Tilepaint and Aquarium Puzzles in Periodic Grids

Yan Gerard[1][0000−0002−2664−0650], Pascal Lafourcade[1][0000−0002−4459−511X], Lola-Baie Mallordy[1][0009−0005−1308−4412], and Léo Robert[2][0000−0002−9638−3143]

[1] Université Clermont Auvergne, LIMOS, France
{yan.gerard,pascal.lafourcade}@uca.fr, lolamallord@gmail.com
[2] Université de Picardie Jules Verne, MIS, France leo.robert@u-picardie.fr

Abstract. We consider two classes of puzzles `Aquarium` and `Tilepaint` connected to the field of Discrete Tomography. The input mixes a tiling of a rectangular grid with projections prescribing the number of squared cells that have to be filled in each row and column. In `Tilepaint`, the tiles are either entirely filled or remain empty while in `Aquarium`, the tiles are filled like aquariums with an horizontal level of water. `Tilepaint` is known to be NP-complete with a proof showing that `Aquarium` is also NP-complete, even with tiles of size at most 3. In this paper, we investigate the complexity of the two puzzles in the special case of regular tilings with small tiles. We show that `Tilepaint` can be solved in polynomial time while `Aquarium` is NP-complete for regular triomino tilings and is polynomial time for some regular domino tilings. As `Aquarium` is NP-complete, we propose, in a second part, a physical zero-knowledge proof (ZKP) protocol for `Aquarium`, using decks of playing cards (using a total of $2mn + 2$ cards for an $m \times n$ grid).

1 Introduction

The aim of this work is to investigate two combinatorial problems at the intersection of Discrete Tomography and tilings. Discrete Tomography focuses on reconstructing a discrete set of points S based on its projections, which count the number of points in S along given straight lines [6, 7]. Tilings, on the other hand, is the antique domain involving partitioning a space —here, a 2D grid— into smaller pieces called *tiles*, each with prescribed shapes.

Previous works have examined the relationship between these two fields by considering tilings of a rectangular grid using dominoes [10] or bars [3], where the goal was to compute a tiling that satisfies certain tomographic constraints. In contrast, our work investigates a different class of problems: given an existing tiling of a grid, the challenge is to select specific tiles or tile subsets that satisfy the prescribed tomographic constraints. As nonograms [13], these problems are variants of the foundational problem of Discrete Tomography i.e the reconstruction of a binary matrix from its row and column sums or equivalently the reconstruction of a lattice set from its horizontal and vertical projections [5, 12].

This variation originates from two recreational puzzles. The first, *Tilepaint*, is a puzzle published by Nikoli[3]. The second, *Aquarium*, was invented in 2004 by Naoki Inaba. For both problems, the input consists of a regular grid of $m \times n$ squares tiled by polyominoes called *tiles* and $m + n$ integers, one for each row and column of the grid.

The aim of *Tilepaint* is to select a set of tiles satisfying the tomographic constraint that the total number of selected squares in each row and column corresponds to the input projections (Fig. 1).

`Tilepaint`(A, H, V)

- `Input:` a tiling of the regular grid of $m \times n$ squares by a set A of polyominoes A_k called *tiles* and a pair of integer vectors $H = (h_i)_{1 \leq i \leq m} \in \mathbb{Z}m$, $V = (v_j)_{1 \leq j \leq n} \in \mathbb{Z}n$ called the horizontal and vertical *projections*.
- `Ouput:` A subset S of the tiles such that for all row and column indices i and j, the number of squares of the tiles of S in row i and column j are equal to h_i and v_j.

The problem *Aquarium* also involves filling certain squares of the tiles with the constraint of having the prescribed number of squares filled in each row and column. But unlike *Tilepaint* where either none or all the squares of each tile are filled, tiles are filled in *Aquarium* as aquariums. It means that for each tile, a water level is chosen below which all the squares of the tile are filled and above which no square of the tile is filled.

`Aquarium`(A, H, V)

- `Input:` a tiling of the regular grid of $m \times n$ squares by a set A of polyominoes A_k called *tiles* and a pair of integer vectors $H = (h_i)_{1 \leq i \leq m} \in \mathbb{Z}m$, $V = (v_j)_{1 \leq j \leq n} \in \mathbb{Z}n$ called the horizontal and vertical *projections*.
- `Ouput:` An horizontal water level l_k for each tile A_k so that for any indices i or j, the number of filled squares *i.e.*, under the water level of its tile is equal to h_i for the row i and to h_j for the column j.

The water level constraint makes the *Aquarium* problem asymmetric. Notice also that when tiles are of height 1, *Tilepaint* and *Aquarium* are equivalent.

In this paper, we consider the problem of reconstructing a solution of `Tilepaint` and `Aquarium` in the special case where the tiling is periodic and has small tiles of size 2 or 3 i.e with dominoes or triominoes. The paper first presents the state of the art and the results. Then it provides the proofs with a polynomial-time algorithm for some domino tilings and some reductions of NP-complete problems for the triomino tilings. We end the paper with a zero-knowledge proof (ZKP) protocol for proving that a solution is known without leaking it.

[3] Nikoli is a well-known Japanese publisher of logic puzzles, famous for popularizing Sudoku.

