<table>
<thead>
<tr>
<th>Title</th>
<th>Computational complexity and an integer programming model of Shakashaka</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Demaine, Erik D.; Okamoto, Yoshio; Uehara, Ryuhei; Uno, Yushi</td>
</tr>
<tr>
<td>Citation</td>
<td>Proceedings of the 25th Canadian Conference on Computational Geometry (CCCG 2013): 31-36</td>
</tr>
<tr>
<td>Issue Date</td>
<td>2013-08</td>
</tr>
<tr>
<td>Type</td>
<td>Conference Paper</td>
</tr>
<tr>
<td>Text version</td>
<td>author</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/10119/11619">http://hdl.handle.net/10119/11619</a></td>
</tr>
<tr>
<td>Rights</td>
<td>Copyrights of the article is maintained by the authors. Erik D. Demaine, Yoshio Okamoto, Ryuhei Uehara and Yushi Uno, Proceedings of the 25th Canadian Conference on Computational Geometry (CCCG 2013), 2013, 31-36.</td>
</tr>
</tbody>
</table>

| Description | |
Computational complexity and an integer programming model of Shakashaka

Erik D. Demaine∗ Yoshio Okamoto† Ryuhei Uehara‡ Yushi Uno§

Abstract

Shakashaka, one of the so-called “pencil-and-paper” puzzles like Sudoku, was proposed by Guten and has been popularized by the Japanese publisher Nikoli. The computational complexity of Shakashaka was not studied so far. We first prove that Shakashaka is NP-complete. Next we formulate Shakashaka as an integer programming problem. We apply the formulation to some concrete instances that appeared in a puzzle book, and solve them by using an IP solver, and show the experimental results; each problem can be solved in a second.

Keywords: integer programming, NP-completeness, pencil-and-paper puzzle, Shakashaka

1 Introduction

The puzzle Shakashaka is popularized by Japanese publisher Nikoli, along with its more famous sibling Sudoku, and several other “pencil-and-paper” puzzles. Shakashaka was proposed in 2008 by Guten, and since then, it has become one of the main puzzles in Nikoli.

An instance of Shakashaka is given as a rectangular board that consists of \(m \times n\) unit squares, and each square is either white or black. To solve the puzzle, each white square is filled by a black isosceles right triangle or left as it is. When a white square is filled by a black isosceles right triangle, it is split into one black isosceles right triangle and one white isosceles right triangle (that is, one of \(\), \(\), \(\), and \(\)). The square filled by a black isosceles right triangle is called a w/b square. Some black squares contain numbers, and each of them specifies the number of w/b squares among four (vertically or horizontally adjacent) neighbors of the black squares. (Each black square without number allows any number of w/b neighbors.) The objective of the puzzle is to fill the white squares satisfying the above constraints starting from the given initial board so that the remaining white area only consists of (empty) squares and rectangles. An example of the puzzle Shakashaka in [1] is shown in Figure 1(a), and its (unique) solution is given in Figure 1(b).

As mentioned in the literature [2], a lot of “pencil-and-paper” puzzles have been shown NP-complete. However, the computational complexity of Shakashaka was not studied so far. In this paper, we first prove that Shakashaka is NP-complete. The proof is done by a reduction of the planar 3SAT. Next we formulate Shakashaka as an integer programming problem. Although integer programming is one of Karp’s 21 NP-complete problems, there are many efficient solver from the practical point of view. According to the talk by Bixby, the performance of the solvers in 2012 runs around \(9.5 \times 10^8\) times faster than one in 1991 [3]. Therefore, once we can formulate the puzzle as a 0-1 integer linear programming problem, we can use these solvers to solve the puzzle efficiently. Some authors proposed integer-programming formulations of several puzzles before, mainly for the didactic purpose [4, 5, 6, 7, 8]. The formulation of Shakashaka is not so straightforward since we have to avoid forming nonrectangular ortho-
nal shapes or nested rectangles. We show that our formulation characterizes the constraints of Shakashaka. We also performed computational experiments, and observed that each instance can be solved in one second.

