On the Computation and Chromatic Number of Colored Domino Tilings

Chris Worman* Boting Yang†

Abstract

A colored domino is a rotatable 2×1 rectangle that is partitioned into two unit squares, which are called faces, each of which is assigned a color. In a colored domino tiling of an orthogonal polygon P, a set of dominoes completely covers P such that no dominoes overlap and so that adjacent faces have the same color. We provide tight bounds on the number of colors required to tile simple and non-simple orthogonal polygons. We also present an algorithm for computing a colored domino tiling of a simple orthogonal polygon.

1 Introduction

Many results concerning domino tilings have focused on "colorless" dominoes, which are simply 2×1 rotatable rectangles (see, for example, [5, 3, 8, 6, 7, 4]). A colored domino is a domino that is partitioned into two unit squares, each of which is assigned a color. Thus a colored domino models the commonly used domino game piece. In a tiling that uses colored dominoes, adjacent faces must have the same color. Results concerning colored domino tilings have only arisen relatively recently [9, 2]. In the colored domino tiling problems studied in [9] and [2], a multiset of dominoes is provided, and in tilings the multiplicity of the dominoes cannot exceed those in the multiset. In [9], the multiplicity of the provided dominoes equals that of the multiplicity of the dominoes used in the tiling. An algorithm is described for computing colored domino tilings of so-called "paths" or "cycles". This algorithm runs in time linear in the number of dominoes used in the tiling. The authors of [9] also consider a colored domino tiling problem where some dominoes have already been positioned on the polygon, and we are asked to decide if the tiling can be completed. This problem is shown to be NP-complete. In [2], Biedl studies two variants of a domino tiling problem. In the first problem, known as *EXACT DOMINO TILING*, the multiplicity of dominoes in the provided set is equal to multiplicity of the dominoes used in the tiling. Biedl shows that *EXACT DOMINO TILING* is NP-complete, even for a very restricted class of polygons known as "caterpillars". It is also shown that *EXACT DOMINO TILING* remains NP-complete when the dominoes are restricted to three colors. In the second domino tiling problem studied in [2], which is called *PARTIAL DOMINO TILING*, not all the dominoes are used in the tiling. It is shown that *PARTIAL DOMINO TILING* is NP-complete, and remains so even for so-called "paths" or "cycles". It is also shown that *PARTIAL DOMINO TILING* is NP-complete when the dominoes are restricted to three colors.

In the problems that we study, the set of dominoes used is not a multiset, and each domino can be used an unlimited number of times. Thus in this kind of colored domino tiling problem, the set of dominoes provides a set of possible "types" of dominoes that can be used in the tiling.

By making a connection between graph coloring and colored domino tiling, we show that if we wish to tile a simple orthogonal polygon, then 2 colors are always sufficient. We also show that if the polygon contains holes, then 4 colors are always sufficient and sometimes necessary. We also describe an algorithm for deciding whether or not a simple orthogonal polygon can be tiled with colored dominoes. This algorithm is constructive and actually computes the tiling if one exists. This algorithm runs in time $O(n)$, where n is the number of dominoes used in the tiling. All our results disallow monochromatic dominoes, which we call *twin* dominoes. The reason for excluding twin dominoes is necessitated from the problem statement: if we allow twin dominoes then color becomes irrelevant and the problem reduces to that of partitioning a polygon into 2×1 rectangles.

Due to space constraints, many details have been omitted in this version of the paper. Complete details can be found in the long version of this paper [10].

