Bounded Wang Tiling

Marek Tyburec
Thursday 12:45-14:15
Physical and Material Engineering
marek.tyburec@fsv.cvut.cz

I. Assignment

A. Problem Statement

Assuming a finite set of *color codes* $\mathcal{C} = \{1, 2, \ \dots \ , n_\mathrm{c}\} \subset \mathbb{N}^+$, the *(Wang) tile* k is a quadruple of the color codes $(c_k^\mathrm{n}, c_k^\mathrm{w}, c_k^\mathrm{s}, c_k^\mathrm{e})$, with $c_k^\mathrm{n}, c_k^\mathrm{w}, c_k^\mathrm{s}, c_k^\mathrm{e} \in \mathcal{C}$ standing for the color codes of the north, west, south, and east edge of the tile k, respectively. Tiles are therefore graphically representable by non-rotatable squares, as shown in Fig. 1. Without loss of generality, we further consider these squares to be unit.

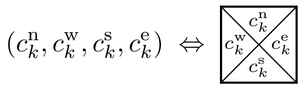
$(c_k^\mathrm{n}, c_k^\mathrm{w}, c_k^\mathrm{s}, c_k^\mathrm{e}) \Leftrightarrow$
c_k^n
c_k^w
c_k^e
c_k^s

Fig. 1: Graphical representation of a Wang tile k.

A *tile set* $\mathcal{T}$ represents a finite collection of n_t tiles.

Tiling $\mathfrak{T}^\mathcal{A}$ of a bounded domain $\mathcal{A} = \mathcal{H} \times \mathcal{W}$, where $\mathcal{H} \subset \mathbb{N}^+$ denotes the set of height coordinates and $\mathcal{W} \subset \mathbb{N}^+$ the set of width coordinates, constitutes an arrangement of copies of the tiles from the tile set $\mathcal{T}$, such that the tiles are placed at the integer lattice points of $\mathcal{A}$, they are contiguous, do not overlap, and cover the entire domain. More formally, tiling is a mapping $M : \mathcal{A} \to \mathcal{T}$ assigning a single tile $k \in \mathcal{T}$ to every $(i, j) \in \mathcal{A}$ coordinate.

A *valid tiling* (Wang tiling) of $\mathcal{A}$, denoted by $\mathfrak{T}_\mathrm{v}^\mathcal{A}$, is a tiling in which the color codes at the shared edges of all pairs of adjoining tiles equal. Therefore, the mapping $M_\mathrm{v} : \mathcal{A} \to \mathcal{T}$ satisfies, in addition to the requirements for M, both

$$c^\mathrm{s}_{M_\mathrm{v}(i,j)} = c^\mathrm{n}_{M_\mathrm{v}(i+1,j)}, \quad \forall i \in \mathcal{H} \setminus \{n_\mathrm{h}\}, \forall j \in \mathcal{W}, \tag{1}$$

and

$$c^\mathrm{e}_{M_\mathrm{v}(i,j)} = c^\mathrm{w}_{M_\mathrm{v}(i,j+1)}, \quad \forall i \in \mathcal{H}, \forall j \in \mathcal{W} \setminus \{n_\mathrm{w}\}, \tag{2}$$

provided that the axes are orientated accordingly to Fig. 2. If M_v exists, we say that the domain $\mathcal{A}$ admits a valid $\mathcal{T}$-tiling, or that it is *tileable* by $\mathcal{T}$.

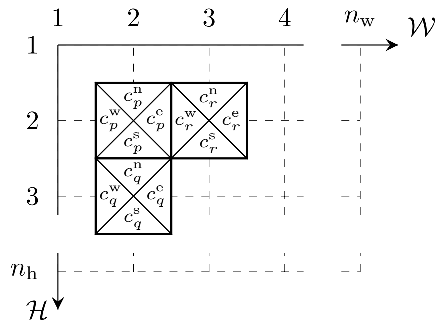

Fig. 2: Color matching among tiles p, q, $r \in \mathcal{T}$.

The *maximum cover* $\mathfrak{T}^\mathcal{A}_{\mathrm{v},\max\mathrm{cov}}$ is a valid tiling of $\mathcal{B}_{\max\mathrm{cov}} := \{\mathcal{B}_{\max\mathrm{cov}} \subseteq \mathcal{A}, \forall \mathcal{B} \subseteq \mathcal{A} : |\mathcal{B}_{\max\mathrm{cov}}| \geq |\mathcal{B}|\}$.

