# Tromino Tilings with Pegs via Flow Networks

Javier T. Akagi[1]

*Facultad Politécnica, Universidad Nacional de Asunción*
*NIDTEC, Campus Universitario, San Lorenzo C.P. 111421, Paraguay*

Eduardo A. Canale[2]

*Facultad de Ingeniería, Universidad de la República*
*Montevideo PC 11300, Uruguay*

Marcos Villagra[3]

*Facultad de Ciencias Exactas y Naturales, Universidad Nacional de Asunción*
*Dpto. de Matemáticas, Campus Universitario, San Lorenzo C.P. 111421, Paraguay*

**Abstract**

A tromino tiling problem is a packing puzzle where we are given a region of connected lattice squares and we want to decide whether there exists a tiling of the region using trominoes with the shape of an L. In this work we study a slight variation of the tromino tiling problem where some positions of the region have pegs and each tromino comes with a hole that can only be placed on top of the pegs. We present a characterization of this tiling problem with pegs using flow networks and show that (i) there exists a linear-time parsimonious reduction to the maximum-flow problem, and (ii) counting the number of such tilings can be done in linear-time. The proofs of both results contain algorithms that can then be used to decide the tiling of a region with pegs in $O(n)$ time.

*Keywords:* tromino tilings, linear-time reduction, parsimonious reduction, maximum-flow, bipartite matchings

## 1 Introduction

### *1.1 Background*

A *polyomino tiling problem* is a packing puzzle where a player attempts to cover a board or region with polyominoes of a given shape. A *polyomino* is a set of square cells joined together by their edges. Of particular interest are polyominoes of two cells called *dominoes*, and polyominoes of three cells called *trominoes* of which there are two types: L-trominoes with the shape of an L and I-trominoes with the shape of an I. From now on, and for the remainder of this work, we will refer to L-trominoes simply as trominoes.

One of the first rigorous studies of polyominoes is due to Golomb [5] who introduced a hierarchy of tiling capabilities by polyominoes. Later, Conway and Lagarias [2] presented an algebraic necessary condition to cover a region with polyominoes. In particular, the study of domino tilings uncovered many tight connections with matrix algebra through the theory of alternating-sign matrices and graph matchings [4].

In computational complexity theory, Moore and Robson [7] showed that deciding the tiling of a given region with trominoes is NP-complete. Demaine and Demaine [3] showed the polynomial-time equivalence

[1] Email:akagi.tada@gmail.com
[2] Email:canale@fing.edu.uy
[3] Email:mvillagra@pol.una.py

Fig. 1. Region with pegs. Aerial view on the left and a 3D view on the right.

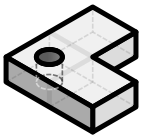

Fig. 2. A p-tromino. The corner cell contains a hole and can only be placed on top of a peg on a region with pegs.

Fig. 3. Placement of a p-tromino. Aerial view on the left and 3D view on the right.

between polyomino tilings, edge-matching puzzles and jigsaw puzzles. Horiyama *et al.* [6] strengthened the NP-completeness result of Moore and Robson and proved that counting the number of tromino packings of a region is #P-complete. More recently, Akagi *et al.* [1] showed that the NP-completeness of deciding a tromino tiling heavily depends on geometrical properties of the region; for example, a region with the shape of an Aztec rectangle admits a polynomial time algorithm that decides a tromino tiling, whereas the problem remains NP-complete for an Aztec rectangle with at least two holes.

### *1.2 Contributions*

In all cited works of the previous paragraphs most of the arguments rely on a (partial) characterization of the tiling problem using a graph-theoretic idea. One such idea is the so-called *dual graph* [1] of a region, which is a graph where each cell of the region is a vertex and there is an edge between two vertices if their corresponding cells are adjacent. There is a natural correspondence between domino tilings and perfect matchings in dual graphs. Likewise, Akagi *et al.* [1] showed that tromino tilings correspond to independent sets in a slight variation of dual graphs called an intersection graph.

