

Labeling Nonograms

Maarten Löffler¹ and Martin Nöllenburg²

1 Department of Computing and Information Sciences, Utrecht University, the Netherlands

m.loffler@uu.nl

2 Algorithms and Complexity Group, TU Wien, Vienna, Austria

noellenburg@ac.tuwien.ac.at

Abstract

Slanted and curved nonograms are a new type of picture puzzles introduced by van de Kerkhof *et al.* (2019). They consist of an arrangement of lines or curves within a frame B , where some of the cells need to be colored in order to obtain the solution picture. Up to two clues are attached as numeric labels to each line on either side of B . In this paper we study the algorithmic problem of optimizing or deciding the existence of a placement of the given clue labels to a nonogram. We provide polynomial-time algorithms for restricted cases and prove NP-completeness in general.

Lines 175

1 Introduction

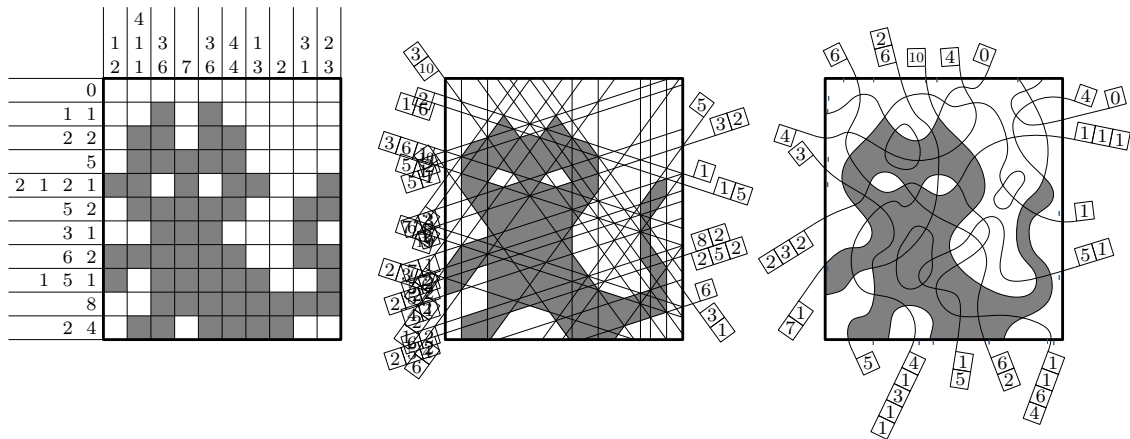


Figure 1 (left) A classic nonogram in solved state. (middle) A slanted nonogram. On the left, we show all possible labels, creating a large amount of overlap. On the right, some possible ways to resolve overlapping labels: extend labels from the same port, extend parallel labels, draw both labels of the same line at the same side, or allow crossings outside the puzzle frame. (right) A curved nonogram, showing a subset of labels (some extended to avoid overlap) which still results in a unique puzzle.

Nonograms, also known as *Japanese puzzles*, *paint-by-numbers*, or *griddlers*, are a popular puzzle type where one is given an empty grid and a set of clues on which grid cells need to be colored, typically resulting in a picture (see Figure 1 (left)). The difficulty of solving nonograms has been studied [2, 6], and remains an active topic of discussion [7].

Van de Kerkhof *et al.* introduced *curved* nonograms, a variant in which the puzzle is no longer played on a grid but on any arrangement of lines or even curves [11]. See also [8, 10, 12]. Of special interest are *slanted* nonograms [10], in which all curves are straight lines, possibly

limited to a fixed number k of orientations (also known as *sloped* nonograms or, in the case of $k = 4$, *tangram* nonograms). Figure 1 illustrates the different variants.

Van de Kerkhof *et al.* describe heuristics to generate puzzles, but they leave open the question of how to attach labels with clues. This is a non-trivial task, as labels could be placed in several valid locations. Each curve enters and leaves the picture *frame* (bold black rectangle in the figures) once, and the information about which incident cells of the arrangement should be filled is summarized in two *clues*, one on each side of the curve (we refer the reader to [11] for a full description of the rules). This gives two logical potential locations for each clue. Furthermore, it may be possible to extend curves outside the frame to make room for the clues, and not all clues need to be given for the puzzle to be solvable.

