Tiling Layouts with Dominoes

Mark Watson[*] Chris Worman[†]

Abstract

We explore the complexity of tiling finite subsets of the plane, which we call *layouts*, with a finite set of tiles. The tiles are inspired by *Wang tiles* and the *domino* game piece. Each tile is composed of a pair of faces. Each face is colored with one of k possible colors. We want to know if a given layout is tileable by a given set of n dominoes. In a tiling, dominoes that touch must do so at like-colored domino faces. We provide an $O(n)$ time algorithm for tiling layouts that are *paths* or *cycles*. We also show that if the layout is partially tiled at the outset of the problem, then the tiling decision problem is NP-complete. We also show that the problem remains NP-complete even if the layout is a *tree*.

1 Introduction

In a geometric tiling problem we wish to fill all or some of the plane with non-overlapping polygons called *tiles*. The tiling problems studied herein are motivated by recent results concerning *Wang tiles*. Wang tiles are non-rotatable unit squares that have colored edges [5]. In a tiling that uses Wang tiles, neighboring tiles must have the same color on adjacent edges. In a typical Wang tiling problem, we are given a finite number of types of tiles and an infinite number of each type, and we are asked to tile some subset of the plane. Berger showed that deciding if the entire plane can be tiled by a given set of Wang tiles is undecidable [2]. Motivated by a connection between Wang tilings and self assembly in DNA computing, researchers have begun to study tiling proper infinite subsets of the plane [1, 3]. In [1], the authors show that the problem of tiling a *ribbon*, which is an infinite "path" in the plane, is undecidable. This result is extended in [3] to show that the problem of tiling a ribbon that is a "cycle" is undecidable.

We study a variation of Wang tiles, which we call *dominoes*, that are 2×1 rectangles that are partitioned into 2 colored faces. Thus unlike Wang tiles, the *faces* are colored rather than the *edges*. Also unlike Wang tiles, we allow rotation of the tiles and we consider finite sets of dominoes. Thus although our tiles have a connection to Wang tiles, they are essentially a generalization of the commonly used domino game piece.

[*]Department of Computer Science, University of Saskatchewan, mdw655@mail.usask.ca

[†]Department of Computer Science, University of Saskatchewan, cmw133@mail.usask.ca

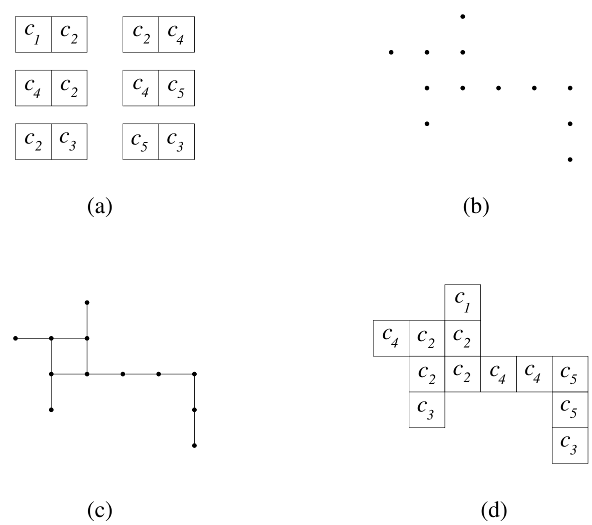

Figure 1: (a) A set of dominoes D realized as orthogonal rectangles with two faces. (b) A layout L. (c) The graph $\mathcal{G}^L$. (d) The realization of a tiling τ on the set D.

2 Terminology

Let $C = \{c_1, c_2, ..., c_k\}$ be a finite set of k colors. A domino d is a pair (c_{d1}, c_{d2}) of colors from C. We will refer to the current set of dominoes that is under consideration as D, and we will refer to the number of dominoes in D as n. We will only consider finite sets of dominoes.

