The Computational Complexity of the KAKURO Puzzle, Revisited

Oliver Ruepp[1] and Markus Holzer[2]

[1] Institut für Informatik, Technische Universität München, Boltzmannstraße 3, D-85748 Garching bei München, Germany
ruepp@in.tum.de

[2] Institut für Informatik, Universität Giessen, Arndtstraße 2, D-35392 Giessen, Germany
holzer@informatik.uni-giessen.de

Abstract. We present a new proof of NP-completeness for the problem of solving instances of the Japanese pencil puzzle KAKURO (also known as Cross-Sum). While the NP-completeness of KAKURO puzzles has been shown before [T. Seta. The complexity of CROSS SUM. *IPSJ SIG Notes*, AL-84:51–58, 2002], there are still two interesting aspects to our proof: we show NP-completeness for a new variant of KAKURO that has not been investigated before and thus improves the aforementioned result. Moreover some parts of the proof have been generated automatically, using an interesting technique involving SAT solvers.

1 Introduction

Pencil puzzles have gained considerable popularity during recent years. The arguably most prominent example is the game of Number Place (jap. *Sudoku*), but there are also many other logic puzzles, which are especially popular in Japan. Here, we are going to take a closer look at the so-called KAKURO puzzle.

KAKURO (also known as Cross-Sum) is a pencil puzzle that could be described as the mathematical version of a crossword puzzle. The difference to the crossword puzzle is that in a KAKURO puzzle, we have to fill numbers into the empty fields instead of letters, and the hints that are used to describe words in a crossword puzzle are also replaced by numbers, namely the sum of the corresponding number sequence.

For a computer scientist, pencil puzzles are especially interesting from the computational complexity point of view. The probably most basic problem is finding a solution for a given puzzle, and in most cases, the corresponding decision problem ("is there a solution?") turns out to be NP-complete. Here NP denotes the class of problems solvable in polynomial time on a nondeterministic Turing machine. To our knowledge, the first result on pencil puzzles is due to Ueda and Nagao [9], who showed that Nonogram is NP-complete. Since then, a number of other pencil puzzles have been analyzed and also found to be NP-complete, too, e.g., Corral [2], Cross-Sum (jap. *Kakkuro*) [7,8], Fillomino [11], Heyawake [4], Slither Link [11], Sudoku [11], to mention a few.

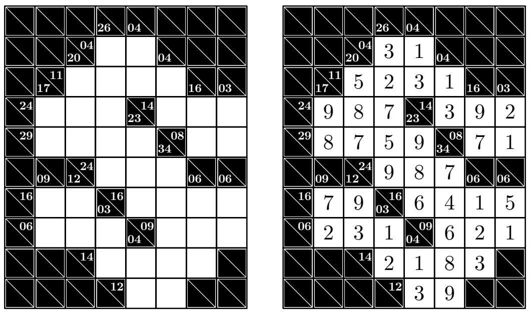
More formally, a Kakuro puzzle is played on a finite, rectangular grid that contains blank cells and black cells. The objective is to fill numbers into the blank cells, according to the following rules:

1. A sum is associated with every horizontal or vertical sequence of white cells. Only maximal sequences are considered, i.e., sequences that do not have a horizontal/vertical neighboring blank cell at either end.
2. Sequences have a minimum length of 2.
3. Only the numbers $1, 2, \ldots, 9$ can be placed into the blank cells.
4. Each horizontal respectively vertical sequence has a black cell left of resp. above its first cell, and that black cell contains as hint the sum that is associated with the sequence. Black fields may contain $0, 1$ or 2 hints.
5. In each horizontal/vertical sequence of cells, every number may occur at most once.
6. The sum of the numbers of a sequence must equal the number that is denoted in the corresponding hint.