How to best label curved nonograms is an interesting open problem; it is most apparent in the case of slanted nonograms, since the rigid structure limits the possible label locations.

Problem statement. In the *nonogram labeling problem* we are given the following input: (i) A *nonogram frame*, which is a simple convex polygon B ; (ii) A set \mathcal{L} of *nonogram lines* passing through B , each $l \in \mathcal{L}$ defining a pair (p_l, q_l) of *ports* in the intersection points of l with B ; and (iii) A pair of non-negative integers (a_l, b_l) for each $l \in \mathcal{L}$, where a_l defines the width of the label above l and b_l defines the width of the label below l .

As output, we desire a *labeling* of \mathcal{L} , constructed as follows: (i) Each label a_l or b_l , $l \in \mathcal{L}$, is assigned to one of the two ports p_l or q_l ; (ii) Each label ℓ is assigned an *extension length* d (which could be 0); (iii) For each label ℓ of a nonogram line l , we draw a *leader* of length d from its assigned port p_l or q_l aligned with the slope of l , and the label itself as an $a_l \times 1$ (or $b_l \times 1$) rectangle, also aligned with the slope of l , and anchored at the end of its leader.

A labeling is *valid* if no two labels overlap each other, no label overlaps an extension leader of another label, and no label intersects the frame. In addition to being valid, we identify three further desired properties.

- **Crossing-free.** We disallow intersections between leaders.
- **Balanced.** We require the two labels of a line $l \in \mathcal{L}$ to be assigned to opposite ports.
- **Compact.** We require all leaders to be of length 0, or the shortest length necessary to avoid an intersection between the label and the frame.

We would like to find a crossing-free compact balanced labeling, but this may not exist (see, e.g., Figure 1 (middle)). We study the computational problem of testing the existence of a solution, or, for some variants, minimizing the total leader length, for several combinations of properties. In some cases, we also restrict the number k of distinct slopes of lines in \mathcal{L} .

Results. First, we observe that a balanced solution which is not crossing-free and not compact always exists: we simply extend the leaders sufficiently far. Since this does not give a satisfactory result, we focus on more restricted variants in the remainder.

In Section 2.1, we show that testing whether a compact solution exists is possible in polynomial time. In Section 2.2, we show that a non-crossing balanced solution of minimal total leader length can be computed in polynomial time, if the assignment of labels to ports is given and $k = 2$. Finally, in Section 3, we show that the problem of testing whether a crossing-free solution exists is NP-complete, even when $k = 2$.

Related work. Labeling nonograms is closely related to the boundary labeling problem in information visualization, where a set of point features in a rectangular frame B is to be annotated with labels (names or short descriptions) that are placed outside B and connected to their features with straight or polygonal leaders [4, 5]. Yet nonogram labeling is different

in several respects: Since the curves/lines in nonograms intersect the frame B in two fixed locations, the possible positions for the clues are very restricted, while in boundary labeling the label can basically be placed anywhere along B as long as the resulting leader lines are valid. Labels in boundary labeling are typically axis-aligned, but the clues in nonograms are aligned with their respective nonogram line or curve. Finally, by extending the nonogram curves beyond the frame to gain extra space, we obtain a new degree of freedom that has been rarely used in boundary labeling, with some exceptions of multi-row labeling [3, 9].

2 Algorithms

2.1 Non-extensible leaders

In our first result, we assume that each label ℓ must be placed as close to the frame B as possible, i.e., ℓ must touch B . This leaves only one degree of freedom for each label ℓ of a nonogram line l , namely whether it is placed at port p_l or q_l .

