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).

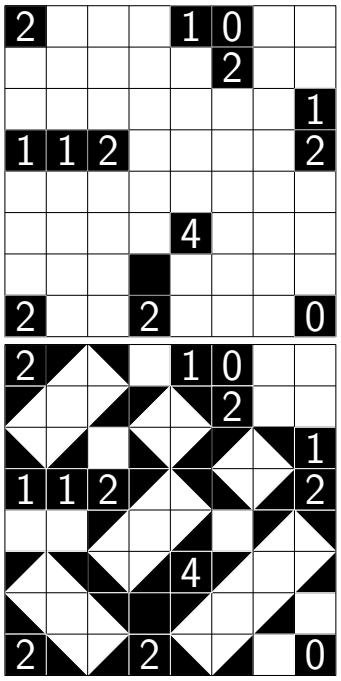

Figure 1: An instance of the puzzle Shakashaka and its solution ([1])

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 9.5×10^8 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 orthogo-

*Computer Science and Artificial Intelligence Laboratory, MIT, edemaine@mit.edu

†Graduate School of Informatics and Engineering, The University of Electro-Communications (UEC), okamotoy@uec.ac.jp

‡School of Information Science, JAIST, uehara@jaist.ac.jp

§Graduate School of Informatics and Engineering, Osaka Prefecture University (OPU), uno@mi.s.osakafu-u.ac.jp

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 $\bar{x_j}$ 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