2 Preliminaries

We here give a formal definition of the puzzle Shakashaka. An instance \( I \) of Shakashaka is a rectangular board of size \( m \times n \). Each unit square is colored either white or black. Some black squares contain a number \( i \in \{0, \ldots, 4\} \). A solution of the instance \( I \) is a mapping from the set of white squares in \( I \) to the set \( \{WH, NW, NE, SW, SE\} \) with the following conditions.

1. Each white square mapped to \( WH \) is left uncolored, and each square mapped to \( NW \) corresponds to the pattern . The squares mapped to \( NE, SW, SE \) correspond to the patterns , , and respectively. A square filled by a black isosceles right triangle is called a w/b square.

2. Each black square that contains the number \( i \) has exactly \( i \) w/b squares among its four neighbors.

3. Each connected white area forms a white rectangle (or square).

Computationally, Shakashaka is a decision problem, where for a given instance we decide whether or not it has a solution. The counting version of Shakashaka counts the number of solutions of a given instance.

3 NP-completeness of Shakashaka

In this section, we prove the following theorem.

Theorem 1 Shakashaka is NP-complete.

The proof is done by a reduction of the planar 3SAT, one of the well-known NP-complete problems [9]. Let \( F \) be an instance of the planar 3SAT. That is, \( F \) consists of the set \( \mathcal{C} = \{C_1, \ldots, C_m\} \) of \( m \) clauses of \( n \) variables \( \mathcal{V} = \{x_1, \ldots, x_n\} \), each clause \( C_i \) consists of three literals, and the graph \( G = (\mathcal{C} \cup \mathcal{V}, \mathcal{E}) \) is planar, where \( \mathcal{E} \) contains an edge \( \{C_i, x_j\} \) if and only if \( x_j \) or \( \overline{x}_j \) is in \( C_i \).

Now we show a reduction of \( F \) to an instance \( I \) of Shakashaka. The key idea is to use the pattern shown in Figure 2 (the outside of the pattern is filled by black squares). For the pattern Figure 2(a), we have two choices shown in Figure 2(b). Essentially, this works as a “wire” to propagate a signal. We consider the big square containing four white squares in 2(b) represents “0,” and the big diamond containing four (different) w/b squares in 2(b) represents “1.” That is, the “wire” pattern propagates a signal using the parity in two different ways. In the context of [2], we need the gadgets of “variable,” “split,” “corner,” and “clause.” We here show the gadgets one by one.

Variable gadget: The variable gadget is depicted in Figure 3 (left). It is easy to see that we have two ways to fill the pattern as Figure 3 (right). It can propagate its value by the wire gadget as in the figure. It is also easy to obtain the negation of the variable.

Split gadget/corner gadget: They are given in a straightforward way as in Figure 4.

Clause gadget: For a clause \( C = \{x, y, z\} \), the clause gadget is given in Figure 5. According to the values of \( x, y, z \), we have eight possible cases. Among them,
only the case \( x = y = z = 0 \) violates the condition of Shakashaka (Figure 6).

The gadgets for wire, variable, split, corner are aligned properly since they can be put into \( 3 \times 3 \) squares. However, at a clause gadget, we have to change the positions of wires to fit the gadgets. To shift the position, we use “parity” gadgets shown in Figure 7 (left). Join-
We formulate Shakashaka in terms of binary integer programming. Remind that an instance \( I \) of Shakashaka consists of a rectangular board of size \( m \times n \). We identify each square by \( (i,j) \in \{1,\ldots,m\} \times \{1,\ldots,n\} \) in the natural way.