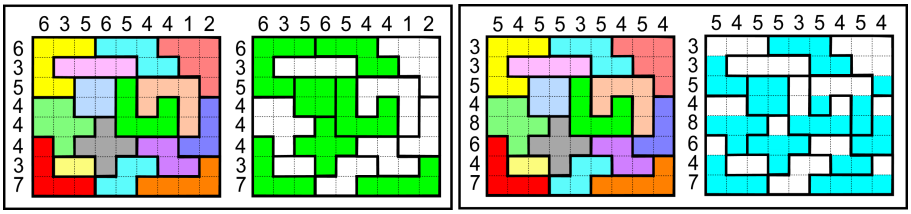


Fig. 1. Instances of `Tilepaint`(A, H, V) and `Aquarium`(A, H, V) and their solutions (left for *Tilepaint* and right for *Aquarium*).

2 State of the Art

The problem `Tilepaint`(A, H, V) has been proved NP-complete in 2022 [8]. The proof is a reduction of `3D-matching` and since it only uses tiles of height 1, it also proves that `Aquarium` is NP-complete. See Fig 2 for the reduction of a small instance of `3D-matching` into a `Tilepaint` instance of height 1. However, this reduction slightly differs from the original in order to show the property that `Tilepaint` remains NP-complete with tiles of size at most 3.

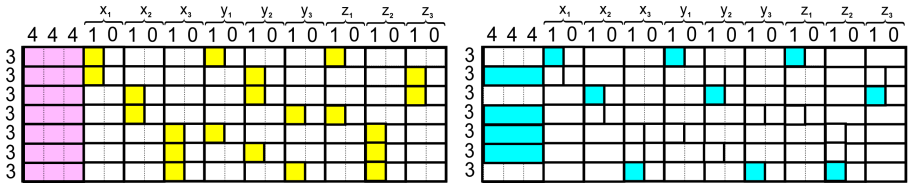


Fig. 2. Reduction of `3D-matching` to `Tilepaint` or `Aquarium`. Left, an instance of `Tilepaint`(A, H, V) encoding the instance of 3D-matching composed of the sets $p_1 = (x_1, y_1, z_1)$, $p_2 = (x_1, y_2, z_3)$, $p_3 = (x_2, y_2, z_3)$, $p_4 = (x_2, y_3, z_1)$, $p_5 = (x_3, y_1, z_2)$, $p_6 = (x_3, y_2, z_2)$, $p_7 = (x_3, y_3, z_2)$. Right, the solution of `Tilepaint` describes the solution $\{p_1, p_3, p_7\}$ of the 3d-matching instance.

The proof of NP-completeness of `Tilepaint` or `Aquarium` encodes the instance of 3d-matching in the tiling which is thus quite irregular, while the projections are fixed. By using a periodic tiling and only variable projections, we can expect different complexities. The first case to investigate is a periodic tiling by $m \times n$ tiles of unit size. In this case, solving `Aquarium` or `Tilepaint` means reconstructing a set of squares with prescribed projections. This problem is one of the milestones of Discrete Tomography. It has been independently proved to be sovable in polynomial time by D. Gale and R. Ryser in 1957 [5, 12].

The Gale strategy is to reduce the problem to a problem of edge selection in a bipartite graph with unit capacities on the edges. Then the problem can

be solved by the initial Gale algorithm or by more general graph algorithms such that Ford-Fulkerson or Hopcroft-Karp algorithms or with a more recent quasi-linear time flow algorithm [1]. It proves the next proposition:

Proposition 1. *Given horizontal, vertical projections and the tiling A of the region by $m \times n$ unit tiles,* `Aquarium` *and* `Tilepaint` *can be solved in time almost linear in the size of the grid ($O(mn)$).*

Let us now consider the extension of the problem to periodic domino and tricomino tilings. Are `Tilepaint` and `Aquarium` still NP-complete in this restricted framework, or do there exist polynomial-time reconstruction algorithms?

3 Complexity Results

We consider four periodic domino tilings that we call the *stack*, the *jail*, the *wall* and the *traffic jam* (Fig. 3).

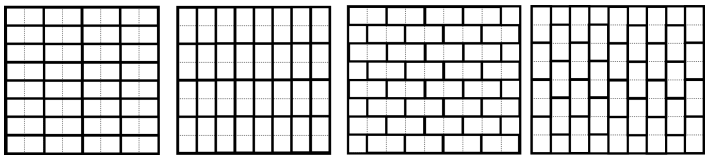


Fig. 3. Four domino tilings. From left to right, the *stack*, the *jail*, the *wall* and the *traffic jam*.

We also consider three periodic triomino tilings drawn in Fig. 4 and respectively called *E*, *Waves*, and *LL*.

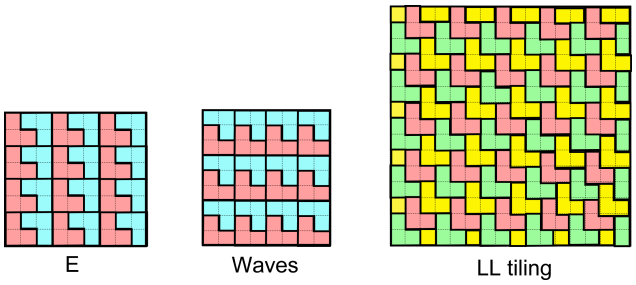


Fig. 4. Three Periodic tilings by triominoes. The *E*, *Waves*, and *LL* tilings.

We summarize our complexity results in Table 1, and express them in the next theorem.

Table 1. Complexities of the periodic instances of `Tilepaint` and `Aquarium`.