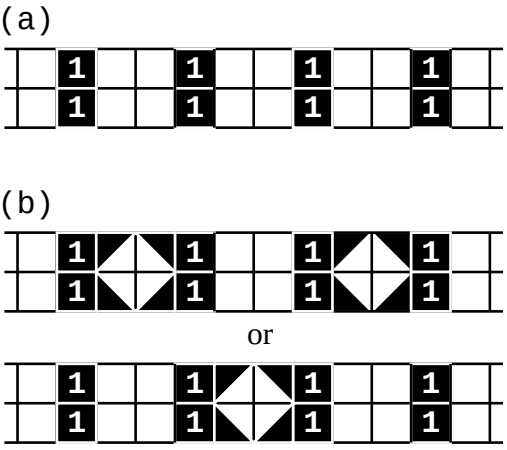

Figure 2: Basic pattern

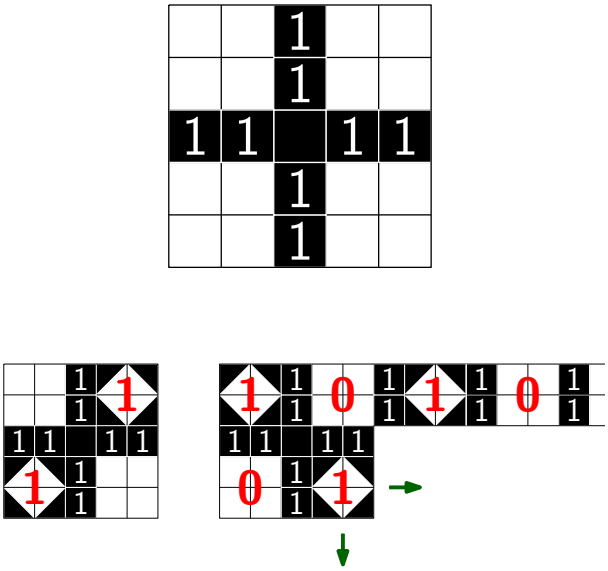

Figure 3: Variable gadget

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,

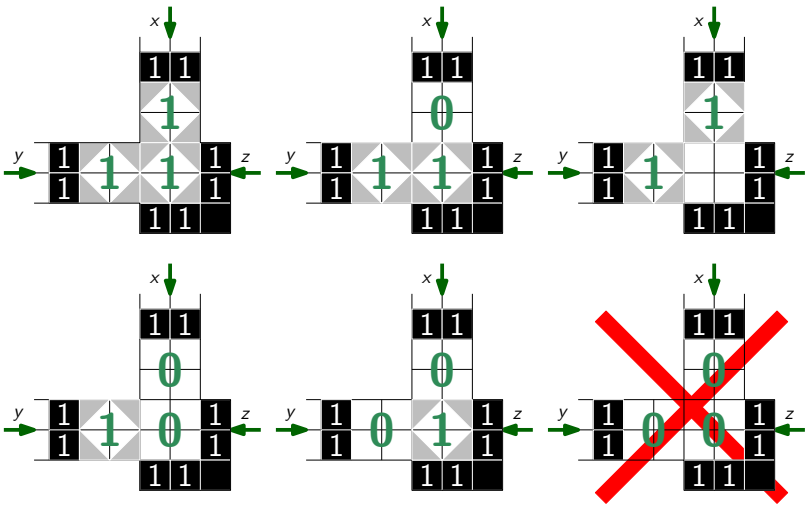

Figure 6: All possibilities of the clause gadget

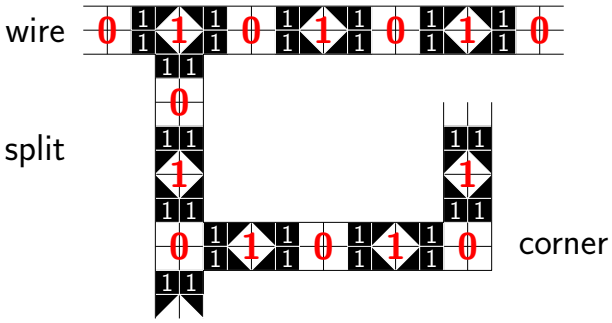

Figure 4: Split and corner gadgets

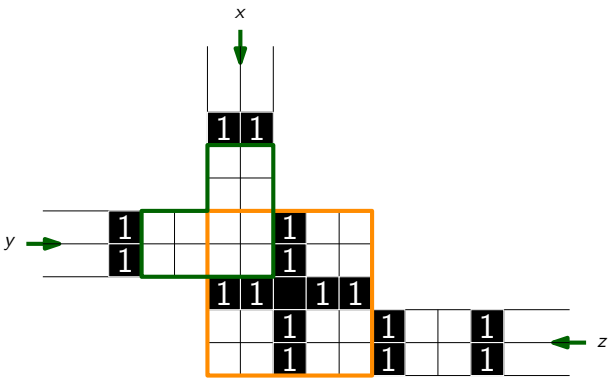

Figure 5: Clause gadget

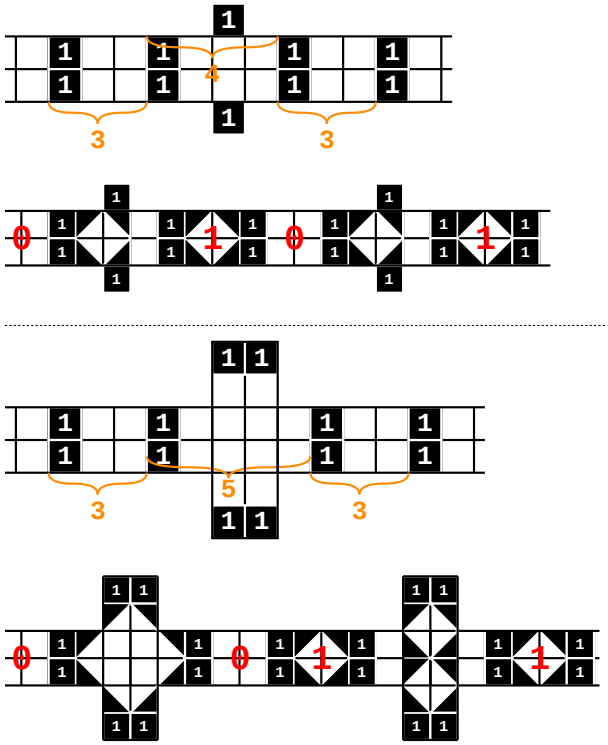

Figure 7: Parity gadgets

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 3×3 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-