An example of a Kakuro puzzle is shown in Figure 1. The NP-completeness of solving these puzzles has originally been shown in [7,8]. The reduction developed there is rather complicated, and derived from a very special problem. In this work, we will instead show an alternative reduction that is based on the problem of planar 3SAT, which is well-known to be NP-complete [5], and constitutes a slight improvement to the original proof. We are going to prove the following theorem:

Theorem 1. *Deciding whether a* Kakuro *puzzle has a solution or not is* NP-*complete, even if the sequence of white cells is at most of length* 4.

We assume the reader to be familiar with the theory of NP-completeness as contained in, e.g., [3].



(a) Example puzzle (b) Solution

Fig. 1. Kakuro example puzzle with its solution

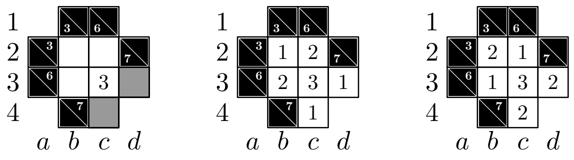
2 Kakuro Is Intractable

To prove Theorem 1, we have to show that the Kakuro problem is contained in NP, and that it is NP-hard. Containment in NP is immediate, since we can guess an assignment of numbers to empty cells and verify that the result is correct.

To show NP-hardness, we will emulate a planar 3SAT circuit using wires, input nodes, NOT gates, and OR gates, and there will also be some supporting gadgets like the phase shifting gadget, a pattern inverter, and a signal shifter. In the gadget diagrams, we will use a chess-like coordinate system to refer to specific cells. Whenever there is a number that is predetermined because of the KAKURO rules, that number will be preinserted into the diagram.

Figure 2 shows the input node gadget. The actual information generated in the gadget is the position of the 1's and 2's, and all in all, there are only two solutions: the sum 6 in a three cell sequence can only be built by summing up the numbers $1, 2, 3$, so we know that the 3 must occur somewhere in the fields $b3, c3, d3$ and somewhere in the fields $c2, c3, c4$. But there can be no 3 at $c2$, because the horizontal sum 3 can only be the result of summing 1 and 2, and the analogous argumentation shows that the 3 is not at $b3$ either. The sum 7 in a sequence of three fields can only be built by summing up the numbers $1, 2, 4$, so no 3 is allowed in the corresponding sequences, thus the 3 cannot be placed at $d3$ and $c4$, and the only position that is left for the 3 is then $c3$, as shown. The alternative solution is created by exchanging the 2's and 1's.

The "output" of this gadget is located at the lower right corner, the corresponding *interface fields* are shown as gray shaded boxes. Gadgets will be connected to each other such that the interface fields overlap. All of our gadgets can be rotated arbitrarily in steps of 90 degrees, as well as mirrored diagonally.

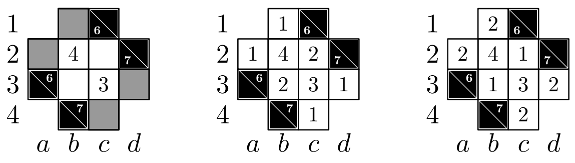


(a) Input node (b) "true" (c) "false"

Fig. 2. The input node (a), with its two solutions (b) and (c)

The wire gadget is depicted in Figure 3. It simply transports the alternating pattern of 1's and 2's. Other gadgets will be attached to both ends such that the interface fields overlap, which explains where the number 4, which is apparently neither defined by horizontal nor vertical hints, comes from: we assume that a horizontal respectively vertical sum of value 7 is defined for the corresponding sequences, and all of our gadgets will be designed such that this will be the case. As an example, imagine how this would work out in a combination of an input node gadget and a wire gadget. Information flow is from the top left corner to the lower right corner. Rotated versions of the gadget will obviously work as well.