Tilings	Stack	Jail	Wall	Traffic	Waves	E	LL
`Tilepaint`	P	P	P	P	P	P	P
`Aquarium`	P	?	P	?	NP-complete	NP-complete	NP-complete

Theorem 1. *(i)* ***`Tilepaint`*** *can be solved in polynomial-time for the Stack, the Jail, the Wall, the Traffic Jam, the Waves, E and LL.*

(ii) ***`Aquarium`*** *can be solved in polynomial-time for the Stack and the Wall.*

(iii) ***`Aquarium`*** *is NP-complete for the Waves, the E and the LL triomino tilings.*

4 Periodic Domino Tilings

We now present the proofs of Theorem 1 for the four considered domino tilings. Our main contribution is to show that the wall instances can be solved in polynomial time.

The stack. It is the easiest case, since one vertical projection out of two is useless. Both instances of `Aquarium` and `Tilepaint` can be reduced to instances with unit tiles and thus can be solved in almost linear time.

The jail. Since `Tilepaint` is symmetrical (direction x and y are equivalent), `Tilepaint` makes no difference between the jail and the stack. Then `Tilepaint` is solvable in polynomial time for the jail. For `Aquarium` the jail and stack instances are different since the vertical tiles can be half-full. Surprisingly, `Aquarium` is challenging. We suspect that for the jail tiling, `Aquarium` is NP-complete but its complexity is an open question.

The wall. As the height of all the aquariums is 1, there is no difference between `Tilepaint` and `Aquarium`. We now show that the domino instances of `Aquarium` and `Tilepaint` with the wall domino tiling can be solved in polynomial time. We distinguish two cases depending on the parity of the number of columns.

(i) If the number of columns is odd, for instance, equal to $2k+1$, each row is made of k tiles of size 2 and a unit tile. It follows that the value of an horizontal projection is odd if and only if the unit tile of the corresponding row is filled. Then the parity of the values of the horizontal projections directly determine the unit tiles which have to be filled and the ones which have to remain empty. Now we remove the unit tiles from the picture and focus our attention on the dominoes. We call rank of a domino the index of the column of its left square so that the ranks of dominoes go from 1 to $2k$. We explain now how to determine the projections of the dominoes of each rank. First, we substract the number of unit squares in the first column from the initial projection v_0 of the first column.

It provides the integer $v'_1 = v_1 - v_0$ counting the number of filled dominoes in the first rank. Then by subtracting v'_1 to v_2, we obtain the integer $v'_2 = v_2 - v'_1$ counting the number of dominoes of rank 2. By repeating this computation in the next ranks, we obtain the number of dominoes of each rank (see Fig. 5 for a detailed example of the algorithm). Knowing the number of dominoes of each rank, we decompose the problem into two independent sub-problems by considering the rows of even and odd indices. Each sub-problem is structured as the stack tiling and thus can be solved in polynomial time. In the case where the two stack instances, with the rows of even indices or the rows of odd indices are both feasible, we aggregate the two solutions to build the final solution. If one of the two instances with even or odd rows is not feasible, then the puzzle admits no solution.

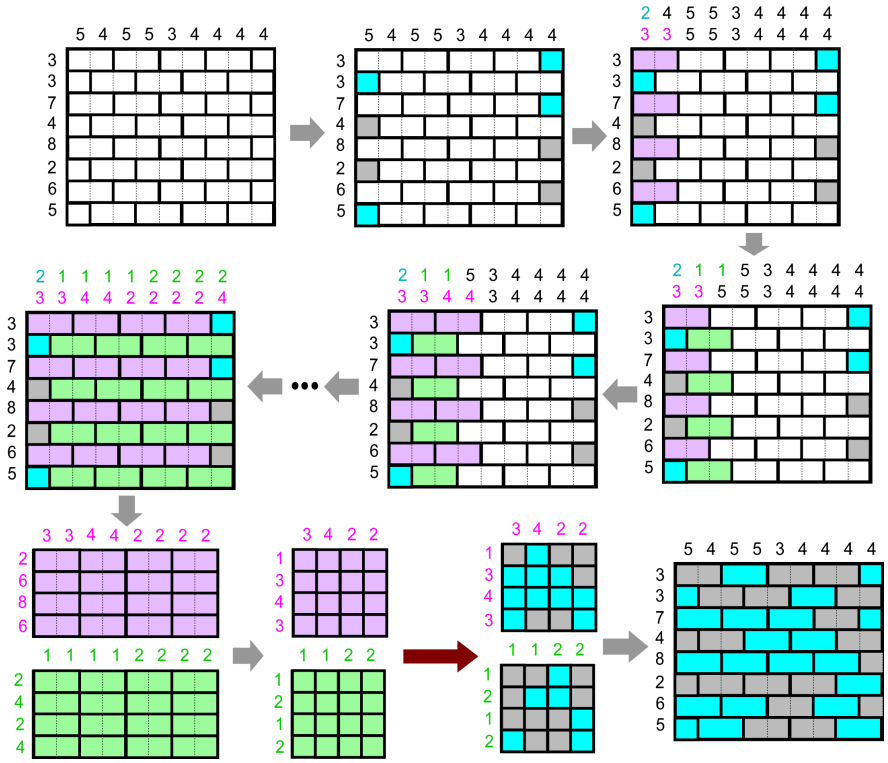


Fig. 5. Algorithm for solving the wall when the number of columns is odd.

