \in F(?^\ell, d)$. Consider any entry j of s. We first deal with the case that $j > \sum_{i=1}^{k} d_i + k - 1$, so s_j belongs to the (possibly empty) segment of zeros at the end of s. By shifting the rightmost block of 1 symbols in s to the right until it overlaps with entry j, a new string $\bar{s} \in F(?^\ell, d)$ can be obtained, with $\bar{s}_j = 1$. Now assume that $j \leq \sum_{i=1}^{k} d_i + k - 1$. If $s_j = 0$, then there must always be a segment of 1 symbols ending directly to the left of j. Shifting this segment to the right by 1, and shifting all symbols to the right of entry j accordingly, yields a new string $\bar{s} \in F(?^\ell, d)$ with $\bar{s}_j = 1$. If $s_j = 1$, then a new string $\bar{s} \in F(?^\ell, d)$ with $\bar{s}_j = 0$ can be obtained by shifting the segment that overlaps with entry j to the right until its leftmost entry is $j + 1$, and shifting all segments to the right of this segment accordingly. As the shift distance is never greater than $\max_{1 \leq i \leq k} d_i$, this is always possible.

Conversely, suppose that $\sum_{i=1}^{k} d_i + k - 1 > \ell - \max_{1 \leq i \leq k} d_i$. Consider the string s, where all segments of 1s have been placed to the left as far as possible, with only one 0 symbol between each pair of consecutive segments, and the string $\tilde{s}$, where all segments of 1s have been placed as far to the right as possible. Let d_t be a maximal element of d. Then its rightmost entry in s is $\sum_{i=1}^{t} d_i + t - 1$ and its leftmost entry in $\tilde{s}$ is $\ell - \sum_{i=t}^{k} d_i - k + t + 1$. Clearly, in any element of $F(?^\ell, d)$, the rightmost entry of d_t cannot be smaller than $\sum_{i=1}^{t} d_i + t - 1$ and its leftmost entry cannot be larger than $\ell - \sum_{i=t}^{k} d_i - k + t + 1$. Put $j = \sum_{i=1}^{t} d_i + t - 1$. Then $j = \sum_{i=1}^{k} d_i - \sum_{i=t+1}^{k} d_i + t - 1 \geq \ell - k - \max_{1 \leq i \leq k} d_i - \sum_{i=t+1}^{k} d_i + t + 1 = \ell - \sum_{i=t}^{k} d_i - k + t + 1$. Therefore, entry j must lie between the left and right boundaries of segment d_j in any member of $F(?^\ell, d)$ and can be fixed at 1, so the *Settle* operation can fix at least one symbol. □

Lemma 4.1 and Lemma 4.2 directly lead to the following properties related to Nonograms:

Corollary 4.3. *Let D be a Nonogram description. If all row and column descriptions satisfy the condition given in Lemma 4.2, the Nonogram is not of simple type.*

Corollary 4.4. *Let D be a Nonogram description. Then D is a Nonogram of simple type with difficulty* 1 *if and only if all row descriptions satisfy the condition given in Lemma 4.1.*

Corollary 4.4 characterizes the simple Nonograms of lowest difficulty. In the next subsection, we turn our attention to the simple Nonograms of asymptotically highest difficulty. High difficulty is obtained by keeping the conditions of Lemma 4.2 satisfied for a large number of lines, and for many sweeps.

4.2 Difficult Nonograms

In this section we will construct certain $m \times n$ Nonograms of the simple type that require approximately $\frac{1}{2}mn$ sweeps, thereby attaining a very high difficulty. More precisely, we show:

Theorem 4.5. *Let m satisfy $m = 8k + 2$ for some integer $k \geq 1$, and take an even integer n with $n \geq 14$. Then there exists an $m \times n$ Nonogram that requires $A(m, n) = (m + 2)(2n - 15)/4 + 10$ sweeps (if $k > 1$). If $k = 1$, so $m = 10$, the Nonogram requires $6n - 37$ sweeps. For square $n \times n$ Nonograms of this special type (so $n = 8k + 2$ with integer $k \geq 2$) we need $(n^2 - \frac{11}{2}n + 5)/2$ sweeps.*

The remaining part of this section is devoted to the construction of these special $m \times n$ Nonograms, and to the proof that the number of sweeps is equal to $A(m, n)$, as mentioned in the theorem. Fig. 4 shows the construction for $m = n = 18$. It is possible to give similar constructions for slightly varied values of m and n, for instance for odd width, but we will not go into detail on this. The slightly different value (2 less than the general formula predicts) if $k = 1$ is explained by a small case difference in the construction, see below.