In this work we present yet another way to characterize tromino tiling problems using this time an idea of flow networks. This characterization, however, applies to a slight variation of the tromino tiling problem where a region contains pegs, see Fig. 1. To cover a region with pegs we use a special type of tromino that we call *p-tromino* which contains a hole in its corner cell as shown in Fig. 2. In a tiling of a region with pegs, only p-trominos can be placed on top of pegs and we assume that there is an infinite supply of p-trominoes and only p-trominoes; that is, there are no regular trominoes with no holes and the hole of a p-tromino cannot be placed on top of a cell with no peg. See Fig. 3 for an example of a correct placement of a p-tromino. We thus define the *P-Tromino Tiling Problem* as the problem of deciding, given a region with pegs, whether there is a tiling of the region with p-trominos. More formally, the decision problem is:

P-Tromino$_{n,k}$

| | | |
|---|---|---|
| INPUT | : | Region $R$ with $n$ cells and a list $P$ of $k$ coordinates of $R$ with pegs. |
| OUTPUT | : | "`Yes`" if there exists a tiling of $R$ with p-trominoes; "`No`" otherwise. |

Now let us recall that in a *maximum-flow problem* we are given a flow network represented by a directed

[1] Note that this is an abuse of the term dual graph used in graph theory. The practice is, however, standard in combinatorics of tilings.

graph and we want to find a flow of maximum value that can be pushed from a source vertex to a sink vertex. Intuitively, the value of a flow is the rate at which some hypothetical material moves through the network. We consider the following decision problem:

MaxFlow$_k$

| | | |
|---|---|---|
| INPUT | : | Flow network $G$, positive integer $k$. |
| OUTPUT | : | "`Yes`" if the value of a maximum flow in $G$ equals $k$; "`No`" otherwise. |

Now we are ready to state the main result of this work.

**Theorem 1.1** *There exists a parsimonious reduction from* P-Tromino$_{n,n/3}$ *to* MaxFlow$_{n/3}$ *that is computable in linear-time.*

The main idea for a proof of Theorem 1.1 is the construction of a flow network representation of a region with pegs such that flow passing through a path in the network, from source to sink, corresponds to the placement of a p-tromino on the region.

As a second result we show that the counting version of P-Tromino$_{n,n/3}$, namely #P-Tromino$_{n,n/3}$, is computable in linear-time.

**Theorem 1.2** #P-Tromino$_{n,n/3}$ *is computable in* $O(n)$ *time.*

The idea of a proof of Theorem 1.2 exploits the structure in the flow network representation of a region with pegs. In particular, we show the existence of induced bipartite subgraphs in the flow network where any maximum matching is a perfect matching of size $n/3$ and each perfect matching agrees with a p-tromino tiling of the region. This reduces our problem of counting p-tromino tilings to counting perfect matchings in a highly structured bipartite graph.

If we are only interested in finding a p-tromino tiling, we can slightly modify our algorithm of Theorem 1.2 for counting perfect matchings and obtain an $O(n)$ time algorithm that finds a p-tromino tiling.

### *1.3 Outline*

The rest of the paper is organized as follows. In Section 2 we introduce the notation used throughout this work and give some precise definitions of tromino tilings and flow networks. In sections 3 and 4 we present our proofs of theorems 1.1 and 1.2, respectively. Finally, we close this paper in Section 5.

## 2 Preliminaries

In this section we review some definitions from tromino tilings and flow networks and present the notation used throughout this work. We use $\mathbb{Z}$ to denote the set of all integers, and $\mathbb{N}$ will denote the set of all natural numbers including 0. We also denote by $[a, b]$ the discrete interval $\{a, a + 1, \ldots, b\}$.

### *2.1 Tromino Tilings*

A region $R$ is a finite union of cells whose interior is connected. If $[a, a + 1] \times [b, b + 1]$ is a cell in $R$, its *coordinate* in $R$ is $(a, b)$. To denote the cell at coordinate $(a, b)$ we use the shorthand notation $R_{(a,b)}$. Two cells are *neighbors* or *adjacent* if the Manhattan distance between the two cells is 1; thus, two cells in diagonal to each other are not adjacent.

A *tromino* is a polyomino of 3 cells with the shape of an L, as we explained in the previous section. Given a tromino $T$, we will refer to the cell in its corner as the *corner cell* of $T$ and the other two cells we will refer to them as the *tips* of $T$. A *cover* or *tiling* of a region $R$ is a set of trominoes covering all cells of $R$ with no overlapping and each tromino is inside $R$. The size of a cover is the number of trominoes in it.

