On bounded Wang tilings

Marek Tyburec* Jan Zeman

Abstract

Wang tiles enable efficient pattern compression while avoiding the periodicity in tile distribution via programmable matching rules. However, most research in Wang tilings has considered tiling the infinite plane. Motivated by emerging applications in materials engineering, we consider the bounded version of the tiling problem and offer four integer programming formulations to construct valid or nearly-valid Wang tilings: a decision, maximum-rectangular tiling, maximum cover, and maximum adjacency constraint satisfaction formulations. To facilitate a finer control over the resulting tilings, we extend these programs with tile-based, color-based, packing, and variable-sized periodic constraints. Furthermore, we introduce an efficient heuristic algorithm for the maximum-cover variant based on the shortest path search in directed acyclic graphs and derive simple modifications to provide a $1/2$ approximation guarantee for arbitrary tile sets, and a $2/3$ guarantee for tile sets with cyclic transducers. Finally, we benchmark the performance of the integer programming formulations and of the heuristic algorithms showing that the heuristics provides very competitive outputs in a fraction of time. As a by-product, we reveal errors in two well-known aperiodic tile sets: the Knuth tile set contains a tile unusable in two-way infinite tilings, and the Lagae corner tile set is not aperiodic.

1 Introduction

Wang tiles, non-rotatable unit squares with colored edges, constitute a formalism introduced by Wang [1] to visualize the $\forall \exists \forall$ decidability problem of predicate calculus. Formulating an equivalent domino problem, Wang considered an infinite number of copies of an arbitrary set of Wang tiles and investigated whether there exists a simply-connected valid tiling of the infinite plane. Moreover, he conjectured in [2] that only the tile sets that form a torus, i.e., cover a periodic simply-connected rectangular domain with identical coloring of the opposite edges, generate infinite valid tilings. Berger [3] disproved the conjecture by finding a tile set that covers the infinite plane aperiodically by exploiting Kahr’s reduction of the Turing halting problem [4] to the origin-constrained domino problem [5]. Hence, the domino problem was proven to be undecidable and, consequently, no general finite algorithm for producing infinite valid tilings exists.

Far less attention has been paid to the finite version of the domino problem, bounded tiling, i.e., searching for a fixed-sized valid tiling generated by an arbitrary tile set. Although the problem is known to be $NP$-complete in general, e.g., [7] or [8, Theorem 7.2.1], most of the available approaches exploit specific properties of particular tile sets, e.g., [9,10,11,12]. However, several closely related works address the tile packing problem for edge-matching puzzles, in which all

---

*Department of Mechanics, Faculty of Civil Engineering, Czech Technical University in Prague, Thákurova 7, 16000 Prague 6, Czech Republic
tiles from the set are placed exactly once; see, e.g., [13, 14, 15] and [16] for an approach aiming at the famous Eternity II puzzle.

In this work, we investigate the bounded Wang tiling problem in its full generality. To this goal, we first survey the most significant aperiodic tile sets in Section 1(a) and applications of Wang tiles in Section 1(b). In Section 1(c), we list available algorithms for generation of Wang tilings. Finally, our aims and contributions appear summarized in Section 1(d).

1.1 Aperiodic tile sets

The originally unexpected property of Wang tile sets—aperiodicity—resulted in a long-term competition among scientists in mathematical logic, computer science, discrete mathematics, and even recreational mathematicians to find the aperiodic tile set of the minimum cardinality [24, Chapter 11]. Starting from the Berger tile set containing 20,426 tiles in 1964 [17, 3], it took almost 50 years until the two sets of 11 tiles were found and proved to be minimal [31]; see Fig. 1 for a graphical overview of the selected historical developments.

In 1966, Läuchli sent to Wang an aperiodic set of 40 tiles over 16 colors, but it remained unpublished until 1975 [13]. Meanwhile, unaware of the Läuchli’s result, Knuth [21] simplified Berger’s set to 92 tiles over 26 colors; and Robinson developed sets of 104 and 52 tiles over 8 colors in 1967 [20], of 56 tiles over 12 colors in 1971 [23], and anticipated an existence of a set of 35 tiles [23].

In 1973, Penrose developed a new approach based on kite and dart tiling, leading to a set of 34 tiles [24]. Robinson, being in contact with Penrose, modified Penrose’s approach to reach

1964 20,426 and 104 Wang tiles by Berger [17, 3]
1966 40 Wang tiles over 16 colors by Läuchli [18]
1967 104 and 52 Wang tiles over 8 colors by Robinson [19, 20]
1968 92 Wang tiles over 26 colors by Knuth [21] (reducible to 86 tiles [22])
1971 56 Wang tiles over 12 colors and 35 tiles by Robinson [23]
1973 34 Wang tiles by Penrose [24]
1973 32 Wang tiles over 16 colors by Robinson [24]
1977 24 Wang tiles over 24 colors by Robinson [24]
1978 16 Wang tiles over 6 colors by Ammann [25]
1987 24 Wang tiles over 9 colors by Grünbaum and Shephard [24]
1995 64 Wang tiles by Senechal [26]
1996 14 Wang tiles over 6 colors by Kari [27]
1996 13 Wang tiles over 5 colors by Čulík [28]
1999 696 Wang tiles by Kari and Papasoglu [29] (deterministic tile set)
2008 104 Wang tiles by Ollinger [12]
2018 19 Wang tiles over 16 colors by Labbé [30] (self-similar tile set)
2021 11 Wang tiles over 5 and 4 colors by Jeandel and Rao [31]

Figure 1: List of aperiodic Wang tile sets.
a reduced set of 32 tiles over 16 colors [24]. Using the same technique together with Penrose rhombs tiling, Grünbaum and Shephard [24] obtained a set of 24 tiles over 9 colors in 1987.

Another two tile sets were discovered by Ammann. In 1978, he used the so-called Ammann bars to reach 16 tiles over 6 colors [25]. Building on the Ammann’s A2 tiling, see, e.g., [24], Robinson obtained a set of 24 tiles over 24 colors in 1977 [24].

Subsequent size reduction of the smallest aperiodic set occurred in 1996, when Kari [27] developed a new method based on Mealy machines multiplying Beatty sequences and presented a set of 14 tiles over 6 colors. Čulík [28], using the same approach, reduced the set even further to 13 tiles over 5 colors.

The search for the minimal aperiodic set was concluded by Jeandel and Rao [31], who used an enumeration approach to find aperiodic sets of 11 tiles over 4 and 5 colors and proved non-existence of an aperiodic set either containing 10 or fewer tiles or labeled by less than 4 colors.

In addition to the original Wang tiles, in 2006, Lagae and Dutré [32] described a subset of the Wang tiles, the corner tiles (we refer to Appendix A for their relation to edge-based Wang tiles), with the adjacency rules stored in the colored corners instead of the edges. In the same year, they constructed multiple aperiodic sets of corner tiles [33], out of which the set of 44 corner tiles over 6 colors was the smallest one. The set was further simplified by Nurmi [34] to 30 corner tiles over 6 colors and both were claimed to be aperiodic.

1.2 Applications of Wang Tiles

Thanks to the property of particular tile sets to generate aperiodic tilings, Wang tiles gained interest among several disciplines. Building on the original purpose of Wang tiles, proofs in the first-order logic [2], they were also used in cellular automata theory [35], topology, group theory [36], and symbolic dynamical systems [37].

In computer graphics, Stam [38] adopted Wang tiles to generate aperiodic textures. After Cohen et al. [9] recognized that stochastic nonperiodic tilings are easier to handle, the Wang-tile-based approach to generating seamless textures became popular, also including the generation of point patterns [9, 39].

In science, Wang tiles and other related aperiodic tilings served as the key tool for understanding the 5-fold symmetry of electron diffraction patterns of quasicrystals [40, 26]. Another application at the nanoscale involved molecular DNA-based realization of Wang tiles, introduced by Winfree et al. [41], which provided a self-assembly of biological nanostructures into aperiodic patterns. The self-assembly process of DNA Wang tiles also powered custom DNA-based computations [42], fueled by Turing completeness of Wang tiles [3, 18].

Beyond the nanoscale, Wang tiles have also been used for efficient compression [43] and reconstruction [44] of nonperiodic microstructures, speeding up numerical analyses of random heterogeneous materials [45, 46]. Furthermore, we have developed a bilevel optimization approach to design modular truss materials based on the corner Wang tiling formalism [47] and a clustering-based method for designing modular structures and mechanisms with continuum topology optimization [48]. Finally, Jilek et al. [49] developed a centimeter-scale self-assembly procedure building on the Wang tiling formalism.

1.3 Wang tiling generation algorithms

To the best of our knowledge, no general approaches to solving the bounded tiling problem have been reported in the literature; the only available results are specific to single families of tile
sets \([9, 10, 11, 12]\), or consider infinite thin strips \([31]\). In what follows, we describe the gist of three tiling algorithms: substitution-based, stochastic, and transducer-based.

**Substitution-based tiling algorithm** Given a tile set \(\mathcal{T}\), substitution is a map \(S : \mathcal{T} \mapsto \mathcal{Z}\) that assigns a tiling \(\mathcal{T}_k\) to each tile \(k \in \mathcal{T}\); we refer the reader to Section 2 for the definitions. Consequently, arbitrary-sized tilings follow from a placed tile \(k\) and a recursively applied substitution rule \([12]\). Hence, the tiling “grows” iteratively. Clearly, such a procedure has a low complexity, but only very specific tile sets allow for such substitution rule that generates valid tilings.

**Stochastic tiling algorithms** In computer graphics, Wang tiles have mostly been used for generating visually appealing yet compressed textures. For this, it is essential to generate these nonperiodic patterns quickly, which is best achieved with stochastic tile sets—usually containing all combinations of edge labels for a given number of colors. For example, in the stochastic tiling algorithm \([9]\), the tiling is generated row-wise, such that the neighbor of any tile that has already been placed can always be selected from at least two tiles at random. This approach was further extended towards the hash-based direct stochastic tiling algorithm \([11]\). Note that stochastic algorithms enable straightforward enforcement of several tile- or edge-based constraints.

**Transducer-based tiling algorithm** The transducer-based tiling algorithm \([31]\) builds on the fact that the 1D domino problem is decidable and can be solved in a polynomial time because the bi-infinite path is formed by an arbitrary cycle in transducer graphs, see Section 2 for clarification. To generate valid tilings of multiple rows, the product of several transducers must be computed. Hence, we must enumerate all feasible valid tilings for a requested height and unit width, and then find a path of the given length in the transducer graph of the just-formed tile set. Obviously, this approach works well for tiling thin strips; however, it is impractical for larger nearly-square domains.

## 1.4 Aims and novelty

In this contribution, we consider the bounded Wang tiling in its general form, thereby allowing arbitrary tile sets and control over the resulting tilings. As follows from the above state-of-the-art survey, no such method has been published yet.

We believe that development of such algorithms is important from multiple reasons. First, we have already investigated modeling and optimization of non-periodic and stochastic microstructures with Wang tilings, e.g., \([43, 44, 45, 50, 46, 47]\). We hope that the extension of our methods to more general tile sets would enable characterizing a broader class of non-periodic conventional materials and meta-materials \([51, 52, 53]\) and thus improve upon the performance of optimized designs.

Apart from emerging applications in materials engineering, we believe that developing a unified methodology is of independent interest, e.g., for the verification of the results available in the literature. Here we justify this claim by finding two errors in well-established aperiodic tile sets.

To do this, we first provide the necessary definitions in Section 2 to make the manuscript self-contained. The subsequent part is devoted to four integer programming formulations for generation of valid tilings: decision variant in Section 3(a) maximum rectangular valid tiling in Section 3(b), maximum-cover in Section 3(c), and maximum adjacency constraint satisfaction in Section 3(d). To allow for a finer control over the resulting tilings, we also include simple
extensions to prescribe tile- and color-based boundary conditions, periodic constraints, and the tile-packing constraint in Section 5(e). We summarize our results in Section 6.

Due to the complexity of the proposed formulations, in Section 4 we propose a heuristic graph-based algorithm to tackle the maximum-cover optimization variant from Section 5(c). The developed algorithm relies on solutions to shortest path problems in directed acyclic graphs, hence possesses a low asymptotic complexity. Further, we show that a slight modification maintains an approximation ratio of 2/3 for the tile sets whose transducer graphs are cyclic.