► **Theorem 2.1.** *Given a nonogram labeling instance, we can decide in polynomial time whether a compact labeling exists. This is true regardless of whether we require it to be balanced.*

Proof. We derive a 2-SAT formula φ that has a satisfying variable assignment if and only if a valid labeling without leader extensions exists. For each nonogram line $l \in \mathcal{L}$ we define two variables x_l^a and x_l^b , where $x_l^a = 1$ ($x_l^a = 0$) indicates that the label above l is assigned to the port p_l (q_l) of l . Similarly, $x_l^b = 1$ ($x_l^b = 0$) indicates that the label below l is assigned to p_l (q_l). It is clear that a variable assignment is in bijection to a port assignment of the labels and it remains to add some clauses to φ to model the valid labelings. For each overlap of a label of a line l with a label of another line l' (we call that a *conflict*), we add a clause that prevents both labels to be selected simultaneously. As an example consider the case that the label above l and the label below l' intersect if both assigned to their ports p_l and $p_{l'}$. Then we add the clause $\neg x_l^a \vee \neg x_{l'}^b$. Now a satisfying assignment for φ corresponds to an assignment of each label to a port of its nonogram line such that no two labels intersect each other; otherwise some clause would not be satisfied. To ensure that the labeling is balanced, we would add the additional clauses $x_l^a \vee x_l^b$ and $\neg x_l^a \vee \neg x_l^b$ for each $l \in \mathcal{L}$.

Solving the 2-SAT instance takes linear time [1] in the size of φ , where the number of clauses of φ is linear in the number of nonogram lines and the number of label conflicts. ◀

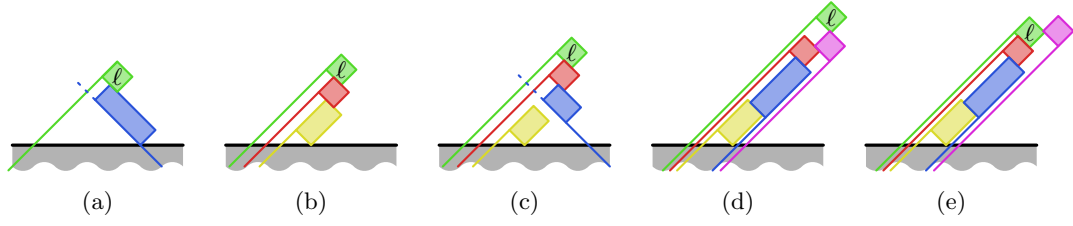
We remark that this 2-SAT model is independent of the type of nonogram lines (or curves) and the shape of B . It depends only on the set of conflicting candidate label positions.

2.2 Fixed side assignment

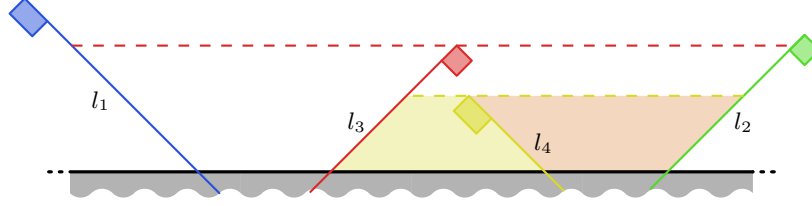
Our second algorithm allows extensible leaders, but disallows leader intersections and assumes that a balanced assignment of the labels to the ports of each nonogram line is given. We further assume that the nonogram lines have slopes ± 1 and that the frame B is a rectangle.

► **Lemma 2.2.** *For a nonogram labeling instance with n lines and a balanced fixed side assignment for each label we can discretize the relevant extension lengths of each label to $O(n^2)$ values such that we can find a labeling of minimum total extension length among this set of extension lengths (if the instance has a solution at all).*

Proof. In a minimum-length labeling, every label ℓ of a nonogram line l should be shifted as close to B as possible without intersecting another label. Hence it either touches B and the



106 ■ **Figure 2** Possible cases of extension lengths for label ℓ in the proof of Lemma 2.2.



130 ■ **Figure 3** Illustration of the dynamic programming recursion.