In this text we present an approach to find $\mathcal{T}_\mathrm{v}^\mathcal{A}$, or, if infeasible, the maximum cover $\mathfrak{T}^\mathcal{A}_{\mathrm{v},\max\mathrm{cov}}$.

B. Problem Categorization

Wang tiling has been extensively studied in the infinite settings, i.e., when $\mathcal{A}$ is the infinite plane. In such settings, the problem is called the domino problem, and was proven to be undecidable by reduction of the Turing machine halting problem [6], which also proved undecidability of the $\forall\exists\forall$ problem of predicate calculus [20]. Moreover, this implies non-existance of a general finite algorithm for production of infinite valid tilings, and existence of aperiodic tile sets that do not allow any simply-connected rectangular valid tilings with equally colored opposite sides [1, 4].

Far less attention has been paid to the finite version of the domino problem, bounded tiling, searching a fixed-sized valid tiling generated by an arbitrary tile set. The problem is only known to be $\mathcal{NP}$-complete in general, we refer to [12] and [13, Theorem 7.2.1], and indeed decidable. Here, we also focus on the $\mathcal{NP}$-hard maximum-cover optimization variant of the problem.

II. Related works

To the author's knowledge, there does not exist any research dealing with the solution of the bounded tiling problem in general settings. Thus, the few methods that are provided implicitly in the literature serve only specific purposes: they either allow for tiling with specific families of tile sets [2, 3, 9, 16], or tiling of infinite thin strips [5]. While it might be preferable to design a polynomial-time algorithm for each family of tile sets independently, there exist non-recursive tile sets not admitting any such tiling algorithm [14].

More frequent in the literature is the closely related (tile) packing problem for edge-matching puzzles, in which all the tiles from a tile set are placed exactly once, see [8, 11, 18], and [19] for an approach aiming at the famous Eternity II puzzle.

A. Substitution-Based Tiling Algorithm

Substitution is a map $S : \mathcal{T} \mapsto \mathfrak{T}$ that assigns to each tile $k \in \mathcal{T}$ a tiling $\mathfrak{T}_k$. Arbitrary-sized tilings are then generated by placing a single tile, and then repeating the substitution rule on all tiles that are already placed [16], so that the tiling "grows" in each iteration. However, only substitution-based tile sets allow for a substitution rule that generates valid tilings.

B. Stochastic Tiling Algorithms

In computer graphics, it is essential to generate visually appealing (nonperiodic) patterns quickly, which is best achieved with stochastic tile sets. For example, in the stochastic tiling algorithm [2] 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 to handle boundary conditions, and named the hash-based direct stochastic tiling algorithm [9].

C. Transducer-Based Tiling Algorithm

The transducer-based tiling algorithm comes from the knowledge that the 1D domino problem is decidable, and can be solved in polynomial time, because the bi-infinite path is formed by an arbitrary cycle in the transducer graph of the tiling [5]. In order to generate valid tilings of multiple rows, it is essential to compute the product of several transducers, i.e., enumerate all feasible valid tilings of requested height and unit width, and then find a given-length path in the transducer graph of just established tile set. Obviously, this approach works well for tiling of thin strips; however, it is impractical for any larger nearly-square domains.

III. Problem solution

A. Design

1) Rectangular Valid Tiling: In this section, we focus on the fundamental problem of finding $\mathfrak{T}_{\mathrm{v}}^{\mathcal{A}}$, or proving it does not exist. From now, we restrict $\mathcal{A}$ to be rectangular in order to simplify notation. However, the presented approach is also applicable to the general unrestricted case.

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

$$x_{i,j,k} = \begin{cases} 0 & \text{if the tile } k \text{ is not placed at } (i,j), \\ 1 & \text{if the tile } k \text{ lies at } (i,j). \end{cases} \tag{3}$$

Consequently, the mapping $M(i,j)$ is expressed as

$$M(i,j) = \sum_{k \in \mathcal{T}} k x_{i,j,k}, \quad \text{s.t.} \sum_{k \in \mathcal{T}} x_{i,j,k} = 1, \quad \forall i \in \mathcal{H}, \forall j \in \mathcal{W}. \tag{4}$$