The construction proceeds as follows. There are k rows with description n, i.e., consisting of only 1s. These rows, the so-called *split rows*, being the $(8i - 1)$th rows $(1 \leq i \leq k)$, are fully fixed in the first *h-sweep*. Furthermore, all columns, except for the first, second and last one, have description 1^{3k+1} (where σ^r denotes a sequence of r copies of a sequence σ). After the first *v-sweep*, the rows immediately above and below the split rows are therefore filled with 0s in all these columns, referred to as the *middle columns*. Any three such rows together, i.e., split row and rows immediately above and below it, form a so-called *3-strip*. Each row above a split row has description 1, each row below a split row has description 1^2. The second 1 can in the third sweep also be fixed easily at the end of the row. So together any 3-strip will —after the third sweep— look like Fig. 5.

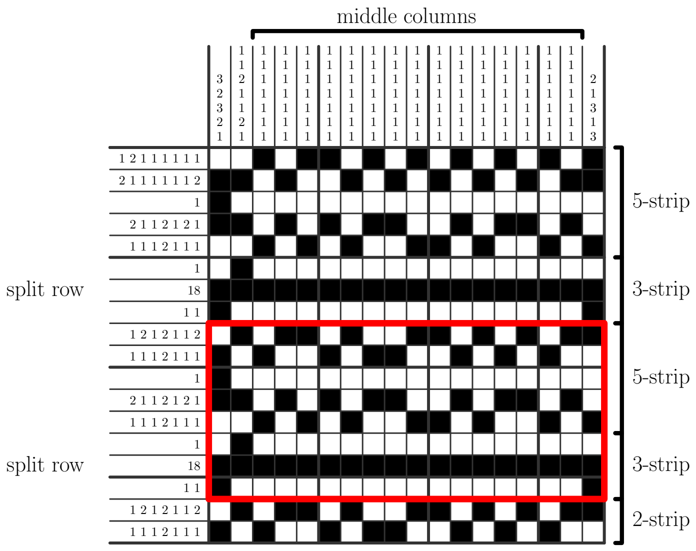


Figure 4: Overview of the construction of an 18 × 18 Nonogram with difficulty 115. The construction can be extended in the vertical direction by inserting consecutive copies of the marked block. Extension in the horizontal direction is straightforward.

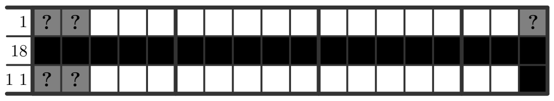


Figure 5: Contents of a 3-strip after the third sweep ($n = 18$). Gray squares denote unknown pixels.

Now the Nonogram is in fact separated by the 3-strips into k parts of height 5, called the *5-strips*, and a final part consisting of the bottom two rows, called the *2-strip*. All these parts must be solved in turn, as will be clear from the sequel.

Note that the 5-strips are all alike, except for the first one, which is used for bootstrapping the solver procedure. Within each 5-strip, the middle row will be filled with 10^{n-1} after the third sweep (its description is a single 1), and then the top two rows of the 5-strip will be solved, largely pixel by pixel, from right to left; after that the bottom two rows of the 5-strip will be solved in a similar fashion, from left to right, again largely pixel by pixel. The traversals from the top two rows to the bottom two rows within each 5-strip, and from each 5-strip to the next 5-strip or the final 2-strip, require special care. These traversals, combined with 5-strip solving, all invert the direction in which pixels are fixed, thereby constituting a zig-zag pattern.

The descriptions for the first, second and last columns are $(32)^k1$, $(112)^k1$

and $2(13)^k$, respectively.

Let us, to begin with, concentrate on the first (and special) 5-strip. The descriptions of its rows are $121^{n/2-3}$, $21^{n/2-3}2$, 1 (as said above), $21^{n/2-7}(21)^2$ and $1^{n/2-6}21^3$, respectively. The other 5-strips have a slightly different description for the first two rows, namely $(12)^2 1^{n/2-7}2$ and $1^3 21^{n/2-6}$. The row descriptions for the final 2-strip are the same: these two rows can be viewed as the top part of a regular 5-strip. In Fig. 4 the resulting solved 18×18 Nonogram is shown; note the two split rows.