Section 5 collects results from the computational assessment of the integer programming formulations (Section 5(a)) and heuristics (Section 5(b)), and on the benchmarking of the periodic tile packing formulation against the algorithm of Lagae and Dutré [14] (Section 5(c)). We close the section with two surprising observations found with integer programming for two well-known aperiodic tile sets: the Knuth [21] tile set of 92 tiles contains a tile unusable in infinite simply-connected valid tilings, Section 5(d), and the Lagae et al. [33] tile set of 44 corner tiles is not aperiodic, Section 5(e). We summarize our results in Section 6.

2 Notation and preliminaries

Considering a finite set of color codes \( \mathcal{C} = \{1, 2, \ldots, n_c\} \subset \mathbb{N} \), the (Wang) tile \( k \) is a quadruple of the color codes \((c^n_k, c^w_k, c^s_k, c^e_k)\), with \( c^n_k, c^w_k, c^s_k, \) and \( c^e_k \) standing for the color codes of the north, west, south, and east edge of the tile \( k \), respectively. Tiles can, therefore, be represented graphically as non-rotatable squares shown in Fig. 2a. Without loss of generality, we further consider these squares to be of the unit size.

A tile set \( \mathcal{T} \) represents a finite collection of \( n_t \) tiles, see Fig. 2b. When \( \forall (c^n, c^w, c^s, c^e) \in \mathcal{C}^4 : (c^n, c^w, c^s, c^e) \in \mathcal{T} \), we call the tile set complete.

Using the notation \( \bullet = \bullet \cap \mathbb{Z}^2 \) to denote an intersection of the set \( \bullet \) with the integer lattice points, tiling \( \mathcal{T}^A \) of a bounded domain \( A \in \mathbb{R}^2 \) is an arrangement of copies of the tiles from \( \mathcal{T} \) such that the tiles are placed at \( \mathcal{A} \), and cover the entire domain \( A \), cf. Fig. 3. More formally, tiling is a mapping \( \mathcal{T}^A : \mathcal{A} \rightarrow \mathcal{T} \) assigning a single tile \( k \in \mathcal{T} \) to every coordinate \((i, j) \in \mathcal{A} \). Consequently, we call tilings \( \mathcal{T}^A \) simply connected iff the domain \( A \) is so.

The tiling \( \mathcal{T}^A \) is rectangular if \( \forall i \in \mathcal{H}, \mathcal{H} = \{1, \ldots, n_h\} \), and \( \forall j \in \mathcal{W}, \mathcal{W} = \{1, \ldots, n_w\} \), it holds that \((i, j) \in \mathcal{A} \). Here, \( \mathcal{H} \) and \( \mathcal{W} \) are the sets of the height and width coordinates.

A valid tiling (Wang tiling) of \( \mathcal{A} \), denoted by \( \mathcal{T}^A_{\text{valid}} \), is a tiling with equal color codes at the shared edges between all pairs of adjoining tiles. Therefore, the mapping \( \mathcal{T}^A_{\text{valid}} : \mathcal{A} \rightarrow \mathcal{T} \) satisfies, in addition to the requirements for \( \mathcal{T}^A \), the additional constraints

\[
\begin{align*}
\hat{c}^n_{\mathcal{T}^A_{\text{valid}}} (i,j) &= \hat{c}^n_{\mathcal{T}^A} (i+1,j), \quad \forall (i,j) \in \mathcal{A} : (i + 1, j) \in \mathcal{A}, \\
\hat{c}^w_{\mathcal{T}^A_{\text{valid}}} (i,j) &= \hat{c}^w_{\mathcal{T}^A} (i,j+1), \quad \forall (i,j) \in \mathcal{A} : (i, j + 1) \in \mathcal{A},
\end{align*}
\]

Figure 2: Graphical representation of (a) a Wang tile \( k \), and of (b) a tile set \( \mathcal{T} \).
provided that the axes are oriented accordingly to Fig. 3. If such a valid tiling exists, we say that the domain \( A \) admits a valid \( T \)-tiling, or that it is tileable by \( T \).

Consider that \( B \subseteq A \) and \( B_{\text{max rect}} \subseteq A \) are simply connected, rectangular, and \( T \)-tileable. Then, the maximum rectangular valid tiling \( T_{A, \text{max rect}} \) is a valid tiling of the domain \( B_{\text{max rect}} \), where \( \{ B_{\text{max rect}} \subseteq A, \forall B \subseteq A : |B_{\text{max rect}}| \geq |B| \} \). Here, the notation \(|*|\) denotes cardinality of the set \(*\).

The maximum cover \( T_{A, \text{max cov}} \) is a valid tiling of \( B_{\text{max cov}} \), where \( B \) and \( B_{\text{max cov}} \) are arbitrary \( T \)-tileable subdomains in \( A \) and \( \{ B_{\text{max cov}} \subseteq A, \forall B \subseteq A : |B_{\text{max cov}}| \geq |B| \} \).

A rectangular valid tiling is said to be periodic, if the color codes at the opposite sides of the rectangle match. If the valid tiling is not periodic, but the considered tile set allows for at least one periodic rectangular tiling, we call it nonperiodic. Finally, if no such periodic pattern exists and the tile set still allows for a valid tiling of the infinite plane, it is referred to as aperiodic. Similarly, the tile set \( T \) is periodic if it permits periodic valid tilings; and aperiodic if all feasible valid tilings are aperiodic.

Transducer graph \([27]\) \( G_{t,h} \) of the tile set \( T \) is a directed (multi-)graph representation of a Mealy machine without any initial nor terminal state. It consists of \(|C|\) states (graph vertices) and \(|T|\) transitions (directed edges) \( E_h \), where

\[
E_h := \bigcup_{k \in T} \left( c_k^w \xrightarrow{c_k^p} c_k^e \right).
\]

For the dual transducer graph \( G_{t,v} \), composed of the dual Wang tiles \([30]\) reflecting \( T \) along the major diagonal of the tiles, the edge set is defined as

\[
E_v := \bigcup_{k \in T} \left( c_k^n \xrightarrow{c_k^g} c_k^s \right).
\]

To illustrate the construction, we include a visual example in Fig. 4.

### 3 Integer programming formulations

In this section, we introduce four integer programming formulations for the generation of valid tilings. The first one, in Section 3(a), develops a decision variant. In the later sections, we investigate the maximum rectangular tiling (Section 3(b)), maximum cover (Section 3(c)), and
Let us now consider the fundamental problem of finding $\mathcal{T}_A$ or proving it does not exist. From now on, we restrict $A$ to be rectangular to simplify notation. However, the presented approach also extends to the general case.

To achieve this, we introduce $\forall (i,j,k) \in \mathcal{H} \times \mathcal{W} \times \mathcal{T}$ a binary decision variable $x_{i,j,k} \in \{0, 1\}$ denoting the placement of the tile $k$ at the $(i,j)$ coordinate such that

$$x_{i,j,k} = \begin{cases} 1 & \text{iff the tile } k \text{ lies at coordinate } (i,j), \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Consequently, mapping $\mathcal{T}_A(i,j)$ is expressed as

$$\mathcal{T}_A(i,j) = \sum_{k \in \mathcal{T}} k x_{i,j,k}, \tag{5}$$

together with the requirement that every $(i,j)$ coordinate is occupied by one tile,

$$\sum_{k \in \mathcal{T}} x_{i,j,k} = 1, \quad \forall (i,j) \in \mathcal{H} \times \mathcal{W}. \tag{6}$$

Similarly, the color codes of a tile placed at $(i,j)$ are expressed using the binary variables as

$$c_{n,\mathcal{T}_A(i,j)} = \sum_{k \in \mathcal{T}} c_{n,k} x_{i,j,k}, \quad \tag{7a} c_{n,\mathcal{T}_A(i,j)} = \sum_{k \in \mathcal{T}} c_{n,k} x_{i,j,k}, \tag{7c}$$

$$c_{w,\mathcal{T}_A(i,j)} = \sum_{k \in \mathcal{T}} c_{w,k} x_{i,j,k}, \quad \tag{7b} c_{w,\mathcal{T}_A(i,j)} = \sum_{k \in \mathcal{T}} c_{w,k} x_{i,j,k}. \tag{7d}$$

Inserting (7) into (1a) and (1b) leads to the horizontal and vertical adjacency constraints expressed in terms of the decision variables, as

$$\sum_{k \in \mathcal{T}} c_{n,k} x_{i,j,k} - \sum_{k \in \mathcal{T}} c_{n,k} x_{i+1,j,k} = 0, \quad \forall (i,j) \in \mathcal{H} \setminus \{n_h\} \times \mathcal{W}, \tag{8a}$$

$$\sum_{k \in \mathcal{T}} c_{w,k} x_{i,j,k} - \sum_{k \in \mathcal{T}} c_{w,k} x_{i,j+1,k} = 0, \quad \forall (i,j) \in \mathcal{H} \times \mathcal{W} \setminus \{n_w\}. \tag{8b}$$

Combining (4), (5), (6), and (8) then provides us with a complete binary linear programming representation of valid tiling $\mathcal{T}_A^\text{valid}$.
For computational reasons, it proved to be advantageous to organize the constraints according to the color codes:

\[
\sum_{k \in T} x_{i,j,k}[c_k^s = \ell] - \sum_{k \in T} x_{i+1,j,k}[c_k^n = \ell] = 0, \quad \forall (i,j,\ell) \in H \setminus \{n_h\} \times W \times C, \quad (9a)
\]

\[
\sum_{k \in T} x_{i,j,k}[c_k^e = \ell] - \sum_{k \in T} x_{i,j+1,k}[c_k^w = \ell] = 0, \quad \forall (i,j,\ell) \in H \times W \setminus \{n_w\} \times C, \quad (9b)
\]

where, in the Iverson notation, \(\sum x_{i,j,k} [c_k = \ell]\) expresses that \(x_{i,j,k}\) is added to the sum if and only if \(c_k = \ell\).

The constraint (9a) requires that the number of tiles at \((i,j)\) with the south edge colored by \(\ell\) equals to the number of tiles at \((i+1,j)\) with the north edge marked by the same \(\ell\), for all \(\ell \in C\).

Because of (6), there are either no tiles with the shared edge colored by \(\ell\), or there is a single tile at each of the coordinates with its common edge labeled by \(\ell\). Analogously to the vertical adjacency constraint, the horizontal constraint (9b) also enforces equality among the number of tiles at \((i,j)\) with the east edge colored by \(\ell\) and the number of tiles at \((i,j+1)\) having the west edge colored by identical \(\ell\).

Finally, combining (4), (6), and (9), while noticing that the constraints (6) naturally propagate with the adjacency constraints from the domain boundaries (compare (10d,10e) with (6)), leads to the binary programming formulation

\[
\text{find } x
\]

\[
\text{s.t. } \sum_{k \in T} x_{i,j,k}[c_k^s = \ell] - \sum_{k \in T} x_{i+1,j,k}[c_k^n = \ell] = 0, \quad \forall (i,j,\ell) \in H \setminus \{n_h\} \times W \times C, \quad (10a)
\]

\[
\sum_{k \in T} x_{i,j,k}[c_k^e = \ell] - \sum_{k \in T} x_{i,j+1,k}[c_k^w = \ell] = 0, \quad \forall (i,j,\ell) \in H \times W \setminus \{n_w\} \times C, \quad (10b)
\]

\[
\sum_{k \in T} x_{i,j,k} = 1, \quad \forall (i,j) \in \{1,n_h\} \times W, \quad (10c)
\]

\[
\sum_{k \in T} x_{i,j,k} = 1, \quad \forall (i,j) \in H \times \{1,n_w\}, \quad (10d)
\]

\[
x_{i,j,k} \in \{0,1\}, \quad \forall (i,j,k) \in H \times W \times T, \quad (10e)
\]

that provides a complete representation of the bounded tiling problem, i.e., all valid tilings solve the integer program, and conversely, all feasible solutions to (10) are valid tilings. Moreover, observe that the problem consists of two totally unimodular constraints if considered independently: (10c,10e) representing row tilings, and (10b,10d) being column tilings. When considered simultaneously, the resulting problem becomes \(\mathcal{NP}\)-complete [7,8].

3.2 Maximum rectangular valid tiling

When a solution to (10) cannot be found in an acceptable time period or when no such solution exists, one can resort to relaxing the requirement of a valid tiling of \(A\) and search for a valid tiling of the largest rectangular subdomain.

