

EVOLOMINO IS NP-COMPLETE

A.V. NIKOLAEV *Communicated by P.P. PETROV*

Abstract: Evolomino is a pencil-and-paper logic puzzle popularized by the Japanese publisher Nikoli (like Sudoku, Kakuro, Slitherlink, Masyu, and Fillomino). The puzzle’s name reflects its core mechanic: the shapes of polyomino-like blocks that players must draw gradually “evolve” in the directions indicated by pre-drawn arrows. We prove, by reduction from 3-SAT, that the question of whether there exists at least one solution to an Evolomino puzzle satisfying the rules is NP-complete. Since our reduction is parsimonious, i.e., it preserves the number of distinct solutions, we also prove that counting the number of solutions to an Evolomino puzzle is #P-complete.

Keywords: computational complexity, NP-complete, #P-complete, pencil-and-paper logic puzzle, polynomial-time reduction, 3-SAT, parsimonious reduction.

1 Introduction

Nikoli is a renowned Japanese publisher specializing in games and, especially, logic puzzles. Established in 1980, it became prominent worldwide with the popularity of Sudoku. The magazine published under the same name by Nikoli was the first puzzle magazine in Japan, and over the years, many new types of puzzles appeared on its pages.

NIKOLAEV, A.V., EVOLOMINO IS NP-COMPLETE.

© 2025 NIKOLAEV A.V.

The research is supported by the P.G. Demidov Yaroslavl State University Project VIP-016.

Received January, 1, 2023, Published December, 31, 2023.

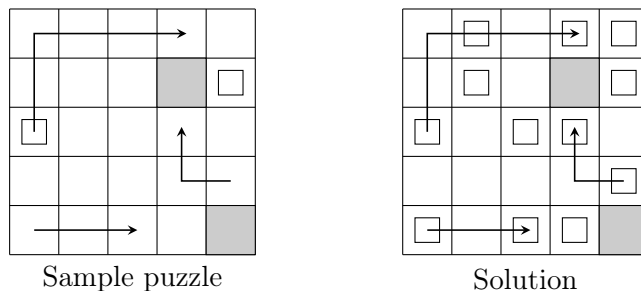


FIG. 1. Example of an EVOLOMINO puzzle

In this paper, we consider a new pencil-and-paper logic puzzle Evolomino that was introduced in the book *768 The Pencil Puzzles 2025* published by Nikoli in 2025 [1].

Here we present the rules of the Evolomino puzzle as they appear on Nicoli's official website [18]:

- Evolomino is played on a rectangular board of white and black cells. Some white cells contain pre-drawn squares and arrows.
- The player must draw squares ($\square$) in some of the white cells.
- A polyomino-like group of squares connected vertically and horizontally is called a *block* (including only one square). Each block must contain exactly one square placed on a pre-drawn arrow.
- Each arrow must pass through at least two blocks.
- The second and later blocks on the route of an arrow from start to finish must progress by adding one square to the previous block without rotating or flipping.

An example of the Evolomino puzzle and its solution from Nicoli's official website [18] is shown in Fig. 1.

The study of the computational complexity of games and puzzles is a rapidly growing area within theoretical computer science. This interest is driven by several factors. On the one hand, from an academic and educational perspective, games and puzzles offer engaging and accessible examples of combinatorial problems, making them valuable tools for teaching computational complexity theory and proof techniques. On the other hand, understanding the complexity of a problem aids in designing efficient algorithms for solving puzzles and in developing winning strategies for combinatorial games involving two or more players.

Many classic games are known to be computationally intractable (assuming $P \neq NP$): one-player puzzles are often NP-complete (as Minesweeper [13], Plumber [16], and Tetris [4]) or PSPACE-complete (as Rush Hour [7]), and two-player games are often PSPACE-complete (as Hex [20] and Reversi [11]) or EXPTIME-complete (as Chess [8] and Go [21]).

In particular, many pencil-and-paper logic puzzles introduced or popularized by Nikoli are NP-complete: Sudoku [27], Kakuro [22, 23], Hashiwokakero [3],

Numberlink [15], Shikaku and Ripple effect [25], Shakashaka [6], Tatamibari [2], and many others. For more results on the computational complexity of games and puzzles, see the surveys by Demaine and Hearn [5] and Kendall et al. [14].

In this paper, we prove that determining whether an Evolomino puzzle has at least one valid solution is NP-complete, via a reduction from 3-SAT. Furthermore, because our reduction preserves the number of solutions, we also establish that counting the total number of solutions to an Evolomino puzzle is #P-complete.

2 Problem statement

Let us begin with a formal definition of the Evolomino puzzle.

EVOLOMINO.