109 extension length is 0 or 1 (depending on which side of l is labeled), or it touches another label
 110 ℓ' blocking it from moving closer to B . This blocking label ℓ' can belong to a line of different
 111 slope, meaning that the extension length of ℓ is given by the intersection point of the two
 112 nonogram lines (possibly $+1$), see Figure 2(a). There are $O(n)$ such intersection points for
 113 ℓ . Or, the lines of ℓ and ℓ' are parallel and of distance at most 2. If the chain of blocking
 114 relations comprises only parallel lines and all of them have the labels on the same side
 115 (Figure 2(b)), then we get a single extension length for ℓ . If the parallel lines come in a group
 116 of right-flipped labels followed by a group of left-flipped labels as in Figures 2(d–e) then any
 117 prefix of the sequence of left-flipped labels can add to the extension length of ℓ , which again
 118 yields $O(n)$ possible extension lengths. Finally, ℓ may be blocked by some chain of parallel
 119 labels, the last of which is blocked by a label of an orthogonal line (Figure 2(c)). Considering
 120 all combinations this last case can give rise to $O(n^2)$ different extension lengths. ◀

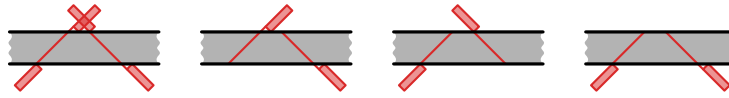
121 ► **Theorem 2.3.** *Given a nonogram labeling instance with n lines and a fixed side assignment*
 122 *for each label, we can decide in $O(n^9)$ time, whether an assignment of an extension length to*
 123 *each label exists such that the resulting labeling is valid. If this is the case we can find one of*
 124 *minimum total extension length.*

125 **Proof.** (Sketch) The algorithm uses dynamic programming and recursively defines $O(n^6)$
 126 independent subinstances, which are bounded by two lines l_1, l_2 , each with $O(n^2)$ possible
 127 extension lengths due to Lemma 2.2, and a horizontal segment through the shorter one of
 128 them (see Figure 3). For subdividing a subinstance recursively, we consider the longest valid
 129 line l' and its label inside the bounded region, for which we have $O(n^3)$ candidates. ◀

131 3 Hardness

132 We will show in this section that the problem of testing whether a valid labeling exists, in
 133 the setting where we are allowed to choose at which port each label is placed and are allowed
 134 to extend the leaders to any desired length, is NP-hard. We will construct an instance with
 135 a rectangular frame which only has lines of slopes 1 and -1 . We will reduce from 3-SAT.

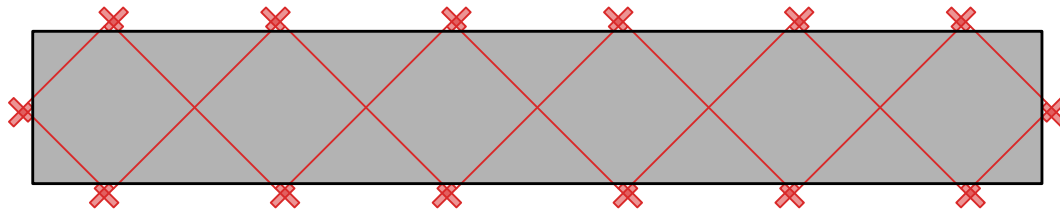
139 **Variables.** The bulk of the construction consists of *cross gadgets*. A cross gadget consists of
 140 two short labels intersecting each other at 90° angles.¹ Figure 4 shows a single cross gadget.



141 ■ **Figure 4** A variable gadget (cross gadget) and its three possible valid solutions.

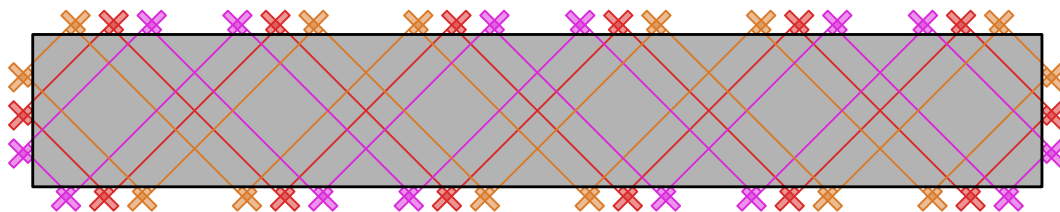
142 Note that that for a cross gadget it is not relevant on which side of the lines the labels
 143 are, and that although labels can be extended, doing so does not change the combinatorial
 144 choices of which combinations of sides are possible.