Without loss of generality, let us assume that the maximum rectangular valid tiling always contains an anchor tile placed at \((1,1)\), i.e.,

\[
\sum_{k \in T} x_{1,1,k} = 1. \quad (11)
\]
On the other hand, all the other coordinates may contain a tile or be empty, thus

$$\sum_{k \in T} x_{i,j,k} \leq 1, \quad \forall (i,j) \in \tilde{A} \setminus (1,1).$$  \hfill (12)

Let us now pick two vertically adjacent coordinates \((i,j)\) and \((i+1,j)\). If there is a tile \(q\) placed at \((i+1,j)\), another tile \(p\) has to be placed at \((i,j)\), as, otherwise, there is no simply-connected rectangular tiling containing both the tiles at \((1,1)\) and at \((i+1,j)\). Validity of the tiling also requires identical color codes at the shared edges. On the other hand, if no tile is placed at \((i+1,j)\), a coordinate \((i,j)\) may be either occupied or empty. The allowed and forbidden combinations are shown in Fig. 5a–5d. Formally stated in terms of the decision variables, these considerations are expressed as

$$\sum_{k \in T} x_{i,j,k}[c_k^p = \ell] - \sum_{k \in T} x_{i+1,j,k}[c_k^q = \ell] \geq 0, \quad \forall (i,j,\ell) \in H \setminus \{n_h\} \times W \times C. \hfill (13)$$

Similar arguments hold also for the coordinates \((i,j)\) and \((i,j+1)\), resulting in the constraints

$$\sum_{k \in T} x_{i,j,k}[c_k^e = \ell] - \sum_{k \in T} x_{i,j+1,k}[c_k^w = \ell] \geq 0, \quad \forall (i,j,\ell) \in H \times W \setminus \{n_w\} \times C. \hfill (14)$$

The allowed and forbidden combinations for this case are shown in Fig. 5e–5h.

The developed constraints (11)–(14) enforce simple connectedness; however, they do not guarantee that the resultant tiling will be rectangular. For any 4 adjacent tiles \(p, q, r,\) and \(s\) placed at \((i,j), (i+1,j), (i,j+1),\) and \((i+1,j+1),\) respectively, these constraints allow for the assemblies shown in Fig. 6. Because the combination (d) cannot appear in any simply-connected rectangular tiling, we must exclude it from the feasible set,

$$\sum_{k \in T} x_{i+1,j,k} + \sum_{k \in T} x_{i,j+1,k} - \sum_{k \in T} x_{i+1,j+1,k} \leq 1, \quad \forall (i,j) \in H \setminus \{n_h\} \times W \setminus \{n_w\}. \hfill (15)$$

Figure 5: Admissible tile placements (a)–(c) and (e)–(g), and forbidden placements (d) and (h) in the maximum rectangular valid tiling formulation.
Finally, combining Eqs. (4), (11), (12), (13), (14), and (15) together with an objective function to maximize \( |\tilde{B}_{\text{max rect}}| \) provides us with the binary maximum rectangular valid tiling optimization program

\[
\begin{align*}
\max \quad & \sum_{i \in \mathcal{H}} \sum_{j \in \mathcal{W}} \sum_{k \in \mathcal{T}} x_{i,j,k} \\
\text{s.t.} \quad & \sum_{k \in \mathcal{T}} x_{i,j,k} [c_k^p = \ell] - \sum_{k \in \mathcal{T}} x_{i+1,j,k} [c_k^p = \ell] \geq 0, \quad \forall (i, j, \ell) \in \mathcal{H} \setminus \{n_h\} \times \mathcal{W} \times \mathcal{C}, \quad (16b) \\
& \sum_{k \in \mathcal{T}} x_{i,j,k} [c_k^w = \ell] - \sum_{k \in \mathcal{T}} x_{i+1,j+1,k} [c_k^w = \ell] \geq 0, \quad \forall (i, j, \ell) \in \mathcal{H} \times \mathcal{W} \setminus \{n_w\} \times \mathcal{C}, \quad (16c) \\
& \sum_{k \in \mathcal{T}} x_{i+1,j,k} + \sum_{k \in \mathcal{T}} x_{i,j+1,k} - \sum_{k \in \mathcal{T}} x_{i+1,j+1,k} \leq 1, \quad \forall (i, j) \in \mathcal{H} \setminus \{n_h\} \times \mathcal{W} \setminus \{n_w\}, \quad (16d) \\
& \sum_{k \in \mathcal{T}} x_{1,1,k} = 1, \quad (16e) \\
& \sum_{k \in \mathcal{T}} x_{i,j,k} \leq 1, \quad \forall (i, j) \in \tilde{\mathcal{A}} \setminus (1, 1), \quad (16f) \\
& x_{i,j,k} \in \{0, 1\}, \quad \forall (i, j, k) \in \mathcal{H} \times \mathcal{W} \times \mathcal{T}. \quad (16g)
\end{align*}
\]

In contrast to (10), a feasible solution to the optimization program (16) can be found in a polynomial time, e.g., by tiling the first row or column of the 1D bounded tiling problem. However, finding an optimal solution to (16) is \( \mathcal{NP} \)-hard, because the optimization problem (16) is reducible to the decision version (10) by fixing the value of the objective function to \( |\tilde{A}| \), which enforces inequalities in (16b), (16c), and (16f), making the constraint (16d) redundant as a consequence.
3.3 Maximum cover

Another option for avoiding the infeasibility of (10) rests in neglecting the requirement of (simple) connectedness, hence allowing for a placement of empty tiles (voids). In this section, we therefore search the maximum cover of $A$, or equivalently a valid tiling of the (possibly disconnected) domain $B_{\text{max cov}} \subseteq A$. For the maximum cover formulation, we assume that any two adjacent tiles satisfy the edge-matching constraints of valid tilings, but these are also satisfied by any of the tile-void, void-tile, or void-void combination, where $\sum_{k \in T} x_{i,j,k} = 0$ for a void located at $(i, j) \in \tilde{A}$.

Thus, each coordinate $(i, j)$ is occupied either by a tile or a void, implying that

$$\sum_{k \in T} x_{i,j,k} \leq 1, \quad \forall (i, j) \in H \times W,$$

and the vertical and horizontal edge matching conditions become

$$\sum_{k \in T} x_{i,j,k} e_k^c = \ell + \sum_{k \in T} x_{i,j+1,k} e_k^w \neq \ell \leq 1, \quad \forall (i, j, \ell) \in H \times W \setminus \{ n_w \} \times C,$$

(18a)

$$\sum_{k \in T} x_{i,j,k} e_k^c = \ell + \sum_{k \in T} x_{i+1,j,k} e_k^w \neq \ell \leq 1, \quad \forall (i, j, \ell) \in H \setminus \{ n_h \} \times W \times C.$$

(18b)

Finally, the combination of Eqs. (17), (18a), (18b) with the objective function to maximize $|\tilde{B}_{\text{max cov}}|$ leads to the binary optimization problem

$$\begin{align*}
\max \quad & \sum_{i \in H} \sum_{j \in W} \sum_{k \in T} x_{i,j,k} \\
\text{s.t.} \quad & \sum_{k \in T} x_{i,j,k} e_k^c = \ell + \sum_{k \in T} x_{i,j+1,k} e_k^w \neq \ell \leq 1, \quad \forall (i, j, \ell) \in H \times W \setminus \{ n_w \} \times C, \\
& \sum_{k \in T} x_{i,j,k} e_k^c = \ell + \sum_{k \in T} x_{i+1,j,k} e_k^w \neq \ell \leq 1, \quad \forall (i, j, \ell) \in H \setminus \{ n_h \} \times W \times C, \\
& \sum_{k \in T} x_{i,j,k} \leq 1, \quad \forall (i, j) \in H \times W, \\
& x_{i,j,k} \in \{0, 1\}, \quad \forall (i, j, k) \in H \times W \times T.
\end{align*}$$

(19a)

(19b)

(19c)

(19d)

(19e)

The program (19) is trivially $\mathcal{NP}$-hard: Requiring the objective function (19a) to be at least $|\tilde{A}|$ implies that

$$\sum_{k \in T} x_{i,j,k} = 1, \quad \forall (i, j) \in H \times W,$$

(20)

i.e., all positions are occupied by a Wang tile. Moreover, (19b) and (19c) require all adjacent tiles to share the color codes at their common edges. Consequently, the resulting tiling is void-free and valid, and solves the $\mathcal{NP}$-complete bounded tiling problem.

3.4 Maximum adjacency constraints satisfaction

Because the decision problem (10) also constitutes a specific instance of the constraint satisfaction problem (CSP), another optimization variant comes from the formulation of the max-CSP problem, maximizing the number of satisfied clauses—color matches in our case.
Therefore, for each vertical and horizontal edge we introduce a new variable \( h_{i,j}^h \in \mathbb{R}_{\geq 0} \), where \((i, j) \in \mathcal{H} \times \mathcal{W} \setminus \mathcal{W}_w \), and \( h_{i,j}^h \in \mathbb{R}_{\geq 0} \), with \((i, j) \in \mathcal{H} \setminus n_h \times \mathcal{W} \), respectively. The adjacency constraints (9) are then relaxed by considering \( h \) instead. Indeed, if \( t \) is a binary function to maximize the number of color matches yields the binary optimization problem among the tiles at \((i, j)\) for \((i, j) \in \mathcal{H} \times \mathcal{W} \setminus \mathcal{W}_w \) respectively. The valid tilings. Thus, in this section, we state some simple extensions to enforce tile- and color-based boundary conditions to solve the tile packing problem [14] and to enforce (variable-sized) periodic boundary conditions.

\[
\sum_{k \in \mathcal{T}} x_{i,j,k}[c_k^p = \ell] - \sum_{k \in \mathcal{T}} x_{i+1,j,k}[c_k^p = \ell] \leq h_{i,j}^h, \quad \forall (i, j, \ell) \in \mathcal{H} \setminus \{n_h\} \times \mathcal{W} \times \mathcal{C},
\]

(21a)

\[
\sum_{k \in \mathcal{T}} x_{i,j,k}[c_k^p = \ell] - \sum_{k \in \mathcal{T}} x_{i,j+1,k}[c_k^p = \ell] \leq h_{i,j}^v, \quad \forall (i, j, \ell) \in \mathcal{H} \times \mathcal{W} \setminus \{n_w\} \times \mathcal{C},
\]

(21b)

instead. Indeed, if \( h_{i,j}^h = 0 \), the edge-matching requirement of the neighboring tiles at \((i, j)\) and \((i + 1, j)\) is satisfied; and it is violated otherwise. Similarly, \( h_{i,j}^v = 0 \) guarantees color matches among the tiles at \((i, j)\) and \((i, j + 1)\).

Finally, rewriting absolute values in (21) by two linear inequalities while supplying an objective function to maximize the number of color matches yields the binary optimization problem

\[
\max_x \sum_{i \in \mathcal{H}} \sum_{j \in \mathcal{W} \setminus \mathcal{W}_w} (1 - h_{i,j}^v) + \sum_{i \in \mathcal{H} \setminus \{n_h\}} \sum_{j \in \mathcal{W}} (1 - h_{i,j}^h)
\]

(22a)

\[
\text{s.t. } \sum_{k \in \mathcal{T}} x_{i,j,k}[c_k^p = \ell] - \sum_{k \in \mathcal{T}} x_{i+1,j,k}[c_k^p = \ell] \leq h_{i,j}^h, \quad \forall (i, j, \ell) \in \mathcal{H} \setminus \{n_h\} \times \mathcal{W} \times \mathcal{C},
\]

(22b)

\[
\sum_{k \in \mathcal{T}} x_{i+1,j,k}[c_k^p = \ell] - \sum_{k \in \mathcal{T}} x_{i,j,k}[c_k^p = \ell] \leq h_{i,j}^v, \quad \forall (i, j, \ell) \in \mathcal{H} \setminus \{n_h\} \times \mathcal{W} \times \mathcal{C},
\]

(22c)

\[
\sum_{k \in \mathcal{T}} x_{i,j,k}[c_k^p = \ell] - \sum_{k \in \mathcal{T}} x_{i,j+1,k}[c_k^p = \ell] \leq h_{i,j}^v, \quad \forall (i, j, \ell) \in \mathcal{H} \times \mathcal{W} \setminus \{n_w\} \times \mathcal{C},
\]

(22d)

\[
\sum_{k \in \mathcal{T}} x_{i,j+1,k}[c_k^p = \ell] - \sum_{k \in \mathcal{T}} x_{i,j,k}[c_k^p = \ell] \leq h_{i,j}^v, \quad \forall (i, j, \ell) \in \mathcal{H} \times \mathcal{W} \setminus \{n_w\} \times \mathcal{C},
\]

(22e)

\[
\sum_{k \in \mathcal{T}} x_{i,j,k} = 1, \quad \forall (i, j) \in \mathcal{H} \times \mathcal{W},
\]

(22f)

\[
x_{i,j,k} \in \{0, 1\}, \quad \forall (i, j, k) \in \mathcal{H} \times \mathcal{W} \times \mathcal{T},
\]

(22g)

that is \( \mathcal{NP} \)-hard due to the reduction to [10] after setting all \( h_{i,j}^v \) and \( h_{i,j}^h \) to zeros. A feasible solution can be found in a polynomial time by finding valid row-column tilings for each row-column, so that either term \( \sum_{i \in \mathcal{H}} \sum_{j \in \mathcal{W} \setminus \mathcal{W}_w} h_{i,j}^v \) or \( \sum_{i \in \mathcal{H} \setminus \{n_h\}} \sum_{j \in \mathcal{W}} h_{i,j}^h \) equals zero.