Given a region $R$ and a list of $k$ positions in $R$ denoting positions of pegs on the board, we define the *tromino tiling with pegs* problem denoted P-Tromino$_{n,k}$, where each tromino comes with a hole in its corner cell and can be placed on the region $R$ only if that corner cell is placed on top of a position with a peg. A tromino with a hole in its corner is called a *p-tromino*. A cover of $R$ with pegs is called a *p-cover*, and P-Tromino$_{n,k}$ is the problem of deciding whether there is a p-cover or not.

### *2.2 Flow Networks*

Given a graph $G = (V, E)$, let $V(G)$ and $E(G)$ denote its set of vertices and edges, respectively. The *degree* of a vertex $v$ in $G$ is the number $d_G(v)$ of vertices adjacent with $v$ in $G$.

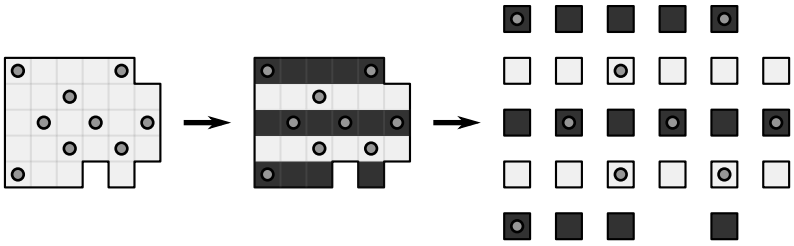

Fig. 4. Coloring of the cells with respect the row number. We start counting from row 0 which is the bottom row of the region.

A *flow network* $G = (V, E)$ is a directed graph where each directed edge $(u, v)$ has an assigned capacity $c(u, v) \geq 0$. If $(u, v)$ is an edge of $G$, then $(v, u)$ is not an edge of $G$. For edges not appearing in $G$ we will usually assign them a capacity of 0. Furthermore, self-loops are not allowed in flow networks. We also assign to each vertex $v \in V(G)$ a capacity $c(v) \geq 0$. Note that we overload our notation and we use $c(u, v)$ with $(u, v) \in E(G)$ to denote capacities on edges and $c(v)$ with $v \in V(G)$ to denote capacities on vertices.

Every flow network $G$ has two distinguished vertices called a *source* denoted $s$ and a *sink* denoted $t$. A path in $G$ from $s$ to $t$ is called an *st-path* and it is denoted $s \rightsquigarrow t$. The in-degree of $s$ and the out-degree of $t$ are always 0. If $v$ is a vertex in $G$, an st-path passing through $v$ is denoted $s \overset{v}{\rightsquigarrow} t$.

A *flow* in $G$ is a function $f : E(G) \to \mathbb{N}$ that satisfies the following three constraints: (i) a *capacity constraint on the edges* $0 \leq f(u, v) \leq c(u, v)$ for all $(u, v) \in E(G)$, (ii) a *capacity constraint on the vertices* $\sum_{(u,v)\in E(G)} f(u, v) \leq c(v)$ for all $v \in V(G)$, and (iii) a *flow conservation constraint* $\sum_{v\in V(G)} f(v, u) = \sum_{v\in V(G)} f(u, v)$ for all $u \in V(G) \setminus \{s, t\}$. If $e$ is an edge of $G$ we call $f(e)$ the flow through $e$, and the flow through an st-path is defined as $f(s \rightsquigarrow t) = \min_{e\in E(G)}\{f(e)\}$. The *value* of a flow $f$ is defined as $|f| = \sum_{v\in V(G)} f(s, v)$. The *maximum-flow problem* is thus the problem of finding a flow of maximum value.

## 3 Proof of Theorem 1.1

Let $R$ be a connected region with $n$ cells and let $P$ be a collection of cells from $R$ that contain pegs, where $|P| = n/3$. We assume that $n$ is a multiple of 3, since, otherwise the problem is trivially unsolvable. The following procedure is an algorithm that takes an instance $(R, P)$ from P-Tromino$_{n,n/3}$ and transforms it to an instance $(G, n/3)$ of MaxFlow$_{n/3}$.

(i) Color each cell $R_{(a,b)} \in R$ according to its row number. A cell in an even row is colored black, and a cell in an odd row is colored white; see Fig. 4.

(ii) Partition the cells of $R \setminus P$ into two sets $B$ and $W$. We define $B$ to be the set of all cells of $R$ that are colored black and do not contain a peg, that is, $B = \{R_{(a,b)} \in R \setminus P : a \text{ is even}\}$. We define $W$ to be the set of all cells of $R$ that are colored white and do not contain a peg, that is, $W = \{R_{(a,b)} \in R \setminus P : a \text{ is odd}\}$.