(ii) If the number of columns $n = 2k$ is even, there are two unit tiles in some rows and zero unit tiles in others. The columns have indices going from 1 to $2k$. For a row with a pair of unit tiles and an odd value of projection, only one of the two unit tiles is filled. It only remains some uncertainty about the side, left or

right, of the unit square which should be filled but it can be determined with two different ways of counting the filled dominoes. Let us denote v'_0 and v'_{2k} the a-priori unknown numbers of unit tiles that have to be filled respectively in the left-most and right-most columns. We also denote t the unknown number of dominoes which have to be filled. By computing the sum of the vertical projections with an odd index, we count the number of filled dominoes t plus the number of unit tiles in the left-most column. Then $\sum_{j=1}^{k} v_{2j-1} = t + v'_0$ while by symmetry, the sum of the vertical projection with an even index $\sum_{j=1}^{k} v_{2j} = t + v'_{2k}$ counts the number of filled dominoes plus the filled unit tiles of the right-most column. It follows that $\sum_{j=1}^{k} v_{2j} - \sum_{j=1}^{k} v_{2j-1} = v'_{2k} - v'_0$ (a). If we now consider a row with an even projection and two unit tiles, by parity, it might contain either two filled unit tiles or zero filled unit tiles. In both cases, the contribution of the row to the difference $v'_{2k} - v'_0$ is null. The only rows contributing to the difference $v'_{2k} - v'_0$ are the ones with an odd projection. They contribute either with a value $+1$ or with a value -1. The difference (a) tells us exactly how many unit tiles (from the rows with an odd projection) have to be filled on the left and how many tiles have to be filled on the right. Then the unit tiles from the rows with an odd projection can be addressed with some basic computations and fully removed from the picture by removing their squares and updating the projections of the remaining tiles. Now let us focus on the unit tiles in the rows having an even projection. We cannot fill one unit tile without filling the other one. Then these pairs of unit tiles are similar to dominoes. By making a bridge from the left column to the right column, the remaining combinatorial problem has a cylinder structure which does not seem to be of any help to solve the problem. (Fig. 6).

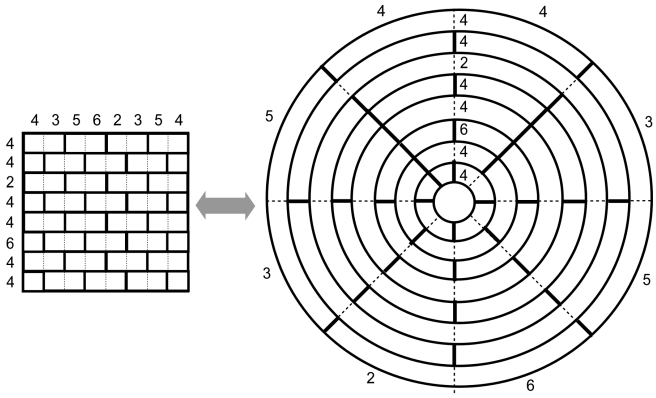


Fig. 6. Cylinder structure of the wall when the number of columns is even.

Our algorithm breaks the cylinder symmetry by assuming that the number of dominoes of a given rank, for instance of rank 1 is fixed (the rank of a domino is the index of the column of its left square). Then by difference with v_2, we obtain

the number of dominoes of rank 2 and so on until the number of dominoes of rank $n-1$. As in the odd case (i), the knowledge of the number of dominoes of each rank allows us to decompose the problem in two sub-problems each one structured as a stack or a stack with some missing squares but it does stop the flow algorithms to solve it. The two instances with the odd or even rows can be solved independently in polynomial time. If one of them has no solution (as illustrated in the second line of Fig. 7), then the assumption on the number of filled dominoes of rank 1 is not valid. Then we test another value. If the two independent instances are feasible, the solutions are aggregated to provide a general solution. For each potential number of filled dominoes of rank 1, the reconstruction routine takes an almost linear time $O(mn)$. Since we have at most m numbers of filled dominoes of rank 1 to test, the total complexity of the reconstruction algorithm is $O(m^2n)$.

It follows that the reconstruction algorithm for the wall is $O(mn)$ if n is odd and $O(m^2n)$ if n is even.

The traffic jam. By symmetry of the wall, we know that `Tilepaint` is polynomial time solvable. For `Aquarium`, the fact that the tiles are vertical and can be half-filled makes the problem non trivial. If all the vertical projections with an even index are null, then the even columns are empty and the non empty columns are tiled as the jail. Then the jail can be reduced to the traffic jam, showing that the complexity for the traffic jam is at least the complexity for the jail.

5 Periodic Triomino tilings

We now prove that `Tilepaint` is polynomial-time for the three considered triomino tilings while `Aquarium` becomes NP-complete.

Polynomial-time algorithms for Tilepaint. We start with the waves triomino tiling (by symmetry, it is equivalent to E, which is not true for Aquarium). We decompose the tiles in two classes denoted L if their shape looks like the letter L and $\bar{L}$ for their symmetric. Two horizontal projections out of three directly provide the number of filled L and $\bar{L}$ in each row. Then if they are consistent, we do not need to take into account the intermediary rows crossing L and $\bar{L}$ triominoes. Concerning the vertical projections, the computation is less trivial. The vertical projection v_1 is equal to $v_1 = 2|L_1| + 1|\bar{L}_1|$ where $|L_1|$ and $|\bar{L}_1|$ are respectively the numbers of filled L and $\bar{L}$ crossing the first column. With $v_2 = |L_1| + 2|\bar{L}_1|$, it leads to $|L_1| = \frac{2}{3}v_1 - \frac{1}{3}v_2$ and $|\bar{L}_1| = \frac{2}{3}v_2 - \frac{1}{3}v_1$. Equivalent expressions hold for the other columns. As we can compute the numbers of filled L and $\bar{L}$ in each row and column, we can reconstruct the sets of filled L and $\bar{L}$ independently as for unit tiles. It solves `Tilepaint`(A, H, V) for the waves and E with time complexity $O(mn)$.