INSTANCE. Given a rectangular board of size $p \times q$, where each cell is colored either white or black. Some of the white cells contain pre-drawn squares ($\square$). The board also includes a set of pre-drawn arrows. Each arrow starts at the center of one cell and ends at the center of another, traversing through the centers of intermediate cells horizontally, vertically or with 90 degree turns. No two arrows may pass through the same cell.

QUESTION. Does there exist at least one solution to the puzzle, i.e., a map from the set of white cells to the set $\{\emptyset, \square\}$ that satisfies the rules of the puzzle:

- each block contains exactly one square placed on the arrow;
- each arrow passes through at least two blocks;
- each subsequent block in the direction of the arrow adds one square to the previous block without rotating or flipping?

We also consider a problem of COUNTING EVOLOMINO, which has the same instance but asks how many distinct solutions the puzzle has.

3 Evolomino is in NP

Lemma 1. $\text{EVOLOMINO} \in \text{NP}$.

Proof. EVOLOMINO is a decision problem, so to prove that it belongs to the class NP, it is sufficient to show that verifying whether a given solution satisfies all the rules of the puzzle is performed in polynomial time.

Let's consider an EVOLOMINO puzzle on a rectangular board of size $p \times q$. If we guess some solution to the puzzle, then checking its correctness will require:

- Rule: "Each block contains exactly one square placed on the arrow."
Since each board cell can belong to no more than one block, we have at most $O(pq)$ blocks. The time required to traverse a single block is linear in its size, which is bounded above by $p \times q$, the size of the entire board. Therefore, the overall time complexity for verifying this rule is $O(p^2q^2)$.

- Rule: “Each arrow must pass through at least two blocks”.
Verifying this rule requires a single pass over each pre-drawn arrow. Since the total length of all arrows is bounded by the number of cells on the board, the overall time complexity for this check is $O(pq)$.
- Rule: “The second and later blocks on the route of an arrow from start to finish must progress by adding one square to the previous block without rotating or flipping”.

We consider a pair of consecutive blocks along the direction of an arrow, both of size at most $O(pq)$. For each such pair, we can attempt to exclude each square of the larger block one by one, then check whether the resulting shape matches the smaller block. This requires traversing both blocks and comparing their structures, resulting in a worst-case complexity of $O(p^2q^2)$ for a pair of two consecutive blocks.

Since the number of consecutive block pairs is bounded by the total length of all pre-drawn arrows, i.e., $O(pq)$, the overall complexity of verifying this rule is $O(p^3q^3)$.

Summing the complexities of verifying all the rules, we conclude that a given solution to the puzzle can be verified in polynomial time $O(p^3q^3)$. Note that this is a fairly rough upper bound, but to prove that $\text{EVOLOMINO} \in \text{NP}$, it is sufficient to be polynomial in the size of the board $p \times q$. $\square$

4 Evolomino is NP-hard

Theorem 1. *EVOLOMINO is NP-hard.*

Proof. To prove that EVOLOMINO is NP-hard, we construct a polynomial-time reduction from NP-complete 3-SAT problem [9, 12] to EVOLOMINO .

3-SAT.

INSTANCE. Given a Boolean formula in conjunctive normal form (CNF), i.e., a collection $C = \{C_1, \dots, C_m\}$ of clauses on a finite set $X = \{x_1, \dots, x_n\}$ of Boolean variables such that $|C_i| = 3$ for all $1 \leq i \leq m$.

QUESTION. Is there a truth assignment for variables X that satisfies all the clauses in C , and therefore satisfies the Boolean formula?

We model each component of 3-SAT with some gadgets on the EVOLOMINO board so that the puzzle has a solution if and only if the corresponding instance of 3-SAT is satisfiable.

Variable and wire gadget. The variable and wire gadget models the truth assignment of a Boolean variable. Its structure is shown in Fig. 2.

Since each block must contain exactly one square placed on an arrow, and each subsequent block along an arrow must progress by adding one square to the previous block without rotating or flipping, the gadget admits only two feasible solutions: two vertical blocks of 1 and 2 squares representing $x = 1$ (Fig. 2 (a)), and two vertical blocks of 3 and 4 squares representing $x = 0$ (Fig. 2 (b)).

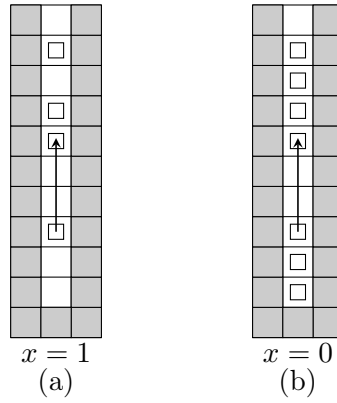


FIG. 2. Variable and wire gadget

Encoding the truth values so that 1 square corresponds to 1 and 3 squares to 0 may seem somewhat counter-intuitive at first, but it will be explained further in the clause gadget. In this framework, we interpret 1 as an “open lock” and 0 as a “closed lock”.