One can verify that after the first three sweeps, the following pixels are fixed: most pixels from the 3-strips (as mentioned above, cf. Fig. 5; the five remaining unknown pixels are used for the traversals within and between the 5-strips), the bottom right pixel of the Nonogram (at 0), the two topmost pixels of the second columns from the left and right (at 01), and the entire middle row from each 5-strip (at 10^{n-1}, as said above). This last filling has the important property that in the middle columns, all 1s are now almost pinned: they must be in either first or second row, fourth or fifth row, and so on. This enforces that all 5-strips must be solved in order, and really after one another.

Concentrating on the first two rows, one can see that after the fourth sweep, only six pixels are fixed. The order, or rather the number of the sweep in which the pixels are found (again for $n = 18$; circles denote black pixels) is shown in Fig. 6. Here, for each two unknown pixels immediately above one another (except for the leftmost two, where this fact is not known yet), exactly one must be 1. This is inferred pixel by pixel, coming from the right, and alternating between top and bottom row, thus contributing to the large number of sweeps needed. The 2s in the descriptions are necessary for the construction of the traversal; this also holds, in several variations, for other rows in 5-strips. Note that in the third sweep no new pixel values are found for these rows.

31	2	㉚	27	㉗	㉖	23	㉒	19	⑱	15	⑭	11	⑩	7	⑥	2	④
㉙	①	29	㉘	28	25	㉔	21	⑳	17	⑯	13	⑫	9	⑧	5	①	④

Figure 6: Order in which pixel values are found for the first two rows.

Finally, let us examine the number of sweeps. From the construction it is clear that the addition of *two* new columns (among the middle columns) increases the number of sweeps by $m+2$. Indeed, in every 5-strip we need an extra 8 sweeps, and the final 2-strip adds another 4; together we get $8k+4 = m+2$ of them. Therefore, $A(m,n)$ should satisfy $A(m,n+2) = A(m,n) + m + 2$.

Furthermore, it is easy to see that every extra 5-strip and its accompanying 3-strip (as shown in Fig. 4) adds $4n + c$ sweeps, for some integer constant c. Careful inspection shows that $c = -30$. We conclude that $A(m,n)$ should satisfy $A(m+8,n) = A(m,n) + 4n - 30$. Using $A(18,18) = 115$ we arrive at the closed formula.

Note that the traversal within the first 5-strip, that reverses the right-left direction into a left-right direction, slightly differs from those in the other 5-strips. This causes the small difference in the number of sweeps for $k = 1$.

5 Generating Nonograms of simple type

In this section we describe an algorithm that produces a series of Nonograms of the simple type of varying difficulty. The algorithm is rather flexible and offers many options that can be customized. We only sketch these options here.

The generated Nonograms should resemble a given gray value image $P \in \{0, \ldots, 255\}^{m \times n}$ We want these Nonograms not to look alike, and therefore maintain a set $\mathcal{L}$ of Nonograms from which a newly generated Nonogram should differ. The new Nonogram is then appended to $\mathcal{L}$.

As a subroutine, the algorithm for generating Nonograms uses a straightforward generalization of the *Difficulty* algorithm from Fig. 2, referred to as *FullSettle*: instead of the difficulty, the *FullSettle* operation returns the set of unknown pixels, where we let the sweeps continue until they make no further progress. Note that in Section 2 the *Difficulty* algorithm was applied to Nonograms of the simple type, where the algorithm terminates by definition, whereas in the current application the Nonograms may not be solved, and termination of *FullSettle* is effected when a sweep does not yield any new fixed pixels.

Furthermore, a function *Init* (P) is used, that returns a 0–1 Nonogram that somehow resembles P, e.g., by applying a threshold operation or a binary edge detection filter to the gray level input image.

```
Generate $(P, \mathcal{L})$ :
    $p \leftarrow$ Init $(P)$; $U \leftarrow$ FullSettle $(p)$;
    while $U \neq \varnothing$ do $p \leftarrow$ Adapt $(p, U, P, \mathcal{L})$; $U \leftarrow$ FullSettle $(p)$; od
    return $(p,$ Difficulty $(p))$;
```

Figure 7: Algorithm that generates a uniquely solvable Nonogram and its difficulty.

