TRIANGLE-PARTITIONING EDGES OF PLANAR GRAPHS, TOROIDAL GRAPHS AND k-PLANAR GRAPHS

Jiawei Gao[1], Ton Kloks, and Sheung-Hung Poon[2]

[1] Software School, Fudan University, 220 Handan Rd., Shanghai, China
gaojw76@gmail.com
[2] Department of Computer Science & Institute of Information Systems and Applications
National Tsing Hua University, No. 101, Sec. 2, Kuang Fu Rd., Hsinchu, Taiwan
spoon@cs.nthu.edu.tw

Abstract. We show that there is a linear-time algorithm to partition the edges of a planar graph into triangles. We show that the problem is also polynomial for toroidal graphs but NP-complete for k-planar graphs, where $k \geqslant 8$.

1 Introduction

To partition the edges into a minimal number of cliques, and to cover them with a minimal number of cliques, are problems that receive a lot of attention lately. Both problems are NP-complete. Covering the edges with a minimal number of cliques remains NP-complete for planar graphs and for graphs with maximal degree at most six [6, 13]. The problem is polynomial for graphs with maximal degree at most five. Also for linegraphs the problem can be solved in polynomial time [20, 21]. Approximating the minimum number of cliques to cover the edges within a constant factor smaller than two remains NP-complete [15]. Approximations with polynomial factors are obtained eg in [2]. A result of Gyárfás implies that the problem of covering the edges with k cliques can be reduced to a kernel with $O(2^k)$ vertices [10, 11]. Recently, it was shown that, under assumption of the exponential time hypothesis, there is no polynomial algorithm which reduces the problem to a kernel of size $O(2^{o(k)})$ [7]. This contrasts the problem of partitioning the edges into cliques. A result of De Bruijn and Erdös implies that there are only two ways to partition the edges of a clique [5]. Actually, the only clique that has a nontrivial partition of the edges into triangles is K_7. Mujuni and Rosamond exploit this fact to derive a fixed-parameter algorithm which reduces the problem to partition the edges into cliques to a kernel of size $O(k^2)$. Fleischer and Wu obtain linear kernels for K_4-free graphs and for planar graphs [8]. Partitioning the edges of a graph into $|E|/3$ triangles remains NP-complete for K_4-free graphs [12, 22].

In this paper, we show that the problem can be solved in linear time for planar graphs. Considering that the triangle covering problem for planar graphs is NP-complete, this is a very surprising result. Moreover, we also show that the problem for toroidal graphs can be solved in polynomial time. However, we find

that the problem is NP-complete for k-planar graphs, which are super classes of planar graphs.

2 Partitioning planar graphs

We say that a graph has a partition if its edges can be partitioned into triangles. We assume that the planar graph G is given together with a plane embedding, *i.e.*, G is a plane graph. In this section, our goal is to show the following theorem.

Theorem 1. *There is a linear time algorithm to partition the edges of a planar graph into triangles.*

First, we need the following definitions.

Definition 1. *Let G be a plane graph. Let T be a triangle in G. The triangle divides the plane into two open regions. We refer to the two regions as the inside and outside of T. If both regions contains vertices of the graph $G - T$ then T is* separating*.*

Definition 2. *Let T be a separating triangle and let $v \in T$. The* inside degree *of v, denoted as $\tilde{d}(v)$, is the number of edges that contain v and of which the other endpoint is inside T. A separating triangle is* even *if the inside degrees of its three vertices are all even.*

Definition 3. *A separating triangle is* outermost *if it has no vertices inside any other separating triangle.*

2.1 The dual graph

Let H be the dual of plane graph G. First the following observation is obvious.

Lemma 1. *When G has a partition then every edge is contained in a triangle. Furthermore, if an edge of G is contained in only one triangle, say with edges e_1, e_2 and e_3, then G has a partition if and only if $G - \{e_1, e_2, e_3\}$ has a partition.*

Disconnnected components of a graph can be edge-partitioned separately. A connected graph can be divided into several biconnected components, each of which again can be handled separately. Thus we may assume from now on that G is biconnected. Furthermore, by Lemma 1, we may also assume that G has no vertices of degree two.

Lemma 2. *If G has a partition, then H is bipartite.*

Proof. Assume that G has a partition. Let C be a cycle in H. There is a one-to-one correspondence between the edges of G and H. The edges of C correspond with a cut set in G. Every triangle of the triangle partition has either all vertices on one side of the cut or it has two vertices on one side, and one vertex on the other side. Thus the cut has an even number of edges. This proves that every cycle of H is even, and so H is a bipartite multigraph. It is easy to check that H has no loops or multiple edges, since G has a partition and no vertices of degree two. □

2.2 Triangle partitioning algorithm

A graph with some odd-degree vertex does not have a triangle partition. By Lemma 2, we also see that a plane graph with non-bipartite dual does not have a triangle partition. Thus from now on, we assume that the given plane graph G satisfying the following conditions:

1. G is biconnected,
2. the dual H of G is bipartite,
3. every vertex in G has even degree at least four, and
4. every edge of G is in at least two triangles.