### 3.5 Extensions

Up to now, we have focused solely on the (re)formulations of the bounded tiling problem, searching for arbitrary valid tilings. However, some potential applications may require finer control over the resulting tilings. Thus, in this section, we state some simple extensions to enforce tile- and color-based boundary conditions to solve the tile packing problem [14] and to enforce (variable-sized) periodic boundary conditions.
3.5.1 Tile-based boundary conditions

At first, we consider boundary conditions in the form of prescribed tiles. As the simplest one, we enforce the placement of a tile \( k \) at \((i, j)\):

\[
x_{i, j, k} = 1, \quad \{i, j, k\} \in H \times W \times T.
\]

(23)

Similarly, we may prevent tile \( k \) from being placed there:

\[
x_{i, j, k} = 0, \quad \{i, j, k\} \in H \times W \times T.
\]

(24)

Placement of an identical tile at the coordinates \((i, j)\) \in \(\tilde{A}\) and \((p, q)\) \in \(\tilde{A}\) requires

\[
x_{i, j, k} - x_{p, q, k} = 0, \quad \{i, p\} \in H, \{j, q\} \in W, \forall k \in T.
\]

(25)

Conversely, different tiles at these coordinates are secured with

\[
x_{i, j, k} + x_{p, q, k} \leq 1, \quad \{i, p\} \in H, \{j, q\} \in W, \forall k \in T.
\]

(26)

3.5.2 Color-based boundary conditions

In addition to the tile-based constraints, we may also enforce specific color codes for individual edges. To do this, the color of the north edge at \((i, j)\) \in \(\tilde{A}\) is set to \(\ell\) by

\[
\sum_{k \in T} x_{i, j, k} [c_{n}^{k} = \ell] = 1, \quad \{i, j, \ell\} \in H \times W \times C.
\]

(27)

On the contrary, we may prevent this color by requiring

\[
\sum_{k \in T} x_{i, j, k} [c_{n}^{k} = \ell] = 0, \quad \{i, j, \ell\} \in H \times W \times C.
\]

(28)

Further, the same color codes at the north edge of \((i, j)\) \in \(\tilde{A}\) and at the west edge of \((p, q)\) \in \(\tilde{A}\) are established with

\[
\sum_{k \in T} x_{i, j, k} [c_{n}^{k} = \ell] - \sum_{k \in T} x_{p, q, k} [c_{w}^{k} = \ell] = 0, \quad \{i, p\} \in H, \{j, q\} \in W, \forall \ell \in C,
\]

(29)

and a different color with

\[
\sum_{k \in T} x_{i, j, k} [c_{n}^{k} = \ell] + \sum_{k \in T} x_{p, q, k} [c_{w}^{k} = \ell] \leq 1, \quad \{i, p\} \in H, \{j, q\} \in W, \forall \ell \in C.
\]

(30)

3.5.3 Periodic tiling

In the domino problem, Wang \[1\] investigated the existence of tile sets admitting infinite aperiodic tilings. Here, we consider a similar setting for the finite domain \(A\): examining periodicity through periodic color-based boundary conditions.

We begin with requiring equal coloring at the fixed opposite domain boundaries,

\[
\sum_{k \in T} x_{1, j, k} [n_{k} = \ell] - \sum_{k \in T} x_{m, h, j, k} [s_{k} = \ell] = 0, \quad \forall (j, \ell) \in W \times C,
\]

(31a)
\[
\sum_{k \in T} x_{i,1,k}[w_k = \ell] - \sum_{k \in T} x_{i,n_t,w,k}[e_k = \ell] = 0, \quad \forall (i, \ell) \in \mathcal{H} \times \mathcal{C}.
\] (31b)

When adding (31) to the decision problem (10), we thus ask for an existence of a fixed-sized periodic Wang tiling.

In a natural generalization, we ask for an existence of finite-sized periodic Wang tilings, thus relying on the maximum rectangular valid tiling formulation (16). Naturally, the domain size is not known in this case. Therefore, we must consider \( \forall (i, \ell) \in \mathcal{H} \times \mathcal{W} \times \mathcal{C} \) constraints of the form

\[
\sum_{k \in T} x_{i,j,k}[e_k \neq \ell] + \sum_{k \in T} x_{i,1,k}[w_k = \ell] - \sum_{k \in T} x_{i,j+1,k}[j < n_t,w] \leq 1,
\] (32a)

\[
\sum_{k \in T} x_{i,j,k}[s_k \neq \ell] + \sum_{k \in T} x_{1,j,k}[n_k = \ell] - \sum_{k \in T} x_{i+1,j,k}[i < n_t,h] \leq 1.
\] (32b)

Here, (32a) prevents a color mismatch of the north edge of \((1, j) \in \tilde{\mathcal{A}}\) and the south edge of \((i, j) \in \mathcal{A}\) iff there is no tile placed at \((i, j + 1) \in \tilde{\mathcal{A}}\). Similarly, in the case of (32b), we prevent a color mismatch of the west edge at \((i, 1) \in \tilde{\mathcal{A}}\) and the east edge at \((i, j) \in \tilde{\mathcal{A}}\) iff the position \((i + 1, j) \in \tilde{\mathcal{A}}\) is empty.

Finally, when adding the constraints (32) to (16), we usually search for the smallest periodic pattern rather than the largest,

\[
\min \sum_{i \in \mathcal{H}} \sum_{j \in \mathcal{W}} \sum_{k \in T} x_{i,j,k}.
\] (33)

### 3.5.4 Tile packing problem

Our last extension constitutes the setting of the tile-packing problem [14]: we require each tile to be placed exactly once yet form a fixed-sized valid tiling,

\[
\sum_{i \in \mathcal{H}} \sum_{j \in \mathcal{W}} x_{i,j,k} = 1, \quad \forall k \in \mathcal{T}.
\] (34)

Note here that this extension requires that \( |\mathcal{T}| = |\tilde{\mathcal{A}}| \) as, otherwise, no solution exists.

#### 4 Heuristic algorithm for the maximum cover tiling problem

In the previous sections, we have introduced several integer programming formulations for the bounded Wang tiling problem and their extensions. Because of the asymptotic complexity of the integer programming formulations, we further develop a simple heuristic algorithm for one of the optimization variants, the maximum cover.

### 4.1 Maximum row cover tilings

Let us start with revising the decision program (10). In this formulation, neglecting any pair of the constraints (10b) with (10d) or (10c) with (10e) provides a totally unimodular constraint matrix, recall Section 3(a). Consequently, such simplified problems are deterministically solvable using the simplex method. Moreover, this setting agrees with the maximum flow problem [15], as (10d) and (10e)
correspond to the flow balances in the source and sink, whereas (10b) and (10c) correspond to the Kirchhoff law equations. Further complexity reduction is possible by recognizing the (shortest) path problem structure, since the source and sink capacities are equal to one, allowing only a single source-to-sink path with positive flow to emerge. Omitting any of these constraint pairs produces valid tilings of (finite) stripes, i.e., of rows or columns. However, the edges shared by the neighboring stripes may not comply with the edge matching rules. Starting with this observation, we first focus on an efficient approach to generate valid tilings of the rows.

As follows from Section 2, any valid tiling of a row can be visualized as a \(|W|\)-long path in the transducer graph \(G_{t.h.}\), recall Section 2. To simplify subsequent developments, we represent the row-tiling problem by a transducer-based directed acyclic graph (DAG) composed of \(|V| + 3\) vertex layers. While both the first and the last layer contain only a single vertex (the source \(s\) and terminal \(t\)), the intermediate layers include \(|C|\) vertices to represent the vertical (east and west) color codes of the tiles, i.e., the states in the transducer graph. The source vertex is connected to all vertices in the second layer, facilitating an arbitrary coloring of the west edge of the first tile, and, similarly, all the vertices in the penultimate layer are linked to the terminal to allow for all colors in the last east edge. The intermediate layers are bridged with the transducer edges \(E_h\); see Fig. 7. Consequently, any \(s \rightarrow t\) path in the yet-established directed graph forms a valid tiling of the row, and conversely, any valid tiling builds a \(s \rightarrow t\) path.

However, because such paths do not exist for tile sets that forbid a valid tiling of the row, we also need to incorporate voids. Clearly, we can add “void” tiles as edges that would interconnect the layers, i.e., any two consecutive layers would form a complete bipartite graph. However, such an approach requires adding at most \(|V||C|^2\) edges to the graph. Therefore, we add supplementary intermediate layers with a single vertex only, symbolizing the “void” tile type, and connect it to all vertices in the preceding and subsequent layer, see the dashed vertices and edges in Fig. 8. Consequently, we generate at most \(2|W||C|\) new edges altogether.

In addition, we assign unitary costs to the edges incoming to the void vertices and zero costs elsewhere. Hence, the \(s \rightarrow t\) path cost is equivalent to the number of voids in the row tiling. Furthermore, because the emergent graph is acyclic and single-sourced, the maximum row-cover tiling is found in \(O(|V| + |E|)\) time using the DAG-shortest-path algorithm [55], where \(V\) denotes the set of the graph vertices and \(E\) the set of the graph edges. In our case, we have

\[
|V| = 2 + (|W| + 1)|C| + |W| = 2 + |W| + |C| + |W||C|, \tag{35a}
\]

\[
|E| = 2|C| + 2|W||C| + |W||T|. \tag{35b}
\]
Thus, the overall asymptotic complexity to generate a maximum row cover tiling evaluates as

$$O(|V| + |E|) = O(2 + |V| + 3|C| + 3|V||C| + |V||T|) = O(|V||C| + |V||T|). \quad (36)$$

Interestingly, the running time (but not the asymptotic complexity) of the DAG-shortest-path algorithm can be improved by recognizing that the topological order of the graph vertices—which is required for the DAG-shortest-path algorithm—is known from the graph construction method in advance.

Any path with total cost $c_t$ contains exactly $c_t$ voids in the row tiling. Because the shortest path algorithm therefore minimizes the number of voids, it generates the maximum row cover as its output. These considerations are summarized below.

**Proposition 4.1.** The shortest path in the graph in Fig. 8 is equivalent to the maximum row cover.

### 4.2 Tiling consecutive rows

Assuming already covered rows $i-1$ and $i+1$, e.g., initially by voids, we aim to generate the maximum cover of the $i$-th row. Interestingly, this only requires a minor modification of the graph in Fig. 8.

For this, we first check the north-east compatibility for each tile $k \in T$ placed at $(i, j)$. Notice that the compatibility is never violated when the neighbors are voids. For color mismatch cases, we remove the edges denoting these incompatible tiles from the graph.

Assume that the rows $(i-1)$ and $(i+1)$ are voids. Then, clearly, inappropriate tiles at the $i$-th row may prevent the vertically-adjacent positions to be populated by tiles. To limit the appearance of such introduced voids, we include a small penalty of $\epsilon = 1/2(|V| + 1)^{-1}$ to the tiles that admit a single vertical neighbor only, and $\epsilon = (|W| + 1)^{-1}$ to tiles not admitting any vertical neighbor. Notice that these costs are selected such that, in the worst case, the total penalty due to these void-preventing weights amounts to $|W|/(|W| + 1) < 1$, i.e., the maximum number of tiles is placed even if the void positions forbid any vertical neighbors. Hence, Proposition 4.1 remains satisfied.