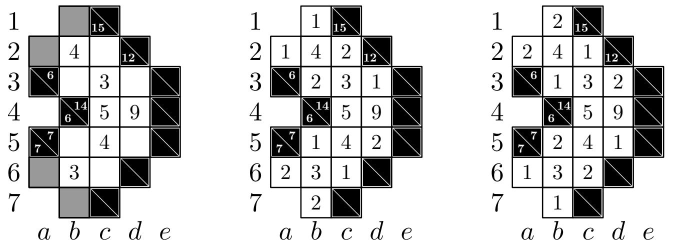
Figure 4 shows the bending device, which allows us to change the direction of signals by 90 degrees. Note that the output of the gadget is inverse to the input, which is an unwanted side effect, but not really a problem since we can simply attach a negation gadget to restore the original signal. The predefined



(a) Wire gadget (b) “true” (c) “false”

Fig. 3. The wire gadget (a), and its two solutions (b) and (c) corresponding to the two different boolean values, respectively

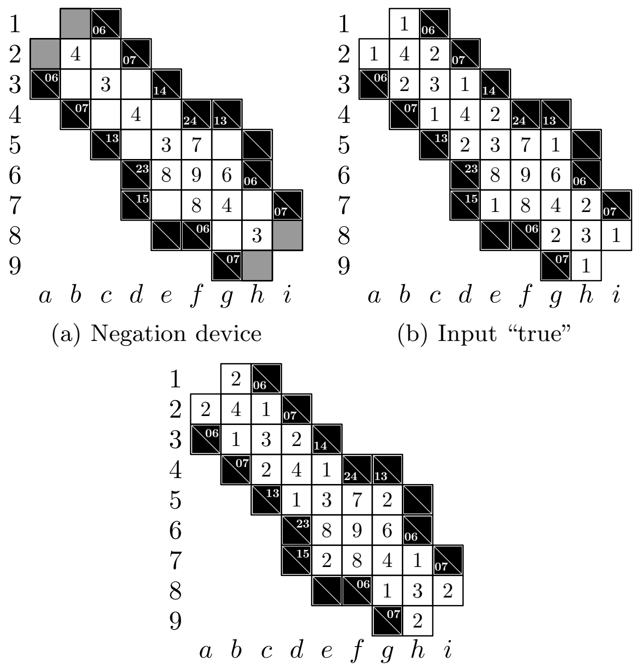
numbers in the diagram can be explained as follows: the horizontal sum 15 which is defined at $c1$ is the smallest possible sum for 5 cells and can only be the result of summing the numbers from 1 to 5. Thus, the highest possible number in this vertical sequence is a 5. Since the horizontal sum defined at $b4$ is 14, we must place this 5 at $c4$, because otherwise there would be a value larger than 10 in the field at $d4$. This also determines the 9 at $d4$, which means that there can only be the values 1 and 2 at $d3$ and $d5$, which in turn determines the 3 and 4 at $c3$ and $c5$.



(a) Bending gadget (b) “true” (c) “false”

Fig. 4. The bending gadget (a) and its two solutions (b) and (c)

The mentioned negation gadget is shown in Figure 5. There are lots of predetermined numbers in the diagram, and for a reader without KAKURO experience it might not be clear why this is the case, so we will explain it in detail. The 4’s and 3’s at $b2$, $c3$ and $h8$ should need no further explaining. The next interesting value is the 7 at $f5$. The reasoning goes as follows: the vertical sum that is defined there is 24, which is the highest value that can the result of summation over three fields, and thus must be the sum of $9, 8$ and 7. But it is not possible to place a 9 or a 8 at $f5$, because then the remaining horizontal sum would be reduced to 4 or 5, and it is not possible to have a sum of such small value in three fields, as the lowest possible sum is $1 + 2 + 3 = 6$. With analogous reasoning, we can determine the numbers at $e6, f6, g6$ and $f7$. The 3 and 4 at $e5$ and $g7$ are forced because the remaining horizontal and vertical sums are 6 and 7, thus the previous argument about the other predefined 3’s and 4’s can be applied. Note that even though the values in the interface fields take on the same values, the pattern of 1’s and 2’s is shifted, which is what we actually wanted to achieve.