We consider two cases, namely, the graph G has separating triangles or not in the following subsections.

Graphs without separating triangles. First, we consider the case where G has no separating triangle. That is, every triangle is a face. Since every edge is in two faces, it follows that every edge is in exactly two facial triangles and that every face is a triangle.

Lemma 3. *Assume that G has no separating triangle. Then G has a partition if and only if every vertex of one color class H_1 of the dual H has degree three.*

Proof. Assume that all vertices in one color class H_1 of H have degree three. Since H is bipartite, H_1 forms a vertex cover of H. All dual faces of vertices in H_1 are triangles and so each vertex of H represents one triangle of G. Therefore, the color class H_1 of H forms a partition of the edges of G into triangles.

Assume that G has a partition. The faces of the triangles in the partition are vertices of H of degree three. Between any two of them, the distance is even, and so they form a color class of H. □

Graphs with separating triangles. Consider a partition $\mathcal{P}$ of the edges of G into triangles. We distinguish three types of separating triangles.

Definition 4. *A separating triangle $S = \{x, y, z\}$ is one of the following three types.*

Type 1: *Either $S \in \mathcal{P}$ or the three edges $\{x, y\}$, $\{x, z\}$ and $\{y, z\}$ are in triangles of $\mathcal{P}$ with the third vertex inside S.*

Type 2: *The three edges $\{x, y\}$, $\{x, z\}$ and $\{y, z\}$ are in triangles of $\mathcal{P}$ of which the third vertex is outside S.*

Type 3: *Some of the edges of $\{x, y\}$, $\{x, z\}$ and $\{y, z\}$ are in triangles of $\mathcal{P}$ with the third vertex inside S and some of them are in triangles with the third vertex outside S.*

The following lemma shows that a separating triangle of Type 3 cannot be a single triangle in any partition.

Lemma 4. *If a separating triangle S is even, then it is of Type 1 or 2 in any partition. If S is not even, it is of Type 3 in any partition.*

Proof. Let $\mathcal{P}$ be a partition and let $S = \{x, y, z\}$ be a separating triangle. Consider the graph G' induced by the vertices inside S, including S. First assume that $S \in \mathcal{P}$. Then S is even, otherwise there is no partition of the edges in G'.

Assume that $S \notin \mathcal{P}$. Assume that $\{x, y\}$ is in a triangle of $\mathcal{P}$ with the third vertex outside S. Assume that the other two edges of S are in triangles of $\mathcal{P}$ with the third vertex inside S. Thus S is of Type 3. Remove the edge $\{x, y\}$ from the graph G'. There is a partition of the edges of $G' - \{x, y\}$ which implies that the degree of x and y is even. Then S is not even in G, a contradiction.

The other cases are similar. This proves that S is even if and only if S is of Type 1 or Type 2 with respect to $\mathcal{P}$. □

We suppose that all even separating triangles have been identified. (In later subsection 2.3, we in fact design a linear-time algorithm to compute them.) Our main algorithm traverses G starting from its outer boundary, and search for all outermost even separating triangles. Our search stops at those outermost even separating triangles when they are reached. Thus the interior of any outermost even separating triangle is considered as being removed since our algorithm does not go into it at the current step. The interior subgraph of each outermost, even, separating, triangle will be dealt with in a later recursive step.

Removing the interiors of even, separating triangles, turns these outmost even separating triangles into triangular faces. Let's denote this new graph as G'. Then G' has no more even separating triangles. We call a face of G', which corresponds to an outermost even separating triangle in G, a *region*.

Remark 1. Lemma 4 generalizes to the regions and faces of G'. Any even region or face is one two types.

Type 1: The region or face is a triangle in the partition $\mathcal{P}$ of G'.
Type 2: All the edges of the boundary are in triangles of $\mathcal{P}$ with the third vertex outside the region.

Notice that even, separating triangles in G that are of Type 1 correspond to regions of G' that are Type 1. Similarly, even, separating triangles in G of Type 2 correspond to regions in G' of Type 2. Of course, faces of G' that are not triangles are automatically Type 2. In the next lemma, we show that the two color classes of the dual of G' correspond with the two types.