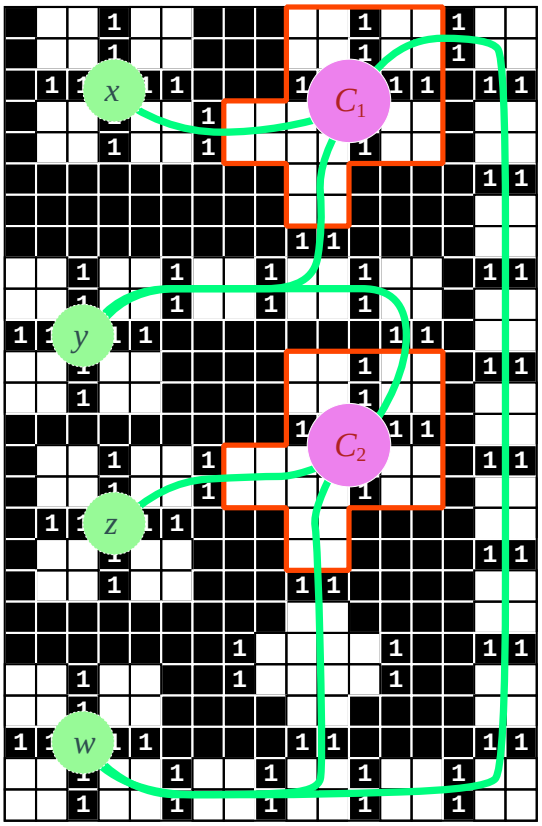

Figure 8: An example for $C_1 = \{x, \bar{y}, w\}$ and $C_2 = \{y, \bar{z}, \bar{w}\}$

ing one of this gadgets in a straightforward way, we can change the position of a wire arbitrarily (Figure 7 (right)). An example of a construction of Shakashaka for the instance $f = C_1 \vee C_2$, where $C_1 = \{x, \bar{y}, w\}$ and $C_2 = \{y, \bar{z}, \bar{w}\}$ is depicted in Figure 8.

The reduction can be done in polynomial time, and it is clear that Shakashaka is in the class NP. Therefore, Shakashaka is NP-complete. We note that our reduction is parsimonious, i.e., preserves the number of solutions. That is, the number of the original CNF is equal to the resulting instance of Shakashaka. Since the counting version of the planar 3SAT is #P-complete [10], we have the following corollary.

Corollary 2 *The counting version of generalized Shakashaka is #P-complete.*

4 Formulation of Shakashaka and experimental results

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):

$$\begin{cases} x[i,j,WH] = 1 & \text{means that } (i,j) \text{ remains white,} \\ x[i,j,NW] = 1 & \text{means that } (i,j) \text{ is filled with } \blacksquare, \\ x[i,j,NE] = 1 & \text{means that } (i,j) \text{ is filled with } \blacksquare, \\ x[i,j,SW] = 1 & \text{means that } (i,j) \text{ is filled with } \blacksquare, \\ x[i,j,SE] = 1 & \text{means that } (i,j) \text{ is filled with } \blacksquare. \end{cases}$$

We construct a linear system $S(I)$ with the variables $x[i, j, *]$ 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:

$$\begin{aligned} x[i,j,WH] + x[i,j,NW] + x[i,j,NE] \\ + x[i,j,SW] + x[i,j,SE] = 1 \end{aligned} \quad (1)$$

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 $\blacksquare$, $\blacksquare$, $\blacksquare$, $\blacksquare$, 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 $\blacksquare$ or $\blacksquare$, we have 45° white area between $(i-1, j)$ and (i, j). Thus $(i-1, j)$ must be $\blacksquare$, $\blacksquare$, or leaving white. Hence, in this case,

$$\begin{aligned} x[i-1,j,WH] + x[i-1,j,NW] + x[i-1,j,NE] \\ + x[i-1,j,SW] + x[i-1,j,SE] = 1 \end{aligned} \quad (2)$$

can be replaced by

$$x[i-1,j,WH] + x[i-1,j,SW] + x[i-1,j,SE] = 1 \quad (3)$$

and we can fix $x[i-1, j, NW] = x[i-1, j, NE] = 0$.