(a) Negation device

(b) Input "true"

(c) Input "false"

Fig. 5. The negation gadget (a) with its solutions (b) and (c)

There is one problem with the negation gadget: the alternating pattern of 3's and 4's that occur in the middle of our gadgets is shifted by 1, and this might be problematic when we try to combine gadgets. What we need here is a phase shifter gadget, and this is shown in Figure 6. The gadget transports the alternating pattern of 1's and 2's without modifying it. The 4 at *b*2 is enforced because there has to be a 4 at *b*1, *b*2 or *b*3 (recall that we can assume that a vertical sum of 7 is defined there), but it cannot be at *b*2 (a horizontal sum of value $6 = 1 + 2 + 3$ will be defined there) and not at *b*4, so the only possible position is *b*3. The next predefined field is *c*3 with a value of 3. That 3 cannot be at *d*3, because then the horizontal sum of 10 of the vertical sequence would be reduced to 7, which is the lowest number that can possibly appear at *d*5 because of the horizontal sum $24 = 9 + 8 + 7$ there. But this would mean that we would have to put a 0 at *d*5, which is not allowed. The 7 at *d*4 is determined because if we would place a 9 or 8 there, the corresponding vertical sum would be reduced to 2 or 1, and such a small value is obviously not possible as the result of summation in the two remaining fields. The argumentation for the other predefined fields is completely analogous.

One of the most important—and usually also most complicated—gadgets needed in a proof like ours is the splitter gadget, which creates a copy of an input signal. The authors tried quite hard to manually create a puzzle that served this purpose, but did not succeed. However, having a very fast KAKURO

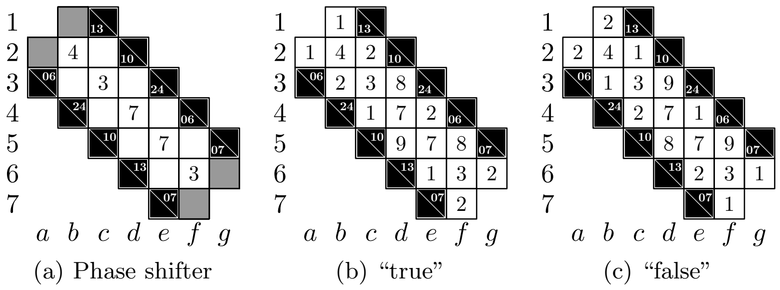


Fig. 6. The phase shifter gadget (a) and the two possible solutions (b) and (c)

solver at disposal (this solver will be described in the latter part of the paper), it was possible to do an exhaustive search for a puzzle that fulfills the desired properties, and this search was indeed successful. The result is shown in Figure 7(a). Note that the output of the puzzle is "inverted," in that the pattern no longer consist of 1's and 2's, but 8's and 9's instead. But it is easy to transform this modified pattern back into the standard one, and this will be explained later.

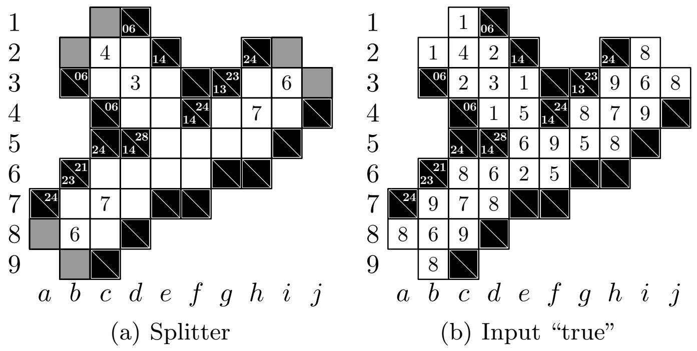
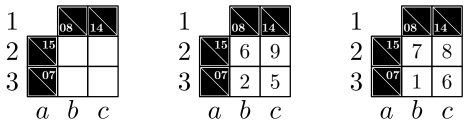


Fig. 7. The splitter gadget (a) duplicates its input. An example is shown in (b).