Consequently, we can build a simple heuristic algorithm, Alg. 1, that requires $|H|$ maximum row-cover iterations, rendering the overall complexity to be $O(|A||C| + |A||T|)$.

Although Alg. 1 usually generates relatively large ratio of the number of placed tiles $|B_{cov}|$ to $|A|$, it probably lacks a guaranteed lower bound. Such bounds can, however, be provided by fairly straightforward modifications introduced next.

![Figure 8: Transducer-based directed acyclic graph for computing the maximum row cover.](image-url)
Proposition 4.3. then reads as Alg. 2, allowing us to state the following, slightly stronger result:

When \( \epsilon = 1/2 \), we penalize the incompatibilities with the cost 2,.

Proof. Consider the maximum row-cover tiling of the odd rows of the initially void domain \( \mathcal{A} \) given in Section 4(a). Then, \( |\mathcal{B}_{\text{cov}}| \geq 1/2 |\mathcal{B}_{\text{max cov}}| \).

Proof. Consider that the maximum row-cover problem alone terminates with \( |\mathcal{B}_{\text{max cov}}| \) tiles. Based on the maximum row-cover property in Proposition 4.1, none of the rows of \( \mathcal{A} \) admit a tiling by more than \( |\mathcal{B}_{\text{max cov}}| \) tiles. Hence, we have \( |\mathcal{B}_{\text{cov}}| \geq 1/2 |\mathcal{H}| |\mathcal{B}_{\text{max cov}}| \) and \( |\mathcal{B}_{\text{max cov}}| \leq |\mathcal{H}| |\mathcal{B}_{\text{max cov}}| \), so that \( |\mathcal{B}_{\text{cov}}| \geq 1/2 |\mathcal{H}| |\mathcal{B}_{\text{max cov}}| \geq 1/2 |\mathcal{B}_{\text{max cov}}| \), where [•] rounds • to the nearest greater or equal integer.

To exploit Proposition 4.2 in Alg. 1 we modify the row processing order to \( \{1, 3, 2, 5, 4, \ldots \} \). Indeed, then each odd row contains exactly \( |\mathcal{B}_{\text{max cov}}| \) tiles. Nevertheless, covering the \( i \)-th (odd) row without acknowledging which tiles are placed in the \( (i - 1) \)-th row may result in an unnecessarily empty \( (i - 1) \)-th row. To avoid such situations, we do not check for compatibility with the \( (i - 1) \)-th row voids, but rather we check using the dual transducer graph with the tiles in the \( (i - 2) \)-th row. For each south color code in the \( (i - 2) \)-th row, we find admissible colors (states) in the dual transducer graph as the states reachable by an edge-long path. Indeed, the reached states are exactly the admissible north colors of compatible tiles in the \( i \)-th row. For the special case of voids in the \( (i - 2) \)-th row, all color codes are assumed to be compatible. Finally, we penalize the incompatibilities with the cost \( \epsilon = 1/2(|\mathcal{B}| + 1)^{-1} \) as before. The final algorithm then reads as Alg. 2 allowing us to state the following, slightly stronger result:

Proposition 4.3. Assume a tile set \( \mathcal{T} \) with the longest path in its transducer graph \( G_{t,h} \) of at least 2. Then, Alg. 2 terminates with \( |\mathcal{B}_{\text{cov}}| \geq 1/2 |\mathcal{B}| \).

Proof. When \( |\mathcal{B}_{\text{max cov}}| = |\mathcal{B}| \), the proof follows directly from Proposition 4.2. For the other cases, the odd rows must contain \( |\mathcal{B}_{\text{max cov}}| \) tiles due to Proposition 4.1. Because these row-covers are maximal, the sequence of consecutive voids in these rows cannot exceed two, as we could have placed an additional tile otherwise, contradicting with the maximum row-cover property. Moreover, without loss of generality, the cost of the shortest path in the \( i \)-th row is at most \( |\mathcal{B}_{\text{max cov}}| + (|\mathcal{B}| - |\mathcal{B}_{\text{max cov}}|) \epsilon \), which occurs when the \( (i - 2) \)-th and \( i \)-th row have the same tile-void patterns. Because the longest void sequence is at most two and the longest path in \( G_{t,h} \) is at least two, we can always place tiles to the north of the voids of the \( i \)-th row.

4.3 \( 1/2 \)-approximation algorithm for general tile sets

In this section, we modify Alg. 1 to maintain the \( 1/2 \) approximation ratio. We start with the following observation:

Proposition 4.2. Consider the maximum row-cover tiling of the odd rows of the initially void domain \( \mathcal{A} \) given in Section 4(a). Then, \( |\mathcal{B}_{\text{cov}}| \geq 1/2 |\mathcal{B}_{\text{max cov}}| \).

Proof. Consider the maximum row-cover problem alone terminates with \( |\mathcal{B}_{\text{max cov}}| \) tiles. Based on the maximum row-cover property in Proposition 4.1, none of the rows of \( \mathcal{A} \) admit a tiling by more than \( |\mathcal{B}_{\text{max cov}}| \) tiles. Hence, we have \( |\mathcal{B}_{\text{cov}}| \geq 1/2 |\mathcal{H}| |\mathcal{B}_{\text{max cov}}| \) and \( |\mathcal{B}_{\text{max cov}}| \leq |\mathcal{H}| |\mathcal{B}_{\text{max cov}}| \), so that \( |\mathcal{B}_{\text{cov}}| \geq 1/2 |\mathcal{H}| |\mathcal{B}_{\text{max cov}}| \geq 1/2 |\mathcal{B}_{\text{max cov}}| \), where [•] rounds • to the nearest greater or equal integer.

To exploit Proposition 4.2 in Alg. 1 we modify the row processing order to \( \{1, 3, 2, 5, 4, \ldots \} \). Indeed, then each odd row contains exactly \( |\mathcal{B}_{\text{max cov}}| \) tiles. Nevertheless, covering the \( i \)-th (odd) row without acknowledging which tiles are placed in the \( (i - 1) \)-th row may result in an unnecessarily empty \( (i - 1) \)-th row. To avoid such situations, we do not check for compatibility with the \( (i - 1) \)-th row voids, but rather we check using the dual transducer graph with the tiles in the \( (i - 2) \)-th row. For each south color code in the \( (i - 2) \)-th row, we find admissible colors (states) in the dual transducer graph as the states reachable by an edge-long path. Indeed, the reached states are exactly the admissible north colors of compatible tiles in the \( i \)-th row. For the special case of voids in the \( (i - 2) \)-th row, all color codes are assumed to be compatible. Finally, we penalize the incompatibilities with the cost \( \epsilon = 1/2(|\mathcal{B}| + 1)^{-1} \) as before. The final algorithm then reads as Alg. 2 allowing us to state the following, slightly stronger result:

Proposition 4.3. Assume a tile set \( \mathcal{T} \) with the longest path in its transducer graph \( G_{t,h} \) of at least 2. Then, Alg. 2 terminates with \( |\mathcal{B}_{\text{cov}}| \geq 1/2 |\mathcal{B}| \).

Proof. When \( |\mathcal{B}_{\text{max cov}}| = |\mathcal{B}| \), the proof follows directly from Proposition 4.2. For the other cases, the odd rows must contain \( |\mathcal{B}_{\text{max cov}}| \) tiles due to Proposition 4.1. Because these row-covers are maximal, the sequence of consecutive voids in these rows cannot exceed two, as we could have placed an additional tile otherwise, contradicting with the maximum row-cover property. Moreover, without loss of generality, the cost of the shortest path in the \( i \)-th row is at most \( |\mathcal{B}_{\text{max cov}}| + (|\mathcal{B}| - |\mathcal{B}_{\text{max cov}}|) \epsilon \), which occurs when the \( (i - 2) \)-th and \( i \)-th row have the same tile-void patterns. Because the longest void sequence is at most two and the longest path in \( G_{t,h} \) is at least two, we can always place tiles to the north of the voids of the \( i \)-th row.

Algorithm 1 Simple maximum cover heuristics

1: function SIMPLEMAXIMUMCOVERHEURISTICS(\( \mathcal{T}, \mathcal{A} \))
2: \( \mathcal{T} \leftarrow \text{initializeVoidTiling}(\mathcal{A}) \)
3: \( G_{t,h} \leftarrow \text{getTransducerGraph}(\mathcal{T}) \)
4: for row \( \leftarrow \{1, \ldots, |\mathcal{H}|\} \) do
5: \( G_{\text{DAG}} \leftarrow \text{constructWeightedDAG}(G_{t,h}, \mathcal{T}, \mathcal{T}, \text{row}) \)
6: \( \text{shortestPath} \leftarrow \text{solveDAGShortestPathProblem}(G_{\text{DAG}}) \)
7: \( \mathcal{T} \leftarrow \text{updateTiling}(\mathcal{T}, \text{shortestPath}, \text{row}) \)
8: end for
9: return \( \mathcal{T} \)
10: end function
4.4 2/3-approximation algorithm for tilesets with cyclic transducers

Another improvement in the approximation factor of Alg. 2 is possible for tile sets with all the states in the transducer graphs $G_{t,h}$ and $G_{t,v}$ being in at least one graph cycle. Notice that this situation occurs for all tile sets that tile the infinite plane.

To this goal, we modify the assignment of costs to graph, and the row processing order to $\{1, 4, 3, 2, 3, 7, 6, 5, 6, \ldots\}$. We begin with (i) tiling the maximum row-cover of the first row. Then, we (ii) find the maximum row-cover of the 4th row such that we penalize possible incompatibilities with the first row based on the dual transducer graph by $\epsilon$. The step (iii) encompasses finding a cover of the 3rd row with penalized incompatibilities with the first row and enforced voids at even positions. Finally, we find the maximum covers for rows 2 and 3. We repeat the procedure for the row numbers iteratively increased by 3, see Alg. 3. Then, we can make the following statement:

**Lemma 4.1.** Consider that all states in the transducer graphs $G_{t,h}$ and $G_{t,v}$ are in at least one graph cycle. Then, Alg. 3 terminates with at least $\frac{2}{3}|\tilde{A}|$ placed tiles.

**Proof.** Since the tile set allows for valid tiling of the row, the $\{1, 4, \ldots\}$ rows are occupied by exactly $|V|$ tiles. The $\{3, 6, \ldots\}$ rows are then populated by at least $\frac{1}{2}|V|$ tiles because each tile from rows $\{4, 7, \ldots\}$ admits a vertical neighbor. Finally, the $\{2, 5, \ldots\}$ rows contain at least the complement of the number of tiles used in the preceding row, because the tiles in the $\{1, 4, 6, \ldots\}$ row admit a south neighbor. Depending on the number of rows, the algorithm places at least

$$|\tilde{B}_{cov}| \geq \min\{|\tilde{A}|, \frac{3}{4}|\tilde{A}|, \frac{2}{3}|\tilde{A}|, \frac{3}{4}|\tilde{A}|, \frac{7}{10}|\tilde{A}|, \frac{2}{3}|\tilde{A}|, \ldots\} = \frac{2}{3}|\tilde{A}|$$

tiles. \qed

4.5 Iterative improvements

Similarly to finding the maximum row covers, we can search for the maximum cover of columns. When combining these two methods, we end up with our final algorithm that has the $O(|\tilde{A}|^2|C| +$...
Proposition 4.4. Alg. 4 runs in a polynomial time and terminates in a finite number of steps.

Proof. We have already shown that finding a maximum row-cover has $O(|W||C| + |W||T|)$ complexity. Further, finding the $2\times$-long paths in the transducer graph possesses the $|C|^2$ complexity and can be run only once prior to the algorithm main loop. Altogether, Alg. 3 requires at most $4/3|H|$ inner iterations so that we have the $O(|\tilde{A}|^2|C| + |\tilde{A}|^2|T| + |C|^2)$ overall complexity.

Regardless of the method at line 2 of Alg. 4, the improving loop runs at most $|\tilde{A}|$ times. Consequently, the algorithm is finite and possesses the $O(|\tilde{A}|^2|C| + |\tilde{A}|^2|T| + |C|^2)$ complexity.

5 Results

Having developed several exact and heuristic methods, this section is devoted to their numerical examination. We begin with assessing the performance of the integer programming formulations in Section 5(a). Then, in Section 5(b) we also relate these results to the outputs of the heuristic algorithms.