The arrow itself functions as a wire gadget, transmitting the signal from start to finish. To prevent interference from other blocks, we fence each arrow on both sides by black cells. Consequently, each arrow has space for only two blocks: one at the beginning and one at the end of the arrow.

Negation gadget. The negation gadget, which inverts the value of a Boolean variable, is shown in Fig. 3 (a). It consists of two variable gadgets representing x and $\bar{x}$ with a single square positioned between them. Since each block must contain exactly one square placed on an arrow, this intermediate square will belong either to the x -block (Fig. 3 (b)), or to the $\bar{x}$ -block (Fig. 3 (c)) in any valid solution, thereby enforcing the inversion of the signal.

Split gadget. The split gadget shown in Fig. 4 (a) is used to duplicate a signal when a single variable appears in multiple clauses. As before, since each block contains exactly one square placed on an arrow and the block shape is preserved in the direction of the arrow without rotating and flipping, the puzzle admits only two feasible solutions: Fig. 4 (b) for splitting the $x = 1$ signal and Fig. 4 (c) for splitting $x = 0$.

Clause gadget. The structure of the clause gadget for a clause $C = \{x \vee y \vee z\}$ is shown in Fig. 5. Since there is a vertical line block of 5 squares at the end of the horizontal arrow, the puzzle rules require that it be preceded by a vertical line block of 4 squares. It is easy to see that such a block can only be placed in one of the three columns corresponding to the literals x , y , and z . However, if a false signal is received for a literal, then the corresponding column is locked from below, because the same block cannot be placed on two different arrows simultaneously. Therefore, the clause gadget has a solution

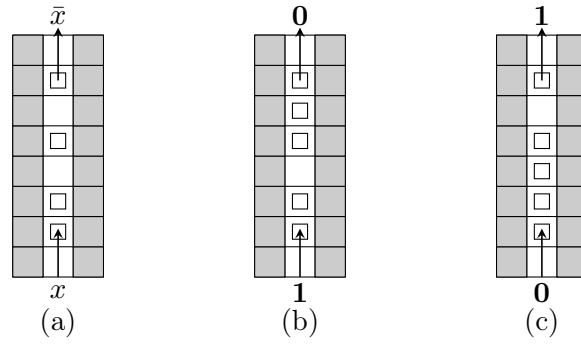


FIG. 3. Negation gadget

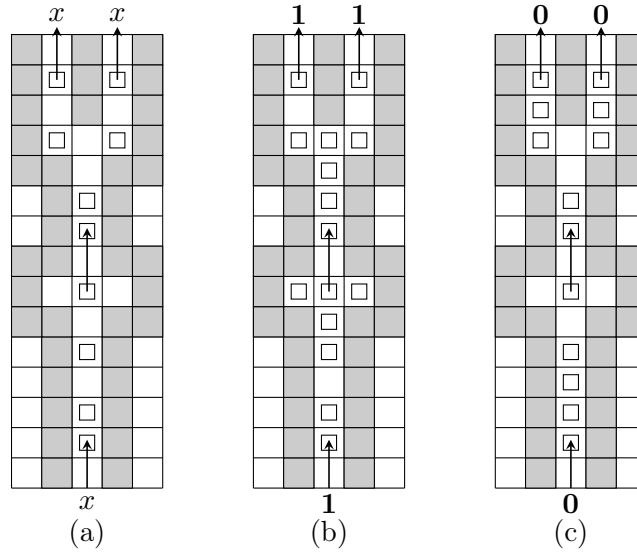


FIG. 4. Split gadget

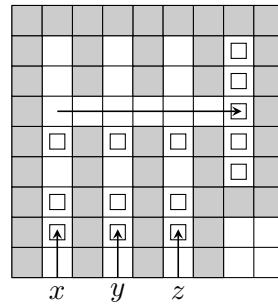


FIG. 5. Clause gadget

if and only if at least one of the literals x , y , or z is satisfied. Examples of feasible cases with one (a), two (b), and three (c) true literals, as well as an infeasible case when all literals are false (d), are shown in Fig. 6.

Crossover gadget. Signals coming from variable gadgets to clause gadgets may cross paths along their routes. To address this issue and ensure correct signal transmission without interference or leakage at wire intersections, we introduce the crossover gadget shown in Fig. 7.

Here, to cross the signals, we use a Γ -shaped polyomino, where the vertical block corresponds to signal x , and the horizontal one to y . The trick is that according to the rules of the EVOLOMINO puzzle, consecutive blocks cannot be rotated in the direction of the arrow: horizontal blocks remain horizontal, and vertical blocks remain vertical. This allows us to preserve the values of the x and y signals, while swapping the order of the wires.

It remains to verify that there are exactly four feasible solutions to the crossover gadget that satisfy the rules of the puzzle, corresponding to the four possible combinations of values for the intersecting signals (see Fig. 8).