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

$$c^{\mathrm{n}}_{M(i,j)} = \sum_{k \in \mathcal{T}} c^{\mathrm{n}}_k x_{i,j,k}, \tag{5a}$$

$$c^{\mathrm{w}}_{M(i,j)} = \sum_{k \in \mathcal{T}} c^{\mathrm{w}}_k x_{i,j,k}, \tag{5b}$$

$$c^{\mathrm{s}}_{M(i,j)} = \sum_{k \in \mathcal{T}} c^{\mathrm{s}}_k x_{i,j,k}, \tag{5c}$$

$$c^{\mathrm{e}}_{M(i,j)} = \sum_{k\in\mathcal{T}} c^{\mathrm{e}}_k x_{i,j,k}. \tag{5d}$$

Inserting (5) into (1) and (2) leads to the horizontal and vertical adjacency constraints expressed in terms of the decision variables only, hence

$$\sum_{k\in\mathcal{T}} c^{\mathrm{s}}_k x_{i,j,k} - \sum_{k\in\mathcal{T}} c^{\mathrm{n}}_k x_{i+1,j,k} = 0, \quad \forall i \in \mathcal{H}\setminus\{n_{\mathrm{h}}\}, \forall j \in \mathcal{W}, \tag{6a}$$

$$\sum_{k\in\mathcal{T}} c^{\mathrm{e}}_k x_{i,j,k} - \sum_{k\in\mathcal{T}} c^{\mathrm{w}}_k x_{i,j+1,k} = 0, \quad \forall i \in \mathcal{H}, \forall j \in \mathcal{W}\setminus\{n_{\mathrm{w}}\}. \tag{6b}$$

Combining (3), (4), and (6) then provides us with a complete binary linear programming representation of M_{v}.

Unfortunately, the adjacency constraints (6) appear to be weak due to their dependence on the actual values of the color codes, and on their order. Thus, we split (6) based on the color codes to provide a tighter representation of the feasible design space and to avoid listing of the actual values, i.e.,

$$\sum_{k\in\mathcal{T}} x_{i,j,k}[c^{\mathrm{s}}_k = \ell] - \sum_{k\in\mathcal{T}} x_{i+1,j,k}[c^{\mathrm{n}}_k = \ell] = 0, \quad \forall i \in \mathcal{H}\setminus\{n_{\mathrm{h}}\}, \forall j \in \mathcal{W}, \forall \ell \in \mathcal{C}, \tag{7a}$$

$$\sum_{k\in\mathcal{T}} x_{i,j,k}[c^{\mathrm{e}}_k = \ell] - \sum_{k\in\mathcal{T}} x_{i,j+1,k}[c^{\mathrm{w}}_k = \ell] = 0, \quad \forall i \in \mathcal{H}, \forall j \in \mathcal{W}\setminus, \forall \ell \in \mathcal{C}\{n_{\mathrm{w}}\}, \tag{7b}$$

where the Iverson notation $\sum_{k\in\mathcal{T}} x_{i,j,k}\,[c^{\mathrm{s}}_k = \ell]$ expresses that $x_{i,j,k}$ is added to the sum if and only if $c^{\mathrm{s}}_k = \ell$.

The constraint (7a) requires that the number of tiles at (i,j) with the south edge colored by ℓ equals to the number of tiles at $(i+1,j)$ with the north edge marked by the same ℓ, for all $\ell \in \mathcal{C}$. Because there are either no tiles with the shared edge colored by ℓ, or a single tile at each of the coordinates, with their common edge labeled by ℓ.

Analogously to the vertical adjacency constraint, also the horizontal adjacency constraint (7b) enforces equality among the number of tiles at (i,j) with the east edge colored by ℓ and the number of tiles at $(i,j+1)$ having the west edge colored by identical ℓ.

Finally, combining (3), and (7), while noticing that the $\sum_{i\in\mathcal{H}}\sum_{j\in\mathcal{W}}\sum_{k\in\mathcal{T}} x_{i,j,k}$ naturally propagates from the domain boundaries, so the number of constraints can be reduced, leads to the binary programming formulation