Now we discuss how we can embed a domino in the integer plane. When embedding a domino in the plane, we interpret a domino $d = (c_{d1}, c_{d2})$ as a 2×1 rectangle that is partitioned into exactly 2 squares. We call these squares *faces* and refer to them as f_1^d and f_2^d. The face f_i^d will be colored color c_{di} (Figure 1(a)). We say that a function $\tau : D \mapsto \mathbb{Z}^2 \times \mathbb{Z}^2$ is a *domino tiling function* if τ has certain properties that we now explain. Firstly, τ must map a domino to a pair of horizontally or vertically aligned points that are 1 unit apart. Furthermore, τ must map dominoes such that adjacent domino faces have the same color. Finally, τ must map dominoes to non-overlapping locations. These properties are formalized below:

1. $\tau(d) = (u, v) \Rightarrow \|uv\| = 1$
2. $\tau(d_i) = (u, v), \tau(d_j) = (w, x)$, and $\|uw\| = 1 \Rightarrow c_{d_i 1} = c_{d_j 1}$
3. $\tau(d_i) = (u, v), \tau(d_j) = (w, x)$, and $\|vx\| = 1 \Rightarrow c_{d_i 2} = c_{d_j 2}$
4. $\tau(d_i) = (u, v)$ and $\tau(d_j) = (u, x) \Rightarrow d_i = d_j$

5. $\tau(d_i) = (u, v)$ and $\tau(d_j) = (w, v) \Rightarrow d_i = d_j$

where $\|uv\|$ is the Euclidean distance between u and v, and $d, d_i, d_j \in D$. We use $\tau(d)$ to determine where a domino d is located. If $\tau(d) = (u, v)$ then f_1^d will be centered at the point u, and f_2^d centered at the point v. We note here that τ may be a partial function, in which case not all dominoes are positioned on the plane.

A *layout* L is a subset of $\mathbb{Z}^2 \times \mathbb{Z}^2$ such that for all $(u, v) \in L$, $\|uv\| = 1$, and for all $(u_i, v_i), (u_j, v_j) \in L$ we have $u_i \neq u_j, v_j$ and $v_i \neq u_j, v_j$. In other words, L consists of pairs of unique points from $\mathbb{Z}^2$ that are either horizontally or vertically aligned (Figure 1(b)). We will also stipulate that $|L| = n$, i.e. the number of dominoes is exactly the right number to cover the layout. Let u and v be two points that are components of some elements of L. Then we say u and v are $adjacent$ in L iff $\|uv\| = 1$. If two members $l_i, l_j \in L$ have adjacent components, then l_i and l_j are also deemed $adjacent$. If τ maps two domino faces to positions that are adjacent in L, then those faces are deemed adjacent. If there exists a domino tiling function τ that is surjective on L for a set of dominoes D then we say that L is *tileable* by D and we refer to τ as a *tiling* of L using D.

Since we are interested in the computation of tilings, we define the following problems:

Definition 1 *In the* **DOMINO TILING** *problem we are given a set D of dominoes and a layout L and we are asked to compute a tiling τ of L if one exists.*

We also study a domino tiling problem where certain dominoes have already been positioned on the layout. Thus the layout is partially tiled with dominoes, and we wish to complete the tiling.

Definition 2 *In the* **PARTIAL DOMINO TILING** *problem, we are given a partial tiling function τ^*, a set D of dominoes, a layout L and we are asked to compute a tiling function τ such that for all $d \in D$ such that $\tau^*(d)$ is defined, we have the following:*

$$\tau^*(d) = (u, v) \quad \Rightarrow \quad \tau(d) = (u, v)$$

In the both the **DOMINO TILING** problem and the **PARTIAL DOMINO TILING** problem, τ may or may not exist.

In Section 3 we study the algorithmic aspects of **DOMINO TILING** with respect to the topology of layouts. We classify the topology of a layout L by using a graph theoretic characterization of L. We define a graph $\mathcal{G}^L = (V, E)$ whose nodes and edges are defined using the components of a layout L in the following way (see Figure 1(c)):

- $V = \{v \mid (v, u) \in L$ or $(u, v) \in L$, for some $u\}$
- $E = \{(u, v) \mid u$ and v are adjacent in $L\}$

We will use the graph properties of $\mathcal{G}^L$ to characterize L. If $\mathcal{G}^L$ is a path or a cycle, for example, then we also say that L is a path or a cycle, respectively.