145 A cross gadget has three possible valid states, and we wish to use them to represent
 146 variables, which have two valid states. Furthermore, we would like to enforce multiple cross
 147 gadgets to represent the same variable. We can achieve both of these properties by connecting
 148 several cross gadgets into a *variable loop*. Figure 5 illustrates a variable loop.



149 ■ **Figure 5** A single variable loop.

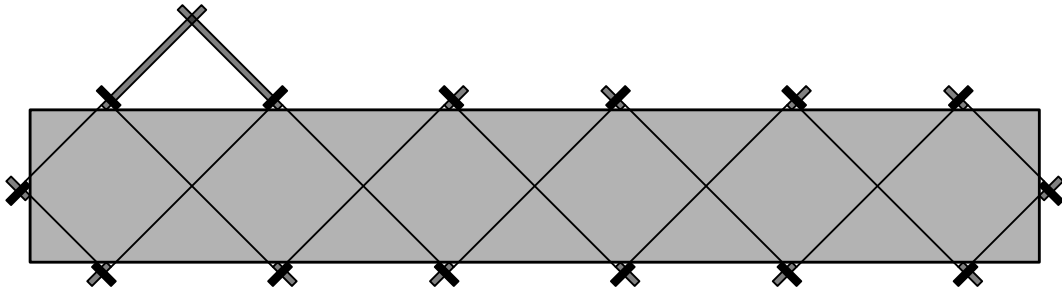
150 By connecting the cross gadgets into a loop, we ensure that only two valid solutions
 151 remain for each cross gadget. Note that we can increase the number of occurrences of a cross
 152 of the same variable by making the frame wider, and we can increase the distance between
 153 consecutive crosses by making the frame higher. We can embed multiple independent loops
 154 next to each other, as illustrated in Figure 6.



155 ■ **Figure 6** Multiple variable loops.

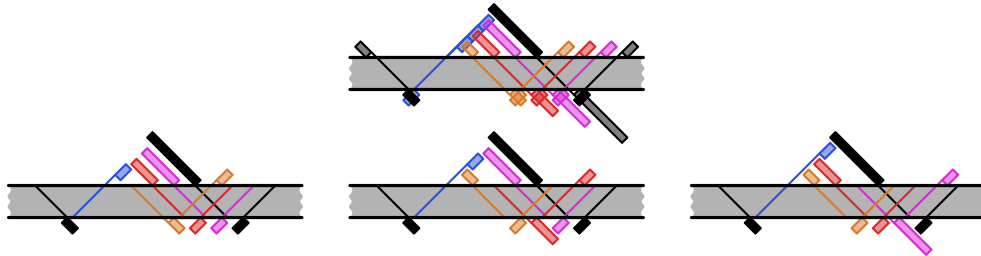
156 In the construction, we will also need to place some labels which cannot be removed. We
 157 create a special variable loop, for which one of the states is disabled by a crossing between a
 158 positive and a negative label. We achieve this by making them longer (see Figure 7). Note
 159 that forced (black) labels are only forced to be on a particular *side* of B ; they can still be
 160 extended, but we will use them in a way where extending black labels is never useful.

136 ¹ Here, we are only using one side of each line $l \in \mathcal{L}$, which corresponds to a setting where not all clues
 137 are present in the puzzle. The construction can easily be adapted to the case where both labels are
 138 present, by placing them at the same side (i.e. not balanced).



161 ■ **Figure 7** A variable loop where one state is impossible; solid black labels are *forced*.

162 **Clauses.** Next, a clause gadget essentially consists of a single *clause* label which is forced
 163 to be at a specific port, but can be extended. Depending on how far it is extended, it will
 164 intersect different variable labels. The clause label is restricted to only three essentially
 165 different positions by two fixed labels. Figure 8 illustrates a clause gadget.