On the other hand, when a black square (i, j) has a number k, it must have k w/b squares as its neighbor. This restriction is described by the following equation:

$$\begin{aligned} x[i-1,j,SW] + x[i-1,j,SE] + x[i+1,j,NW] \\ + x[i+1,j,NE] + x[i,j-1,NE] + x[i,j-1,SE] \\ + x[i,j+1,NW] + x[i,j+1,SW] = k, \end{aligned} \quad (4)$$

where $x[i, j, *]$ is regarded as 0 if (i, j) is black. We also fix $x[i-1, j, NW] = x[i-1, j, NE] = x[i+1, j, SW] = x[i+1, j, SE] = x[i, j-1, NW] = x[i, j-1, SW] = x[i, j+1, NE] = x[i, j+1, SE] = 0$ 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 ◺. In the case, the upper triangle can be orthogonal if and only if either $x[i, j+1, SE] = 1$ ($(i, j+1)$ is ◢) or $x[i+1, j+1, SW] = 1$ ($(i, j+1)$ is ◺). Therefore, we obtain the following constraint:

$$x[i, j, SW] \leq x[i, j+1, SE] + x[i+1, j+1, SW]. \quad (5)$$

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 $(i, j+1)$ is ◹, 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:

$$x[i, j, SW] + x[i+1, j+1, SW] \leq x[i, j+1, WH] + 1. \quad (6)$$

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. □

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 $x[i, j, WH] = x[i+1, j, WH] = x[i, j+1, WH] = 1$. Then, $(i+1, j+1)$ must be ◢ or must remain white. Thus we add the following constraints (for all possible directions):

$$x[i, j, WH] + x[i+1, j, WH] + x[i, j+1, WH] \leq x[i+1, j+1, WH] + x[i+1, j+1, SE] + 2. \quad (7)$$

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.*

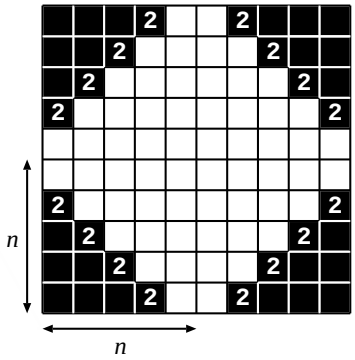

Figure 9: An attificial example of the puzzle Shakashaka of size n.

Constraint E (Exclusion of Nested White Rectangles): The last problem is that the linear system so far may produce nested rectangles. We suppose that both of (i, j) and $(i+k, j+k)$ are ◪. Then, to avoid the nest, we have to have ◢ between them. That is, we have to have ◢ at $(i+k', j+k')$ for some $0 < k' < k$. And it is not difficult to see that this is a necessary and sufficient condition to avoid nested rectangles. This observation gives us the following constraint:

$$x[i, j, NW] + x[i+k, j+k, NW] \leq \sum_{0<k'<k} x[i+k', j+k', SE] + 1. \quad (8)$$

Combining all Propositions and observations above, we conclude the following.

Theorem 6 *Let I be an instance of Shakashaka, and $S(I)$ be the linear system that consists of the constraints above. Then, a feasible solution of $S(I)$ gives a solution of I, and vice versa.*

We here show our experimental results. We used SCIP 3.0.0[1] (Binary: Windows/PC, 32bit, cl 16, intel 12.1: statically linked to SoPlex 1.7.0, Ipopt 3.10.2,CppAD 20120101.3) as an IP solver [11], that runs on a laptop machine (Intel Core2 Duo P8600@2.40GHz with RAM 4GB on Windows Vista Business SP2). Each of the ten instances at `nikoli.com`[2], was solved in one second in our experiments (Table 1).