The strategy for LL is similar in that we decompose the tiling in three classes of tiles (all of them are L triominoes) as shown with the colors red, green and yellow in Fig. 4. Assume that the number of filled yellow tiles is known in the

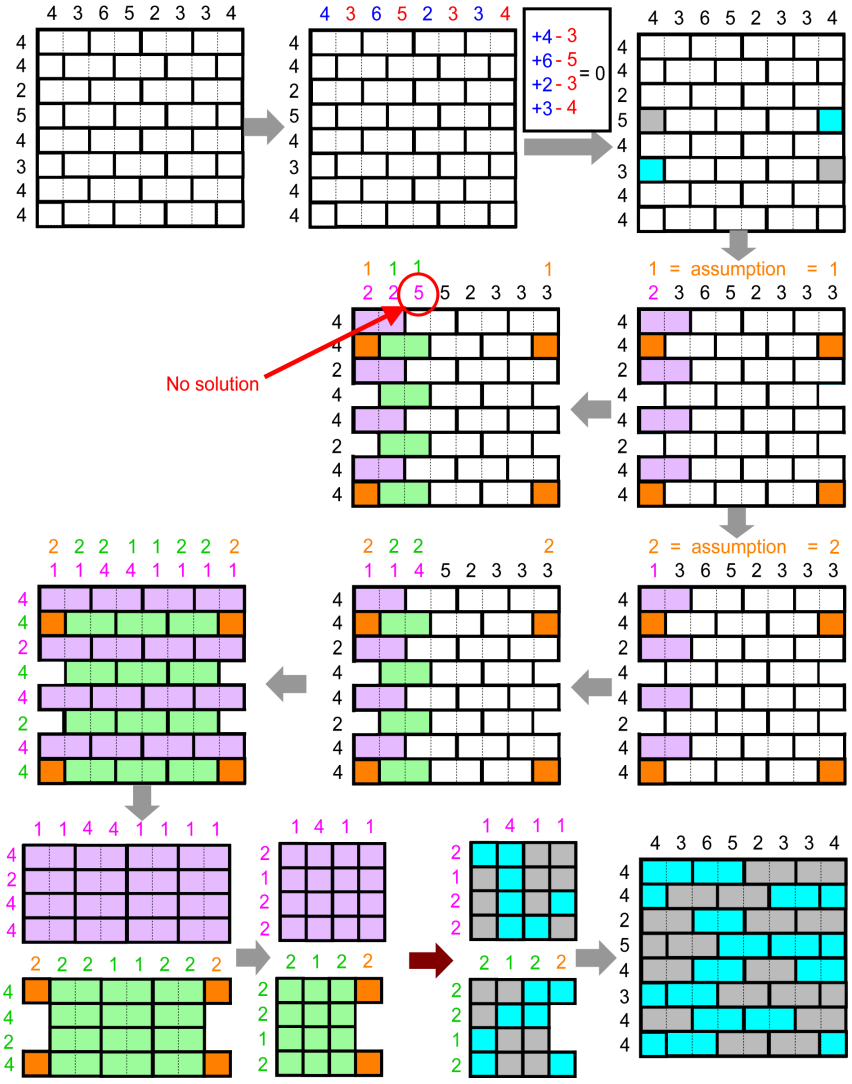


Fig. 7. Algorithm for solving the wall when the number of columns is even. Above, the difference between the vertical projections of the even and odd columns provides the difference between the filled unit cells in the left-most and rightmost columns by considering uniquely the rows of odd horizontal projections. It allows us to remove from the unit tiles of the rows with an odd projection (by updating the projections). Then the algorithm requires to fix the number of dominoes filled with a given rank *i.e.*, with a left square in a given column, here the left-most. It can either lead to a contradiction as in the second row of the figure or to a solution, through a decomposition into two sub-problems.

left-most column. By difference, we get the number of filled green tiles in this first column. Then the second vertical projection provides the number of filled red tiles in the second column and so on. By this same reasoning, we obtain the number of filled tiles of each color in each column. It works in the same way for the rows: with an assumption on the number of filled green tiles in the lowest row, we obtain all the other projections for each set of colored tiles. Then the filled tiles of each color can be computed in almost linear time (Proposition 1) as performed for unit tilings. If there exists a solution for each color, then we obtain a solution. If a color is not feasible, then there is no global solution with the considered assumptions on the leftmost and lowest projections. In other words, by assuming the number of green filled tiles in the left-most column and lowest row, we can determine in polynomial time whether it provides solutions. As the number of values to test is at most m for the left-most column and n for the lowest row, it provides a polynomial time algorithm of time complexity $O(m^2n^2)$ for solving any instance in the LL tiling.