Pseudo-code for the algorithm *Generate* is shown in Fig. 7. The main ingredient is the function *Adapt* $(p, U, P, \mathcal{L})$ that returns a Nonogram p' that is equal to p, except for (at least) one pixel that is 0 in p but is 1 in p'. Note that, since the number of black pixels strictly increases, the loop in *Generate* indeed terminates: an all black Nonogram is certainly uniquely solvable. Also note that upon entering *Adapt* at least one $(i, j) \in U$ satisfies $p_{ij} \neq 0$; indeed, if *FullSettle* $(p) \neq \varnothing$, it cannot be the case that all the unknown pixels must be 1. The function *Adapt* proceeds as in Fig. 8.

Here suitable non-negative parameters α, β and γ must be chosen. So we want the Nonogram to have a small amount of unknowns, we would like the changed pixel to be dark in the original image, and many Nonograms from $\mathcal{L}$ to be white in that particular pixel. If $\alpha = \gamma = 0$, the final Nonogram will resemble the original P, but will usually be quite dark. However, if $\beta = 0$, resemblance will be worse. High γ-values ensure diversity. Clearly, in particular if $\mathcal{L}$ is large,

```
$Adapt\,(p, U, P, \mathcal{L})$ :
    $min \leftarrow \infty$;
    for all $(i, j) \in U$ (in random order) do
        if $p_{ij} = 0$ then
            $p_{ij} \leftarrow 1$; % try new image, that differs in one pixel
            $value \leftarrow \alpha \cdot |FullSettle\,(p)| + \beta \cdot P_{ij} + \gamma \cdot \sum_{L \in \mathcal{L}} L_{ij}$;
            if $value < min$ then $min \leftarrow value$; $(k, \ell) \leftarrow (i, j)$; fi
            $p_{ij} \leftarrow 0$; % restore original image
        fi
    od
    $p_{k\ell} \leftarrow 1$;
    return $p$;
```

Figure 8: Algorithm that slightly adapts an image p.

it might be hard or even impossible to guarantee that the generated Nonogram sufficiently differs from those in $\mathcal{L}$.

In this way we get Nonograms of different difficulty, but usually quite hard ones. In order to obtain Nonograms of more varying and usually lower difficulty, the following algorithm can be used:

```
$Vary\,(p, P, \mathcal{L}, depth)$ :
    $\mathcal{M} \leftarrow \varnothing$; $d \leftarrow 0$;
    while $d < depth$ and $p$ has white pixels do
        $min \leftarrow \infty$; $U \leftarrow \{$white pixels in $p\}$;
        for all $(i, j) \in U$ (in random order) do
            $p_{ij} \leftarrow 1$; $value \leftarrow \alpha \cdot |FullSettle\,(p)| + \beta \cdot P_{ij} + \gamma \cdot \sum_{L \in \mathcal{L}} L_{ij}$;
            if $value < min$ then $min \leftarrow value$; $(k, \ell) \leftarrow (i, j)$; fi
            $p_{ij} \leftarrow 0$;
        od
        $p_{k\ell} \leftarrow 1$; $d \leftarrow d + 1$;
        if $|FullSettle\,(p)| = 0$ then $\mathcal{M} \leftarrow \mathcal{M} \cup \{(p, Difficulty\,(p))\}$; fi
    od
    return $\mathcal{M}$;
```

Figure 9: Algorithm that generates Nonograms of varying difficulty.

The algorithm returns a set of at most *depth* uniquely solvable Nonograms together with their difficulties, whose sets of black pixels strictly include that of the original p, and can therefore in general be expected to have lower difficulty. Note that each Nonogram added to $\mathcal{M}$ has at least one black pixel more than its predecessor. In fact, in practice uniquely solvable Nonograms are encountered in nearly every iteration.

Fig. 10, Fig. 12 and Fig. 13 contain some examples. All pictures in Fig. 10 are of size 32×32, while those in Fig. 12 and Fig. 13 are of size 30×38. The first picture is the original gray value image, from which the second is obtained

