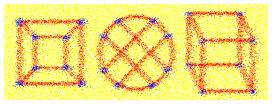
Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 22, no. 1, pp. 23–49 (2018)
DOI: 10.7155/jgaa.00457

Parameterized Complexity of 1-Planarity

Michael J. Bannister[1] *Sergio Cabello*[2] *David Eppstein*[3]

[1]Dept. of Mathematics and Computer Science, Santa Clara University, USA
[2]Faculty of Mathematics and Physics, University of Ljubljana, Slovenia
[3]Dept. of Computer Science, University of California, Irvine, USA

Abstract

We consider the problem of drawing graphs with at most one crossing per edge. These drawings, and the graphs that can be drawn in this way, are called 1-planar. Finding 1-planar drawings is known to be NP-hard, but we prove that it is fixed-parameter tractable with respect to the vertex cover number, tree-depth, and cyclomatic number. Special cases of these algorithms provide polynomial-time recognition algorithms for 1-planar split graphs and 1-planar cographs. However, recognizing 1-planar graphs remains NP-complete for graphs of bounded bandwidth, pathwidth, or treewidth.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
May 2017	September 2017	October 2017	October 2017	October 2017

Published:
January 2017

Article type:
Regular paper

Communicated by:
M. Bekos, M. Kaufmann, F. Montecchiani

The research of Bannister and Eppstein was supported in part by the National Science Foundation under grants 0830403, 1217322, 1618301, and 1616248, and by the Office of Naval Research under MURI grant N00014-08-1-1015. The research of Cabello was supported in part by the Slovenian Research Agency, program P1-0297, projects J1-4106 and J1-8130, and within the EUROCORES Programme EUROGIGA (project GReGAS) of the European Science Foundation. We also gratefully acknowledge the Slovenian Research Agency for travel funds allowing the authors to meet and perform this research. A preliminary version of this paper was presented at the 13th International Symposium on Algorithms and Data Structures (WADS 2013) and appears in *Lecture Notes in Computer Science* 8037, pp. 97–108.

E-mail addresses: mbannister@fastmail.fm (Michael J. Bannister) sergio.cabello@fmf.uni-lj.si (Sergio Cabello) eppstein@uci.edu (David Eppstein)

1 Introduction

1-planar graphs are the graphs that can be drawn in the plane with at most one crossing per edge. They were introduced by Ringel in 1965 [30] and have since been extensively studied from the point of view of basic properties such as their colorings [5, 9], edge density [3, 6, 29, 31], characterization by forbidden subgraphs [25, 26], and embeddings on nonplanar surfaces [4, 33]. In graph drawing, 1-planarity has more recently become of interest as a way of generalizing planar drawings in a controlled way that does not lead to too much visual complexity. Works in this area have compared 1-planarity to other forms of controlled crossings such as RAC (right-angle-crossing) graphs [16], found an algorithmic characterization of the 1-planar drawings that can be straightened to have all edges represented by straight line segments [23], and studied the transformation of rotation systems into 1-planar drawings [2, 15]. However, until now there have been no published algorithms for finding 1-planar drawings of arbitrary graphs. Unfortunately, testing 1-planarity is NP-hard in general [20,26], even for graphs obtained from planar graphs by adding a single edge [7], so we cannot expect it to be solved by an algorithm whose running time is a polynomial of the input size.

Because of the difficulty of recognizing 1-planar drawings and their usefulness in graph drawing, it becomes of interest to study the complexity of algorithms for testing 1-planarity that are not fully polynomial. An important tool for this sort of study is *parameterized complexity* [13, 18], according to which we consider additional numeric parameters (other than the numbers of edges and vertices) that measure the complexity of an input graph, and seek algorithms whose running time is the product of a polynomial in the input size and a non-polynomial function of the other parameter or parameters. If this can be accomplished, the result will in general be an algorithm that solves the problem correctly on all graphs, that can be relied on to be efficient for graphs that have small values of the parameter, and that has a performance that degrades gracefully as the parameter increases.

In this paper we study for the first time the parameterized complexity of 1-planarity. We provide the following results:

- We show that testing 1-planarity and finding 1-planar drawings are fixed-parameter tractable when they are parameterized by the *vertex cover number*. Our algorithm uses a polynomial kernel, a polynomial-time transformation of any instance to an equivalent instance with size polynomial in the parameter.

- We use our vertex cover number parameterization to provide a polynomial time recognition algorithm for 1-planar *split graphs*. A split graph is a graph whose vertices can be partitioned into a clique and an independent set [19, 34]. We prove that the 1-planar split graphs have bounded vertex cover number, allowing our algorithm to run in polynomial time instead of fixed-parameter-tractable time for this case.