Complexity of reduction and board size. Let's consider an instance of the 3-SAT problem that consists of a collection of clauses $C = \{C_1, \dots, C_m\}$ over Boolean variables $X = \{x_1, \dots, x_n\}$.

Let's represent a CNF formula as a bipartite incidence graph with n variables in one part and m clauses in the other. An edge (x, C) indicates that the variable x appears in clause C . Hence, we obtain a bipartite graph with $n+m$ vertices and $3m$ edges, based on which we construct an EVOLOMINO board. An example of an incidence graph of a CNF formula is shown in Fig. 9 with solid edges corresponding to non-inverted variables and dashed edges to inverted variables.

Let's estimate the total number of gadgets required:

- n variable and wire gadgets, one for each Boolean variable $x_1, \dots, x_n$;
- m clause gadgets, one for each clause $C_1, \dots, C_m$;
- $3m - n$ split gadgets, to assign a wire to each edge;
- at most $3m$ negation gadgets, one for each edge, for an inverted variable;
- at most $\binom{3m}{2}$ crossover gadgets, one for each crossing of edges.

Regarding the crossing number, we note that the incidence graph can always be drawn on the rectangular grid so that no three edges are crossing at a single grid cell, and any pair of edges crosses at most once. More precise estimates of the crossing number of a bipartite graph can be obtained from graph theory (see, for example, Turán's brick factory problem [24, 28]).

Since each gadget, except for the wire, has a constant size, an instance of the 3-SAT problem with m clauses and n variables can be reduced to an EVOLOMINO puzzle on a board of size $O(m^2 + n) \times O(m^2 + n)$. As for the wires, they only need to transmit signals from the variable gadgets to

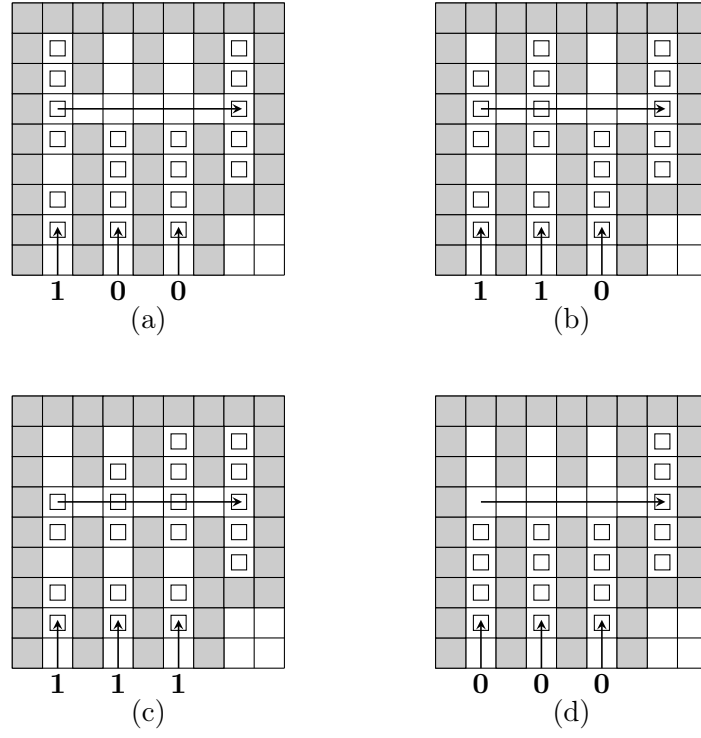


FIG. 6. Three feasible cases (a), (b), (c), and one infeasible case (d) of a clause gadget