**Algorithm 3 2/3-approximation algorithm**

```plaintext
function MAXIMUM_COVER_APPROXIMATION067(T, A)
    $\tilde{T} \leftarrow$ initializeVoidTiling(A)
    $G_{t,h}, G_{t,v} \leftarrow$ getTransducerGraphs(T)
    setRowsNotVisited()
    for row $\leftarrow \{1, 4, 3, 2, 3, 7, 6, 5, 6, \ldots \}$ do
        if mod(row−1,3)== 0 then
            $G_{DAG} \leftarrow$ constructWeightedDAGFromDTransducer($G_{t,h}, G_{t,v}, \tilde{T},$ row, 2)
        else if mod(row−2,3)== 0 then
            if rowVisited(row)==false then
                $G_{DAG} \leftarrow$ constructWeightedDAGFromDTransducer($G_{t,h}, G_{t,v}, \tilde{T},$ row, 1)
            $G_{DAG} \leftarrow$ removeTilesAtEvenPositions($G_{DAG}$)
        else
            $G_{DAG} \leftarrow$ constructWeightedDAG($G_{t,h}, \tilde{T}, \tilde{T},$ row)
        end if
    end for
    return $\tilde{T}$
end function
```
Extensions of the integer programs are investigated in subsequent sections. First, we demonstrate the usefulness of the packing constraint by comparing the efficiency of the solution to the tile-packing problem using our method with the times reported by Lagae and Dutré [14], Section 5(c). Subsequently, we also present two unexpected discoveries revealed when testing formulations: the Knuth [21] tile set contains a tile unusable in infinite tilings, Section 5(d), and the Lagae et al. [33] tile set of 44 corner tiles lacks aperiodicity, Section 5(e).

We implemented all the methods described above in C++. As the integer programming solver, we used the state-of-the-art optimizer Gurobi 9.5.0 [56] dynamically linked to the compiled binary. Numerical tests were evaluated on a personal laptop running the Ubuntu 18.04 operating system equipped with 24 GB of RAM and Intel® Core® i5-8350U CPU clocked at 1.70GHz.

5.1 Integer programming formulations

In this section, we investigate the performance of all integer programming formulations from Section 3, i.e., the decision program (10), the maximum rectangular tiling (16), the maximum cover (19), and the maximum adjacency constraint satisfaction problem (22).

We are unaware of any standard sets for bounded tiling problems except for the specific, mostly aperiodic tile sets listed in the literature, recall Section 1(a). Hence, we consider a set of benchmark problems consisting of five aperiodic tile sets (11 tiles over 4 colors by Jeandel and Rao [31], 13 tiles over 5 colors by Čulík [28], 14 tiles over 6 colors by Kari [27], 16 tiles over

Algorithm 4 Final maximum cover heuristics

```
1: function FINALMAXIMUMCOVERHEURISTICS(\mathcal{T}, \mathcal{A})
2: \mathcal{I} ← \text{generateInitialCover}(\mathcal{T}, \mathcal{A})
3: \mathcal{G}_{t,b}, \mathcal{G}_{t,v} ← \text{getTransducerGraphs}(\mathcal{T})
4: numVoidsOld ← \infty
5: method ← “columns”
6: while numVoidsOld − getNumVoids(\mathcal{I}) > 0 do
7: numVoidsOld ← getNumVoids(\mathcal{I})
8: if method == “rows” then
9:  for row ← \{1, \ldots, |\mathcal{H}|\} do
10: \mathcal{G}_{\text{DAG}} ← \text{constructSimpleDAG}(\mathcal{G}_{t,b}, \mathcal{I}, \text{row})
11: shortestPath ← \text{solveDAGShortestPathProblem}(\mathcal{G}_{\text{DAG}})
12: \mathcal{I} ← \text{updateTiling}(\mathcal{I}, \text{shortestPath, row})
13: method ← “columns”
14: end for
15: else
16:  for column ← \{1, \ldots, |\mathcal{W}|\} do
17: \mathcal{G}_{\text{DAG}} ← \text{constructSimpleDAG}(\mathcal{G}_{t,v}, \mathcal{I}, \text{column})
18: shortestPath ← \text{solveDAGShortestPathProblem}(\mathcal{G}_{\text{DAG}})
19: \mathcal{I} ← \text{updateTiling}(\mathcal{I}, \text{shortestPath, column})
20: method ← “rows”
21: end for
22: end if
23: end while
24: return \mathcal{I}
25: end function
```
Figure 9: New tile sets (a) Finite1 of 7 tiles over 4 colors, and (b) Finite2 of 16 tiles over 16 colors used in our algorithmic tests.

6 colors by Ammann [24], and 56 tiles over 12 colors by Robinson [23]), two stochastic tile sets introduced in computer graphics (8 tiles over 2 colors by Cohen et al. [9] and a set of 16 tiles over 4 edge colors by Lagae and Dutré [32]), two periodic tile sets (10 tiles over 4 colors by Wang [18] and the set of 30 tiles over 17 edge colors by Lagae et al. [33] and Nurmi [34]). In addition, in Fig. 9 we introduce two tile sets that do not allow for a valid tiling of the infinite domain.

For all these tile sets, we aimed at generating valid tilings sized, respectively, 20 × 20, 25 × 25, and 30 × 30. The running time of the Gurobi solver was limited to 300 seconds for the single-threaded mode.

The results shown in Tab. 1 illustrate that the performance of the decision program (10) surpasses any of the candidate variants. However, it failed to find an existent feasible solution in the time limit four times. In these cases, the output of the optimization problems 16, 19, 22.

| Tile set        | Size  | Dec. prog. [10] | Max. rect. [16] | Max. cov. [19] | Max. CSP [22] |
|-----------------|-------|----------------|----------------|---------------|---------------|
| Aperiodic1 (11/4) [21] | 20 × 20 | 0.111 | 300.070 *150 | 300.083 *606 | 300.071 *1135 |
|                | 25 × 25 | 90.810 | 300.070 *125 | 300.082 *612 | 300.078 *1184 |
|                | 30 × 30 | 300.069 *infeasible | 300.097 *861 | 300.089 *1628 |
| Aperiodic2 (13/5) [25] | 20 × 20 | 0.114 | 145.300 *400 | 300.055 *399 | 300.059 *742 |
|                | 25 × 25 | 178.357 | 300.070 *125 | 300.082 *612 | 300.078 *1184 |
|                | 30 × 30 | 300.069 *infeasible | 300.097 *861 | 300.089 *1628 |
| Aperiodic3 (14/6) [27] | 20 × 20 | 275.359 | 181.171 *400 | 300.058 *397 | 300.058 *752 |
|                | 25 × 25 | 300.057 *infeasible | 300.072 *100 | 300.086 *619 | 300.086 *1178 |
|                | 30 × 30 | 300.073 *infeasible | 300.092 *90 | 300.107 *863 | 300.104 *1610 |
| Aperiodic4 (16/6) [28] | 20 × 20 | 0.142 | 171.136 *400 | 176.584 *400 | 71.141 *760 |
|                | 25 × 25 | 0.196 | 300.063 *100 | 300.251 *577 | 300.085 *1030 |
|                | 30 × 30 | 0.251 | 300.265 *60 | 300.132 *794 | 300.115 *1616 |
| Aperiodic5 (56/12) [23] | 20 × 20 | 0.440 | 300.107 *20 | 300.214 *350 | 302.442 *688 |
|                | 25 × 25 | 0.400 | 300.155 *25 | 300.354 *553 | 300.197 *1055 |
|                | 30 × 30 | 0.648 | 300.228 *30 | 300.434 *795 | 301.102 *1569 |
| Stochastic1 (8/2) [29] | 20 × 20 | 0.066 | 0.101 *400 | 0.046 *400 | 4.195 *760 |
|                | 25 × 25 | 0.091 | 0.125 *625 | 0.089 *625 | 5.508 *1200 |
|                | 30 × 30 | 0.116 | 0.225 *900 | 0.110 *900 | 10.021 *1740 |
| Stochastic2 (16/4) [33] | 20 × 20 | 0.114 | 0.107 *400 | 0.129 *400 | 3.226 *760 |
|                | 25 × 25 | 0.141 | 0.175 *625 | 0.210 *625 | 6.118 *1200 |
|                | 30 × 30 | 0.183 | 0.237 *900 | 0.283 *900 | 6.846 *1740 |
| Periodic1 (10/4) [13] | 20 × 20 | 0.121 | 107.454 *400 | 117.982 *400 | 34.656 *760 |
|                | 25 × 25 | 0.153 | 274.813 *625 | 300.066 *584 | 224.734 *1200 |
|                | 30 × 30 | 0.193 | 300.977 *81 | 302.606 *824 | 300.087 *1628 |
| Periodic2 (30/17) [34] | 20 × 20 | 0.236 | 109.800 *400 | 252.700 *400 | 88.721 *1200 |
|                | 25 × 25 | 0.325 | 300.103 *125 | 300.222 *545 | 300.150 *1017 |
|                | 30 × 30 | 0.473 | 300.158 *30 | 300.300 *786 | 300.204 *1521 |
| Finite1 (7/4) [21] | 20 × 20 | 0.066 | infeasible | 300.025 *120 | 300.051 *378 | 300.076 *725 |
|                | 25 × 25 | 0.086 | 300.038 *125 | 300.054 *585 | 300.061 *1108 |
|                | 30 × 30 | 0.105 | 300.046 *108 | 300.069 *826 | 300.080 *1628 |
| Finite2 (16/16) | 20 × 20 | 0.100 | infeasible | 300.273 *40 | 300.207 *326 | 300.077 *684 |
|                | 25 × 25 | 0.133 | 300.067 *50 | 300.111 *493 | 300.094 *1029 |
|                | 30 × 30 | 0.168 | 300.084 *30 | 300.131 *690 | 300.128 *1525 |

Table 1: Benchmark results. Values marked by an asterisk denote a premature termination of the integer programming solver.
provided at least some output. Interestingly, the decision problem (10) also was more efficient in the case of proving that the domain $|A|$ lacks $T$-tilability.

Comparison of the optimization variants hints that the maximum cover (19) and the maximum adjacency constraint satisfaction (22) problems scale better than the maximum rectangular tiling (16). Indeed, generating any smaller rectangular domain remains $NP$-complete, preventing any polynomial-time approximation algorithm to exist. On the other hand, both the formulations (19) and (22) admit simple heuristics, recall Section 3 allowing the solver to obtain higher-quality feasible solutions faster.

### 5.2 Heuristic algorithms

Second, we compare the performance of the maximum cover formulation (19) solved with the heuristic Alg. 4 supplied with three different initial coverings, i.e., based on Algs. 1, 2 and 3.

Alg. 4 ran sequentially. In order to limit the dependence of the heuristic algorithm on the ordering of tiles, we randomized the edge order in the directed acyclic graphs. Thus, we evaluated Alg. 4 100 times for each of the tested option, and listed the best, worst, and mean results in Tab. 2.