We will also use a graph characterization of a set of dominoes. Using a set D of dominoes, we build an undirected pseudograph[1] whose nodes correspond to colors and whose edges correspond to dominoes. Given a set of dominoes D that use the colors $C = \{c_1, c_2, ..., c_k\}$, we define the *domino graph* $\mathcal{G}^D = (V, E)$:

- $V = C$
- $E = ((u, v) \mid (u, v) \in D)$

Thus each domino from D corresponds to exactly one edge in E, while each color from C is represented by a single node in V.

3 Tiling Paths and Cycles

In this Section, we focus on the **DOMINO TILING** problem where the provided layout L is a path or a cycle. The following Lemma provides the motivation for our algorithm for computing tilings of layout that are paths or cycles.

Lemma 1 *A layout L, which is a path, is tileable by D iff $\mathcal{G}^D$ contains an Euler trail or circuit.*

Proof. Recall that an Euler trail (resp. circuit) of a graph G is a path (resp. cycle) that uses every edge of G exactly once.

($\Rightarrow$) We define a graph $G = (V_G, E_G)$ using τ:

- $V_G = \{v \mid v$ is a face of a domino from $D\}$
- $E_G = \{(u, v) \mid$ faces u and v are adjacent under $\tau\}$

The graph G will be isomorphic to the layout graph $\mathcal{G}^L$ since each face lies a point from a component of L, and faces are only adjacent if they are one unit apart. Since L is a path, $\mathcal{G}^L$ is also a path, and hence G is a path. Recall that G has exactly one edge for each domino in D. If the faces that are at the endpoints of the path G are different colors then G describes a path in $\mathcal{G}^D$. If they are the same colors then they describe a cycle in $\mathcal{G}^D$. Moreover, this path or cycle will be Eulerian since it uses every edge (i.e. domino) exactly once.

($\Leftarrow$) Let $\Sigma = (e_1, e_2, ..., e_n)$ be an Euler trail or circuit from $\mathcal{G}^D$ that is given by its edges. We will use Σ to position dominoes from D onto L, thus computing τ.

Now we describe the procedure for positioning the dominoes of D onto L by using Σ. We begin by specifying a member of L that is at an endpoint of the path $\mathcal{G}^L$ to be `NEXT`. Then we perform the following procedure for $i = 1, 2, ..., n$:

1. Place the domino that corresponds to the edge e_i at location `NEXT` according to e_i. Now location `NEXT` is deemed occupied.

[1] A $pseudograph$ is a graph that allows both loops and multiple edges between nodes.

2. NEXT $\leftarrow$ the next unoccupied member of L that is adjacent to NEXT.

We are assured that adjacent dominoes positioned using this procedure have like-colored faces since consecutive edges in Σ have a common endpoint[2]. We are also guaranteed that all dominoes from D are used since Σ is Eulerian and edges in $\mathcal{G}^D$ have a one-to-one correspondence with the dominoes in D. $\square$

We can easily extend this proof to deal with layouts that are cycles. The only difference is that NEXT is initially any member of L, and we update NEXT cyclically around the cycle L. Thus we have the following:

Lemma 2 *A layout L, which is a cycle, is tileable by D iff $\mathcal{G}^D$ contains an Euler circuit.*

It is well-known that Euler circuits and paths can be computed on a graph $G = (V, E)$ in $O(|V| + |E|)$ time. Recall that nodes in $\mathcal{G}^D$ correspond to colors, while edges correspond to dominoes. Furthermore, since every domino is colored with at most 2 colors we have that $k \in O(n)$. Thus we have the following:

Theorem 3 **DOMINO TILING** *can be solved in $O(n)$ time if L is a path or a cycle.*

4 Tiling Partially Tiled Layouts

In this Section we show that the decision version of **PARTIAL DOMINO TILING** is NP-complete. We use the following problem in our proof, which was shown to be NP-complete in [4]:

Definition 3 *In the* **3,4-SAT** *problem we are given a boolean expression ϕ in CNF with exactly 3 variables per clause and each variable appears at most 4 times in ϕ, and we are asked to decide whether ϕ is satisfiable.*

Theorem 4 **PARTIAL DOMINO TILING** *is NP-complete.*

Proof. **PARTIAL DOMINO TILING** is in NP since we can "guess" a tiling, and in polynomial time we can verify that the tiling is valid.