2 Basic Definitions

A *colored domino* is a rotatable 2×1 rectangle that has been partitioned into two colored unit squares, which we refer to as the *faces* of the colored domino. Since all dominoes considered herein will be colored, we will often refer to "colored dominoes" simply as "dominoes". All polygons that we consider are orthogonal polygons with integer coordinates. We adopt terminology similar to that found in [9, 2], and refer to such polygons as *layout polygons*. We refer to the set of unit squares induced by the integer grid within a layout polygon P as the *squares* in P, which are denoted by $\rho(P)$. A *tiling* is a placement of dominoes from a set D onto a lay-

*Department of Computer Science, University of Regina, {worman2c, boting}@cs.uregina.ca

†Research supported by NSERC.

out polygon P such that each square in P is covered by exactly one domino face, and adjacent faces of dominoes have the same color. We say that $p, q \in \rho(P)$ are *matched* under a tiling if p and q are covered by the same domino in the tiling. A tiling of a layout polygon P therefore describes a perfect matching of $\rho(P)$.

We are concerned with tilings that disallow twin dominoes. Motivated by this restriction, we define D_k as follows: D_1 is a singleton containing a solitary twin domino, and for $k \geq 2$, we define D_k to be the set of $\binom{k}{2}$ non-twin dominoes over k colors.

Definition 2.1 k-tileable: *A layout polygon P is* k-tileable *if and only if there exists a tiling of P using dominoes from D_k.*

Extending this notion slightly, we define a *k-tiling* to be a tiling that uses dominoes from D_k.

3 Leaves, Corners, and Color Partitions

Definition 3.1 Leaf: *Let L be a 2×1 subrectangle of P. Then L is a* leaf *of P if L contains a square p such that p is adjacent to exactly one square in $\rho(P)$.*

Definition 3.2 Corner: *Let C be a 2×2 subrectangle of P. Then C is a* corner *of P if C contains a square p such that p is adjacent to only two other squares in $\rho(P)$.*

The following two Lemmas about leaves and corners will be essential to our algorithm, and will also be useful when we obtain lower bounds on the number of colors needed to tile a non-simple layout polygon.

Lemma 1 *Let P be a k-tileable layout polygon for $k \geq 2$ that contains a leaf L. In any k-tiling of P there is only one domino that intersects with L.*

Lemma 2 *Let P be a k-tileable layout polygon for $k \geq 2$ that contains a corner C. In any k-tiling of P there are only two dominoes that intersect with C.*

A key observation about leaves and corners is their existence in simple layout polygons.

Lemma 3 *If P is a simple layout polygon such that $|\rho(P)| \geq 2$, then P contains a leaf or a corner.*

The next set of definitions provides a connection between graph coloring and colored domino tiling. If P is a layout polygon that has been tiled with colored dominoes, then we refer to the color of the domino face that is covering the square $p \in \rho(P)$ as $color(p)$.

Definition 3.3 Color Partition: *Let P be a k-tileable layout polygon, and let τ be a k-tiling of P. We define the color partition of P with respect to τ to be a partition $P_1, P_2, ..., P_m$ of P that satisfies each of the following conditions:*

1. *P_i is a connected set of like-colored squares from $\rho(P)$.*
2. *If p_i and p_j are adjacent squares from $\rho(P)$ that are in different regions from the color partition, then $color(p_i) \neq color(p_j)$.*

Definition 3.4 Color Partition Graph: *Let $P_1, P_2, ..., P_m$ be the color partition associated with a k-tiling τ of a k-tileable layout polygon P. We define the* color partition graph *to be a graph with vertices $v_1, v_2, ..., v_m$. There is an edge (v_i, v_j), $i \neq j$, in the color partition graph if the partitions P_i and P_j share an edge on their boundaries.*

Notice that if we color the nodes of a color partition graph G according to the colors of the corresponding color partitions, then we have a proper k-coloring of G. Likewise, a k-coloring of G defines a k-tiling of the original polygon. Thus a k-coloring of a color partition graph corresponds exactly to a k-tiling. This is the key observation that is exploited in the following section.

4 Colored Domino Tilings and Graph Coloring

In this section we obtain tight bounds on the number of colors needed to tile simple and non-simple layout polygons.