- We show testing 1-planarity and finding 1-planar drawings are fixed-parameter tractable when they are parameterized by the *tree-depth*. For this problem we use a kernel of non-polynomial size.
- We use our tree-depth parameterization to provide a polynomial time recognition algorithm for 1-planar *cographs*. A cograph is a graph with no four-vertex path as an induced subgraph [10]. We prove that the 1-planar cographs have bounded tree-depth, allowing our algorithm to run in polynomial time instead of fixed-parameter-tractable time for this case.
- We design a fixed-parameter tractable algorithm for 1-planarity when it is parameterized by the *cyclomatic number* (the minimum number of edges that must be removed from the graph to make it into a forest). Again, our algorithm is based on a kernelization for the problem.
- We prove that the problem of testing 1-planarity remains NP-complete for graphs of bounded bandwidth. Therefore, it is unlikely that there exists a fixed-parameter tractable algorithm for 1-planarity when parameterized by bandwidth, pathwidth, treewidth, or clique-width.

Although our primary motivation is in understanding the complexity of 1-planarity, our research on the vertex cover and tree-depth parameters has a secondary purpose as well. For certain graph parameters, general theorems are known that guarantee the existence of an inexplicit fixed-parameter tractable algorithm (with unknown dependence on the parameter). We would like to explore the circumstances in which these algorithms can be made explicit. In particular, the graphs of bounded vertex cover number and the graphs of bounded tree-depth are well-quasi-ordered under induced subgraphs [28]. This means that for any graph recognition problem closed under induced subgraphs (as 1-planarity is) and for any fixed bound on vertex cover or tree-depth, there is a finite set of forbidden induced subgraphs that can be used to characterize the problem. By searching for these forbidden subgraphs, we may obtain a linear time recognition algorithm. However, the theorems that prove these results do not imply any computable bound on the size of these forbidden subgraphs or on the dependence on the parameter of these linear time algorithms. In contrast, for 1-planarity with these parameters we provide explicit algorithms whose dependence on the parameter is known and computable, albeit impractically large.

2 Vertex cover number

The *vertex cover number* k of an undirected graph G is the minimum number of vertices needed to touch all of the edges of G. This number is central to the theory of parameterized complexity, to the point where Guo et al. call it "the *Drosophila* of fixed-parameter algorithmics" [21]. After much earlier work on the problem, the best fixed-parameter tractable algorithms for computing the vertex cover number, parameterized by this number, take time $O(1.2738^k + kn)$ [8]. We will show that, when parameterized by vertex cover number, 1-planarity is also

fixed-parameter tractable, using a standard technique, kernelization, whereby we replace an instance graph with an equivalent instance of size bounded by a function of the kernel. Although the vertex cover number is a weaker parameter than the tree-depth that we consider later (a graph of vertex cover number k has tree-depth at most $k+1$), we begin with this parameter for two reasons. First, for this parameter we achieve stronger results, namely a polynomial kernel, than we do for the other parameters that we consider. And second, the simplicity of this case makes it an appropriate warm-up for the other parameters.

Before developing a parameterized algorithm for 1-planarity on graphs of small vertex cover, we start with an exact exponential-time algorithm, based on a separator decomposition of 1-planar graphs. We define a *π-curve* (for a given 1-planar drawing of a graph G) to be a simple closed curve in the plane that intersects the drawing only at vertices and crossings of G. In particular, the curve must not intersect the interior of any edge away from a crossing.

We first use cycle separators to argue the following:

Lemma 1 *In any 1-planar drawing of G there exists a π-curve, with π of length $O(\sqrt{n})$, that has at most $2|E|/3$ edges in the interior and at most $2|E|/3$ edges in the exterior. We call such curve a* balanced separating curve *for the drawing.*

Proof: The existence of such curve follows from the result of Miller [27] on simple cycle separators for embedded biconnected planar graphs. Miller considers graphs in which the vertices, edges, and faces may be weighted, with weights summing to 1. He proves that if the graph has n vertices, each face has a constant number of edges, and no face has weight greater than $2/3$, then there exists a simple cycle of $O(\sqrt{n})$ edges whose inside and outside both have total weight at most $2/3$.

Consider the planarization G_P of the 1-planar drawing of G, where each intersection is replaced by a vertex. Let Γ be the bipartite vertex-face incidence graph of G, which has a vertex for each vertex or face of G_P, and an edge for each incident pair of a vertex and a face of G_P; then Γ is planar and biconnected, and inherits a planar embedding from G_P. The faces of this embedding correspond to the edges of G_P, and are all quadrilaterals. Each face of Γ corresponds to an edge in G_P, which is either an uncrossed edge in G or half of a crossed edge in G. We assign weight $1/|E(G)|$ (where $E(G)$ denotes the edge set of G) to the faces of Γ that correspond to uncrossed edges in G, and we assign weight $1/(2|E(G)|)$ to the faces of Γ that correspond to crossed edges in G. We assign weight zero to the vertices and edges of Γ. These weights add to 1, with no faces having weight more than $2/3$, as Miller requires. We apply Miller's result to find a short cycle separator of the resulting triangulated graph. This cycle separator forms a π-curve α that splits the faces of Γ in a balanced way: the inside and outside of α each have total face weight at most $2/3$. □

For more on separators in 1-planar and k-planar graphs, see Grigoriev and Bodlaender [20] and Dujmović et al. [14].