We will reduce **3,4-SAT** to **PARTIAL DOMINO TILING**. We will construct a layout L, a set of dominoes D, and a partial tiling function τ^* according to the boolean expression ϕ such that a total tiling function τ will exist iff ϕ is satisfiable.

The colors we use for the faces of dominoes come from different aspects of ϕ. For each variable v_i we introduce a unique color c_{v_i}. For each clause w_i we introduce a unique color c_{w_i}. We also introduce a unique color c_{n_i} for each not operator that appears in ϕ. Finally, we have one color called c_t that represents "true", and another color called c_f that represents "false". Now we describe how L is constructed from a collection of other layouts that correspond to different aspects of ϕ.

[2] Recall that these endpoints, i.e. the vertices of $\mathcal{G}^D$, are the colors from C.

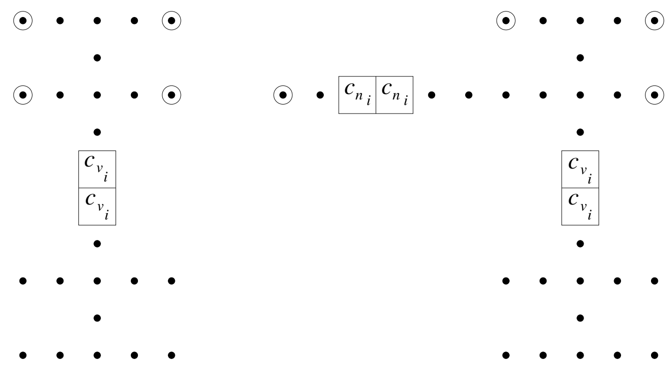

Figure 2: Left, a variable layout for an unnegated variable. Right, a variable layout with one negated output.

We will represent variables that are never negated in ϕ by constructing a layout and adding some dominoes to D. The layout that represents an unnegated variable v_i is shown in Figure 2. Four points from this layout have been emphasized. We will refer to these points are *terminals* since layouts that represent clauses will be positioned at these points. A domino (c_{v_i}, c_{v_i}), which we will refer to as a *variable domino*, has been positioned on this layout. The position of each variable domino will be reflected in our partial tiling function τ^*. We refer to the portion of a variable layout that is above the variable domino as the *value zone*, while the portion below is called the *reservoir*. For each unnegated variable in ϕ we will also add a collection of five (c_t, c_t) dominoes and five (c_f, c_f) dominoes to D. These dominoes will be used to "transmit" the truth value of a variable to a clause. We will also add two important dominoes of the form (c_t, c_{v_i}) and (c_f, c_{v_i}) to D. These dominoes will be used to "set" the truth value of a variable.

Negated variables will be handled in a very similar fashion to unnegated variables. We simply insert *negation dominoes* near the terminal on the layout for an unnegated variable (Figure 2). These dominoes will be of the form (c_{n_i}, c_{n_i}). The fact that such negation dominoes exist will be reflected in τ^*. For each negation domino we add two dominoes (c_{n_i}, c_t) and (c_{n_i}, c_f) to D.