Definition 4.1 Chromatic Number of P: *The* chromatic number *of a layout polygon P, denoted $\chi(P)$, is the minimum number $k \geq 2$ for which P is k-tileable.*

Lemma 4 *If P is a k-tileable simple layout polygon with a k-tiling τ, for $k \geq 2$, then the color partition graph G associated with τ is a tree.*

Since trees are 2-colorable, we have the following:

Theorem 5 *If P is a k-tileable simple layout polygon for $k \geq 2$, then $\chi(P) = 2$.*

Now we turn our attention to non-simple layout polygons, i.e. layout polygons that contain one or more holes. We can show that the layout polygon depicted in Figure 1 requires at least 4 colors in any k-tiling for $k \geq 2$.

Lemma 6 *There exists a non-simple layout polygon that requires at least four colors in any k-tiling for $k \geq 2$.*

If we combine Lemma 6 with the fact that any color partition graph of a layout polygon is planar, and is therefore 4-colorable, we obtain the following:

Theorem 7 *Let P be a k-tileable layout polygon for $k \geq 2$. Then $\chi(P) \leq 4$, and this bound is tight.*

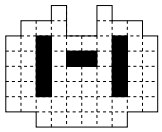

Figure 1: A layout polygon P that requires at least 4 colors to tile it. P contains 3 holes, which are depicted as black rectangles.

5 Computing Tilings of Simple Polygons

Now we turn our attention to an algorithm for the following tiling problem regarding simple layout polygons:

Definition 5.1 TWINLESS TILING: *In the TWINLESS TILING problem, we are given a simple layout polygon P, and we are asked to decide whether or not P is k-tileable for $k \geq 2$, and if so, we are to compute such a tiling.*

Our algorithm for the *TWINLESS TILING* problem operates on the following decomposition of the layout polygon.

Definition 5.2 Leaf-Corner Decomposition: *A* leaf-corner decomposition *of a layout polygon P, denoted $\mathcal{L}(P)$, is defined recursively as follows: if P does not contain a leaf or a corner, then $\mathcal{L}(P) := \emptyset$. Otherwise, let X be a leaf or corner of P, and let $P_1, P_2, ..., P_l$ be the simple layout polygons obtained by removing X from P. Then we define $\mathcal{L}(P) := \{X\} \bigcup \mathcal{L}(P_1) \bigcup \mathcal{L}(P_2) \bigcup ... \bigcup \mathcal{L}(P_l)$.*

We say that a leaf-corner decomposition $\mathcal{L}(P)$ *covers* P if and only if the union of all the members in $\mathcal{L}(P)$ is P.

Lemma 8 *Given a simple layout polygon P and a leaf-corner decomposition $\mathcal{L}(P)$, if $\mathcal{L}(P)$ does not cover P then P is not k-tileable for $k \geq 2$.*

The main objective of our algorithm is to compute a perfect matching of $\rho(P)$ that has a special property. We ensure that the computed perfect matching of $\rho(P)$ does not contain any so-called twin-forcing arrangements:

Definition 5.3 Twin-Forcing Arrangement: *Let P be a layout polygon with a perfect matching M of $\rho(P)$. A* twin-forcing arrangement *is a set of four squares $p, q, r, s \in \rho(P)$, such that p and q are adjacent and matched with each other, while r and s are adjacent, but r is not matched with s.*

Lemma 9 *Let P be a simple layout polygon. P is k-tileable with matching M, for $k \geq 2$, if and only if M is a perfect matching of $\rho(P)$ that does not contain a twin-forcing arrangement.*

Let C be a corner, from a leaf-corner decomposition $\mathcal{L}(P)$ of a simple layout polygon P. We can show that C must be either be matched "vertically" or "horizontally". The key to our algorithm is noticing that the matching of a leaf or a corner may "force" neighboring leaves and corners to be matched a certain way if we are to avoid twin-forcing arrangements. We say that two members X and Y of a leaf-corner decomposition are *adjacent* if there exists two squares $p, q \in \rho(P)$, such that $p \in X$, $q \in Y$, and p and q are adjacent.