Lemma 5. *Assume that G' has a partition and let H' be its dual. Let H_1 and H_2 be the two color classes of H'. Then all the vertices of H_1 are of one of the two Types 1 or 2 and all the vertices of H_2 are of the opposite type.*

Proof. Notice that the vertices of H' along any path alternate between the two types and, since H' is connected, this proves the theorem. □

We use Lemma 5 to partition G', and we can see that there are at most two ways to partition G'. Then when we proceed to process a recursive step for a subgraph inside an outmost even separating triangle S of G, if triangle S is

labeled Type 1 in G', then the related subgraph inside S to be processed in this recursive step includes S; if triangle S is labeled Type 2, the interior subgraph of S to be processed does not include S. Since all recursive steps are processed on separate subgraphs of G, it is clear that the whole recursive procedure runs in linear time. Thus the last remaining task for us is to find all even separating triangles of G in linear time, which will be done in next subsection.

2.3 Finding even separating triangles

Definition 5. *Let* $G = (V, E)$ *be a plane graph. A level decomposition partitions the vertices into levels* $L_1, L_2, \ldots$ *defined as follows.*

(a) L_1 *is the outerface of* G *and,*
(b) *for* $i > 1$, L_i *is the outerface of*

$$G - \bigcup_{j=1}^{i-1} L_j.$$

Given a plane graph G, a level decomposition can be obtained in linear time [17] (see also [3, 18]). Notice that any consecutive sequence of k levels induces a k-outerplanar graph. It is well-known that k-outerplanar graphs have treewidth at most $3k + 1$ (see eg [4]) and therefore they have $O(k^3 n)$ triangles. Each level induces an outerplanar graph. A graph is outerplanar if and only if each biconnected component is a tree of cycles, that is, neighboring cycles in the tree have one edge in common and there are at most two cycles incident with each edge (see eg [14]). Next, we show how to find all even separating triangles of G in linear time.

Lemma 6. *All even separating triangles of a plane graph* G *can be found in linear time.*

Proof. By Lemmas 3 and 5 the bottleneck in the computation is the finding of the even separating triangles of G. We describe below how all even separating triangles can be determined in linear time.

Consider a level decomposition. The outerface L_1 is outerplanar. Each biconnected component of L_1 induces a tree of cycles. Assume there is a cycle in this tree of cycles which is a triangle T. Assume that $L_1 \neq T$ and that the inside of T is nonempty. Then T is a separating triangle. Using the clockwise orientation of each neighborhood we can determine if it is even.

Consider a cycle C of L_1. Assume that the inside of C is nonempty. Then it contains a component of L_2. First create a list of triangles that have at least one vertex of C and at least one vertex of the part of L_2 which is inside C. Since this graph has treewidth at most 7 we can make a list of these triangles in linear time. Check which triangles are even and have a nonempty interior using the clockwise orientation of the neighborhoods.

When all even, separating triangles are determined that contain at least one vertex of L_1, then the vertices of L_1 are deleted, and the algorithm continues with the remaining graph in a similar manner as described above.

Using some suitable data structures this algorithm can be implemented to run in linear time. This proves the lemma. □

With the triangle partition algorithm in Section 2.2 and Lemma 6, we thus have a linear time algorithm to partition the edges of plane graph G into triangles. This completes the proof of Theorem 1.

3 Partitioning toroidal graphs

A graph is *toroidal* if it can be embedded on the torus. Toroidal graphs [23] generalize planar graphs in many ways dramatically. For example, cliques with up to seven vertices are toroidal. By the graph minor theorem toroidal graphs are characterized by a finite collection of forbidden minors or topological obstructions. By Kuratowski's or Wagner's theorem, for planar graphs this obstruction set has only two elements. For toroidal graphs these obstructions are still not completely known. One has identified 16,629 forbidden minors and 239,322 forbidden topological obstruction [9]. For some subclasses the full set is known, see eg [1, 9].

It is convenient to consider drawings of toroidal maps using a rectangular or a square piece of paper. Opposite edges of the paper are point-by-point identified (in the same direction); an edge of the graph which runs out on the right edge of the square, comes back in on the left edge of the square, and similarly edges wrap around on the top- and bottom-edge of the square. As an example one may have a look at the embedding of K_7 on a torus, ie, a representation of Császár's, or Szilassi's polyhedron. Let G be a toroidal embedding of a graph. We distinguish the following types of cycles in G.

Contractible cycles These are the boundaries of areas homeomorphic to open discs, or faces.

Noncontractible and nonseparating cycles Consider the drawing of the graph on a square piece of paper. These cycles consist of a path connecting the top- and bottom-edge (with identified edges), or the left- and right-edge (with identified edges). The removal of these cycles reduces the graph to a planar graph, drawn on a cylinder.

Noncontractable, separating cycles These are the cycles whose removal separate the graph into an inside component and an outside component, just as in the planar case.