We represent each clause w_i by exactly three layouts that will be adjacent to the layouts for the variables that are in the clause w_i. The three layouts for clause w_i are shown in Figure 3. The emphasized points in Figure 3 are called *terminals*. There are three dominoes of the form (c_{w_i}, c_{w_i}) that have been positioned on these clause layouts. We will refer to these dominoes as *clause dominoes*. The positions of clause dominoes will be reflected in τ^*. The portion of the layout that is to the left of the clause domino will be referred to as the *reservoir* of the clause layout. Notice that

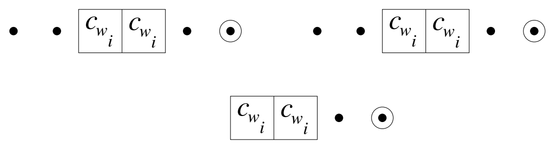

Figure 3: The three layouts that represent a clause.

one of the clause layouts does not have a reservoir. Each of these layouts corresponds to exactly one of the variables that is in clause w_i. Each of them will be positioned so that their terminal is adjacent to one of the four terminals in the layout for the variable they correspond with. For this to be accomplished, clause layouts may need to be rotated. For each clause w_i, we add three dominoes of the form (c_{w_i}, c_t) and two dominoes of the form (c_{w_i}, c_f) to D. Figure 4 shows a variable layout that has two clause layouts attached to it.

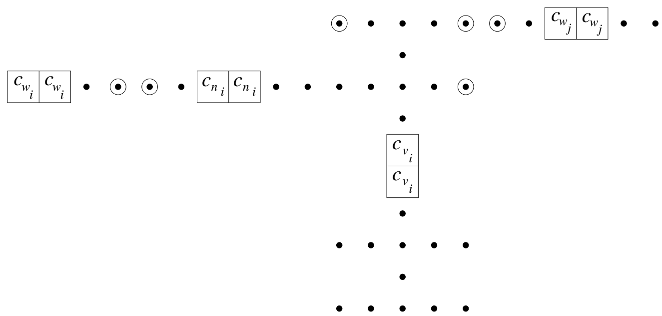

Figure 4: Variable v_i appears negated in clause w_i and unnegated in clause w_j.

Now suppose that there exists some tiling function τ that is a solution to **PARTIAL DOMINO TILING** for L, D, and τ^* as described above. We will show that if τ exists then ϕ is satisfiable. Specifically, we will use a truth assignment of ϕ that is implied by τ. First let us examine the dominoes that have been positioned next to variable dominoes. Variable dominoes are only adjacent to two other dominoes from D. Recall that the variable domino for v_i is colored c_{v_i}. In our construction of D, we placed exactly two dominoes that have faces with color c_{v_i}. These two dominoes are of the form (c_t, c_{v_i}) and (c_f, c_{v_i}). Thus we are guaranteed that these two dominoes are adjacent to the variable domino for v_i. We will assign a truth value to variable v_i in ϕ according to which of these two dominoes gets placed in the value zone of the variable layout for v_i: if (c_t, c_{v_i}) is positioned in the value zone then $v_i =$ *true*, and $v_i =$ *false* otherwise. Now consider negation dominoes. The i^{th} negation domino is only adjacent to two other dominoes, and due our construction, these two dominoes will be (c_{n_i}, c_t) and (c_{n_i}, c_f). Now consider clause dominoes. Due to our construction, all dominoes with faces colored c_{w_i} must be positioned next to a corresponding clause domino. This ensures that all clause layout reservoirs get filled by either (c_{w_i}, c_t) or (c_{w_i}, c_f) dominoes. Also, since there are only two dominoes of the form (c_{w_i}, c_f), we are guaranteed that at least one layout component for each clause will have a face of color c_t on its terminal. This corresponds to each clause from ϕ being satisfied. We still must show that if a clause layout has a c_t face on its terminal then the variable associated with that terminal has either been set of true, or it has been set to false and it appears negated in the clause. First consider the case where a variable v_i appears unnegated in clause w_i and suppose a clause layout for w_i has a c_t face on the terminal associated with v_i. Since this terminal is adjacent to a terminal in the v_i's variable layout, all the positions in v_i's value zone will be occupied by dominoes with c_t faces. The same argument works for clause terminals that are covered by c_f faces. Thus for unnegated variables, truth values are properly propagated to clauses. We can easily extend this argument for negated variables by noticing that negation dominoes simply "flip" the truth value. Thus we have that every clause is satisfied

Now suppose that ϕ is satisfiable. We will compute a tiling function τ from a satisfying truth assignment of ϕ.