$$\begin{aligned}
&\text{find } \mathbf{x} && \text{(8a)}\\
&\text{s.t. } \sum_{k\in\mathcal{T}} x_{i,j,k}[c^{\mathrm{s}}_k = \ell] - \sum_{k\in\mathcal{T}} x_{i+1,j,k}[c^{\mathrm{n}}_k = \ell] = 0, \quad \forall i \in \mathcal{H}\setminus\{n_{\mathrm{h}}\}, \forall j \in \mathcal{W}, \forall \ell \in \mathcal{C}, && \text{(8b)}\\
&\sum_{k\in\mathcal{T}} x_{i,j,k}[c^{\mathrm{e}}_k = \ell] - \sum_{k\in\mathcal{T}} x_{i,j+1,k}[c^{\mathrm{w}}_k = \ell] = 0, \quad \forall i \in \mathcal{H}, \forall j \in \mathcal{W}\setminus\{n_{\mathrm{w}}\}, \forall \ell \in \mathcal{C}, && \text{(8c)}\\
&\sum_{k\in\mathcal{T}} x_{i,j,k} = 1, \quad \forall i \in \{1, n_{\mathrm{h}}\}, \forall j \in \mathcal{W}, && \text{(8d)}\\
&\sum_{k\in\mathcal{T}} x_{i,j,k} = 1, \quad \forall i \in \mathcal{H}, \forall j \in \{1, n_{\mathrm{w}}\}, && \text{(8e)}\\
&x_{i,j,k} \in \{0,1\}, \quad \forall i \in \mathcal{H}, \forall j \in \mathcal{W}, \forall k \in \mathcal{T}, && \text{(8f)}
\end{aligned}$$

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 (8) represent valid tilings. Moreover, it can be seen that the problem consists of two totally unimodular constraints—the rows (8c) and (8e), and columns (8b) and (8d) tilings—making, however, the resulting problem $\mathcal{NP}$-complete in combination.

2) Maximum Cover: An option to avoid the infeasibility of (8) rests in neglecting the requirement of (simple) connectedness. Hence, in this section, we search the maximum cover of $\mathcal{A}$, or equivalently a valid tiling of the (possibly disconnected) domain $\mathcal{B}_{\max\mathrm{cov}} \subseteq \mathcal{A}$.

For the maximum cover formulation, we assume that any two adjacent tiles satisfy the adjacency constraints of valid tilings, but these are also satisfied by any of the tile-void, void-tile, or void-void combination; here, we denote $(i,j) \in \mathcal{A}$ without any tile as a void.

Thus, each coordinate (i,j) is occupied either by a tile, or by a void, i.e.,

$$\sum_{k\in\mathcal{T}} x_{i,j,k} \leq 1, \quad \forall i \in \mathcal{H}, \forall j \in \mathcal{W}, \tag{9}$$

and for any two adjacent coordinates (i,j) and $(i+1,j)$ there can not happen a situation in which the color codes at shared edges of the adjacent tiles (if any) do not match,

$$\sum_{k\in\mathcal{T}} x_{i,j,k}[c^{\mathrm{e}}_k = \ell] + \sum_{k\in\mathcal{T}} x_{i,j+1,k}[c^{\mathrm{w}}_k \neq \ell] \leq 1, \quad \forall i \in \mathcal{H}, \forall j \in \mathcal{W}\setminus\{n_{\mathrm{w}}\}, \forall \ell \in \mathcal{C}. \tag{10}$$

The same principle is also valid in the case of any two horizontally adjacent coordinates (i, j) and $(i, j+1)$, hence

$$\sum_{k\in\mathcal{T}} x_{i,j,k}[c_k^{\mathrm{s}} = \ell] + \sum_{k\in\mathcal{T}} x_{i+1,j,k}[c_k^{\mathrm{n}} \neq \ell] \leq 1, \quad \forall i \in \mathcal{H} \setminus \{n_{\mathrm{h}}\}, \forall j \in \mathcal{W}, \forall \ell \in \mathcal{C}. \tag{11}$$

Finally, combining Eqs. (9), (10), (11) with the objective function to maximize $|\mathcal{B}_{\max\mathrm{cov}}|$ provides us with the binary optimization problem

$$\max_{\mathbf{x}} \sum_{i\in\mathcal{H}}\sum_{j\in\mathcal{W}}\sum_{k\in\mathcal{T}} x_{i,j,k} \tag{12a}$$

$$\text{s.t.} \sum_{k\in\mathcal{T}} x_{i,j,k}[c_k^{\mathrm{e}} = \ell] + \sum_{k\in\mathcal{T}} x_{i,j+1,k}[c_k^{\mathrm{w}} \neq \ell] \leq 1, \quad \forall i \in \mathcal{H}, \forall j \in \mathcal{W} \setminus \{n_{\mathrm{w}}\}, \forall \ell \in \mathcal{C}, \tag{12b}$$

$$\sum_{k\in\mathcal{T}} x_{i,j,k}[c_k^{\mathrm{s}} = \ell] + \sum_{k\in\mathcal{T}} x_{i+1,j,k}[c_k^{\mathrm{n}} \neq \ell] \leq 1, \quad \forall i \in \mathcal{H} \setminus \{n_{\mathrm{h}}\}, \forall j \in \mathcal{W}, \forall \ell \in \mathcal{C}, \tag{12c}$$

$$\sum_{k\in\mathcal{T}} x_{i,j,k} \leq 1, \quad \forall i \in \mathcal{H}, \forall j \in \mathcal{W}, \tag{12d}$$

$$x_{i,j,k} \in \{0, 1\}, \quad \forall i \in \mathcal{H}, \forall j \in \mathcal{W}, \forall k \in \mathcal{T}. \tag{12e}$$