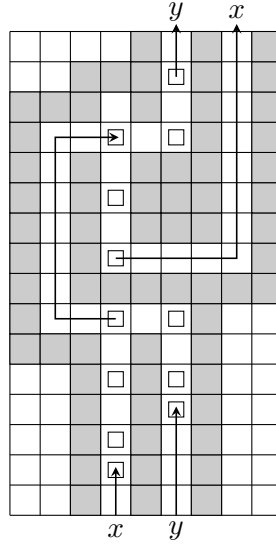


FIG. 7. Crossover gadget

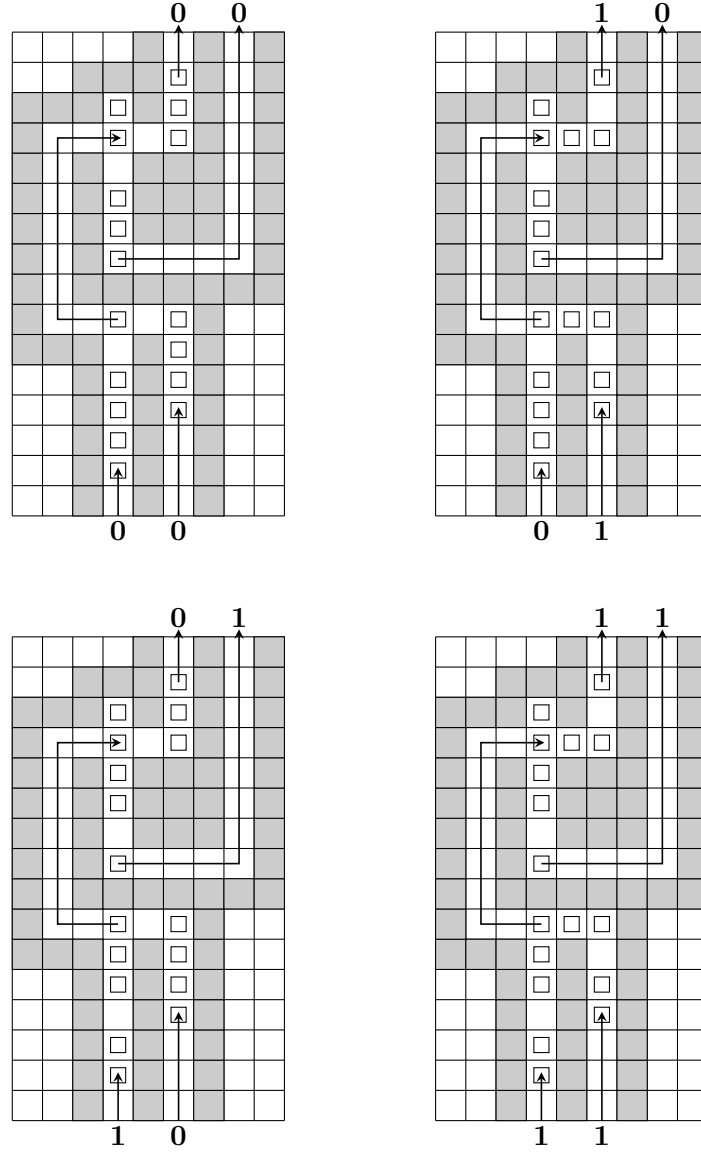


FIG. 8. Four feasible solutions for a crossover gadget

the clause gadgets, so their length is proportional to the overall board size. Therefore, the reduction is performed in polynomial time.

Note that both the board size and the complexity of the reduction can be significantly reduced if we exclude the crossover gadgets and consider the reduction from the NP-complete PLANAR 3-SAT problem [17] where the incidence graph of a Boolean formula can be embedded on a plane, i.e. drawn without edge-crossings. For related results on upper bounds for the area of rectangular grid drawings of planar graphs, see Rahman et al. [19].

However, we decided that if the crossover is possible, it would be correct to consider a more general construction.

An example of constructing the entire puzzle board. An example of constructing the EVOLOMINO board for the 3-SAT instance $C_1 \wedge C_2$, where $C_1 = (x_1 \vee \bar{x}_2 \vee x_3)$ and $C_2 = (x_2 \vee \bar{x}_3 \vee \bar{x}_4)$ is illustrated in Fig. 9.

The board size is 42×21 and includes:

- 4 variable and wire gadgets for x_1, x_2, x_3 , and x_4 ;
- 2 clause gadgets for C_1 and C_2 ;
- 2 split gadgets for x_2 and x_3 ;
- 3 negation gadgets for (x_2, C_1) , (x_3, C_2) , and (x_4, C_2) ;
- 1 crossover gadget for (x_2, C_2) and (x_3, C_1) .

An example of a puzzle solution that satisfies all the clauses and corresponds to the truth assignment $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0$ is shown in Fig. 10.

Completing the proof. It remains to note that all gadgets, except for the clause gadgets, always admit a valid solution, while the clause gadget is solvable if and only if at least one of the literals is true. Thus, the puzzle has a solution if and only if the corresponding instance of the 3-SAT problem is satisfiable. This makes the EVOLOMINO problem NP-hard, and since, by Lemma 1, it also belongs to the class NP, we conclude that EVOLOMINO is NP-complete. $\square$

Moreover, once we fix a truth assignment in an instance of 3-SAT, the filling pattern of the resulting instance of EVOLOMINO is uniquely determined. Thus, our reduction is parsimonious, i.e., it establishes a bijection between the solution sets of the two problems. Consequently, the number of satisfying truth assignments to the original CNF formula equals the number of solutions to the resulting EVOLOMINO puzzle. Since the COUNTING 3-SAT is #P-complete [26], we obtain the following corollary.

Corollary 1. COUNTING EVOLOMINO is #P-complete.

Let us recall that the complexity class #P, introduced by Valiant [26], contains the counting versions of NP decision problems. For more information on #P-complete problems and parsimonious reduction, see the book by Goldreich [10].