If a variable v_i is set to *true* then we position domino (c_{v_i}, c_t) in v_i's variable layout value zone along with five (c_t, c_t) dominoes. We also place (c_{v_i}, c_f) along with five (c_f, c_f) dominoes in v_i's reservoir. Placing the five (c_t, c_t) dominoes in v_i's value zone will ensure that all unnegated terminals correspond to clauses that have a (c_{w_i}, c_t) domino at their terminal. If a variable has a negated terminal, then this forces the (c_{n_i}, c_t) to be adjacent to the variable layout's terminal, which in turn forces the (c_{n_i}, c_f) domino to be adjacent with the terminal for the corresponding clause. This causes the clause to have a (c_{w_i}, c_f) domino at its terminal. Since ϕ is satisfiable, this can only occur at at most two variable layouts. Recall that we've only added two dominoes of the form (c_{w_i}, c_f) for each clause. This implies that variable layouts that correspond to a variable that has been set to true can be tiled.

If the variable v_i is false, we do the exact opposite: we fill the value zone with (c_{v_i}, c_f) and five (c_f, c_f) dominoes, and the reservoir with (c_{v_i}, c_t) and five (c_t, c_t) dominoes. We can argue that such layouts can be tiled by using the reverse of the argument given above. The (c_f, c_f) dominoes in the value zone will force unnegated terminals to be associated with clauses that have a (c_{w_i}, c_f) at their terminal. Since ϕ is satisfiable, at most two variables will have this property, which corresponds to the fact that their are only two (c_{w_i}, c_f) dominoes per clause. $\square$

The layout L constructed in the proof of Theorem 4 is disconnected in the sense that $\mathcal{G}^L$ is disconnected. We can easily augment L so that it is connected. We can achieve this by "daisy chaining" the variable layouts with fixed position dominoes: the variable layout for v_i is connected to the variable layout for v_{i+1} by a path that contains fixed position dominoes. It is easily seen that this new connected layout is also a tree in the sense that $\mathcal{G}^L$ is a tree. Thus we have the following:

Corollary 5 **PARTIAL DOMINO TILING** *is NP-complete even if L is a tree.*

5 Conclusions and Future Work

We've shown that domino tilings of layouts that are either paths or cycles can be computed in $O(n + k)$ time, where n is the number of dominoes and k is the number of colors on the domino faces. We've also shown that if a layout is partially tiled at the outset of the problem, then the problem is NP-complete, even if the layout is a tree.

Many questions regarding domino tilings remain open. We are interested in the time complexity of **DOMINO TILING** where L is a tree or a graph. Also, we have only considered domino tiling problems where $n = |L|$. Many interesting problems arise when we consider so-called *imperfect tilings* where $n > |L|$. In this situation, we wish to find subsets of D that can tile L. It would also be interesting to explore the similarities and differences between domino tilings and Wang tilings.

6 Acknowledgements

We would like to thank Dr. Mark Daley, Dr. Grant Cheston, and Dr. Mark Keil for providing useful insights into the problems discussed herein.

References

[1] L. Adleman, J. Kari, L. Kari, and D. Reishus. On the decidability of self-assembly of infinite ribbons. *Proceedings of FOCS 2002, IEEE Symposium on Foundations of Computer Science*, pages 530–537, 2002.

[2] R. Berger. The undecidability of the domino problem. *Memoirs of the American Mathematical Society*, (66):1–72, 1966.

[3] J. Kari. Infinite snake tiling problems. In M. Ito and M. Toyama, editors, *Developments in Language Theory, 6th International Conference, DLT 2002, Kyoto, Japan, September 18-21, 2002, Revised Papers*, volume 2450 of *Lecture Notes in Computer Science*, pages 67–77. Springer, 2003.

[4] C. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, (8):85–89, 1984.

[5] H. Wang. Proving theorems by pattern recognition. ii. *Bell Systems Technical Journal*, (40):1–41, 1961.