B. *Description And Implementation of Heuristics*

In this section, we develop and implement a simple $\mathcal{O}(|\mathcal{A}||\mathcal{T}|^2)$ heuristics for the maximum cover formulation (12). In order to develop the heuristics, let us return to the decision program (8). As it was already outlined, neglecting any of the pair of constraints (8b), (8d), or (8c), (8e), leads to the totally unimodular constraint matrix, so that the problem is solvable deterministically in a polynomial time. Visually, such relaxations result in valid tilings of (finite) stripes, i.e., rows or columns. However, validity of the tilings is not guaranteed in the edges shared by the neighboring stripes. Starting with this observation, we first focus on formulation of an efficient approach to generate valid tilings of the stripes.

Assuming a valid tiling of a single row, the problem is represented by the graph: the tiles in the tile set are placed in layers, repeated exactly n_{w} times. The layers (copies of the tile set) are connected by directed arcs from the layer j to $j+1$, $\forall j \in \mathcal{W} \setminus n_{\mathrm{w}}$, such that the tiles are connected if and only if the east color of the tile at the j-th column matches the west color of the tile at $(j+1)$-th column. In this directed graph, all paths of the length $n_{\mathrm{w}} - 1$ clearly represent valid tilings of the row. Further, we add a source vertex, and connect it to all the tiles in the first layer. Similarly, a terminal vertex is added, and all tiles in the n_{w}-th layer are connected to the terminal vertex.

Assigning zero costs to all the edges and recognizing that the graph is acyclic and single-sourced, due to the added source vertex, a valid tiling of a row is found in $\mathcal{O}(|\mathcal{V}| + |\mathcal{E}|)$ time using the DAG-shortest-path algorithm, where $\mathcal{V}$ denotes the set of the graph vertices and $\mathcal{E}$ the set of the graph edges. In our case, we have

$$|\mathcal{V}| = 2 + n_{\mathrm{w}}|\mathcal{T}|, \tag{13a}$$

$$|\mathcal{E}| = 2|\mathcal{T}| + (n_{\mathrm{w}} - 1)k_{\mathcal{T}}, \tag{13b}$$

where $k_{\mathcal{T}}$ is a constant specific to the tile set $\mathcal{T}$ denoting the number of valid row-tilings when $n_{\mathrm{h}} = 2$. Consequently, we have $k_{\mathcal{T}} = |\mathcal{T}|^2$ in the worst case, so that the overall asymptotic complexity to generate a valid tiling of a row reads as

$$\mathcal{O}(|\mathcal{V}| + |\mathcal{E}|) = \mathcal{O}(2 + n_{\mathrm{w}}|\mathcal{T}| + 2|\mathcal{T}| + (n_{\mathrm{w}} - 1)|\mathcal{T}|^2) = \mathcal{O}(n_{\mathrm{w}}|\mathcal{T}|^2). \tag{14}$$

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 is known from the graph construction method in advance.

Having outlined the initial idea, we now extend the approach to handle the cases when valid tilings of a row do not exist. In such a case, it is sufficient to interconnect the incompatible tiles among the j-th and $(j+1)$-th layers, $\forall j \in \mathcal{W} \setminus n_{\mathrm{w}}$, and assign to them the cost 1[1]. Because tilings obtained by this method may not be valid, the results need to be post-processed to generate valid tilings of some subdomain. Here, we process edges and vertices in the shortest path, such that for each vertex representing a tile the cost of the incoming edge is determined. If the cost equals to one, the void is placed; a tile corresponding to the vertex is placed otherwise. Consequently, any shortest path of the total cost c_{t} contains exactly c_{t} color mismatches, and c_{t} voids in the post-processed row tiling. Because the shortest path algorithm explicitly minimizes the number of voids, the maximum cover of the row is reached.