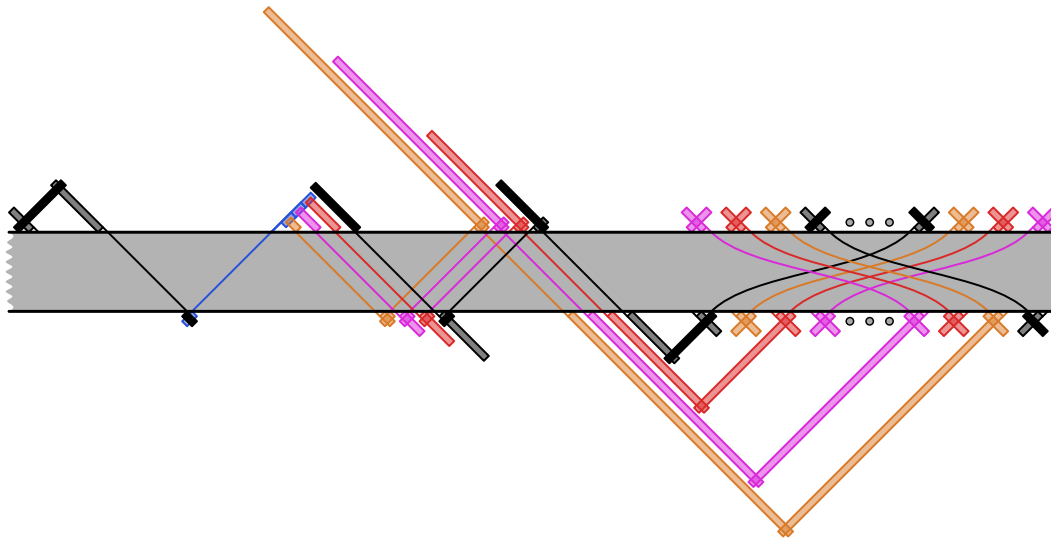
166 ■ **Figure 8** Clause gadget and three possible valid solutions.

167 For a clause label, it is important on which side of the line the label is placed: it must be
 168 faced towards the variable labels. We need to connect the clause gadget to the correct literals
 169 of the three variables involved in the clause, as well as to the fixed loop on two sides. For this,
 170 we need to make some very long labels. Figure 9 illustrates how a clause gadget is connected,
 171 showing only the relevant labels. Note that the variables may need to be connected to two
 172 different groups of cross gadgets in the variable loops.

174 ► **Theorem 3.1.** *Given a nonogram instance without side assignment and extensible leaders,*
 175 *it is NP-complete to decide whether a valid labeling exists.*

176 — References —

- 177 1 Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear-time algorithm for
 178 testing the truth of certain quantified boolean formulas. *Information Processing Letters*,
 179 8(3):121–123, 1979. doi:10.1016/0020-0190(79)90002-4.
- 180 2 Kees Joost Batenburg and Walter A. Kosters. On the difficulty of nonograms. *ICGA*
 181 *Journal*, 35(4):195–205, 2012. URL: <https://doi.org/10.3233/ICG-2012-35402>, doi:
 182 10.3233/ICG-2012-35402.
- 183 3 Michael A. Bekos, Michael Kaufmann, Katerina Potika, and Antonios Symvonis. Multi-
 184 stack boundary labeling problems. In S. Arun-Kumar and Naveen Garg, editors, *Foundations of Software Technology and Theoretical Computer Science (FSTTCS'06)*, volume
 185 4337 of *LNCS*, pages 81–92. Springer, 2006. doi:10.1007/11944836_10.



173 ■ **Figure 9** Connecting a clause gadget to the correct literals for clause $\neg\text{purple} \vee \text{red} \vee \neg\text{orange}$.