From Tab. 2, it follows that the initialization with the cover from Alg. 1 is the most efficient for the tested tile sets, both in terms of speed and performance. The remaining two initializations seem to be fairly comparable on average. While for Alg. 1 at least 82% of tiles were always placed, only more than 60% followed from Alg. 2. Using Alg. 3, we obtained at least 70% tile

| Tile set         | Size | Alg. 4            | Alg. 4            | Alg. 4            |
|------------------|------|-------------------|-------------------|-------------------|
|                  |      | with Alg. 1        | with Alg. 2        | with Alg. 3       |
|                  |      | $t$ ($|S|$ | $t$ ($|S|$ | $t$ ($|S|$ |
|                  |      | min | avg | max | min | avg | max | min | avg | max |
| Aperiodic1 (11/4) | 20x20 | 0.024 | 0.049 | 0.065 | 0.018 | 0.010 | 0.006 | 0.008 | 0.009 | 0.016 | 0.018 | 0.020 |
|                  | 25x25 | 0.034 | 0.065 | 0.102 | 0.021 | 0.014 | 0.007 | 0.010 | 0.011 | 0.018 | 0.020 | 0.022 |
|                  | 30x30 | 0.044 | 0.103 | 0.146 | 0.024 | 0.017 | 0.010 | 0.013 | 0.015 | 0.023 | 0.025 | 0.026 |
| Aperiodic2 (13/5) | 20x20 | 0.034 | 0.066 | 0.110 | 0.022 | 0.014 | 0.008 | 0.010 | 0.013 | 0.021 | 0.024 | 0.025 |
|                  | 25x25 | 0.044 | 0.085 | 0.132 | 0.025 | 0.016 | 0.010 | 0.014 | 0.017 | 0.025 | 0.028 | 0.030 |
|                  | 30x30 | 0.054 | 0.104 | 0.156 | 0.029 | 0.017 | 0.011 | 0.016 | 0.020 | 0.029 | 0.031 | 0.032 |
| Stochastic1 (8/2) | 20x20 | 0.014 | 0.025 | 0.037 | 0.009 | 0.012 | 0.006 | 0.011 | 0.014 | 0.022 | 0.025 | 0.027 |
|                  | 25x25 | 0.024 | 0.036 | 0.052 | 0.013 | 0.016 | 0.010 | 0.013 | 0.017 | 0.025 | 0.028 | 0.030 |
|                  | 30x30 | 0.034 | 0.047 | 0.064 | 0.017 | 0.018 | 0.012 | 0.015 | 0.020 | 0.028 | 0.031 | 0.033 |
| Stochastic2 (16/4) | 20x20 | 0.013 | 0.026 | 0.037 | 0.009 | 0.012 | 0.007 | 0.011 | 0.013 | 0.022 | 0.024 | 0.027 |
|                  | 25x25 | 0.023 | 0.035 | 0.050 | 0.012 | 0.015 | 0.010 | 0.013 | 0.017 | 0.025 | 0.028 | 0.030 |
| Periodic1 (10/4) | 20x20 | 0.024 | 0.036 | 0.051 | 0.013 | 0.018 | 0.011 | 0.014 | 0.018 | 0.027 | 0.029 | 0.031 |
|                  | 25x25 | 0.034 | 0.045 | 0.066 | 0.017 | 0.020 | 0.012 | 0.016 | 0.020 | 0.030 | 0.032 | 0.034 |
| Periodic2 (30/17) | 20x20 | 0.013 | 0.025 | 0.036 | 0.009 | 0.012 | 0.008 | 0.011 | 0.013 | 0.020 | 0.023 | 0.026 |
|                  | 25x25 | 0.024 | 0.034 | 0.049 | 0.013 | 0.017 | 0.011 | 0.015 | 0.019 | 0.028 | 0.031 | 0.034 |
|                  | 30x30 | 0.034 | 0.044 | 0.059 | 0.017 | 0.021 | 0.012 | 0.017 | 0.021 | 0.030 | 0.033 | 0.036 |
| Finetile1 (7/4)   | 20x20 | 0.023 | 0.035 | 0.046 | 0.013 | 0.017 | 0.011 | 0.015 | 0.019 | 0.028 | 0.030 | 0.033 |
|                  | 25x25 | 0.033 | 0.044 | 0.054 | 0.017 | 0.021 | 0.012 | 0.017 | 0.021 | 0.030 | 0.033 | 0.036 |
|                  | 30x30 | 0.043 | 0.054 | 0.065 | 0.021 | 0.025 | 0.013 | 0.019 | 0.024 | 0.033 | 0.035 | 0.038 |
| Finetile2 (16/16) | 20x20 | 0.016 | 0.028 | 0.039 | 0.011 | 0.014 | 0.009 | 0.012 | 0.016 | 0.025 | 0.028 | 0.030 |
|                  | 25x25 | 0.026 | 0.037 | 0.049 | 0.015 | 0.018 | 0.012 | 0.017 | 0.021 | 0.030 | 0.033 | 0.036 |
|                  | 30x30 | 0.036 | 0.047 | 0.059 | 0.019 | 0.023 | 0.014 | 0.019 | 0.023 | 0.032 | 0.035 | 0.038 |

Table 2: Numerical tests of the maximum-cover heuristics, Alg. 4 initialized based on Algs. 1, 2, and 3. Best mean runs are highlighted in bold.
When comparing Tab. 1 with Tab. 2, a few patterns emerge. First, the heuristic algorithm always generates valid tilings if (any of) the stochastic tile sets are used. For aperiodic and periodic tile sets, Gurobi required a considerably longer time to reach feasible solutions of a similar quality, but usually surpassed the developed algorithms in the time limit of 300 s. In the case of Alg. 1, it can be seen that the resulting covers are very competitive to the outputs of (19) and also obtained in much shorter times.

5.3 Periodic tile packing problem

As the second numerical example, we consider the periodic tile packing problem investigated in computer graphics applications [14]. Considering a complete edge tile set, Lagae and Dutr é searched for a periodic square valid tiling with each tile from the tile set used exactly once. Clearly, such tilings not only contain the entire (textural) information stored in individual tiles but also maintain compatibility with the traditional periodic arrangement.

| Tile set               | Time (10 31 34) | Time Lagae and Dutr é [14] |
|------------------------|-----------------|---------------------------|
| Stochastic edge (16/2) | < 1 sec.        | < 1 sec.                  |
| Stochastic edge (81/3) | < 1 sec.        | < 1 sec.                  |
| Stochastic edge (256/4)| 9 sec.          | 140 days                  |
| Stochastic edge (625/5)| 4 days          | -                         |

Table 3: Periodic tile packing problem: comparison of core times needed to find a single feasible solution by integer programming (second column) and by the backtracking method (third column) proposed in Lagae and Dutr é [14] to find a feasible solution.
While Lagae and Dutrée [14] proposed a backtracking-based algorithm to generate periodic packings, we rely here on a solution to the decision program (10) supplemented with the packing (34) and fixed periodicity (31) constraints. The resulting core times spent in the search for a single feasible solution (Tab. 3) illustrates the higher effectiveness of our method.

5.4 Unusable tile in the Knuth tile set

One of the oldest aperiodic tile sets, containing 92 tiles over 26 colors, is from Knuth [21, Exercise 5 in Section 2.3.4.3]. Generating valid tilings from the Knuth tile set using the decision program (10) together with the tile-based boundary conditions, recall Section 3(e)(i), led to an unexpected observation that enforced placement of the \( \beta US \) tile makes the program (10) infeasible under certain circumstances.

After a careful investigation, it indeed turned out that there is not any \( 2 \times 2 \) valid tiling with the \( \beta US \) tile placed at \((2, 2)\). Moreover, there is also not any \( 4 \times 3 \) valid tiling with the \( \beta US \) tile placed at \((3, 1)\). Thus, using the maximum-cover optimization variant (19) and the \( \beta US \) tile enforced at the respective coordinate, there are exactly 31 optimal solutions with the objective function equal to 3, and 498 optimal solutions with the objective function equal to 11.

Consequently, the \( \beta US \) tile can appear only in the strip of at most 3 consecutive infinite columns and does not allow for simply-connected valid tilings of the infinite plane. In a private communication, Knuth confirmed the issue and discovered another 5 tiles that are unnecessary but usable in infinite valid tilings, allowing for a possible reduction of the tile set to 86 tiles. For more information, we refer the interested reader to Knuth’s discussion about the reduced tile set [22, Exercise 221 in Section 7.2.2.1].

5.5 Periodicity of the Lagae corner tile set

Analogously to the Wang tiles, with the connectivity information stored in the edges, Lagae and Dutrée [32] introduced corner tiles with colored corners. As Wang [18] noted in 1975, these formalisms are interchangeable if the (infinite) domino problem is considered, because every set of Wang tiles can be represented by sets of corner tiles with greater or equal cardinality [33]. However, corner tiles avoid the so-called corner problem of Wang tiles in computer graphics [32], motivating Lagae et al. [33] to develop conversion methods for transforming Wang tiles to corner tiles, and vice versa. A direct product of these conversions are aperiodic tile sets of corner tiles [33].

Two of these methods, called horizontal and vertical translations, were used to convert the Ammann set of 16 Wang tiles over 6 colors [24] to the set of 44 corner tiles over 6 colors, and the resulting isomorphic corner tile sets were claimed aperiodic [33]. In 2016, Nurmi [34] noticed that, in this set, 14 tiles are unusable in infinite valid tilings, and reduced the corner tile set to 30 tiles over 6 colors. Quite surprisingly, neither Lagae et. al. nor Nurmi recognized that the tile set forms a torus, and is therefore periodic, as we show next.

Instead of developing a new formulation for another type of tiles, we first notice that corner tiles are actually a subset of Wang tiles, and therefore every set of corner tiles can be represented by a set of Wang tiles with the same cardinality, see Appendix A. For these tiles, we solve the rectangular tiling formulation (16) with periodic boundary conditions (32) and an objective function to find the smallest tiling (33). As its output, we receive the optimal value of 6 and 12 optimal periodic rectangular tilings of the size \( 2 \times 3 \). Not surprisingly, all these solutions follow from only two periodic patterns shown in Fig. 11 by translations over the infinite plane.
Figure 11: Rectangular periodic valid tilings. Translating a $2 \times 3$ rectangle over the infinite valid tiling generated from (a) or (b) leads to 6 different periodic patterns of the same size. Consequently, the tile set allows for 12 periodic rectangles of the size $2 \times 3$.

Having revealed the smallest periodic solutions, it remains to be shown why the Lagae conversion methods failed. Lagae et al. \cite{33} mentioned that their methods lack bijectiveness in general but they assumed it was not the case here. Therefore, we believe it is useful to state the conditions under which the methods are bijective and show that they are not satisfied for the Ammann tile set.

**Lemma 5.1.** The horizontal translation method from Lagae et al. \cite{33} is bijective iff the dual transducer graph $G_{T,v}$ of the input tile set $T$ does not contain any parallel arcs.

**Proof.** The horizontal translation method is formally a mapping $T \times T \rightarrow \mathcal{T}_{\text{corner}}$ that generates $\forall (p,q) \in T^2 : c_p = c_q^w$ a corner tile $(c_p^w, c_p^e, c_q^w, c_q^e)$. To be bijective, the cardinality of the output needs to be equal to the cardinality of the input, and the mapping has to produce unique output for each input. Consequently, all the tiles $p \in T$ in the original tile set must be uniquely determined by $c_p^w$ and $c_p^e$, as the color codes of the vertical edges of $T$ are avoided in the construction of $\mathcal{T}_{\text{corner}}$.

Let us now consider that the dual transducer graph contains a parallel arc connecting the state $c_p^w$ with $c_q^w$. Then, there may exist two tiles colored by $(c_p^w, c_p^e, c_q^w, c_q^e)$ and $(c_q^w, c_q^e, c_p^w, c_p^e)$ that are indistinguishable in $\mathcal{T}_{\text{corner}}$, which contradicts the bijection. For the other option, if the transducer graph does not contain any parallel arcs, then each $c_p^w$, $c_p^e$ identifies with a single arc labeled by $c_q^w|c_q^e$, i.e., with a single tile, which completes the proof.

Rotating the tile set by 90 degrees, the arguments in Lemma 5.1 provide us with the conditions for the bijectiveness of the vertical translation method:

**Lemma 5.2.** The vertical translation method of Lagae et al. \cite{33} is bijective iff the transducer graph $G_{T,h}$ of the input tile set $T$ does not contain any parallel arcs.

For the Ammann tile set, we obtain the transducer graph $G_{T,h} = G_{T,v}$ shown in Fig. 12. Clearly, there exist parallel arcs $1 \rightarrow 0$. Moreover, using the same approach, we can show that the horizontal translation method also fails for the Robinson tile set of 24 tiles over 24 colors \cite{24}, contrary to the claims in \cite{33}, and the corresponding corner tile set is also periodic.

6 Conclusions

In this contribution, we investigated methods generating bounded Wang tilings for arbitrary tile sets. To this goal, we developed an $\mathcal{NP}$-complete binary linear programming formulation \cite{10}. 
as well as its \( \mathcal{NP} \)-hard optimization variants relaxing some of the initial assumptions: tilability of the entire rectangular domain leading to the maximum rectangular tiling formulation (16), simple-connectedness to the maximum-cover program (19), and tiling validity to the maximum constraint satisfaction problem (22). In addition, we supplemented these formulations with extensions enabling control over individual tiles and their colors, including a tile-packing constraint to enable generation of periodic tile packings, and introduced the variable-sized periodic constraints to facilitate computation of the smallest periodic patterns.

Motivated by the \( \mathcal{NP} \)-hardness of all optimization formulations, we developed simple yet efficient heuristic algorithms for the maximum-cover variant (19). These algorithms rely on the fact that generating the maximum row cover is equivalent to finding the shortest path in a directed acyclic graph. Moreover, well-chosen costs of the graph edges also maintain color matches with the neighboring rows. Thus, in the simplest case, a heuristic solution follows from a sequential generation of row-cover tilings. Moreover, with simple modifications to the row processing order, we showed how to provide a \( 1/2 \) approximation factor for general tile sets and a \( 2/3 \) guarantee for tile sets whose transducer graphs are cyclic.