Let us now consider that the rows $i-1$ and $i+1$ are tiled (either by valid tilings or by voids), and we aim to generate the maximum cover of the i-th row such that the minimum number of voids is used. Quite remarkably, such a modification only requires us to modify some of the costs in the already created graph: for each tile $k \in \mathcal{T}$ in the j-th column we check whether the south color code of the tile at $(i-1, j)$ is not compatible with c_k^{n}, and whether the north color code of the tile at $(i+1, j)$ is not compatible with c_k^{s}, where compatibility refers to the tile-void, void-tile, void-void, or tile-tile with matching colors. If any of the compatibility conditions is not satisfied, the edges incoming to the vertex that corresponds to the tile k at the j-th layer are assigned the cost 1.

```
initialize a tiling of $\mathcal{A}$ with voids
for all rows do
    generate maximum cover tiling of the row
end
```

Algorithm 1: Simple maximum-cover based heuristics.

Consequently, the heuristics to find the maximum cover is written in pseudo-code as and requires n_h iterations in the inner loop. Consequently, the overall asymptotic complexity equals $\mathcal{O}(|\mathcal{A}||\mathcal{T}|^2)$.

Similarly to finding maximum cover of rows, one can search for the maximum cover of columns. Combining these two methods, we end up with a slightly improved heuristics that runs in $\mathcal{O}(|\mathcal{A}|^2|\mathcal{T}|^2)$ time in the worst case.

```
initialize a tiling of $\mathcal{A}$ with voids
set maximum cover method to rows
set improvement to infinity
while positive improvement do
    if maximum cover by rows then
        for all rows containing at least one void do
            generate maximum cover tiling of the row
        end
        set maximum cover method to columns
    else
        for all columns containing at least one void do
            generate maximum cover tiling of the column
        end
        set maximum cover method to rows
    end
    set improvement to the difference in the number of voids in the this and the previous iteration
end
```

Algorithm 2: Improved maximum-cover based heuristics.

IV. Experiments

A. Benchmark Settings

The described methods were implemented in C++, and evaluated on a personal laptop Acer Aspire VN7-591G running the Ubuntu 18.04 operating system, equipped with 8 GB of RAM and Intel® Core™ i5-4210H CPU clocked at 2.90 GHz. The source code was compiled with the g++ compiler, version 4.8, and linked to the Gurobi 7.5 optimizer library.

Binary linear programs (8) and (12) were solved by the Gurobi solver, which was allowed to utilize all the cores and threads available, two and four, respectively. In the case of the (improved) heuristics, Algorithm 2, the algorithm ran sequentially except for a simple parallelism inside the DAG-shortest-path algorithm, supporting the non-deterministic behavior of the algorithm[2]. Therefore, the heuristics is evaluated 100 times for each tested tile set, and the best, worst, and average results are shown.

For the benchmark instances, we have selected five well-known aperiodic tile sets, two stochastic tile sets, two periodic tile sets, and two tile sets that do not allow for a valid tiling of the finite domain. For all the tile sets, we aimed at generating valid tilings of the sizes 20×20, 25×25, and 30×30, respectively. The running time of the Gurobi solver was limited by 300 s.

B. Results

The benchmark results are shown in Table I.

C. Discussion

The results shown in Table I imply that the decision program (8) surpasses the maximum cover formulation (12) when the domain $\mathcal{A}$ is $\mathcal{T}$-tileable and no time limit is imposed. Although the speed dominance may seem unexpected at the first sight, the reason happens to be quite simple: the maximum cover formulation builds a larger decision tree and requires more constraints. Limiting the Gurobi optimizer runtime, as was our case, (12) provides a partial cover of the domain-to-be-tiled at least, contrary to the prematurely terminated (8).

A slightly more interesting results arise from the comparison of the maximum cover formulation (12) with the heuristics. First, it shall be noted that the heuristics always generates valid tilings if (any of) the stochastic tile sets are used; in this case, the heuristics finds a feasible solution to both (8) and (12) faster than the Gurobi optimizer. For aperiodic and periodic tile sets, Gurobi required considerably longer time to reach the solutions of a similar quality, but usually surpassed the heuristics in the

[1]The asymptotic complexity of the algorithm remains; however, the bound value $k_\mathcal{T} = |\mathcal{T}|^2$ is now active for all tile sets.

[2]Non-deterministic behavior of the DAG-shortest-path algorithm was introduced by randomizing the order of the edges in the graph representation of the problem.