Lemma 7. *Let* G *be a toroidal embedding of a graph. Assume that* G *has a partition* $\mathcal{P}$*. Assume that all triangles are contractible. Then the dual is bipartite. Furthermore, the triangles of* $\mathcal{P}$ *consist of one color class of the dual.*

Proof. All triangles of G are faces. Since every edge is in exactly one triangle, any path in the dual alternates between faces that are in $\mathcal{P}$ and faces that are not in $\mathcal{P}$. Thus all cycles of the dual are even. Furthermore, every path between two faces of $\mathcal{P}$ has even length, so $\mathcal{P}$ consists of exactly the faces of one color class of the dual. □

Consider representation of G on a rectangular planar region. Consider a left to right ordering of the nonseparating triangles that wrap around the top- and bottom-end of the region. Let T_1 and T_2 be two triangles with T_1 left of T_2 in this order. Possibly T_1 and T_2 have some vertices in common, but we assume that $T_1 \neq T_2$. The *piece* $G(T_1, T_2)$ consists of the vertices and edges that are in the region between T_1 and T_2.

Definition 6. *Consider a piece* $G(T_1, T_2)$. *A* bridge *is either an edge or a path of length two between two vertices, on in* $T_1 \setminus T_2$ *and the other in* $T_2 \setminus T_1$*, which wraps around the right- and left edge of the plane region.*

Consider two vertices $x \in T_1 \setminus T_2$ and $y \in T_2 \setminus T_1$. Assume that x and y are adjacent such that the edge $\{x, y\}$ is embedded in $G(T_1, T_2)$. A bridge between x and y of length two, together with the edge $\{x, y\}$ creates a nonseparating triangle. Similarly, a path of length two from x to y embedded in the piece $G(T_1, T_2)$ together with a bridge which is an edge, is a nonseparating triangle.

Theorem 2. *There exists a polynomial-time algorithm which checks if the edges of a toroidal graph can be partitioned into triangles.*

Proof (Sketch). Separating triangles are treated in exactly the same manner as in the planar case or, alternatively, via dynamic programming using Tarjan's decomposition tree [24]. Tarjan describes an $O(nm)$ algorithm to find a binary decomposition tree which decomposes a graph using clique separators. Using dynamic programming on this decomposition tree we can obtain, for each even separating triangle T, a table with boolean entries which tells us whether the graph G_T induced by the triangle and the inside has a partition $\mathcal{P}$ with

(a) $T \in \mathcal{P}$, and
(b) all the edges of T are in triangles of $\mathcal{P}$ with some vertex outside T.

The algorithm determines the feasible partitions of pieces $G(T_1, T_2)$ by dynamic programming. For each T_i it has a boolean value which indicates if there is a partition with T_i as a triangle, and for each edge in T_i whether there is a partition with the edge in a triangle with a third vertex inside the piece or outside the piece.

Consider a piece $G(T_1, T_2)$. The table also needs to keep track of the triangles in partitions that use some vertex of T_1 and some vertex of T_2 are a bridge. For any two vertices $x \in T_1 \setminus T_2$ and $y \in T_2 \setminus T_1$, for which there is a triangle which uses a bridge, there are at most n such triangles. Furthermore, at most one of them can be an element of a partition. The algorithm builds a table which lists all partitions of the edges of the piece into triangles. For each pair of vertices x

and y an entry of the table contains the information whether a triangle is used in the partition that uses a bridge of length one or two from x to y. Triangles that use bridges cut the piece into parts.

The pieces are processed as follows. The triangles that use a bridge cut the piece into smaller strips. Each strip is bounded on the top and bottom by paths of length one or two. The table for the piece is computed using dynamic programming on the strips. For each edge in the border of the strip, the information is kept whether the border is a triangle in the partition, or which edges are in triangles with the third vertex inside or outside the piece.

By dynamic programming the algorithm computes a table for all pieces in order of increasing size. To write down the dynamic programming algorithm is a standard technique. We omit the cumbersome write-up of all details. □

4 NP-completeness for k-planar graphs

A graph is k-planar if it has an embedding in the plane such that every edge crosses at most k other edges. Note that 0-planar graphs are simply planar graphs. In this section, we show that the partition problem for k-planar graphs is NP-complete for all $k \geqslant 8$.

Theorem 3. *The triangle edge partition problem for* k*-planar graph is NP-complete, where* $k = 8$.

First we show that the problem is in NP. Suppose we are given a triangle partition of the edges of the given graph G. We can easily verify that whether the given partition is a triangle partition of the edges of G.