(iii) Construct a flow network $G = (V, E)$ as follows.
- (a) Let $s$ be a source vertex and $t$ be a sink vertex.
- (b) Each cell $R_{(a,b)}$ in $R$ is a vertex $(a, b)$ in $V$.
- (c) Add an edge from $s$ to each vertex $(a, b)$ whose corresponding cell is colored black, that is, $R_{(a,b)} \in B$.
- (d) Add an edge from each vertex $(a, b)$ whose corresponding cell is colored white to $t$.
- (e) For each black cell $R_{(a,b)} \in B$ and each cell with a peg $R_{(c,d)} \in P$, there is an edge $((a, b), (c, d))$ in $E$ if and only if $R_{(a,b)}$ and $R_{(c,d)}$ are adjacent in the region $R$; see for example Fig. 5.
- (f) For each cell with a peg $R_{(c,d)} \in P$ and for each white cell $R_{(a,b)} \in W$, there is and edge $((c, d), (a, b))$ in $E$ if and only if $R_{(c,d)}$ and $R_{(a,b)}$ are adjacent in the region $R$.
- (g) All edges in items (e) and (f) have a capacity of 1, all edges coming from $s$ and all edges going to $t$ have capacity 1, and all remaining missing edges have a capacity of 0.
- (h) Each vertex $(a, b) \in V(G) \setminus \{s, t\}$ has $c((a, b)) = 1$ and $c(s) = c(t) = \infty$.

In this work we refer to the graph $G$ obtained from the procedure above as the *region network* of $R$, see for example Fig. 6. It is easy to see that any region network has $n + 2$ vertices and at most $2n/3 + 4n/3$ edges, and can be constructed in $O(n)$ time. Also note that a region network is a dual graph with two new vertices, some edges removed, and direction given to the remaining edges.

**Proposition 3.1** *The value of a maximum flow in $G$ equals $|R|/3$ if and only there is a p-cover of $R$.*

Proposition 3.1 follows from lemmas 3.3 and 3.4 below. First, however, we show a technical lemma con-

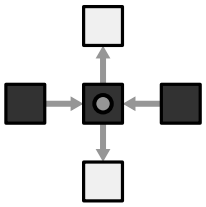

Fig. 5. Each cell with a peg has at most 2 incoming edges from black cells and at most 2 outgoing edges to white cells. The adjacent cells on the sides of a cell with a peg in the region $R$ are always of the same color, for example, a black cell with a peg always has black horizontal neighbors and white vertical neighbors.

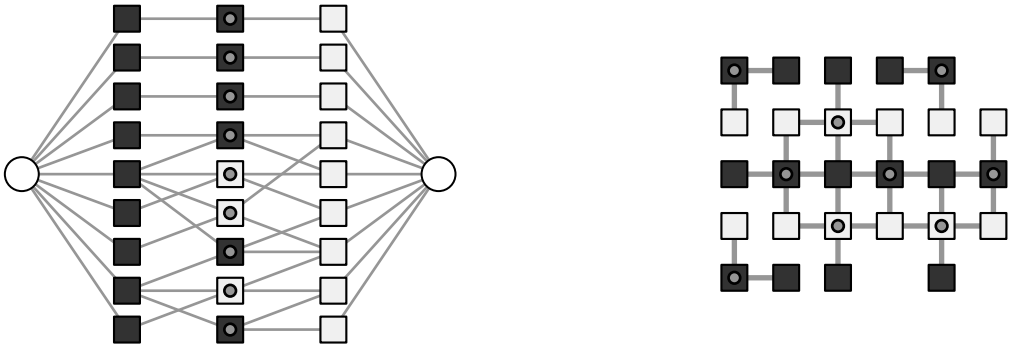

Fig. 6. Region network of the region on the right. Note that there are no edges (incoming or outgoing) between cells with no pegs. Similarly, there are no edges between cells with pegs.

cerning the uniqueness of st-paths in the region network.