Tile set	Size	**Decision program** (8)		**Maximum cover IP** (12)		**Maximum cover heuristics**			
		Time [s]	*Objective*	*Time [s]*	*Objective*	*Avg. time [s]*	*Min. obj.*	*Avg. obj.*	*Max. obj.*
Aperiodic1 (11/4) [5]	20×20	0.183	0	300.000	*396	0.190	359	367.86	378
	25×25	262.455	0	300.000	*596	0.386	559	572.60	588
	30×30	300.000	*infeasible	300.155	*841	0.621	810	821.41	838
Aperiodic2 (13/5) [22]	20×20	0.177	0	300.097	*399	0.290	350	361.24	370
	25×25	300.103	*infeasible	300.145	*604	0.566	553	563.10	573
	30×30	300.140	*infeasible	300.202	*877	0.947	796	808.33	823
Aperiodic3 (14/6) [7]	20×20	300.090	*infeasible	300.148	*398	0.290	359	370.20	381
	25×25	300.111	*infeasible	300.156	*602	0.479	565	577.34	596
	30×30	300.146	*infeasible	300.208	*869	0.893	808	829.62	845
Aperiodic4 (16/6) [4]	20×20	0.191	0	70.203	400	0.262	353	366.29	381
	25×25	0.302	0	300.218	*546	0.458	546	571.56	595
	30×30	0.450	0	300.293	*785	0.810	802	823.83	866
Aperiodic5 (56/12) [17]	20×20	1.029	0	300.660	*350	1.181	343	357.94	376
	25×25	0.859	0	300.869	*553	2.308	543	559.96	585
	30×30	1.245	0	301.182	*795	4.400	784	806.01	850
Stochastic1 (8/2) [2]	20×20	0.121	0	0.109	400	0.064	400	400.00	400
	25×25	0.157	0	0.153	625	0.094	625	625.00	625
	30×30	0.216	0	0.193	900	0.131	900	900.00	900
Stochastic2 (16/4) [10]	20×20	0.184	0	0.182	400	0.109	400	400.00	400
	25×25	0.225	0	0.305	625	0.164	625	625.00	625
	30×30	0.342	0	0.449	900	0.240	900	900.00	900
Periodic1 (10/4) [21]	20×20	0.150	0	68.063	400	0.194	344	354.59	376
	25×25	0.220	0	192.673	625	0.367	536	555.01	592
	30×30	0.348	0	300.162	*773	0.603	770	793.83	823
Periodic2 (30/17) [15]	20×20	0.346	0	71.149	400	0.384	359	379.52	399
	25×25	0.576	0	300.561	*545	0.611	557	591.85	618
	30×30	0.871	0	300.828	*786	1.094	827	850.63	887
Finite1 (7/4)	20×20	0.083	infeasible	300.082	*377	0.127	350	359.27	368
	25×25	0.107	infeasible	300.091	*585	0.233	552	560.13	572
	30×30	0.151	infeasible	300.108	*807	0.387	792	805.28	816
Finite2 (16/16)	20×20	0.143	infeasible	300.173	*326	0.322	327	341.22	354
	25×25	0.208	infeasible	300.293	*493	0.663	517	529.21	540
	30×30	0.283	infeasible	300.299	*690	1.132	742	760.77	781

TABLE I: Benchmark results. Values marked by an asterisk denote a premature termination of the solver.

time limit of 300 s. Finally, for the tile sets that do not allow valid tiling of $\mathcal{A}$, heuristics reached solutions of slightly better quality than Gurobi did, and the computational demands of the heuristics were significantly lower.

Yet another patterns arise in the measured data: all the worst-case heuristic solutions obtained were greater than $0.85 OPT$[3], where OPT refers to the optimal objective function value, and the speed and solution quality of the developed heuristics scaled better to larger domain sizes than Gurobi did.

V. Conclusion

Focusing on the bounded Wang tiling problem, this research project formulated its $\mathcal{NP}$-complete formulation, and its optimization variant, the $\mathcal{NP}$-hard maximum cover formulation. Aiming to speed-up the solution process, we have also developed and implemented a DAG-shortest-path-based heuristics for the maximum cover formulation.

Performance of the formulations and heuristics was assessed on 11 distinct tile sets. Preliminary numerical results imply that the decision formulation outperforms the optimization variant, and hint at the future potential of the heuristics.

References

[1] Robert Berger. *The undecidability of the domino problem.* Number 66. American Mathematical Society (AMS), 1966. doi: 10.1090/memo/0066. URL https://doi.org/10.1090/memo/0066.