Next we show that the triangle edge-partition problem is NP-hard for k-planar graphs. We reduce the planar one-in-three 3SAT problem [16] to this problem. We reduce from the 3SAT problem. The input instance for the planar one-in-three 3SAT problem is a set $\{x_1, x_2, \ldots, x_n\}$ of n variables, and a Boolean formula $F = c_1 \wedge c_2 \wedge \ldots \wedge c_m$ of m clauses, where each clause consists of exactly three literals, such that the variable clause graph of the input instance is planar. The planar one-in-three 3SAT problem asks for whether there exists a truth assignment to the variables so that each clause in given formula F has exactly one true literal and exactly two false literals. In the following, we will describe the construction of variable gadgets, literal gadgets, and clause gadgets, respectively, for our polynomial-time reduction. In the construction, we repeatedly use the following construction unit, called an ω-tube. An ω-*tube* of length ℓ and of width ω is a graph consisting of an integer grid of vertices $\{(x, y)\}$ with $0 \leqslant x < \ell$ and $0 \leqslant y < \omega$ for some positive integers ℓ and ω. The edge set of the ω-tube is formed by performing the following steps: (Note that the plus operations relating to y indices here are all modulo ω.)

1. Connect an edge between (x, y) and $(x, y+1)$ for $0 \leqslant x < \ell$ and $0 \leqslant y < \omega$;
2. Connect an edge between (x, y) and $(x+1, y)$ for $0 \leqslant x < \ell - 1$ and $0 \leqslant y < \omega$; and

3. Connect an edge between (x, y) and $(x + 1, y + 1)$ for $0 \leqslant x < \ell - 1$ and $0 \leqslant y < \omega$.

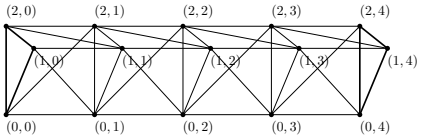

Fig. 1. A variable gadget, which is a 3-tube.

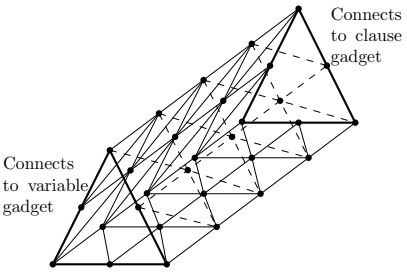

Fig. 2. A literal gadget, which is a 6-tube.

See Figure 1 for an example of a 3-tube. The polygons $\{(0, 0), (0, 1), \ldots, (0, \omega - 1)\}$ and $\{(\ell - 1, 0), (\ell - 1, 1), \ldots, (\ell - 1, \omega - 1)\}$ at the both ends of the ω-tube are called the *end polygons* of the tube.

4.1 Variable gadget

A variable gadget is a 3-tube of length N. See Figure 1 as an example. First we show in the following lemma that a variable gadget has exactly two partitions.

Lemma 8. *A variable gadget has exactly two partitions.*

Proof. A 3-tube has only 3 types of triangles:

1. α*-triangle:* $\{(x, y), (x, y + 1), (x + 1, y + 1)\}$,
2. β*-triangle:* $\{(x, y), (x + 1, y), (x + 1, y + 1)\}$, and
3. γ*-triangle:* $\{(x, 0), (x, 1), (x, 2)\}$.

A triangle partition of a variable gadget must contain at least one α-triangle, or one β-triangle, because γ-triangles do not contain any $((x, y), (x + 1, y + 1))$ edge. If a partition contains an α-triangle, then the partition nearly only contains α-triangles, except for one end γ-triangle $\{(0, 0), (0, 1), (0, 2)\}$. If a partition contains a β-triangle, then the partition nearly only contains β-triangles, except for one end γ-triangle $\{(N - 1, 0), (N - 1, 1), (N - 1, 2)\}$. □

The former partition corresponds to the true value for the variable, and thus is called the *true partition*; the latter partition corresponds to the false value for the variable, and thus is called the *false partition*.

Later on, if variable v appears as a literal v in clause c, we plan to delete an α-triangle in the variable gadget of v and a hexagonal hole is thus formed. Such a hole will identify with an end-hexagon of a literal gadget later on. If variable v appears as a literal $\bar{v}$ in clause c, we plan to delete a β-triangle to create a hexagonal hole to connecting a literal gadget later on.

4.2 Literal gadget

A literal gadget is a subgraph that connects a variable gadget to a clause gadget. We form a literal gadget as a 6-tube of length M. See Figure 2 for an example. It serves the function of propagating a partition from one of its ends to the other.

Of the two end-hexagons of the literal gadget, one end merges with a hexagonal hole of a variable gadget as mentioned previously, the other will connect to a clause gadget later on. In one partition of the literal gadget, it contains all α-triangles of the literal gadget; however, in such a so-called partition, the edges of the clause-end hexagon has not been included in any triangle of this partition. This corresponds to the false value of this literal, and thus such a partition is called the *false partition* of the literal gadget. In the other partition of the literal gadget, it contains all β-triangles of the literal gadget; however, in such a so-call partition, the edges of the variable-end hexagon has not been included in any triangle of this partition. This corresponds to the true value of this literal, and thus such a partition is called the *true partition* of the literal gadget.