**Lemma 3.2** *Let $f$ be a flow in $G$ with $|f| = n/3$. For any $R_{(a,b)}$ there exists an unique path $s \overset{(a,b)}{\rightsquigarrow} t$ with $f(s \overset{(a,b)}{\rightsquigarrow} t) = 1$.*

**Proof.** The flow passing through $(a, b)$ is upper bounded by 1, either because $c((a, b)) = 1$ if $R_{(a,b)} \in P$ or because the in-degree or out-degree is 1 if $R_{(a,b)} \in B$ or $R_{(a,b)} \in W$, respectively. Therefore, if there is flow passing through $(a, b)$ we have that there is an unique path $s \overset{(a,b)}{\rightsquigarrow} t$ with $f(s \overset{(a,b)}{\rightsquigarrow} t) = 1$. In order to finish the proof we need to show that $f(s \overset{(a,b)}{\rightsquigarrow} t) = 1$ for all $R_{(a,b)}$. Indeed, suppose otherwise that there exists $(a, b)$ such that for any path $s \overset{(a,b)}{\rightsquigarrow} t$ the flow through $(a, b)$ is 0. Then, if $R_{(a,b)} \in B$ (respectively $P$, $W$), the flow through the edge-cut $(\{s\}, \{s\}^c)$ (respectively $(\{s\} \cup B, P \cup W \cup \{t\})$, $(\{s\} \cup B \cup P, W \cup \{t\})$) will be of value $n/3 - 1$, which contradicts the fact that the flow has value $n/3$. □

**Lemma 3.3** *If the value of a maximum flow in $G$ equals $n/3$, then the region $R$ with $k = n/3$ pegs has a p-cover.*

**Proof.** Let $f$ be a flow of maximum value in $G$, with $|f| = n/3$. Let $R_{(a,b)}$ be a cell with a peg. By Lemma 3.2 we known that there is a unique path $s \overset{(a,b)}{\rightsquigarrow} t$ with a flow of 1 going through it. Moreover, from our construction of $G$, a path $s \overset{(a,b)}{\rightsquigarrow} t$ always has the form

$$s \to (c, d) \to (a, b) \to (c', d') \to t, \tag{1}$$

where $R_{(c,d)}$ is a black cell with no peg and $R_{(c',d')}$ is a white cell with no peg.

Now we can construct a p-cover of $R$ using the following "p-tromino placement rule": place the tromino $T = R_{(c,d)} \cup R_{(a,b)} \cup R_{(c',d')}$ on top of $R_{(a,b)}$. In Fig. 7 we show a correspondence between a flow in the region network and a tromino cover.

To finish the proof we need to show that our placement of the trominoes is a p-cover of $R$. This follows from Lemma 3.2 since for each cell $R_{(a,b)}$ the vertex $(a, b)$ appears in exactly one st-path. □

**Lemma 3.4** *If the region $R$ with $k = n/3$ pegs has a p-cover, then the value of a maximum flow in $G$ equals $n/3$.*

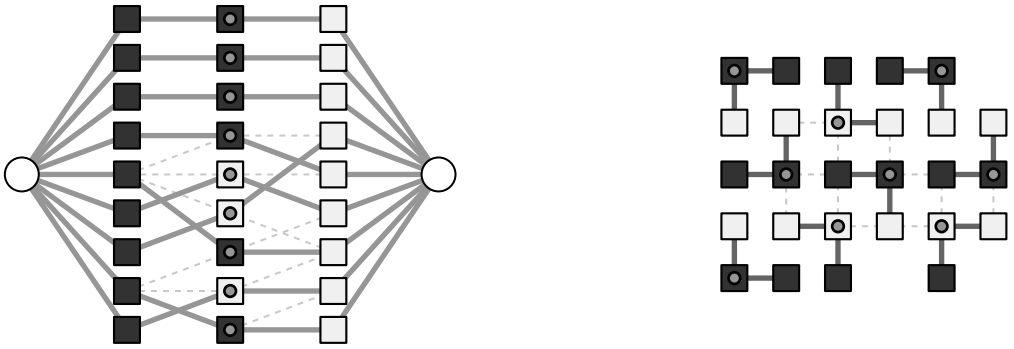

Fig. 7. Maximum flow in a region network on the left. Dashed lines are the edges of the graphs with no flow, and solid lines are the edges with flow. The right subfigure shows the region network over the region.