**Variables:** For each white square \((i,j)\), we will use five 0-1 variables \(x[i,j,WH], x[i,j,NW], x[i,j,NE], x[i,j,SW], \) and \(x[i,j,SE]\). Each of them takes either 0 or 1, and it takes 1 in the following cases, respectively (otherwise, it takes 0):

\[
\begin{align*}
    x[i,j,WH] = 1 & \quad \text{means that } (i,j) \text{ remains white},
    x[i,j,NW] = 1 & \quad \text{means that } (i,j) \text{ is filled with } \square,
    x[i,j,NE] = 1 & \quad \text{means that } (i,j) \text{ is filled with } \blacktriangleleft,
    x[i,j,SW] = 1 & \quad \text{means that } (i,j) \text{ is filled with } \blacktriangledown,
    x[i,j,SE] = 1 & \quad \text{means that } (i,j) \text{ is filled with } \blacklozenge.
\end{align*}
\]

We construct a linear system \( S(I) \) with the variables \( x[i,j,\ast] \) such that the solutions of the instance \( I \) of Shakashaka are in bijection with the solutions of \( S(I) \). To this end, we set up five types of linear constraints as described below.

**Constraint A (At Most One Triangle in Each White Square):** In a solution of \( I \), each white square either remains white, or filled with one of the four black isosceles right triangles. We map this condition to the following linear equality:

\[
x[i,j,WH] + x[i,j,NW] + x[i,j,NE] + x[i,j,SW] + x[i,j,SE] = 1
\]

for each \( i \) and \( j \) where \((i,j)\) is a white square.

**Proposition 3** Let \( S_A(I) \) be the linear system that consists of Constraint A. Then any feasible solution of \( S_A(I) \) gives the mapping from each white square to one of \( \square, \blacktriangleleft, \blacktriangledown, \blacklozenge \), or a white blank.

**Constraint B (Neighbors of Black Squares):** Next we look at the black squares \((i,j)\). First we consider the case that \((i,j)\) contains no number. In this case, \((i,j)\) gives some restrictions to its white neighbor. For example, suppose that \((i-1,j)\) is white. Then, if \((i-1,j)\) is \( \square \) or \( \blacktriangledown \), we have 45° white area between \((i-1,j)\) and \((i,j)\). Thus \((i-1,j)\) must be \( \square \), \blacktriangleleft, or leaving white. Hence, in this case,

\[
x[i-1,j,WH] + x[i-1,j,NW] + x[i-1,j,NE] + x[i-1,j,SW] + x[i-1,j,SE] = 1
\]

\( \text{for each } i \text{ and } j \text{ where } (i,j) \text{ is a white square.} \)

\[ (2) \]

\[
x[i-1,j,WH] + x[i-1,j,NW] + x[i-1,j,NE] + x[i-1,j,SW] + x[i-1,j,SE] = 1
\]

\( \text{for each } i \text{ and } j \) where \((i,j)\) is a white square.

\[ (3) \]

\[
x[i-1,j,WH] + x[i-1,j,NW] + x[i-1,j,NE] + x[i-1,j,SW] + x[i-1,j,SE] = 1
\]

\( \text{for each } i \text{ and } j \) where \((i,j)\) is a white square.

\[ (4) \]
where \(x[i,j,*]\) is regarded as 0 if \((i,j)\) is black. We also fix \(x[i-1,j,SW] = x[i-1,j,NE] = x[i+1,j,SW] = x[i+1,j,SE] = x[i,j-1,NW] = x[i,j-1,SW] = x[i,j+1,NE] = x[i,j+1,SE] = 0\) to avoid the 45° white angle.

**Constraint C (Sequences of Triangles):** Next we turn to the restrictions to make each connected white area a rectangle. Assume \(x[i,j,SW] = 1\). That is, \((i,j)\) is filled as \(\mathbb{W}\). In the case, the upper triangle can be orthogonal if and only if either \(x[i,j+1,SE] = 1\) \((i,j+1)\) is \(\mathbb{L}\) or \(x[i+1,j,SW] = 1\) \((i,j+1)\) is \(\mathbb{R}\). Therefore, we obtain the following constraint:

\[
x[i,j,SW] \leq x[i,j+1,SE] + x[i+1,j+1,SW]. \tag{5}
\]

Moreover, when \(x[i,j,SW] = x[i+1,j+1,SW] = 1\), \((i,j+1)\) must remain white, or \(x[i,j+1,WH] = 1\). (When \((i,j)\) is \(\mathbb{R}\), we have a parity problem; we cannot enclose this area by extending this pattern. The other cases are also inhibited.) This implies the following constraint:

\[
x[i,j,SW] + x[i+1,j+1,SW] \leq x[i,j+1,WH] + 1. \tag{6}
\]

We add the similar constraints for other directions. Then, we have the following proposition.