4.3 Clause gadget

A clause gadget for a clause c is formed by simply identifying the clause-end hexagons of the three corresponding literal gadgets. See Figure 4.3 for an example. Because the clause end-hexagons of the three literal gadgets of clause gadget

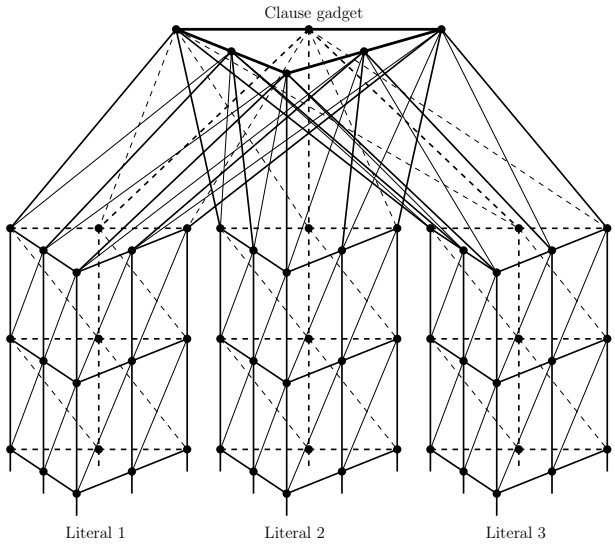

Fig. 3. The structure of a clause gadget.

are identified as one hexagon H, the edges of H lies in the triangle partition of

exactly one literal gadget among the three literal gadgets for clause c, but not in partitions of the other two literal gadgets. This means that one literal gadget has the true partition and the other two have false partitions. That is to say, exactly one of the three literals is true and the other two literals are false. Moreover, if variable v has the true value, the variable gadget is partitioned as the true partition. For clauses with literal v, the literal gadget is partitioned as the true partition, and the hexagon of the clause gadget is partitioned in the partition of the literal gadget of v whereas the variable-end hexagon of this literal gadget is partitioned in the partition of the variable gadget of v. For clauses with literal $\bar{v}$, the literal gadget is partitioned as the false partition, and the clause gadget hexagon is not partitioned by current literal gadget whereas the variable-end hexagon of this literal gadget is partitioned in the partition of this literal gadget. If variable v have the false value, the variable gadget is partitioned as the false partition. For clauses with literal $\bar{v}$, the literal gadget is partitioned as the true partition and the hexagon of the clause gadget is partitioned in the partition of the literal gadget of $\bar{v}$ whereas the variable-end hexagon of this literal gadget is partitioned in the partition of the variable gadget of v. For clauses with v, the literal gadget is partitioned as the false partition, and the clause gadget hexagon is not partitioned by current literal gadget whereas the variable-end hexagon of this literal gadget is partitioned in the partition of this literal gadget. Hence, the whole constructed graph has a triangle partition if and only if there is a truth assignment for all variables so that for any clause in formula F, exactly one literal is true and the other two literals are false. Therefore, the planar 1-in-3 3SAT problem has a solution if and only if the constructed graph has a triangle partition. This completes our reduction proof. However, we still need to obtain the constant k for the k-planarity of the constructed graph, which is the following lemma. Its proof is in the appendix.

Lemma 9. *The graph constructed above is 8-planar.*

5 Concluding remarks

In our result, we show that the triangle partition problem is linear-time solvable for planar graphs, but NP-complete for 8-planar graphs. We leave open the question whether the triangle partition problem for k-planar graphs, where $1 \leqslant k \leqslant 7$, is polynomial-time solvable or NP-complete.

We do not know about many complexity results on clique partitions in toroidal graphs, let alone for graphs with arbitrary (fixed) genus. Our results indicate that the partitioning problem can be solved in polynomial time for graphs of fixed genus. One of the questions that remain open is whether the same holds for partitions into cliques of arbitrary, bounded size.

References

1. Archdeacon, D., C. Bonnington, N. Dean, N. Hartsfield and K. Scott, Obstruction sets for outer-cylindrical graphs, *Journal of Graph Theory* **38**, pp. 42–64, 2001.