[2] Michael F. Cohen, Jonathan Shade, Stefan Hiller, and Oliver Deussen. Wang tiles for image and texture generation. *ACM Transactions on Graphics*, 22(3):287, jul 2003. doi: 10.1145/882262.882265. URL https://doi.org/10.1145/882262.882265.

[3] Alexandre Derouet-Jourdan, Shizuo Kaji, and Yoshihiro Mizoguchi. A linear algorithm for Brick Wang tiling. *CoRR*, abs/1603.04292, 2016. URL http://arxiv.org/abs/1603.04292.

[4] Branko Grünbaum and Geoffrey Colin Shephard. *Tilings and patterns.* Freeman, 1987.

[5] Emmanuel Jeandel and Michaël Rao. An aperiodic set of 11 Wang tiles, 2015. URL http://arxiv.org/abs/1506.06492.

[6] Andrew S Kahr, Edward F Moore, and Hao Wang. Entscheidungsproblem reduced to the ∀∃∀ case. *Proceedings of the National Academy of Sciences*, 48(3):365–377, 1962.

[7] Jarkko Kari. A small aperiodic set of Wang tiles. *Discrete Mathematics*, 160(1-3):259–264, nov 1996. doi: 10.1016/0012-365x(95)00120-l. URL https://doi.org/10.1016/0012-365x(95)00120-l.

[3]We are able to prove that the approximation ratio is 0.67 at least.

[8] S. Z. Kovalsky, D. Glasner, and R. Basri. A Global Approach for Solving Edge-Matching Puzzles. *SIAM Journal on Imaging Sciences*.

[9] Ares Lagae and Philip Dutré. A procedural object distribution function. *ACM Transactions on Graphics*, 24(4):1442–1461, oct 2005. doi: 10.1145/1095878.1095888. URL https://doi.org/10.1145/1095878.1095888.

[10] Ares Lagae and Philip Dutré. An alternative for Wang tiles. *ACM Transactions on Graphics*, 25(4):1442–1459, oct 2006. doi: 10.1145/1183287.1183296. URL https://doi.org/10.1145/1183287.1183296.

[11] Ares Lagae and Philip Dutré. The tile packing problem. *Geombinatorics*, 17(1):8–18, July 2007. URL http://www.uccs.edu/~geombina/2007.htm.

[12] Harry R Lewis. *Complexity of solvable cases of the decision problem for the predicate calculus*, page 35–47. IEEE, Oct 1978. doi: 10.1109/SFCS.1978.9.

[13] H.R. Lewis and C.H. Papimitriou. *Elements of the Theory of Computation*. Prentice-Hall Software Series. Pearson Education Canada, 1981. ISBN 9780132734172.

[14] Dale Myers. Nonrecursive tilings of the plane. II. *Journal of Symbolic Logic*, 39, 06 1974. doi: 10.2307/2272641.

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

[16] Nicolas Ollinger. Two-by-two substitution systems and the undecidability of the domino problem. *Logic and Theory of Algorithms*, pages 476–485, 2008.

[17] Raphael M. Robinson. Undecidability and nonperiodicity for tilings of the plane. *Inventiones Mathematicae*, 12(3):177–209, sep 1971. doi: 10.1007/bf01418780. URL https://doi.org/10.1007/bf01418780.

[18] Chris Russell Rui Yu and Lourdes Agapito. Solving Jigsaw Puzzles with Linear Programming. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 139.1–139.12. BMVA Press, September 2016. ISBN 1-901725-59-6. doi: 10.5244/C.30.139. URL https://dx.doi.org/10.5244/C.30.139.

[19] Fabio Salassa, Wim Vancroonenburg, Tony Wauters, Federico Della Croce, and Greet Vanden Berghe. MILP and Max-Clique based heuristics for the Eternity II puzzle. 2017. URL https://arxiv.org/pdf/1709.00252.pdf.

[20] Hao Wang. Dominoes and the AEA case of the decision problem. *Symposium on the Mathematical Theory of Automata*, pages 23–55, 1962.

[21] Hao Wang. Notes on a class of tiling problems. *Fundamenta Mathematicae*, 82(4):295–305, 1975. URL http://eudml.org/doc/214668.

[22] Karel Čulík. An aperiodic set of 13 Wang tiles. *Discrete Mathematics*, 160(1-3):245–251, nov 1996. doi: 10.1016/s0012-365x(96)00118-5. URL https://doi.org/10.1016/s0012-365x(96)00118-5.