As usual, we begin by explaining the predetermined numbers in the puzzle. The 4 at $c2$ should be clear, the argumentation is the same as in the previous cases. At $d1$, a vertical sum of value 6 is defined, which means that a 3 must appear either at $d3$ or $d4$. But it cannot be at $d4$, because the vertical sum defined there is 6, which means that another 3 at $e4$ would be enforced, which is not allowed. Analogous argumentation shows that the 7 belonging to the sum 24 that is defined at $a7$ can only be at $c7$. Determining the position for the 7 belonging to the horizontal sum defined at $f4$ is a little bit more complicated: there is no obvious reason why it cannot be at $g4$, so we have to fill out the puzzle until a conflict arises. If we assume the 7 is placed at $g4$, then we would have a 6 at $g5$ and another 7 at $h5$, which already makes a sum of 13, and the remaining value of the horizontal sum there is $28 - 13 = 15$. But 15 can only be the result of $9 + 6$ or $8 + 7$, and both 6 and 7 already occur at $g5$ resp. $h5$, which means that it is not possible to solve the puzzle any more, and so the 7 has to be at $h4$.

By now, we already know a lot about this sub-puzzle: the sum of fields $e3$ and $e4$ will be 6, thus the remaining horizontal sum for fields $e5$ and $e6$ is $14 - 6 = 8$. For the fields $e5$ and $f5$, we have a sum of $28 - 13 = 15$ left, because the numbers at $g5$ and $h5$ will always build the sum 13. With analogous reasoning, we see that the fields $e6$ and $e7$ must contain a sum of $21 - 14 = 7$. All in all, we have a sub-puzzle in the fields $e5, f5, e6, f6$ that corresponds to the puzzle shown in Figure 8.



(a) Core puzzle (b) Solution 1 (c) Solution 2

Fig. 8. The center of the splitter gadget (a), and its two solutions (b) and (c)

It is easy to see that this puzzle has exactly two solutions: the vertical sum 14 defined at $c2$ means that the value at $c3$ is at least 5, but the horizontal sum 7 also tells us that the value cannot be bigger than 6. Looking at cell $e3$ in the splitter puzzle, we see that there is either a 1 or a 2, and this obviously determines the solution chosen in the center of the gadget because of Rule 5. This solution in turn determines the numbers chosen at $h5$ and $d6$ (in the original gadget), effectively synchronizing all values. Thus, we have shown that the splitter gadget performs as desired. One example of the full solution is shown in Figure 7(b).

What we still need now is some gadget to transform the 8/9 output pattern into a corresponding 1/2 pattern. In addition to that, we need to shift the pattern vertically or horizontally by one cell, to assure that it can be connected to other gadgets. A gadget that has this effect is shown in Figure 9. The diagram shows a gadget that shifts the signal down vertically by one cell, but by mirroring it on the diagonal we can obtain a gadget that shifts the signal right by one cell. As usual, rotated versions of the device will also work.

The number 7 at $b2$ can be explained with standard reasoning, as well as the 6 at $c3$. The first more difficult number is the 7 at $d4$: because of the horizontal sum of 24 defined there, a 7 has to appear either at $d4$ or $e4$. But it cannot be at $e4$, because then, there would be a 9 or 8 at $d4$, and we already know that there will be a 8 resp. 9 at $d3$, and this means that the sum of fields $d3$ and $d4$ would already be 17, leaving 0 for $d5$, which is of course not allowed. With the 7 fixed at $d4$, we also know that there can only be the values 1 or 2 at $d5$, and this in turn determines the 3 at $e5$. The 7 at $h8$ is determined there because the horizontal sum that is defined there can only be the result of adding 9 and 7, but if the 9 were placed here, the vertical sum would be reduced to 5, which is too small for the three remaining fields. This also forces the 9 at $g8$. The remaining 3's and 4's should need no explicit explanation, since the can all be explained with the standard argumentation.