Aquarium is NP-complete. We now prove that $\texttt{Aquarium}(A, H, V)$ is NP-complete with the regular triomino tilings. We reduce a problem of Discrete Tomography with three colors that we denote $\texttt{DT3}(H, H', V, V')$ [4]. The input of $\texttt{DT3}(H, H', V, V')$ consists of a rectangular region of $m \times n$ unit squares with two vertical projections $V \in \mathbb{Z}n$ and $V' \in \mathbb{Z}n$ and two horizontal projections $H \in \mathbb{Z}m$ and $H' \in \mathbb{Z}m$. The output consists of two disjoint subsets *Blue* and *Red* whose horizontal and vertical projections are respectively H and V for *Blue* and H' and V' for *Red*. The top rectangle of Fig. 8 shows an instance of $\texttt{DT3}(H, H', V, V')$ and its solution. The third color that we call *white* is the one of the squares which are neither in *Blue* nor in *Red*. The class of problems $\texttt{DT3}(H, H', V, V')$ has been proved to be NP-complete in 2012 in [4]. The two problems $\texttt{DT3}(H, H', V, V')$ and $\texttt{Aquarium}(A, H, V)$ on triomino tiling are connected by using the property that in Aquarium a triomino can have three states: empty, half-filled, or full. The strategy of the reduction is to put these three states in correspondence with the three colors of $\texttt{DT3}(H, H', V, V')$.

We consider the wave triomino tiling. We encode an instance of $\texttt{DT3}(H, H', V, V')$ with m rows and n columns in an instance of $\texttt{Aquarium}(A, H^A, V^A)$ with $3m$ rows and $2n$ columns. With indices i going from 0 for the bottom rows to $m-1$ or $3m-1$ for the top rows, we choose $h^A_{3i} = 0$, $h^A_{3i+1} = h_i + h'_i$, $h^A_{3i+2} = 2h_i$, and for j going from 0 to $n-1$, we have $v^A_{2j} = v_j$ and $v^A_{2j+1} = 2v_j + v'_j$. The null rows require that all the L triominoes are empty. Then a solution of this aquarium instance has exactly $v^A_{2j} = v_j$ full tiles in the pair of columns of indices $2j$,$2j+1$. The difference $v^A_{2j+1} - 2v^A_{2j} = v'_j$ gives the number of half-filled tiles in the pair of columns $2j$,$2j+1$. For the rows, we have exactly $\frac{h_{3i+2}}{2} = h_i$ full tiles in the triplet of rows of indices $3i$, $3i+1$, $3i+2$. The number of half-full tiles is $h_{3i+1} - \frac{h_{3i+2}}{2} = h_i + h'_i - \frac{2h_i}{2} = h'_i$. Then if we consider the full tiles and the half-filled tiles, their number on each pair of columns and triplet of rows fits exactly with the number of expected blue and red squares of a solution of $\texttt{DT}(3, H, H', V, V')$. It shows that $\texttt{DT}(3, H, H', V, V')$ admits a solution if and

only if $\texttt{Aquarium}(A, H^A, V^A)$ is feasible. The NP-hardness of $\texttt{DT}(3, H, H', V, V')$ implies the NP-hardness of $\texttt{Aquarium}(A, H^A, V^A)$.

By well choosing the rows and columns with null projections, a similar reduction can be replicated for the E and LL triomino tilings.

6 ZKP protocol for Aquarium

Recall that a Zero-Knowledge Proof protocol describes the interactions between a prover Pv, who possesses a secret, and a verifier Vf, who wants to be convinced that Pv indeed has the solution without revealing the solution itself. In our context, we aim to construct a protocol that convinces Vf that Pv has the solution to a given `Aquarium` instance without leaking the solution.

Our protocol is split into two phases: (i) the construction of Pv's solution done by Vf where the level constraint is enforced; (ii) the verification of the tomographic constraints. In total, for a grid of size $n \times m$ and A its set of polyominos, our protocol needs $2mn + 2$ cards, and uses $\sum_{a \in A}(\text{height}(a) - 1)$ AND protocols, $\sum_{a \in A} \text{height}(a)$ copy protocols and $n + m$ shuffles. We formally describe the different phases of our protocol.

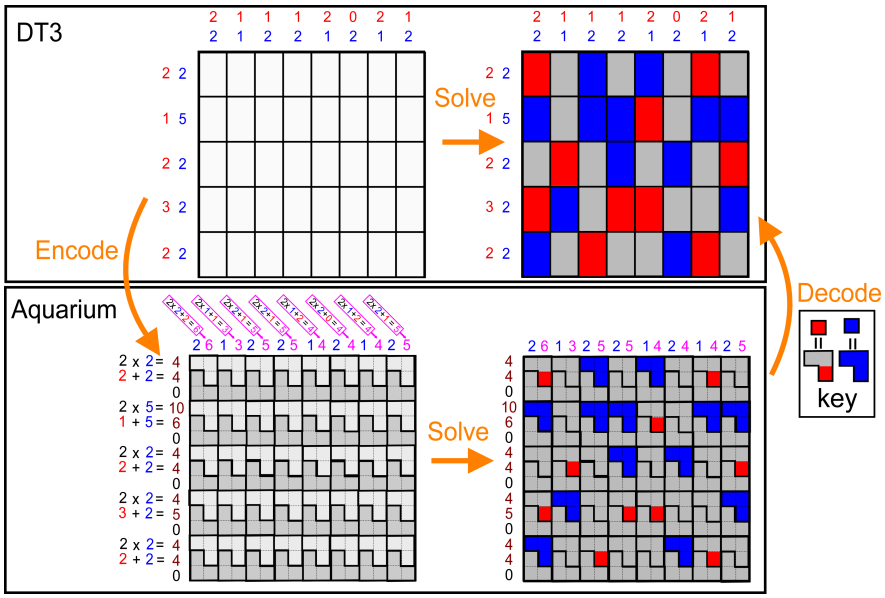


Fig. 8. An instance of $\texttt{DT}(3, H, H', V, V')$, its encoding in $\texttt{Aquarium}(A, H, V)$ and the corresponding solutions.