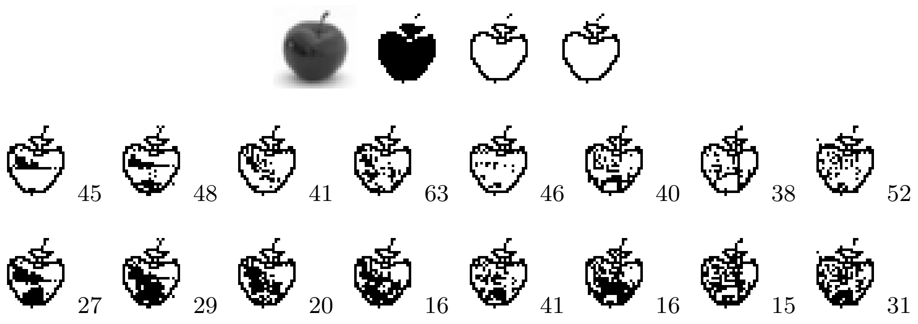


Figure 10: Nonograms created for the apple input image. Top row (left to right): grey level input image; thresholded image; result of edge detection; removal of white lines; Middle row: Nonograms created using the *Generate* algorithm; Bottom row: Final Nonograms created by the *Vary* algorithm.

by thresholding (aiming at 35 % black pixels). For the third picture, an edge detection filter is applied in Fig. 10 and Fig. 12. For the fourth picture (the third in case of Fig. 13), empty lines were addressed (in Fig. 10 this is visible near the top of the picture; in Fig. 12 near the ear). The pictures in the middle row are Nonograms of the simple type that have been generated consecutively by the *Generate* algorithm, and can therefore be expected not to look alike entirely; the numbers indicate the difficulties; the parameters of *Generate* were set at $\alpha = \gamma = 8$ and $\beta = 1$. The pictures in the bottom row are obtained from those immediately above them by the *Vary* algorithm with $\mathit{depth} = 60$; the final Nonogram generated in the main loop is depicted (other ones could also have been chosen). The set $\mathcal{L}$ contains the Nonograms from the second line created so far. Note that usually the difficulty decreases steadily during this process, but certainly not always, as is illustrated in Fig. 11: note the steep rise from difficulty 47 to 69 in the top right part of the graph. And clearly, the Nonograms from Fig. 13 are easier than those from Fig. 12; they also look more alike. The results demonstrate that by combining preprocessing of the input image with the *Generate* and *Vary* algorithms, a varied set of Nonograms can be generated that also have a varying difficulty.

6 Conclusions and further research

Nonograms are interesting study objects, due to their links with both combinatorial optimization and logic reasoning, as well as their rich variety of combinatorial properties. In this paper, we focused on the set of *simple* Nonograms, which can be solved by a series of reasoning steps involving only a single column or row at a time. We proposed a *difficulty measure* for this class, which corresponds roughly with the solution strategy followed by human puzzlers and has favourable computational properties.

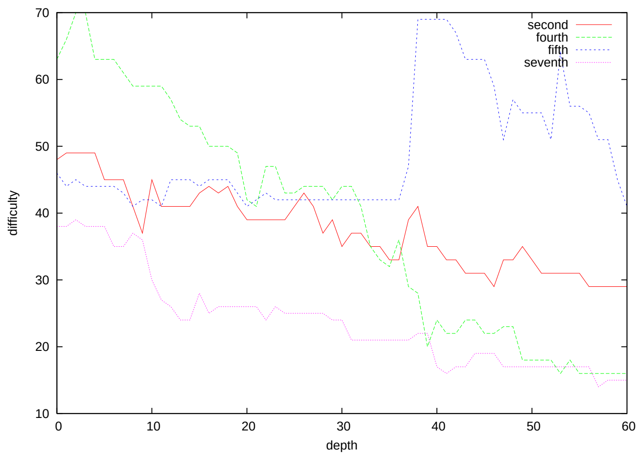


Figure 11: Evolving difficulty during runs of *Vary* for second, fourth, fifth and seventh image in the second and third row from Fig. 10.