A puzzle that emulates an OR-gate is shown in Figure 10. Its basic layout is identical to that of the splitter gadget, and it was found using the same

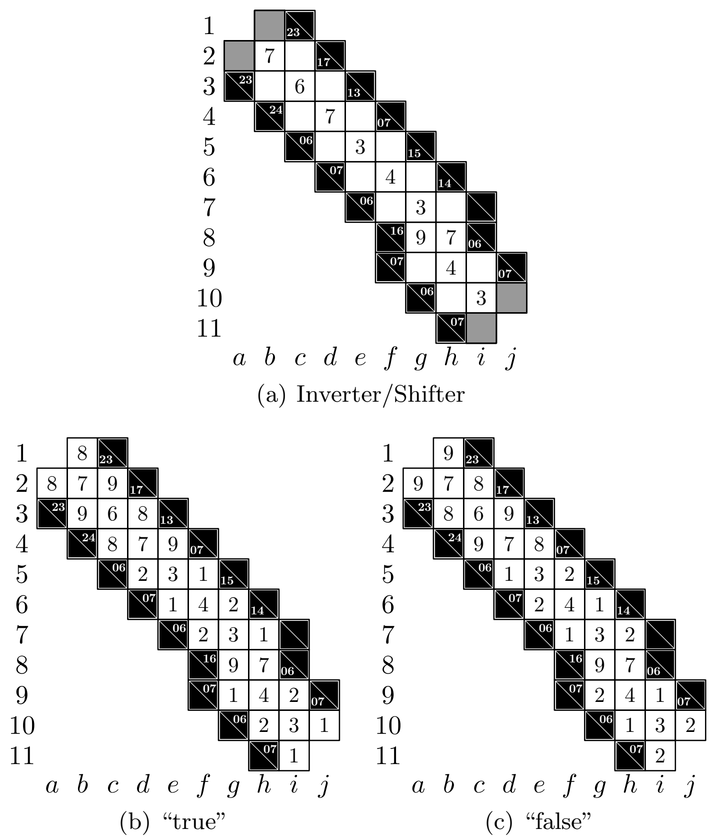


Fig. 9. The inverter/shifter gadget (a) and solutions (b) and (c)

exhaustive search method. The device does not produce an output signal that corresponds to the OR of two input signals, but instead it takes three input signals and enforces that not all of these signals are “false.”

The predefined numbers at $c7$ and $h4$ should need no further explaining. The placement of the 3 at $d3$ is not obvious, it could also be placed at $e3$ without a direct resulting conflict. In that case however, a 4 would be enforced at $e4$, and the vertical sum so far is 7, which leaves $12 - 7 = 5$. But 5 can only be partitioned as $5 = 1 + 4$ or $5 = 2 + 3$, and both 3 and 4 already occur in the vertical sequence, so this is where the conflict occurs.

As in the case of the splitter gadget, a sub-puzzle in the fields $e5, f5, e6, f6$ emerges, and this sub-puzzle is shown in Figure 11(a). It is easy to derive the three shown solutions to this gadget: there are 4 possibilities to partition the 6 that is defined at $a2$: $1 + 5$, $2 + 4$, $4 + 2$, $5 + 1$. By simply trying all of these, we see that a 1 is impossible at $c2$, which prevents a solution in the last case, while all the other cases lead to a valid solution.

To prove that the gadget works as described we would need to verify that there is no solution if all the input patterns are “false,” and that there is a

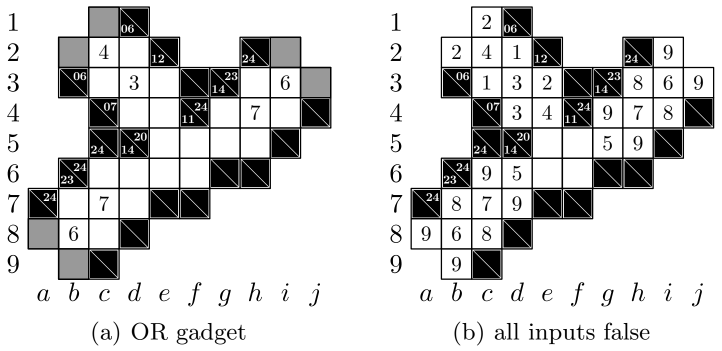


Fig. 10. The OR gadget (a), and the situation when all inputs are false (b)