We also try another instance at `nikoli.com`, which was prepared for a competition. The board has size 31 × 45, the level is Extreme, and the number of white squares is 1230. A solution was obtained in 2.63 seconds.

The other examples are artificial ones (see Figure 9); for each $n = 1, 2, \ldots$, the board of size $2n \times 2n$ consists of $4 \times \sum_{i=1}^{n-1} i = 2n(n-1)$ black squares, and $4 \times (n-1)$

[1] `http://scip.zib.de/`

[2] `http://www.nikoli.com/ja/puzzles/shakashaka/`

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

Table 1: Experimental results for the instances at `nikoli.com`

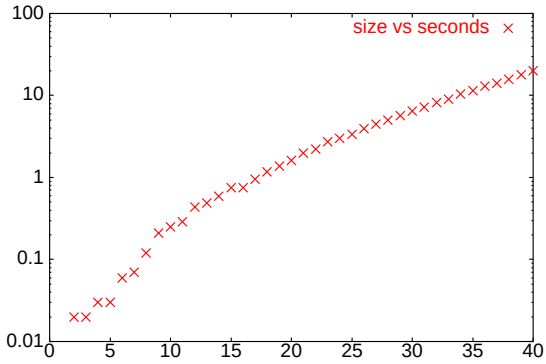

Figure 10: Seconds for the artificial examples ($n = 2, 3, \ldots, 40$).

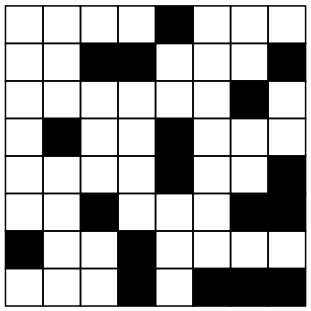

Figure 11: An instance of Shakashaka without number

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

[1] Nikoli, Shakashaka 1, vol.151, Pencil and Paper Puzzle Series, Nikoli, Jan. 2012.

[2] R.A. Hearn and E.D. Demaine, Games, Puzzles, and Computation, A K Peters Ltd., 2009.

[3] R. Bixby, "Presolve for Linear and Mixed-Integer Programming," Proc. of 24th RAMP Symposium, pp.193–200, 2012.

[4] A. Bartlett, T.P. Chartier, A.N. Langville, and T.D. Rankin, "Integer Programming Model for the Sudoku Problem," J. of Online Mathematics and its Applications, vol.8, Articule ID 1798, 2008.

[5] R.A. Bosch, "Painting by Numbers," Optima, vol.65, pp.16–17, 2001.

[6] M.J. Chlond, "Classroom Exercises in IP Modeling: Su Doku and the Log Pile," INFORMS Transactions on Education, vol.5, pp.77–79, 2005.

[7] W.J.M. Meuffles and D. den Hertog, "Puzzle—Solving the *Battleship* Puzzle as an Integer Programming Problem," INFORMS Transactions on Education, vol.10, no.3, pp.156–162, 2010.

[8] L. Mingote and F. Azevedo, "Colored Nonograms: An Integer Linear Programming Approach," Proceedings of EPIA 2009 LNAI vol. 5816, pp.213–224, Springer-Verlag 2009.

[9] D. Lichtenstein, "Planar Formulae and Their Uses," SIAM J. on Computing, vol.11, no.2, pp.329–343, 1982.

[10] N. Creignou and M. Hermann, "Complexity of generalized satisfiability counting problems," Information and Computation, vol.125, pp.1–12, 1996.

[11] T. Achterberg, "SCIP: Solving Constraint Integer Programs," Mathematical Programming Computation, vol.1, pp.1–41, 2009.