Lemma 10 *Let P be a simple layout polygon with a perfect matching M of $\rho(P)$, and a leaf-corner decomposition $\mathcal{L}(P)$ that covers P. $\mathcal{L}(P)$ satisfies the following rules with respect to M if and only if M does not contain a twinforcing arrangement.*

1. *For each edge e of each leaf $L \in \mathcal{L}(P)$, there is at most one member of $\mathcal{L}(P)$ that both intersects e and is adjacent to L.*
2. *If $C_i, C_j \in \mathcal{L}(P)$ are two corners that share and entire edge, then C_i and C_j are both matched horizontally or they are both matched vertically.*
3. *If $L, C_i \in \mathcal{L}(P)$ are a leaf and a corner such that L shares and entire vertical (resp. horizontal) edge with C_i, then C_i is matched vertically (resp. horizontally).*
4. *If $X, Y \in \mathcal{L}(P)$ are both adjacent to a corner $C_i \in \mathcal{L}(P)$, such that both X and Y intersect the same vertical (resp. horizontal) edge of C_i, then C_i is matched horizontally (resp. vertically).*

Now we can describe our algorithm for solving the *TWINLESS TILING* problem.

Theorem 11 *The TWINLESS TILING problem can be solved in $O(|\rho(P)|)$ time, or equivalently, $O(n)$ time, where n is the number of dominoes needed in the tiling.*

Proof. The algorithm begins by constructing a leaf-corner decomposition $\mathcal{L}(P)$ of P. If $\mathcal{L}(P)$ does not cover P then the algorithm halts and reports that P is not k-tileable for $k \geq 2$. The correctness of this step of the algorithm is guaranteed by Lemma 8.

Then the algorithm uses rules (1)-(4) of Lemma 10 to construct a boolean expression ϕ, which is an instance of the 2SAT problem. For each corner $C_i \in \mathcal{L}(P)$ we create two variables h_i and v_i, which correspond to C_i being matched horizontally and vertically, respectively. In order to ensure that exactly one matching of C_i is chosen, we add the following clauses to ϕ: $(h_i \vee v_i)$ and $(\neg h_i \vee \neg v_i)$. If rule (1) is violated for any leaf in $\mathcal{L}(P)$, we add the unsatisfiable clause $(false)$ to ϕ. We encode rules (2)-(4) as clauses by inspecting the neighbors of each leaf and corner in $\mathcal{L}(P)$. If the condition of rule

(2) is satisfied, then we add the following clauses to ϕ: $(\neg h_i \vee h_j)$, $(\neg h_j \vee h_i)$, $(\neg v_i \vee v_j)$, and $(\neg v_j \vee v_i)$. If the condition of rule (3) is satisfied, we add the following clause to ϕ: (v_i) (resp. (h_i)). For rule (4) we conditionally add the following clause: (h_i) (resp. (v_i)).

By construction, ϕ is satisfiable if and only if $\mathcal{L}(P)$ conforms to rules (1)-(4) of Lemma 10. Furthermore, a satisfying assignment of ϕ corresponds exactly to a perfect matching M of $\rho(P)$, and by Lemma 10, M does not contain a twin-forcing arrangement. Thus by Lemma 9, ϕ is satisfiable if and only if P is k-tileable. Thus we can decide if P is k-tileable for $k \geq 2$ by solving the 2SAT problem on ϕ.

To construct a tiling of P, we observe that M is a matching of some 2-tiling (see full version for details). We compute the colors of the dominoes in a 2-tiling of P as follows. We color an arbitrary square $p \in \rho(P)$ using the color black. Then we proceed using DFS, only visiting those squares that are uncolored. When we visit a square q, we assign a color to q based upon the color of a neighboring square: if a neighbor is colored black, and q is matched with this neighbor, then q is colored white, otherwise q is colored black. Since DFS is used to visit the squares, each square is assigned a color, and we are guaranteed that no twin dominoes arise since M is the matching of a 2-tiling.