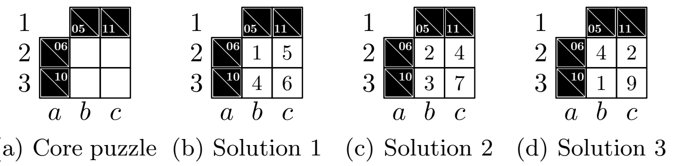


Fig. 11. The core puzzle of the OR gadget (a) has three solutions (b), (c), and (d)

solution in all other cases. This is a cumbersome exercise, so we will show the former property and one example for the latter property.

Now let us assume that all input values are "false," which means that the number 2 resp. 9 appears in the interface fields. The situation is shown in Figure 10(b). When we fill out the puzzle, there will be a 2 at $e3$, which means that only the solutions 1 and 3 shown in Figure 11 are possible for the core puzzle. But there will also be a 5 at $g5$, which means that solution 1 is also not possible, and solution 3 will also not be possible since there will be a 9 at $c6$.

Next, let us see what happens if we change the upper left input to "true," in which case a solution should be possible. Then there would be a 1 at $e3$, which means that we have to choose solution 2 for the core puzzle. But with this core puzzle solution, there will be no restrictions on the other two input values, because, e.g., at $g5$ and $h5$ the only possible values are 5 and 9 or 6 and 8, neither of which conflicts with the core puzzle solution. This means that arbitrary input values are allowed at the upper right corner, and with analogous reasoning, we can show that the same holds for the lower left input. Thus, we have already shown that all input combinations where the upper left input is "true" are valid solutions. Showing that all remaining input combinations lead to valid solutions is done completely analogous.

It is clear now that we have developed all the gadgets that are needed to emulate arbitrary planar 3SAT instances, and thus we have proven the first part of Theorem 1. Moreover, the result also constitutes an improvement to the original proof in [7,8], in that the maximum sequence length of white cells needed is 4 instead of 5. This can be achieved by replacing the bending gadget with a splitter gadget and simply attaching an input node to the superfluous output, effectively discarding the unneeded signal. This finally proves the second part of our main result.

The reduction developed herein is parsimonious, which means that we have also proven a stronger result: Checking a Kakuro puzzle for unique solvability is DP-complete under randomized polynomial time many-one reductions by employing Valiant and Vazirani's [10] result on unique satisfiability. Here DP denotes the class of differences of any two NP sets [6]—note that DP is equal to the second level of the Boolean Hierarchy over NP.

3 Automatic Gadget Generation

As has been mentioned before, some of the gadgets that have been used for the proof have been generated automatically, using an efficient Kakuro solver. That solver actually translates the Kakuro instance into a SAT instance, and applies a SAT-solver to the result, in our case Minisat [1]. That principle has become a standard approach in recent years, and SAT-solvers as well as constraint-based solvers have been used to tackle many pencil puzzles.

What we have done now to find gadgets that implement certain functionalities is the following: we knew how we would like our signals to be transmitted, i.e., the shape of the wire gadgets was already known. Furthermore, we knew that all of the gadgets we were looking for (splitter, OR) have three wire gadgets connected, so we tried to find a shape that connects three wires and is as small as possible. Figure 12 shows the puzzle shape we came up with.