2. Ausiello, G., P. Creszensi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela and M. Protasi, *Complexity and Approximation: Combinatorial optimization problems and their approximability properties*, Springer, 1999.
3. baker, B., Approximation algorithms for NP-complete problems, *Journal of the ACM* **41**, pp. 153–180, 1994.
4. Bodlaender, H., A partial k-arboretum of graphs of bounded treewidth, *Theoretical Computer Science* **209**, Elsevier, pp. 1–45, 1998.
5. de Bruijn, N. and P. Erdös, On a combinatorial problem, *Indagationes Mathematicae* **10**, pp. 421–423, 1948.
6. Chang, M. and H. Müller, On the tree-degree of graphs, *proceedings WG* Springer LNCS 2204, pp. 44–54, 2001.
7. Cygan, M. M. Pilipczuk and M. Pilipczuk, Known algorithms for edge clique cover are probably optimal. Manuscript ArXiV: 1203.1754v1, 2012.
8. Fleischer, R. and X. Wu, Edge clique partition of K_4-free and planar graphs, *Proceedings CGGA*, Springer LNCS 7033, pp. 84–95, 2011.
9. Gagarin, A., W. Myrvold and J. Chambers, The obstructions for toroidal graphs with no $K_{3,3}$'s, *Discrete Mathematics* **309**, Elsevier, pp. 513–520, 2009.
10. Gramm, J., J. Guo, F. Hüffner and R. Niedermeier, Data reduction, exact, and heuristic algorithms for clique cover, *Proceedings ALENEX*, SIAM, pp. 86–94, 2006.
11. Gyárfás, A., A simple lowerbound on edge clique covering by cliques, *Discrete Mathematics* **85**, pp. 103–104, 1990.
12. Holyer, I., The NP-completeness of some edge-partition problems, *SIAM J. Comput*, **10**, pp. 713–717, 1981.
13. Hoover, D., Complexity of graph covering problems for graphs of low degree, *JCMCC* **11**, pp. 187–208, 1992.
14. Kloks, T., *Treewidth - Computations and Approximations*, Springer, Lecture Notes in Computer Science 842, 1994.
15. Kou, L., L. Stockmeyer and C. Wong, Covering edges by cliques with regard to keyword conflicts and intersection graphs, *Comm. ACM* **21**, pp. 135–139, 1978.
16. Laroche, P., Planar 1-in-3 satisfiability is NP-complete, *Comptes rendus de l'Acade'mie des sciences*, Se'rie 1, Mathe'matique, pp. vol. 316, no4, pp. 389-392, 1993.
17. Lipton, R. and R. Tarjan, A separator theorem for planar graphs, *SIAM Journal on Applied Mathematics* **36**, SIAM, pp. 177–189, 1979.
18. Lipton, R. and R. Tarjan, Applications of a planar separator theorem, *SIAM Journal on Computing* **9**, SIAM, pp. 615–627, 1980.
19. Mujuni, E. and F. Rosamond, Parameterized complexity of the clique partition problem, *Proceedings CATS*, Australian Computer Society, **77**, pp. 75–78, 2008.
20. Orlin, J., Contentment in graph theory: covering graphs with cliques, *Proceedings of the Nederlandse Academie van Wetenschappen, Amsterdam, Series A* **80**, pp. 406–424, 1977.
21. Pullman, N., Clique covering of graphs IV. Algorithms, *SIAM Journal on Computing* **13**, pp. 57–75, 1984.
22. Shaohan, M., W. Wallis and W. Lin, The complexity of the clique partition number problem, *Nineteenth Southeastern Conference on Combinatorics, Graph Theory and Computing*, Congr. Numer. **67**, pp. 59–66, 1988.
23. Surhone, L., M. Tennoe (ed.) and S. Henssonow (ed.), *Toroidal graph*, Betascript Publishing, 2010.
24. Tarjan, R., Decomposition by clique separators, *Discrete Mathematics* **55** pp. 221–232, 1985.
25. Valiant, L., The complexity of computing the permanent, *Theoretical Computer Science* **8**, pp. 189–201, 1979.

Appendix

A. Proof of Lemma 9.

Proof. Along either a variable gadget or a clause gadget, the intra-gadget edges may intersect. Apart from these intersections, intersections between different gadgets may occur too. Hence, by following the planar structure of variable clause graph of the planar one-in-three 3SAT instance, we make the clause gadgets and the conjunction positions between variable gadgets and literal gadgets reasonably far apart so that the intersection situation for one of such positions does not interfere other positions.

First we consider the intersection situation around the clause gadget for a clause c. We can draw the neighborhood of a clause gadget so that the intersection structure around this clause gadget only involves the edges incident to the vertices of the hexagon of the clause gadget. This intersection structure is shown in Figure 4. In the figure, we can see that the number of intersections for each