5 Conclusion

In this paper, we have proved, via a reduction from 3-SAT, that the EVOLOMINO puzzle is NP-complete. Furthermore, since our reduction is parsimonious and preserves the number of distinct solutions, the corresponding counting problem, COUNTING EVOLOMINO, is also #P-complete.

Evolomino is a relatively new puzzle for which little is known from a theoretical computer science perspective. Several directions for future research appear promising. On the one hand, the development of algorithms for the

general NP-hard case of EVOLOMINO, such as integer linear programming models or backtracking approaches. On the other hand, the NP-completeness result implies only that some instances, in particular those that correspond to the 3-SAT problem, are computationally intractable (assuming $P \neq NP$). It would be worthwhile to investigate additional constraints or special cases under which subproblems of EVOLOMINO become polynomially solvable.

References

- [1] *768 The Pencil Puzzles 2025*. The Pencil Puzzle series. Nikoli, 2025. URL: <https://nikolibookshop.stores.jp/items/66f4704da0789100010f3a6f>.
- [2] A. Adler, J. Bosboom, E. D. Demaine, M. L. Demaine, Q. C. Liu, and J. Lynch. Tatamibari Is NP-Complete. In M. Farach-Colton, G. Prencipe, and R. Uehara, editors, *10th International Conference on Fun with Algorithms (FUN 2021)*, volume 157 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 1:1–1:24, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. DOI: [10.4230/LIPIcs.FUN.2021.1](https://doi.org/10.4230/LIPIcs.FUN.2021.1).
- [3] D. Andersson. Hashiwokakero is NP-complete. *Information Processing Letters*, 109(19):1145–1146, 2009. DOI: [10.1016/j.ipl.2009.07.017](https://doi.org/10.1016/j.ipl.2009.07.017).
- [4] R. Breukelaar, E. D. Demaine, S. Hohenberger, H. J. Hoogeboom, W. A. Kosters, and D. Liben-Nowell. Tetris is hard, even to approximate. *International Journal of Computational Geometry & Applications*, 14(01n02):41–68, 2004. DOI: [10.1142/S0218195904001354](https://doi.org/10.1142/S0218195904001354).
- [5] E. D. Demaine and R. A. Hearn. Playing games with algorithms: algorithmic combinatorial game theory. In M. H. Albert and R. J. Nowakowski, editors, *Games of No Chance 3*, Mathematical Sciences Research Institute Publications, pages 3–56. Cambridge University Press, 2009.
- [6] E. D. Demaine, Y. Okamoto, R. Uehara, and Y. Uno. Computational complexity and an integer programming model of Shakashaka. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E97.A(6):1213–1219, 2014. DOI: [10.1587/transfun.E97.A.1213](https://doi.org/10.1587/transfun.E97.A.1213).
- [7] G. W. Flake and E. B. Baum. Rush Hour is PSPACE-complete, or “Why you should generously tip parking lot attendant”. *Theoretical Computer Science*, 270(1):895–911, 2002. DOI: [10.1016/S0304-3975\(01\)00173-6](https://doi.org/10.1016/S0304-3975(01)00173-6).
- [8] A. S. Fraenkel and D. Lichtenstein. Computing a perfect strategy for $n \times n$ chess requires time exponential in n . *Journal of Combinatorial Theory, Series A*, 31(2):199–214, 1981. DOI: [10.1016/0097-3165\(81\)90016-9](https://doi.org/10.1016/0097-3165(81)90016-9).