**Proof.** Let $C$ be a p-cover of size $n/3$. We construct a maximum flow $f$ whose value is $n/3$.

Take any tromino $T \in C$. With no loss of generality, suppose that $T =$  and that its corner cell $R_{(a,b)}$ is colored white. Hence, the upper cell $R_{(c,d)}$ is black and its left tip $R_{(c',d')}$ is white. Thus we have a path $s \overset{(a,b)}{\rightsquigarrow} t$ in the network graph $G$ of the form $s \to (c,d) \to (a,b) \to (c',d') \to t$. For all edges $e$ of this path we assign a flow $f(e) = 1$. We do this for all other trominos in $C$.

First we show that $f$ is a flow with value $|f| = n/3$. Since $C$ is a cover, there are no overlapping trominoes and all cells are covered. This is equivalent to say that each vertex of $G$ that is not the source $s$ or the sink $t$ appears in exactly one st-path $s \rightsquigarrow t$ with $f(s \rightsquigarrow t) = 1$. This implies that $f$ is a flow because each vertex has an incoming and outgoing flow of 1. Furthermore, since we have $n/3$ trominoes in $C$, there are $n/3$ such st-paths, and hence, $|f| = n/3$.

To finish the proof, we show that $|f| = n/3$ is maximum. This follows directly by considering the cut $(\{s\}, \{s\}^c)$ consisting on the $n/3$ edges incident with the source $s$, and any flow is bounded by the capacity of any cut. □

With lemmas 3.3 and 3.4 we finish our proof of Proposition 3.1. Now we show that our reduction is parsimonious.

**Lemma 3.5** *The reduction is parsimonious.*

**Proof.** Let $H$ be our reduction presented in this section and let us fix a region $R$ and a list of pegs $P$. Recall that $H$ is parsimonious if it preserves the number of yes-certificates. That is, the sets

$$\{C : (R,P) \in \text{P-Tromino}_{n,n/3} \text{ and } C \text{ is a p-cover of } (R,P)\}$$

and

$$\{f : (H(R,P), n/3) \in \text{MaxFlow}_{n/3},\ f \text{ is a flow in } H(R,P) \text{ with } |f| = n/3\}$$

are of the same size.

The existence of a bijective function between both sets above follows from the following argument. By our p-tromino placement rule of Lemma 3.3, to any unique st-path in $H(R,P)$ of the form of Eq.(1) with flow passing through it, there is exactly one p-tromino $T \in C$; a different st-path corresponds to a different p-tromino $T' \in C$. Likewise, for any $T \in C$ there is exactly one st-path of the form of Eq.(1) with a flow of 1 passing through the vertices corresponding to the cells in $T$. □

## 4 Proof of Theorem 1.2

Let us start by stating some trivial facts about matchings in graph theory. (i) The number of perfect matchings of a graph is the product of the number of perfect matchings of its components. (ii) The number of perfect matchings of a cycle is 0 or 2 whenever the cycle is odd or even, respectively. (iii) The number of perfect matchings of the union of $k$ cycles is $2^k$ if the cycles are all even and 0 if some of them are odd. (iv) If a graph has a *leaf* $v$, i.e, a node with an unique incoming edge $(v,w)$, then that edge should be in any perfect matching of the graph.

Let $G = (P \cup B \cup W \cup \{s,t\}, E)$ be the region network that is constructed with our reduction of Section 3. From $G$, let us consider the following two graphs $G' = (P \cup B \cup \{s,t'\}, E')$ and $G'' = (P \cup W \cup \{s',t\}, E'')$ given by $E' = E(G - W - \{t\}) \cup P \times \{t'\}$ and $E'' = E(G - B - \{s'\}) \cup \{s'\} \times P$, with distinguished sources

**Input:** graph $H$
**Output:** number of perfect matchings of $H$
1: $M \leftarrow \emptyset$; $H' \leftarrow H$;
2: **while** there is a leaf $v$ in $H'$ with unique incoming edge $(w, v)$ **do**
3: $M \leftarrow M \cup \{(v, w)\}$;
4: $H' \leftarrow H' - v - w$;
5: **end while**
6: Let $H_1, \ldots, H_k$ be the connected components of $H'$, where $k = 0$ if $V(H') = \emptyset$;
7: **if** there is $i$ such that $H_i$ has an odd cycle or an isolated vertex **then**
8: **return** 0
9: **else**
10: **return** $2^k$
11: **end if**

Algorithm 1: Counting perfect matchings in $H$.

and sinks $s, t'$ and $s', t$, respectively. Note that $G$ has a flow of value $n/3$ if and only if there exists a flow in $G'$ and a flow in $G''$ both with value $n/3$. Furthermore, the number of flows of value $n/3$ in $G$ equals the number of flows with value $n/3$ in $G'$ and $G''$ multiplied.