- 187 4 Michael A. Bekos, Michael Kaufmann, Antonios Symvonis, and Alexander Wolff. Boundary
188 labeling: Models and efficient algorithms for rectangular maps. *Computational Geometry*
189 *Theory and Applications*, 36(3):215–236, 2007. doi:10.1016/j.comgeo.2006.05.003.
- 190 5 Michael A. Bekos, Benjamin Niedermann, and Martin Nöllenburg. External labeling
191 techniques: A taxonomy and survey. *Computer Graphics Forum*, 38(3):833–860, 2019.
192 doi:10.1111/cgf.13729.
- 193 6 Daniel Berend, Dolev Pomeranz, Ronen Rabani, and Ben Raziel. Nonograms: Combina-
194 torial questions and algorithms. *Discrete Applied Mathematics*, 169:30–42, 2014. URL:
195 <https://doi.org/10.1016/j.dam.2014.01.004>, doi:10.1016/j.dam.2014.01.004.
- 196 7 Yen-Chi Chen and Shun-Shii Lin. A fast nonogram solver that won the TAAI 2017 and
197 ICGA 2018 tournaments. *ICGA Journal*, 41(1):2–14, 2019. URL: [https://doi.org/10.](https://doi.org/10.3233/ICG-190097)
198 [3233/ICG-190097](https://doi.org/10.3233/ICG-190097), doi:10.3233/ICG-190097.
- 199 8 Tim K. de Jong. The concept and automatic generation of the curved nonogram puzzle.
200 Master’s thesis, 2016. URL: <https://dspace.library.uu.nl/handle/1874/337632>.
- 201 9 Andreas Gemsa, Jan-Henrik Haunert, and Martin Nöllenburg. Multi-row boundary-labeling
202 algorithms for panorama images. *ACM Trans. Spatial Algorithms and Systems*, 1(1):1:1–
203 1:30, 2015. doi:10.1145/2794299.
- 204 10 Raphael J. Parment. Generation of sloped nonograms. Master’s thesis, 2015. URL: <https://dspace.library.uu.nl/handle/1874/323196>.
- 205 11 Mees van de Kerkhof, Tim de Jong, Raphael Parment, Maarten Löffler, Amir Vaxman, and
206 Marc J. van Kreveld. Design and automated generation of japanese picture puzzles. *Comput.*
207 *Graph. Forum*, 38(2):343–353, 2019. URL: <https://doi.org/10.1111/cgf.13642>, doi:
208 [10.1111/cgf.13642](https://doi.org/10.1111/cgf.13642), doi:10.1111/cgf.13642.
- 209 12 Mees A. van de Kerkhof. Improved automatic generation of curved nonograms. Master’s
210 thesis, 2017. URL: <https://dspace.library.uu.nl/handle/1874/357864>.
- 211

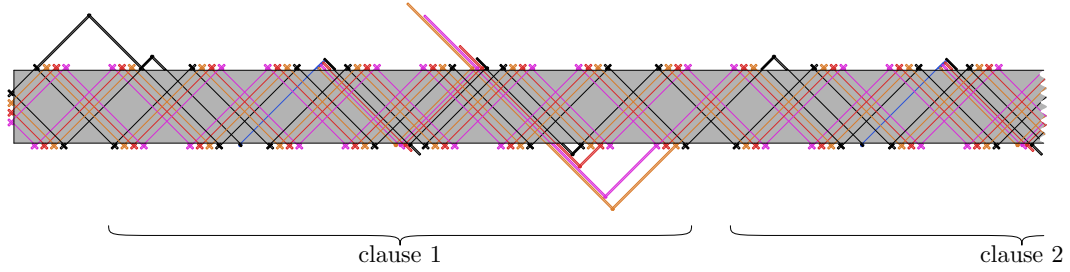
212 A Additional details from Section 3

213 **The global picture.** Globally, we embed the different clauses horizontally next to each
214 other. Each clause, including its connections, covers a horizontal distance of a constant
215 number of variable zig-zags. These connections are placed between the variable loops, so

100:8 Labeling Nonograms

216 they do not interfere. Figure 10 shows how the first clause and the beginning of the second
217 clause could look globally, without hiding any labels.

218 The total horizontal distance covered will be $O(nm)$. The vertical distance is $O(n)$.



219 ■ **Figure 10** The global picture.