- [9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, 1979.
- [10] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008. DOI: [10.1017/CB09780511804106](https://doi.org/10.1017/CB09780511804106).
- [11] S. Iwata and T. Kasai. The Othello game on an $n \times n$ board is PSPACE-complete. *Theoretical Computer Science*, 123(2):329–340, 1994. DOI: [10.1016/0304-3975\(94\)90131-7](https://doi.org/10.1016/0304-3975(94)90131-7).
- [12] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, editors, *Complexity of Computer Computations. The IBM Research Symposia Series*. Springer, Boston, MA. Pages 85–103, Boston, MA. Springer US, 1972. DOI: [10.1007/978-1-4684-2001-2_9](https://doi.org/10.1007/978-1-4684-2001-2_9).
- [13] R. Kaye. Minesweeper is NP-complete. *The Mathematical Intelligencer*, 22(2):9–15, 2000. DOI: [10.1007/BF03025367](https://doi.org/10.1007/BF03025367).
- [14] G. Kendall, A. Parkes, and K. Spoerer. A survey of NP-complete puzzles. *ICGA Journal*, 31(1):13–34, 2008. DOI: [10.3233/ICG-2008-31103](https://doi.org/10.3233/ICG-2008-31103).
- [15] K. Kotsuma and Y. Takenaga. NP-completeness and enumeration of Number Link puzzle. *IEICE Tech. Rep.*, 109(465):1–7, 2010.
- [16] D. Král’, V. Majerech, J. Sgall, T. Tichý, and G. Woeginger. It is tough to be a plumber. *Theoretical Computer Science*, 313(3):473–484, 2004. DOI: [10.1016/j.tcs.2002.12.002](https://doi.org/10.1016/j.tcs.2002.12.002).
- [17] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982. DOI: [10.1137/0211025](https://doi.org/10.1137/0211025).
- [18] Nikoli Co., Ltd. Evolomino, 2025. URL: <https://www.nikoli.co.jp/en/puzzles/evolomino/>.
- [19] M. Rahman, S.-i. Nakano, and T. Nishizeki. Rectangular grid drawings of plane graphs. *Computational Geometry*, 10(3):203–220, 1998. DOI: [https://doi.org/10.1016/S0925-7721\(98\)00003-0](https://doi.org/10.1016/S0925-7721(98)00003-0).
- [20] S. Reisch. Hex ist PSPACE-vollständig. *Acta Informatica*, 15(2):167–191, 1981. DOI: [10.1007/BF00288964](https://doi.org/10.1007/BF00288964).
- [21] J. Robson. The complexity of Go. In *Proceedings of the IFIP 9th World Computer Congress on Information Processing*, volume 9, pages 413–417, 1983.
- [22] O. Ruepp and M. Holzer. The computational complexity of the Kakuro puzzle, revisited. In P. Boldi and L. Gargano, editors, *Fun with Algorithms*, pages 319–330. Springer Berlin Heidelberg, 2010. DOI: [10.1007/978-3-642-13122-6_31](https://doi.org/10.1007/978-3-642-13122-6_31).
- [23] T. Seta. *The complexities of puzzles, CROSS SUM and their another solution problems (ASP)*. PhD thesis, Department of Information Science, the Faculty of Science, the University of Tokyo, 2002.
- [24] L. A. Székely. Turán’s brick factory problem: The status of the conjectures of Zarankiewicz and Hill. In R. Gera, S. Hedetniemi, and C. Larson, editors, *Graph Theory: Favorite Conjectures and Open*

- Problems – 1*, pages 211–230, Cham. Springer International Publishing, 2016. DOI: [10.1007/978-3-319-31940-7_13](https://doi.org/10.1007/978-3-319-31940-7_13).
- [25] Y. Takenaga, S. Aoyagi, S. Iwata, and T. Kasai. Shikaku and Ripple Effect are NP-complete. *Congressus Numerantium*, 216:119–127, 2013.
- [26] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979. DOI: [10.1137/0208032](https://doi.org/10.1137/0208032).
- [27] T. Yato and T. Seta. Complexity and completeness of finding another solution and its application to puzzles. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 86-A:1052–1060, 2003.
- [28] K. Zarankiewicz. On a problem of P. Turán concerning graphs. *Fundamenta Mathematicae*, 1(41):137–145, 1955. DOI: [10.4064/fm-41-1-137-145](https://doi.org/10.4064/fm-41-1-137-145).

ANDREI VALERIEVICH NIKOLAEV
P.G. DEMIDOV YAROSLAVL STATE UNIVERSITY,
SOVETSKAYA STR. 14,
150003, YAROSLAVL, RUSSIA
Email address: andrei.v.nikolaev@gmail.com