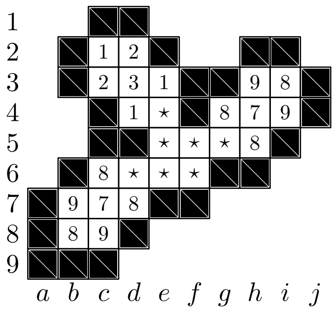


Fig. 12. The search pattern used for the OR gadget

The actual search then worked like this: the fields containing a $\star$ symbol have been used as wildcards, which means that we simply exhaustively generated all possible puzzles by filling in numbers greater or equal to 1 into the fields containing the $\star$ symbol. Note that the input to the puzzle solver consists only of the sums of number sequences, not of the numbers themselves. Thus, even though the search mask shown in Figure 12 looks as though some numbers were fixed, they actually are not, and there might be solutions where the numbers turn out differently.

So after generating all the puzzles by filling in numbers, we ran the KAKURO solver on each of them, and had the solver find alternative solutions. The first step then was filtering out all puzzles that did not have the correct total number of solutions, which is 7 for the OR gate. It is clear that the OR gadget must have exactly 7 solutions, since all possible input value assignments are allowed except for all input values being “false.” After filtering out merely by number of solutions, 2472 potential candidates for the OR gadget were left. For the puzzles that had 7 solutions, we then looked at the values of those solutions at the critical positions $c2$, $i3$ and $b8$, sorting out puzzles that had redundant solutions. Thus, we were able to find not only one, but a handful of puzzles that fulfilled the requirements, and we chose one of those puzzles as our gadget.

Automatically finding gadgets for the splitter device was done analogously. The program source codes used will be made available upon request.

4 Conclusion

In this paper we have developed a new proof showing that the pencil puzzle KAKURO is NP-complete. Though the result was known before, our proof is still interesting, because it is based on a reduction from a better known problem, and also because some parts of the proof have been generated automatically. Furthermore, we were able to show that the KAKURO puzzle remains NP-complete if only sequences of white cells of length at most 4 are used. It is not clear how the complexity of the problem is affected if number sequences have length at most 3, but observe, that Seta has shown in [7] that KAKURO puzzles with sequences that have length at most 2 can be efficiently solved in linear time.

References

1. Eén, N., Sörensson, N.: An extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 502–518. Springer, Heidelberg (2004)
2. Friedman, E.: Corral puzzles are NP-complete. Technical report, Stetson University, DeLand, FL 32723 (2002)
3. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1990)
4. Holzer, M., Ruepp, O.: The troubles of interior design-a complexity analysis of the game heyawake. In: Crescenzi, P., Prencipe, G., Pucci, G. (eds.) FUN 2007. LNCS, vol. 4475, pp. 198–212. Springer, Heidelberg (2007)
5. Lichtenstein, D.: Planar formulae and their uses. SIAM Journal on Computing 11(2), 329–343 (1982)
6. Papadimitriou, C.H., Yannakakis, M.: The complexity of facets (and some facets of complexity). Journal of Computer and System Sciences 28(2), 244–259 (1984)
7. Seta, T.: The complexities of puzzles, cross sum and their another solution problems (asp). Senior thesis, Univerity of Tokyo, Deptartment of Information Science, Faculty of Science, Hongo 7-3-1, Bunkyo-ku, Tokyo 113, Japan (February 2001)
8. Seta, T.: The complexity of CROSS SUM. IPSJ SIG Notes, AL-84, 51–58 (2002) (in Japanese)

9. Ueda, N., Nagao, T.: NP-completeness results for NONOGRAM via parsimonious reductions. Technical Report TR96-0008, Department of Computer Science, Tokyo Institut of Technology, Ôokayama 2-12-1 Meguro Tokyo 152-8552, Japan (May 1996)
10. Valiant, L.G., Vazirani, V.V.: NP is as easy as detecting unique solutions. Theoretical Computer Science 47(3), 85–93 (1986)
11. Yato, T.: Complexity and completeness of finding another solution and its application to puzzles. Master's thesis, Univerity of Tokyo, Deptartment of Information Science, Faculty of Science, Hongo 7-3-1, Bunkyo-ku, Tokyo 113, Japan (January 2003)