**Lemma 4.1** *The number of flows in $(G', s, t')$ of value $n/3$ is the same as the number of perfect matchings in $H = G' - s - t'$.*

**Proof.** It suffices to prove that the map that takes a flow $f$ and assigns a set $M$ of edges with flow equals to 1 is a bijection between the set of flows of value $n/3$ and perfect matchings in $H$. The map is clearly one to one, hence, we only need to check that $M$ is a perfect matching and that any perfect matching is the image of a flow $f$. In order to check the former we need to prove that there are no adjacent edges in $M$. Indeed, if for instance, $(x, y)$ and $(x', y)$ are two such edges, then the flow conservation constraint cannot be satisfied on vertex $y$ because the capacity of edge $(y, t')$ is 1. Therefore, the edges in $M$ form a matching in $H$, which is also perfect because it has value $|B|$. Finally, let $M$ be a given a perfect matching on $H$. If we define the flow $f$ to be 1 for the edges in the matching $M$ and for the edges incident with $s$ and $t'$, but 0 otherwise, then clearly $f$ is a flow on $(G', s, t')$ with image $M$. □

In Algorithm 1 we present a method for counting perfect matchings on the graph $H$ of Lemma 4.1. Let us prove the correctness of Algorithm 1.

The **while** loop of lines 2–5 has one key invariant, namely that *variable $M$ is always a matching of $H$ and $M$ should be contained in any possible perfect matching of $H$*. Another invariant is that *the number of vertices of $H'$ in $P$ is the same as the number of vertices of $H'$ in $B$*. The **while** loop always finishes because those edges incident to pending vertices must be in any perfect matching.

In lines 3–4 inside the **while** loop, Algorithm 1 erases those pending vertices together with its unique neighbor, since its incident edge should be on any perfect matching of $H'$. Thus, the new graph $H'$ obtained contains a perfect matching if and only if $H$ does.

Let us call $\hat{H}$ the graph $H'$ at the end of the loop. By the condition of line 2 of the **while** loop, the graph $\hat{H}$ has no leafs.

**Lemma 4.2** *$\hat{H}$ either has some isolated vertices or is the union of cycles.*

**Proof.** By the construction of graph $H$, a vertex $v$ of $H$ has degree $d_H(v) \leq 2$ if $v \in P$ and $d_H(v) \leq 4$ if it is in $B$. The same holds for vertices on $H'$ (and then on $\hat{H}$), since $H'$ is a subgraph of $H$.

Let $\hat{P}$ (resp. $\hat{B}$) be those vertices of $\hat{H}$ in $P$ (resp. in $B$), i.e., $\hat{P} = V(\hat{H}) \cap V$ and $\hat{B} = V(\hat{H}) \cap B$. Notice that by the second invariant $|\hat{P}| = |\hat{B}|$.

We need to prove that if $\hat{H}$ has no isolated vertices then it is the union of cycles, i.e., all its vertices have degree 2. Indeed, since all vertices have degree greater than 1, all vertices in $\hat{P}$ should have degree 2. Now

$$2|\hat{P}| = \sum_{v \in \hat{P}} 2 = \sum_{v \in \hat{P}} d_{\hat{H}}(v) = |E(\hat{H})| = \sum_{v \in \hat{B}} d_{\hat{H}}(v) \geq \sum_{v \in \hat{B}} 2 = 2|\hat{B}|.$$

However, $|\hat{P}| = |\hat{B}|$, and hence, the inequality should be an equality which happens only if $d_{\hat{H}}(v) = 2$ for all $v \in \hat{B}$. Therefore, the vertices of $\hat{H}$ have degree 2, and $\hat{H}$ should be the union of some number of cycles. □

From Lemma 4.2 it follows that the connected components of line 6 in Algorithm 1 are all cycles or have some isolated vertex. Therefore if either $\hat{H}$ has an isolated vertex or an odd cycle, then there will be no perfect matching; otherwise, $\hat{H}$ will have $k$ even cycles and, therefore, $2^k$ perfect matchings. This proves the correctness of Algorithm 1.