We illustrated the effectiveness of these methods on a collection of 11 tile sets. Generating tilings sized \( 20 \times 20, 25 \times 25, \) and \( 30 \times 30 \) revealed that the decision program (10) is the most efficient for our test problems. However, when a time limit is imposed or if the tile set does not allow for valid tiling of the entire domain, then the maximum cover (19) and maximum adjacency constraint satisfaction problems (22) appear to be similarly efficient. The remaining formulation—maximum rectangular tiling (16)—exhibits the worst scalability.

The usefulness of integer programming extensions was demonstrated by means of three illustrative problems: showing a better solution efficiency to the tile packing problem than the Lagae and Dutrée [14] backtracking approach, revealing an unusable tile in the Knuth [21] tile set, and proving that the Lagae et al. [33] tile set of corner tiles lacks aperiodicity. For the latter case, we also included an explanation for why the tile set construction method failed.

Furthermore, we tested the performance of the heuristic algorithms against the Gurobi solver tackling integer problem (19) for 300 s. Testing revealed that the simplest setup of Alg. 4 initialized with the cover generated by Alg. 1 produces the best results, obtained faster and (in most cases) competitive with the outputs of Gurobi. Somewhat surprisingly, the variants with guaranteed lower bounds exhibited slightly worse performance on average.

Having summarized our contributions, we believe that this work has not only introduced new methods that can possibly be applied to materials engineering, but also a simple and quite extensible framework to verify theoretical results on Wang tilings.

![Figure 12: Transducer graph of the Ammann set of 16 Wang tiles over 6 colors.](image-url)
A Corner tiles represented as Wang tiles

Each corner tile in the corner tile set is defined by a quadruple of color codes \((c_{nw}^k, c_{sw}^k, c_{se}^k, c_{ne}^k)\), with \(c_{nw}^k, c_{sw}^k, c_{se}^k,\) and \(c_{ne}^k\) denoting the colors of the northwest, southwest, southeast, and northeast corner of the \(k\)-th tile. Similarly to Wang tiles, corner tiles are assembled such that the color codes at the adjoining corners match.

This is, however, also maintained if we denote their edges by labels in the form of tuples of the corner codes, \((c_{nw}^k, c_{ne}^k)\), \((c_{nw}^k, c_{sw}^k)\), \((c_{sw}^k, c_{se}^k)\), and \((c_{ne}^k, c_{se}^k)\), each of which denotes a single edge label of the north, west, south, and east edge, respectively. Consequently, we can compute unique color codes as

\[
\begin{align*}
  c_{nw}^k &= c_{nw}^k + c_{sw}^k n_{vc}, \\
  c_{sw}^k &= c_{nw}^k + c_{se}^k n_{vc}, \\
  c_{se}^k &= c_{sw}^k + c_{ne}^k n_{vc}, \\
  c_{ne}^k &= c_{se}^k + c_{ne}^k n_{vc},
\end{align*}
\]

where \(n_{vc}\) stands for the number of colors used in the corner tile set. Graphical illustration of the corner-edge tile equivalence is shown in Fig. 13.

Data accessibility Source code available at: https://gitlab.com/tyburec/tilopt.

Authors’ Contributions M.T.: conceptualization, methodology, software, writing – original draft, J.Z.: methodology, writing – review & editing, supervision, funding acquisition.

Competing Interests The authors declare there is no competing interest.

Funding This research was funded by the Czech Science Foundation, project No. 19-26143X.

Acknowledgments We thank Stephanie Krueger for proofreading the initial draft of this manuscript.

References

[1] Wang H. 1963 Dominoes and the AEA case of the decision problem. Symposium on the Mathematical Theory of Automata pp. 23–55.

[2] Wang H. 1961 Proving theorems by pattern recognition–II. Bell System Technical Journal 40, 1–41.

[3] Berger R. 1966 The undecidability of the domino problem. Memoirs of the American Mathematical Society.

[4] Turing AM. 1937 On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society 2, 242, 230–265.
[5] Davis M. 1958 *Computability & Unsolvability*. Dover Books on Computer Science Series. Dover Publications.

[6] Kahr AS, Moore EF, Wang H. 1962 Entscheidungsproblem reduced to the ∀∃∀ case. *Proceedings of the National Academy of Sciences* **48**, 365–377.

[7] Lewis HR. 1978 pp. 35–47. In *Complexity of solvable cases of the decision problem for the predicate calculus*, pp. 35–47. IEEE.

[8] Lewis H, Papimtiou C. 1981 *Elements of the theory of computation*. Prentice-Hall Software Series. Pearson Education Canada.

[9] Cohen MF, Shade J, Hiller S, Deussen O. 2003 Wang tiles for image and texture generation. *ACM Transactions on Graphics* **22**, 287–294.

[10] Derouet-Jourdan A, Kaji S, Mizoguchi Y. 2019 A linear algorithm for brick Wang tiling. *Japan Journal on Industrial and Applied Mathematics* **36**, 749–761.

[11] Lagae A, Dutré P. 2005 A procedural object distribution function. *ACM Transactions on Graphics* **24**, 1442–1461.

[12] Ollinger N. 2008 Two-by-two substitution systems and the undecidability of the domino problem. In *Logic and Theory of Algorithms* pp. 476–485. Springer Berlin Heidelberg.

[13] Kovalsky SZ, Gläser D, Basri R. 2015 A global approach for solving edge-matching puzzles. *SIAM Journal on Imaging Sciences* **8**, 916–938.

[14] Lagae A, Dutré P. 2007 The Tile Packing Problem. *Geombinatorics* **17**, 8–18.

[15] Rui Yu CR, Agapito L. 2016 Solving jigsaw puzzles with linear programming. In Richard C. Wilson ERH, Smith WAP, editors, *Proceedings of the British Machine Vision Conference (BMVC)* pp. 139.1–139.12. BMVA Press.

[16] Salassa F, Vancroonenburg W, Wauters T, Della Croce F, Berghe GV. 2017 MILP and max-clique based heuristics for the Eternity II puzzle. arXiv: 1709.00252.

[17] Berger R. 1964 *The Undecidability of the Domino Problem*. PhD thesis Harvard University.

[18] Wang H. 1975 Notes on a class of tiling problems. *Fundamenta Mathematicae* **82**, 295–305.

[19] Robinson RM. 1967 Seven polygons which permit only nonperiodic tilings of the plane. *Notices of American Mathematical Society* p. 835.

[20] Poizat B. 1980 Une théorie finielement axiomatisable et superstable. *Groupe d’étude des théories stables* **3**, 1–9.

[21] Knuth DE. 1968 *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. Addison-Wesley Educational Publishers Inc.

[22] Knuth DE. 2018 *The Art of Computer Programming, Volume 4B, Fascicle 5: The: Mathematical Preliminaries Redux; Backtracking; Dancing Links*. Addison-Wesley Professional.

[23] Robinson RM. 1971 Undecidability and nonperiodicity for tilings of the plane. *Inventiones Mathematicae* **12**, 177–209.
[24] Grünbaum B, Shephard GC. 2016 *Tilings and patterns*. Mineola, New York: Dover Publications.

[25] Robinson RM. 1978 Undecidable tiling problems in the hyperbolic plane. *Inventiones Mathematicae* 44, 259–264.

[26] Senechal M. 1996 *Quasicrystals and Geometry*. Cambridge University Press.

[27] Kari J. 1996 A small aperiodic set of Wang tiles. *Discrete Mathematics* 160, 259–264.

[28] Čulík K. 1996 An aperiodic set of 13 Wang tiles. *Discrete Mathematics* 160, 245–251.

[29] Kari J, Papasoglu P. 1999 Deterministic aperiodic tile sets. *Geometric and Functional Analysis* 9, 353–369.

[30] Labbé S. 2019 A self-similar aperiodic set of 19 Wang tiles. *Geometriae Dedicata* 201, 81–109.

[31] Jeandel E, Rao M. 2021 An aperiodic set of 11 Wang tiles. *Advances in Combinatorics* pp. 1–37.

[32] Lagae A, Dutrée P. 2006 An alternative for Wang tiles: Colored edges versus colored corners. *ACM Transactions on Graphics* 25, 1442–1459.

[33] Lagae A, Kari J, Dutrée P. 2006 Aperiodic sets of square tiles with colored corners. Report CW.

[34] Nurmi T. 2016 From checkerboard to cloverfield: Using Wang tiles in seamless non-periodic patterns. In *Bridges Finland Conference Proceedings*.

[35] Kari J. 1990 Reversibility of 2D cellular automata is undecidable. *Physica D: Nonlinear Phenomena* 45, 379–385.

[36] Conway J, Lagarias J. 1990 Tiling with polyominoes and combinatorial group theory. *Journal of Combinatorial Theory, Series A* 53, 183–208.

[37] Mozes S. 1989 Tilings, substitution systems and dynamical systems generated by them. *Journal d'Analyse Mathématique* 53, 139–186.

[38] Stam J. 1997 *Aperiodic texture mapping*. European Research Consortium for Informatics and Mathematics. Tech. rep., R046.

[39] Hiller S, Deussen O, Keller A. 2001 Tiled blue noise samples. In *Proceedings of the Vision Modeling and Visualization Conference* pp. 265–272 Stuttgart, Germany.

[40] Radin C. 1987 Low temperature and the origin of crystalline symmetry. *International Journal of Modern Physics B* 1, 1157–1191.

[41] Winfree E, Liu F, Wenzler LA, Seeman NC. 1998 Design and self-assembly of two-dimensional DNA crystals. *Nature* 394, 539–544.

[42] Seeman NC, Mao C, LaBean TH, Reif JH. 2000 Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. *Nature* 407, 493–496.
[43] Novák J, Kučerová A, Zeman J. 2012 Compressing random microstructures via stochastic Wang tilings. *Physical Review E* **86**, 4–7.

[44] Doškár M, Novák J, Zeman J. 2014 Aperiodic compression and reconstruction of real-world material systems based on Wang tiles. *Physical Review E* **90**, 062118.

[45] Doškár M, Novák J. 2016 A jigsaw puzzle framework for homogenization of high porosity foams. *Computers & Structures* **166**, 33–41.

[46] Doškár M, Zeman J, Rypl D, Novák J. 2020 Level-set based design of Wang tiles for modelling complex microstructures. *Computer-Aided Design* **123**, 102827.

[47] Tyburec M, Zeman J, Doškár M, Kružík M, Lepš M. 2020 Modular-topology optimization with Wang tilings: An application to truss structures. *Structural and Multidisciplinary Optimization* **63**, 1099–1117.

[48] Tyburec M, Doškár M, Zeman J, Kružík M. 2022 Modular-topology optimization of structures and mechanisms with free material design and clustering. *Computer Methods in Applied Mechanics and Engineering*. Accepted.

[49] Jílek M, Somr M, Kulich M, Zeman J, Přeučil L. 2021 Towards a passive self-assembling macroscale multi-robot system. *IEEE Robotics and Automation Letters* **6**, 7293–7300.

[50] Doškár M, Zeman J, Jarušková D, Novák J. 2018 Wang tiling aided statistical determination of the Representative Volume Element size of random heterogeneous materials. *European Journal of Mechanics – A/Solids* **70**, 280–295.

[51] Coulais C, Teomy E, de Reus K, Shokef Y, van Hecke M. 2016 Combinatorial design of textured mechanical metamaterials. *Nature* **535**, 529–532.

[52] Yang W, Liu Q, Gao Z, Yue Z, Xu B. 2018 Theoretical search for heterogeneously architected 2D structures. *Proceedings of the National Academy of Sciences* **115**, E7245–E7254.

[53] Nežerka V, Somr M, Janda T, Vorel J, Doškár M, Antoš J, Zeman J, Novák J. 2018 A jigsaw puzzle metamaterial concept. *Composite Structures* **202**, 1275–1279.

[54] Knuth DE. 1992 Two notes on notation. *The American Mathematical Monthly* **99**, 403–422.

[55] Korte B, Vygen J. 2006 *Combinatorial Optimization*. Springer-Verlag.

[56] Gurobi Optimization, LLC. 2022 Gurobi Optimizer Reference Manual. [http://gurobi.com](http://gurobi.com)