**Proposition 4** Let \(S_C(I)\) be the linear system that consists of Constraints A, B, and C, and fix any feasible solution of \(S(I)\). Then, each angle on the boundary of each connected white area given by the mapping is 90°.

**Proof.** It is not difficult to check that no 45° white angle is left alone between two black areas by the restrictions. \(\square\)

**Constraint D (Exclusion of Concave Corners):** By Proposition 4, any feasible solution to Constraints A, B, and C produces the pattern consists of white polyomino. However, this does not exclude a concave corner yet. By Equation 5, no w/b square forms a part of a concave corner. Thus, a concave corner may be produced by only white squares. We suppose that \(x[i,j,WH] = x[i+1,j,WH] = 1\). Then, \((i+1,j+1)\) must be \(\mathbb{W}\) or must remain white. Thus we add the following constraints (for all possible directions):

\[
x[i,j,WH] + x[i+1,j,WH] + x[i,j+1,WH] \leq x[i+1,j+1,WH] + x[i+1,j+1,SE] + 2. \tag{7}
\]

We now have the following proposition.

**Proposition 5** Let \(S_D(I)\) be the linear system that consists of Constraints A, B, C, and D, and fix any feasible solution of \(S(I)\). Then each connected white area given by the mapping is a convex orthogonal shape.
Table 1: Experimental results for the instances at nikoli.com

<table>
<thead>
<tr>
<th>Problem</th>
<th>Size</th>
<th>Level</th>
<th># of white squares</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10x10</td>
<td>Easy</td>
<td>76</td>
<td>0.02</td>
</tr>
<tr>
<td>2</td>
<td>10x10</td>
<td>Easy</td>
<td>77</td>
<td>0.03</td>
</tr>
<tr>
<td>3</td>
<td>10x10</td>
<td>Easy</td>
<td>82</td>
<td>0.03</td>
</tr>
<tr>
<td>4</td>
<td>10x18</td>
<td>Easy</td>
<td>131</td>
<td>0.07</td>
</tr>
<tr>
<td>5</td>
<td>10x18</td>
<td>Medium</td>
<td>156</td>
<td>0.09</td>
</tr>
<tr>
<td>6</td>
<td>10x18</td>
<td>Medium</td>
<td>144</td>
<td>0.07</td>
</tr>
<tr>
<td>7</td>
<td>14x24</td>
<td>Medium</td>
<td>297</td>
<td>0.21</td>
</tr>
<tr>
<td>8</td>
<td>14x24</td>
<td>Hard</td>
<td>295</td>
<td>0.19</td>
</tr>
<tr>
<td>9</td>
<td>20x36</td>
<td>Hard</td>
<td>645</td>
<td>0.84</td>
</tr>
<tr>
<td>10</td>
<td>20x36</td>
<td>Hard</td>
<td>632</td>
<td>0.91</td>
</tr>
</tbody>
</table>

Figure 10: Seconds for the artificial examples ($n = 2, 3, \ldots, 40$).

black squares contain the number 2 as shown in the figure. Each of them has a unique solution. For $n = 40$, the solution is obtained in 19.86 seconds. The computation times seem to be proportional to $1.18^n$.

5 Concluding Remarks

In this paper, we show that Shakashaka is NP-complete. In our reduction, the black squares contain 1 only, and no other numbers are used. An interesting question is to determine the computational complexity of Shakashaka that contains no black squares with numbers. A nontrivial example is given in Figure 11, which has a unique solution. There are two natural questions in this Shakashaka puzzle. How many black squares are required to have a unique solution for $m \times n$ board? Can this restricted Shakashaka be solved in linear time?

References