Before considering the time taken by our algorithm, we make an important note concerning the input to our algorithm. The time taken by algorithms that operate on polygons is typically expressed in terms of the number of vertices that the polygon has. This presents a problem for the *TWINLESS TILING* problem since we can describe a layout polygon with $O(1)$ vertices that contains an arbitrary number of squares. Although we can deal with this problem in a number of different ways, we adopt the following assumption: the layout polygon P is described by $\rho(P)$ in the input in the form of an adjacency list of squares and neighbors. Using this assumption, we can access the neighbors of each square in $O(1)$ time since each square has at most four neighbors.

$\mathcal{L}(P)$ can be computed in $O(|\rho(P)|)$ time as follows. First we identify all the leaves and corners in P by examining each square and its neighbors. Place each of these leaves and corners in a list called T . Then remove the next item from T and call it X. If any of the squares in X are marked, then ignore X, otherwise add X to $\mathcal{L}(P)$ and mark all the squares in X. After marking the squares in X, add any new leaves or corners that are created to the list T . This procedure stops after $O(|\rho(P)|)$ steps. Computing ϕ can clearly be accomplished in $O(|\rho(P)|)$ time by examining the $O(1)$ neighbors of each leaf and corner from $\mathcal{L}(P)$. Notice that ϕ has $O(|\rho(P)|)$ clauses. Thus a satisfying assignment of ϕ can be computed in $O(|\rho(P)|)$ time using the algorithm from [1]. Assigning colors to the squares in $\rho(P)$ uses DFS and hence can be accomplished in $O(|\rho(P)|)$ time. □

6 Open Problems

We are interested in characterizing the layout polygons that require at most 3 colors in any k-tiling for $k \geq 2$. Since the submission of this paper, we have shown that the *TWINLESS TILING* problem is NP-complete for non-simple polygons.

7 Acknowledgements

We would like to thank Therese Biedl for useful comments regarding the results presented in section 4. We would also like to thank an anonymous referee whose comments lead to an improvement of the algorithm presented in Section 5.

References

[1] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inf. Process. Lett.*, 8(3):121–123, 1979.

[2] T. Biedl. The complexity of domino tiling. Unpublished Manuscript.

[3] G. Csizmadia, J. Czyzowicz, L. Gasieniec, F. Kranakis, and J. Urrutia. Domino tilings of orthogonal polygons. In *Canadian Conference on Computational Geometry (CCCG'99)*, pages 154–157, 1999.

[4] J. Czyzowicz, E. Kranakis, and J. Urrutia. Domino tilings and two-by-two squares. In *Canadian Conference on Computational Geometry (CCCG'97)*, 1997.

[5] C. Kenyon and R. Kenyon. Tiling a polygon with rectangles. In *33rd Fundamentals of Computer Science (FOCS)*, pages 610–619, 1992.

[6] M. L. N. Elkies, G. Kuperberg and J. Propp. Alternating sign matrices and domino tilings, part 1. *Journal of Algebraic Combinatorics 1*, pages 111–132, 1992.

[7] M. L. N. Elkies, G. Kuperberg and J. Propp. Alternating sign matrices and domino tilings, part 2. *Journal of Algebraic Combinatorics 1*, pages 219–234, 1992.

[8] J. Propp. A reciprocity theorem for domino tilings. *The Electronic Journal of Combinatorics*, 8, 2001.

[9] C. Worman and M. Watson. Tiling layouts with dominoes. In *Proceedings of the 16th Canadian Conference on Computational Geometry (CCCG'04)*, pages 86–90, 2004.

[10] C. Worman and B. Yang. On the computation and chromatic number of colored domino tilings. http://www.cs.uregina/∼worman2c/cccg-05-domino-paper.ps, 2005.