Algorithm 1 runs in linear time maintaining the degree of each vertex of $H'$ and a list of the leafs of $H'$. Since the degree is upper bounded by 4, computing the degrees takes at most $2 \times 4n/3$ time, and in that run it is possible to build a list of leafs. Given a list of leafs, the condition of the **while** loop uses constant time, while the deletion of vertices in line 5 also takes constant time because of the bound on the degrees. After line 5 is executed, the degrees are refreshed at those vertices incident with vertex $w$, and if one of them becomes a leaf, it will be added to the list of leafs. All theses operations require constant time. The **while** loop is executed at most $n/3$ times. The connected components of $\hat{H}$ are also computed in linear time. Finally, the number of flows in $G$ is obtained by multiplying the number of matchings in $H$ times the number of matchings in $H'' = G'' - s' - t$.

## 5 Concluding Remarks and Open Problems

In this work we showed how to characterize the tromino tiling problem with pegs using flow networks. We called this characterization the *region network* representation of a region with pegs. Then we showed that perfect matchings in bipartite subgraphs of a region network correspond to p-covers in a region with pegs. Thus, the number of perfect matchings in such bipartite graphs give us the number of p-covers and we showed that counting these perfect matchings can be done in linear-time. All these results present a tight connection between tromino tilings with pegs, maximum flows in flow networks and perfect bipartite matchings, thus giving us a new way to understand tromino tiling problems.

Below we give a couple of open problems that we believe are challenging and will deepen our understanding of tromino tilings in general.

(i) In Theorem 1.1 we showed that our reduction holds only when the number of pegs is $n/3$ and $n$ is a multiple of 3. Suppose now that beside p-trominoes, we also have normal (with no hole) trominoes at our disposal. A natural question to ask is what will happen if we have $n/3-k$ pegs and $k$ is given. Intuitively, if $k$ is "small," we can construct in linear-time a region network and run a polynomial-time algorithm for maximum flow to place p-trominoes on pegs. Then, for the squares that remain to be covered we use normal trominoes and, since $k$ is close to 0, we can find a tiling by brute force and thus in constant-time. On the other hand, if $k$ is "large," we have only a few pegs on the region and our brute force approach does not give us an efficient algorithm and in general should be NP-complete. What are the values of $k$ for which the P-Tromino tiling problem is NP-complete or admits a polynomial-time algorithm? We believe the P-Tromino tiling problem is fixed-parameter tractable with $k$ as a parameter.

(ii) The P-Tromino tiling problem, as defined in this paper, is given by a region with pegs and trominoes with holes. Now let us suppose that we interchange those roles and this time the trominoes have pegs in their corner cells and the region has holes. Suppose further that we can place a tromino on any hole on the region, even if a tip of the tromino is covering another hole. This version of the problem is similar to a p-tromino tiling but it is not the same, because now the tips of the trominoes can cover other holes on the region. Here we need to change our construction of a region network to allow edges between cells with pegs. In this new version of the problem we can also ask: how does the number of holes on the region affects the computational complexity of the tiling problem? If the number of holes on the region is close to $n$, then the tiling problem is NP-complete. Then, for which number of holes on the region does the problem admits a polynomial-time algorithm?

## References

[1] Javier T. Akagi, Carlos F. Gaona, Fabricio Mendoza, Manjil P. Saikia, and Marcos Villagra. Hard and easy instances of L-tromino tilings. *Theoretical Computer Science*, 815:197 – 212, 2020.

[2] J.H Conway and J.C Lagarias. Tiling with polyominoes and combinatorial group theory. *Journal of Combinatorial Theory, Series A*, 53(2):183 – 208, 1990.

[3] Erik D. Demaine and Martin L. Demaine. Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity. *Graphs and Combinatorics*, 23(S1):195–208, 2007.

[4] Noam Elkies, Greg Kuperberg, Michael Larsen, and James Propp. Alternating-sign matrices and domino tilings (part I). *Journal of Algebraic Combinatorics*, 1(2):111–132, 1992.

[5] Solomon W. Golomb. Tiling with polyominoes. *Journal of Combinatorial Theory*, 1(2):280–296, 1966.

[6] Takashi Horiyama, Takehiro Ito, Keita Nakatsuka, Akira Suzuki, and Ryuhei Uehara. Complexity of tiling a polygon with trominoes or bars. *Discrete & Computational Geometry*, 58(3):686–704, 2017.

[7] C. Moore and J.M. Robson. Hard tiling problems with simple tiles. *Discrete & Computational Geometry*, 26(4):573–590, 2001.