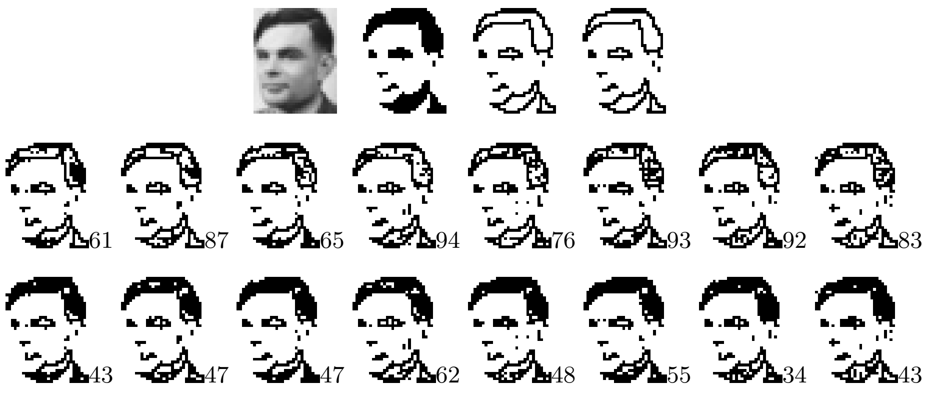


Figure 12: Nonograms created for the input image of Alan Turing. Top row (left to right): grey level input image; thresholded image; result of edge detection; removal of white lines; Middle row: Nonograms created using the *Generate* algorithm; Bottom row: Final Nonograms created by the *Vary* algorithm.

First, we described a family of $m \times n$ Nonograms that have asymptotically maximal difficulty, up to a constant factor. An interesting question remains if the difficulty can still be increased to cmn, where $c \in (\frac{1}{2}, 1]$. In the second part of the paper, we briefly described an algorithm for *generating* Nonograms of varying difficulty. The basic steps of this algorithm allow for a broad spectrum of variants, each yielding different types of Nonograms. We intend to explore such possible extensions, and their properties, in future work.

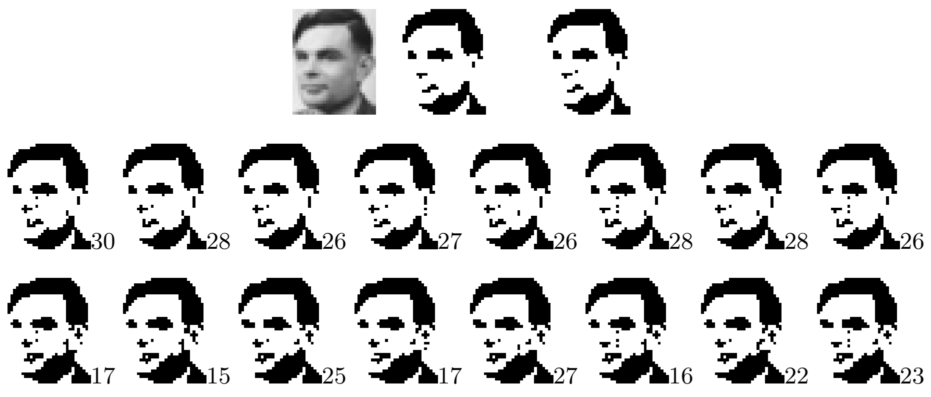


Figure 13: Nonograms created for the input image of Alan Turing. No edge detection filter is used. Top row (left to right): grey level input image; thresholded image; removal of white lines; Middle row: Nonograms created using the *Generate* algorithm; Bottom row: Final Nonograms created by the *Vary* algorithm.

References

[1] K.J. Batenburg and W.A. Kosters. A discrete tomography approach to Japanese puzzles. In *Proceedings of the 16th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC)*, pages 243–250, 2004.

[2] K.J. Batenburg and W.A. Kosters. Solving Nonograms by combining relaxations. *Pattern Recognition*, 42:1672–1683, 2009.

[3] E.G. Ortiz-Garcia, S. Salcedo-Sanz, J.M. Leiva-Murillo, A.M. Perez-Bellido, and J.A. Portilla-Figueras. Automated generation and visualization of picture-logic puzzles. *Computers and Graphics*, 31:750–760, 2007.

[4] S. Salcedo-Sanz, E.G. Ortiz-Garcia, A.M. Perez-Bellido, J.A. Portilla-Figueras, and X. Yao. Solving Japanese puzzles with heuristics. In *Proceedings IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 224–231, 2007.

[5] S. Salcedo-Sanz, J.A. Portilla-Figueras, E.G. Ortiz-Garcia, A.M. Perez-Bellido, and X. Yao. Teaching advanced features of evolutionary algorithms using Japanese puzzles. *IEEE Transactions on Education*, 50:151–156, 2007.

[6] S. Simpson. Website Nonogram solver [accessed 18.1.2010] `www.comp.lancs.ac.uk/~ss/nonogram/links.html`, 2008.

[7] N. Ueda and T. Nagao. NP-completeness results for Nonogram via parsimonious reductions, preprint, 1996.

[8] G.J. Woeginger. The reconstruction of polyominoes from their orthogonal projections. *Information Processing Lettersbibtex*, 77:225–229, 2001.