6.1 Setup phase

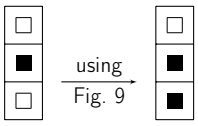
Before getting involved in the protocol, Pv and Vf need to agree on public parameters. The grid of an $\texttt{Aquarium}(A, H, V)$ instance is composed of m rows and n columns, with A, H and Vf being public values of the polyominoes (*e.g.*, the tiles), the numbers on the left and upper edges of the grid, respectively.

The grid has no commitment yet, the prover Pv will commit its solution in the *construction phase* (where Vf knows that all the tiles are correctly filled). The next step is to verify that the tomographic constraints are respected, if so, the protocol does not abort and Vf is convinced that Pv has the solution.

6.2 Construction phase

The prover Pv wants to select some squares of the tiles. The selected squares are blackened while the remaining squares are left white. There are two steps, Pv indicates the level separating the black and white squares (without Vf knowing its height) then Vf will fill the region up to the given level.

Level Indicator

1. Pv and Vf consider the leftmost commitments of each row. Then, the prover Pv commits a black value on a given cell to represent the level; the other cells are committed to white. Note that if the region is empty (*i.e.*, no black cell), then Pv simply commits white value on each cell.
2. The verifier Vf will spread the level downward: 

We use the technique of [11] to enforce that every cell under the level indicator will be coloured in black. Basically, for each adjacent commitments (from top to bot), Vf computes an AND operation with the Mizuki-Sone AND protocol [9] on the second commitment (and keeps the first commitment in its initial state). The general sub-routine is illustrated below in Figure 9.

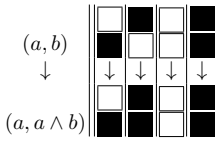


Fig. 9. Subroutine of [11] for modifying each consecutive cell where $a, b \in \{0, 1\}$ represents the colours of the leftmost and rightmost cells, respectively.

Spreading the Level The next step is to propagate the configuration of the leftmost cells to the rest of the tile (done via the Mizuki-Sone copy protocol [9]). Formally, for this step, For

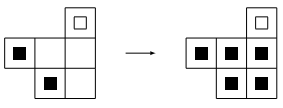


Fig. 10. Copies of right to leftmost commitment to fill each row.

each row of length ℓ, apply the copy protocol of the leftmost commitment to output ℓ identical commitments to form the committed row as described in Figure 10. The previous two steps are repeated for each tile.

6.3 Verification Phase

Verifying that Pv has indeed the solution is done by checking the numbers for each row are respected. This means that Vf needs to check that the numbers on each row/column is correct, to be convinced that Pv has committed a solution.

The verification of this rule is straightforward, each leftmost card of commitments will be used to verify the row constraints while the rightmost card will be used for column constraints. This approach has the advantage of using cards only once, so no input-preserving technique is needed (thus reducing the number of steps and shuffles). Formally, Pv and Vf apply the following steps:

1. For each row i, Vf picks the leftmost card of each commitment. Then Pv and Vf shuffle them. When Vf reveals the cards, if the number of $\boxed{\clubsuit}$ is equal to h_i (the number at the left of row i) then Vf continues, otherwise aborts.
2. For each column j, Vf picks the rightmost card of each commitment. Then Pv and Vf shuffle them. When Vf reveals the cards, if the number of $\boxed{\heartsuit}$ is equal to $m - v_j$ (v_j the number indicated at the top of column j) then Vf continues, otherwise aborts.

6.4 Security Proofs

Our protocol provides completeness, soundness and is zero-knowledge. Note that the sub-protocols used from the literature have been proven secure *i.e.*, they are correct, complete, sound and zero-knowledge.

Theorem 2 (Completeness). *If* Pv *knows the solution of an Aquarium instance, then* Pv *can convince* Vf*.*

Proof. Suppose that Pv has the solution, we want to show that the protocol will not abort, hence that Pv is able to prove the number rule. Let S be a solution, S is a valid grid respecting all the constraints.

During the construction phase, Pv indicates the water level of S for each room to Vf. As S is a valid grid, Pv can always choose the top left cell of the region (if the region is empty, then Pv simply puts white commitments) which indicates the level. Hence, when Vf spreads the level downward with the AND protocol and fills each row with the copy protocol, Vf ends up with a room coloured in exactly the same way as Pv from the correctness of the AND and copy protocols. As this routine is repeated for each room, Vf will colour each region following the "level" rule, so Pv can construct the solution S with this routine. Furthermore, since S is the solution, the "number" rule is enforced. Therefore, the protocol will not abort at the number rule verification.

Finally, the protocol will not abort so the completeness is verified.

Theorem 3 (Soundness). *If* Pv *does not provide a solution of the* $p \times q$ *Aquarium grid, then* Vf *always rejects.*

Proof. We prove the contrapositive of this statement. Suppose V accepts, meaning the verification passes.

When Vf checks the number of $\boxed{\clubsuit}$ for each row, this number corresponds to the number of black commitments in the row (a black commitment is represented as $\boxed{\clubsuit}\boxed{\heartsuit}$) since the leftmost cards are picked. For the columns, the complementary of each commitment is checked meaning that the number of $\boxed{\heartsuit}$ is considered. Since the number of $\boxed{\heartsuit}$ cards corresponds to the difference of the total number of rows in the grid and the column indicating number, the complement of each commitments is equal to the column indicating number. Since the rightmost cards are picked, this corresponds to the solution of the grid.