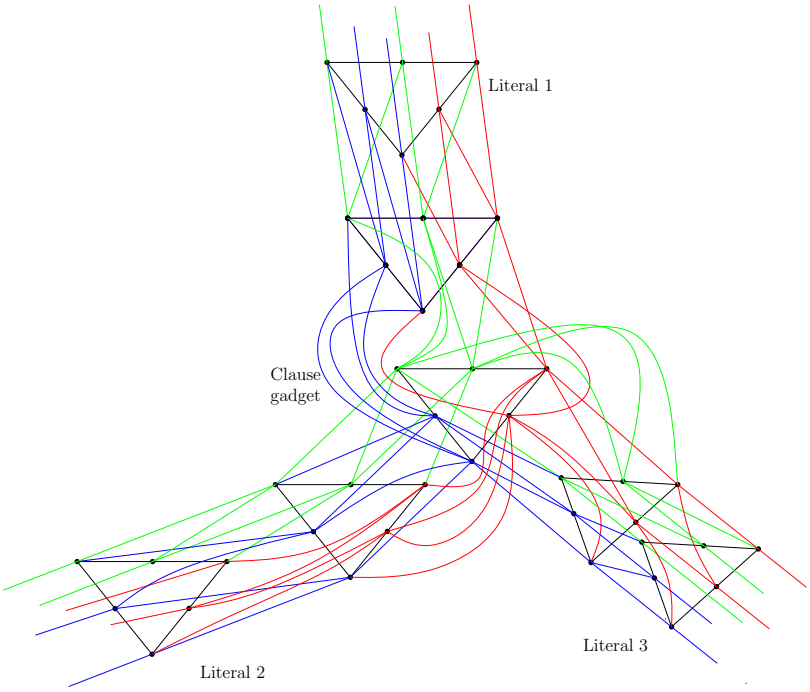

Fig. 4. The intersection structure around a clause gadget.

edge in this structure is at most 8.

Next we consider the intersection situation around the conjunction position between a variable gadget and a literal gadget. Again we try to draw the neigh-

borhood of such a connecting position so that the intersection structure around the connecting hexagon only involves the edges incident to the vertices of the connection hexagon between the variable gadget and the clause gadget. This intersection structure is shown in Figure 5. In the figure, we can see that the

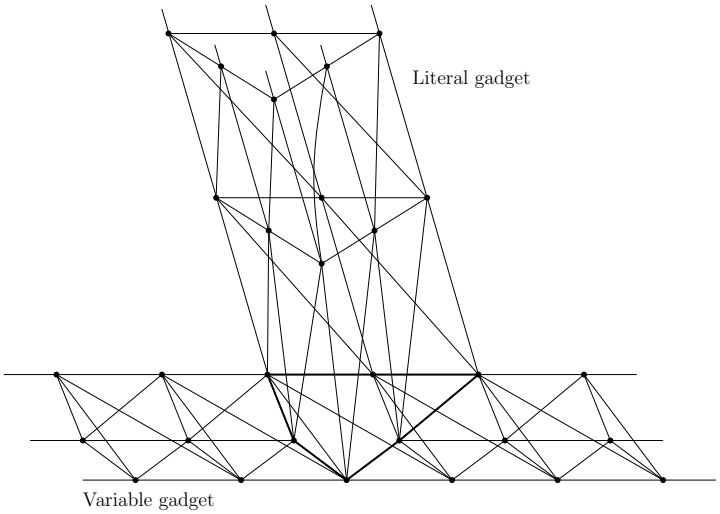

Fig. 5. The intersection structure around the conjunction hexagon of a variable gadget and a literal gadget.

number of intersections for each edge in this structure is at most 5.

Apart from these two intersection structures, we also need to consider the intra-gadget intersection structures inside a variable or literal gadget. Along a variable gadget, at those positions where no connection to literal gadgets occur, we construct the intersection structure as shown in Figure 6. In the figure, we

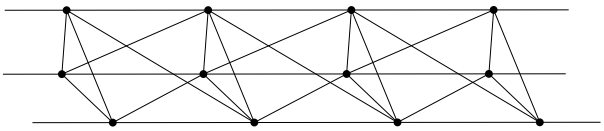

Fig. 6. The intra-gadget intersection structure along a variable gadget.

can see that the number of intersections for each edge in this structure is at most 3.

Along a literal gadget, at those positions where no connection to variable gadgets or clause gadgets occur, we construct the intersection structure in the literal gadget as shown in Figure 7. In the figure, we can see that the number of

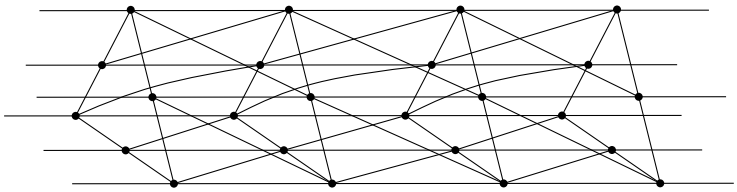

Fig. 7. The intra-gadget intersection structure along a literal gadget.

intersections for each edge in this structure is at most 5.

All in all, by combining the above local intersection structures altogether, we can obtain a drawing for the constructed graph such that each edge can contains at most 8 intersections. Hence, the constructed graph in our reduction is 8-planar. □