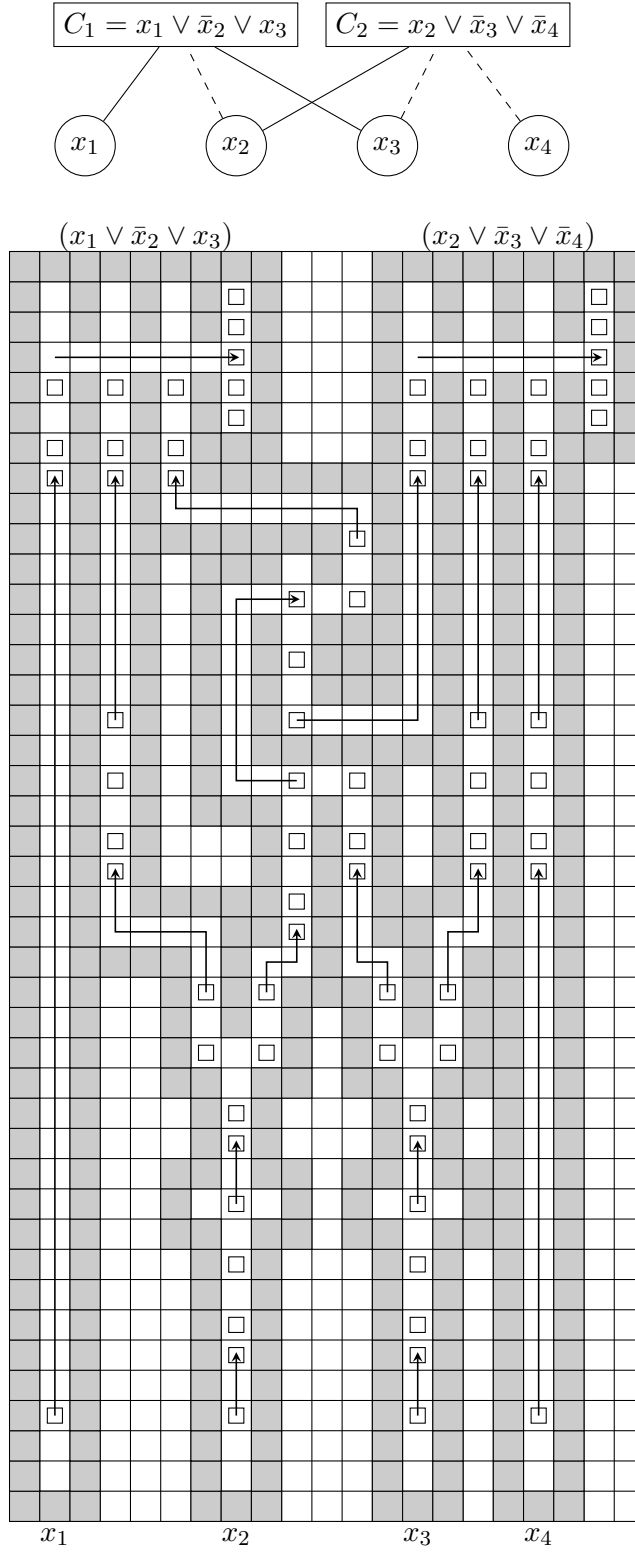


FIG. 9. An example of constructing an EVOLOMINO puzzle for the CNF formula $(x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4)$

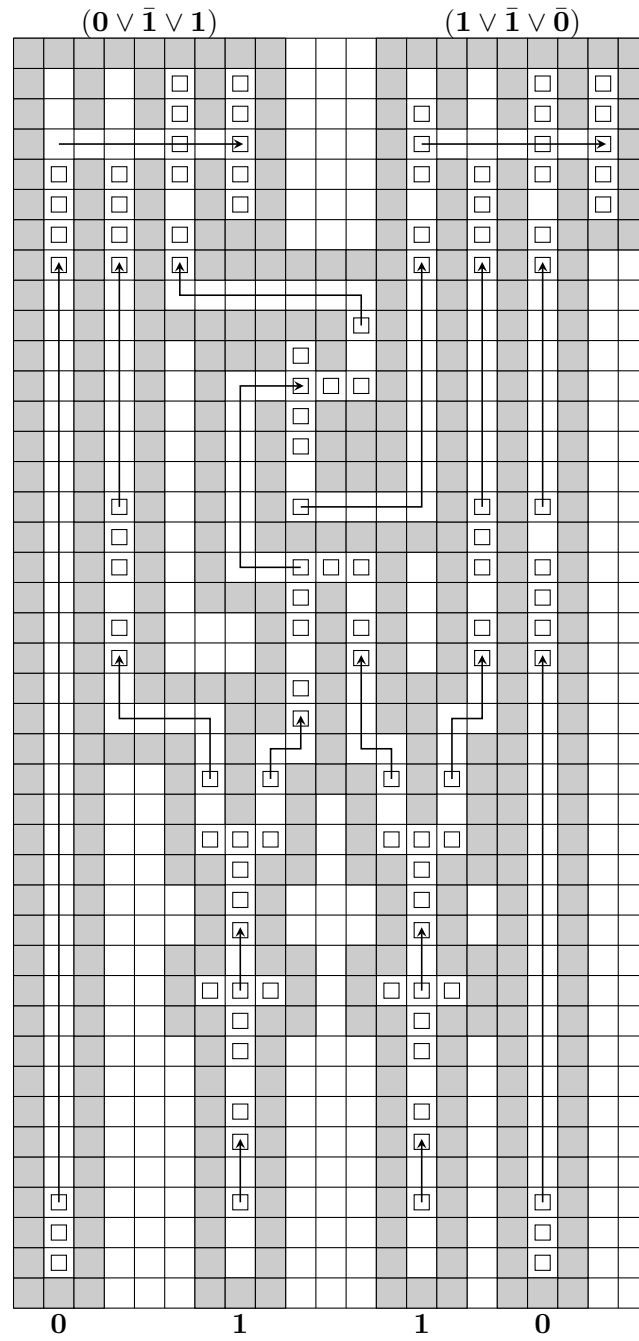


FIG. 10. Solution to the puzzle corresponding to the CNF formula $(x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4)$ under the truth assignment $x_1 = 0, x_2 = 1, x_3 = 1, x_4 = 0$