Notice that we omit the construction phase in this analysis. Indeed, since Vf constructs Pv's solution, then no invalid configuration can occur (*e.g.*, a filled black row with a white cell in it, or a white cell below a black cell in the same room). This claim is straightforward since the only input of the prover is during the level indicator phase. At this step, the prover chooses the values of the commitments (only one black value and the other white). Even if the prover puts multiple black values on the commitments, the sub-routine of [11] *smooths* the final values of the sequence. This comes from the fact that only the topmost black cell is taken into account in the final configuration. This last point allows us to let Pv making its own commitments and not using heavier sub-protocols, such as the chosen-pile protocol [2], that would result in worst complexity (as the number of shuffles and cards would increase).

Theorem 4 (Zero-knowledge). Vf *learns nothing about* Pv*'s solution of the given grid* G*.*

Proof. Suppose that Pv has a grid S, we want to show that Vf will not have learned anything about S at the end of the protocol. For this, we prove that the interaction betwenn Pv and Vf can be simulated by a simulator Sim that does not know Pv's solution. It is sufficient to show that all distributions of face-up cards can be simulated by Sim.

The verification phase is the only step where cards are revealed. Since pile-scramble shuffle is used, on each row and each column, all outputs have the same probability to occur (given a specific initial number of $\boxed{\heartsuit}$ and $\boxed{\clubsuit}$ which corresponds to the row/column indicating number).

Thus, the verifier Vf learns nothing about the solution.

7 Conclusion

We focused our attention on instances of `Tilepaint` and `Aquarium` on regular tilings (see results in Table 1). As shown for domino and triomino tilings, the `Tilepaint` puzzles seem to be polynomial time solvable (for a fixed tiling) with

a polynomial whose degree is linear in the size of the tiles. On the other hand, `Aquarium` remains NP-complete for periodic triomino tilings and the question of its complexity is open for some of the most simple domino tilings such as the jail and the traffic jam.

Finally, we proposed a ZKP protocol for `Aquarium` and proved its security. Our method could be extended to `Tilepaint`, since it is also NP-complete for irregular tiling and works in a similar way. An interesting future work would be to design ZKP protocols on other `Aquarium` tilings.

References

1. Chen, L., Kyng, R., Liu, Y.P., Peng, R., Gutenberg, M.P., Sachdeva, S.: Maximum flow and minimum-cost flow in almost-linear time. In: 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS). pp. 612–623 (2022). https://doi.org/10.1109/FOCS54457.2022.00064
2. Dumas, J.G., Lafourcade, P., Miyahara, D., Mizuki, T., Sasaki, T., Sone, H.: Interactive physical zero-knowledge proof for Norinori. In: Du, D.Z., Duan, Z., Tian, C. (eds.) Computing and Combinatorics. LNCS, vol. 11653, pp. 166–177. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26176-4_14
3. Dürr, C., Ch., E.G., Rapaport, I., Rémila, E.: Tiling with bars under tomographic constraints. CoRR **cs.DS/9903020** (1999), https://arxiv.org/abs/cs/9903020
4. Dürr, C., Guiñez, F., Matamala, M.: Reconstructing 3-colored grids from horizontal and vertical projections is np-hard: A solution to the 2-atom problem in discrete tomography. SIAM J. Discret. Math. **26**(1), 330–352 (2012). https://doi.org/10.1137/100799733, https://doi.org/10.1137/100799733
5. Gale, D.: A theorem on flows in networks. Pacific J. Math. **7**, 1073–1082 (1957). https://doi.org/doi:10.2140/pjm.1957.7.1073., https://msp.org/pjm/1957/7-2/pjm-v7-n2-p04-s.pdf
6. Gardner, R.J.: Geometric Tomography. Encyclopedia of Mathematics and its Applications, Cambridge University Press (1995)
7. Herman, G.T., Kuba, A.: Discrete Tomography - Foundations, Algorithms and Applications. Birkhauser (1999)
8. IWAMOTO, C., IDE, T.: Five cells and tilepaint are np-complete. IEICE Transactions on Information and Systems (Mar 2022). https://doi.org/10.1587/transinf.2021fcp0001
9. Mizuki, T., Sone, H.: Six-card secure AND and four-card secure XOR. In: Deng, X., Hopcroft, J.E., Xue, J. (eds.) FAW 2009. LNCS, vol. 5598, pp. 358–369. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02270-8_36
10. Picouleau, C.: Reconstruction of domino tiling from its two orthogonal projections. Theor. Comput. Sci. **255**(1-2), 437–447 (2001). https://doi.org/10.1016/S0304-3975(99)00312-6, https://doi.org/10.1016/S0304-3975(99)00312-6
11. Robert, L., Miyahara, D., Lafourcade, P., Mizuki, T.: Physical ZKP protocols for nurimisaki and kurodoko. Theor. Comput. Sci. **972**, 114071 (2023). https://doi.org/10.1016/J.TCS.2023.114071, https://doi.org/10.1016/j.tcs.2023.114071
12. Ryser, H.J.: Combinatorial properties of matrices of zeros and ones. Canbadian Journal of Mathematics **9**, 371–377 (1957)
13. Ueda, N., Nagao, T.: Np-completeness results for nonogram via parsimonious reductions (1996), https://api.semanticscholar.org/CorpusID:3040166