Lemma 2 *Testing 1-planarity of an n-vertex graph G takes time $2^{O(n)}$.*

Proof: If the graph has more than $4n-8$ edges, we immediately return that it is not 1-planar [29]. Otherwise, we proceed with a divide and conquer algorithm, as follows.

For the divide and conquer algorithm, let us consider a more general problem. Given an n-vertex graph $G=(V,E)$ and a subset $F\subseteq E$, is there a 1-planar drawing of G where no edge of F participates in any crossing? We define a predicate $\phi(G,F)$ that is true when such drawing exists, and false otherwise. We can compute $\phi(G,F)$ recursively by trying all possible balanced separating curves and edge partitions, as follows.

Consider a cyclically-ordered sequence $\Pi=u_0,\ldots,u_k,u_0$ of distinct elements that are either vertices of G or pairs of edges of G, with each pair disjoint from F, representing the set of vertices and crossings that might appear in a π-curve. Let G_P be the "planarization" of G, an abstract graph formed by replacing each edge pair in Π by a degree-four vertex. Let $E_1\sqcup E_2$ be a partition of the edges of G_P. Let F' be formed from F by adding to it all of the edges incident to crossing vertices in G_P. Let H be a wheel graph in which a single central vertex u is connected to each vertex or crossing point in Π (represented by a vertex in G_P), and these vertices of G_P are connected to each other in the cyclic order given by Π. For $i=1,2$, let G_i be a graph with edge set $E(H)\cup E_i$ with its vertex set consisting of the endpoints of these edges. Let $F_i=E(H)\cup(F'\cap E_i)$. Then, both $\phi(G_1,F_1)$ and $\phi(G_2,F_2)$ are true if and only if G has a 1-planar drawing where no edge of F participates in a crossing and there is a π-curve separating E_1 from E_2 and passing through the vertices and crossings in Π in order. (Such a drawing might uncross one of the crossing vertices in G_P, but this does not prevent the resulting drawing from being 1-planar.) Moreover, in linear time we can combine drawings of G_1 and G_2 certifying that $\phi(G_1,F_1)$ and $\phi(G_2,F_2)$ are true to obtain a 1-planar drawing of G certifying that $\phi(G,F)$ is true.

Because of the existence of balanced separating curves for 1-planar drawings we have

$$\phi(G,F) \;=\; \bigvee_{\pi,E_1,E_2} \big(\phi(G_1,F_1)\wedge\phi(G_2,E_2)\big),$$

where π ranges over all cyclic sequences of $O(\sqrt{n})$ distinct vertices and edge pairs in G, and E_1,E_2 ranges over all partitions of the graph G_P for the given cyclic sequence such that E_1 and E_2 have at most $2|E|/3$ edges each. This means that $\phi(G,F)$ can be obtained solving $O(n^{\sqrt{n}}2^{|E|})=2^{O(n)}$ subproblems, each with at most $2|E|/3+O(\sqrt{n})$ edges. We thus get, when $|E|$ is larger than some constant, the recursion

$$T(|E|)\leq 2^{O(n)}T\big(2|E|/3+O(\sqrt{n})\big),$$

which solves to $T(|E|)\leq 2^{O(n)}$. □

Next, we turn to kernelization, the technique we use to transform this exact but exponential algorithm into a fixed-parameter tractable algorithm. Our kernelization depends on the 1-planarity properties of certain complete bipartite graphs:

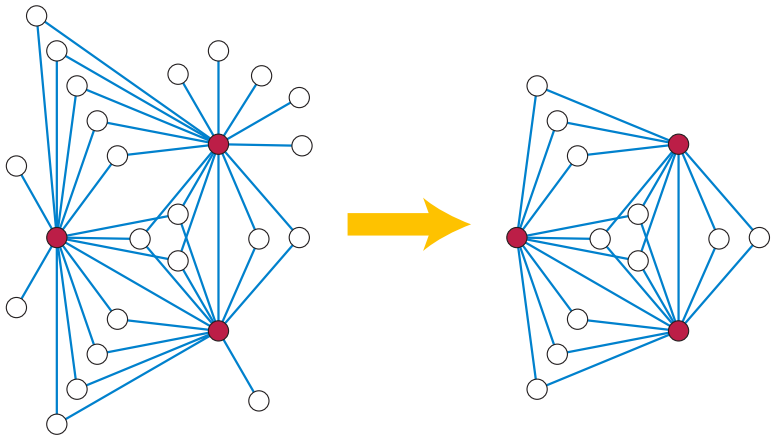

Figure 1: Kernelization for vertex cover number k: remove degree-one vertices, and reduce each $K_{2,i}$ subgraph (with two cover vertices on one side of the bipartition) to $K_{2,\min\{i,2k-3\}}$. Here $k = 3$, so the $K_{2,i}$ subgraphs are reduced to $K_{2,3}$.

Lemma 3 (Czap and Hudák [11]) *A complete bipartite graph is 1-planar if and only if it is of the form $K_{1,n}$, $K_{2,n}$, $K_{3,3}$, $K_{3,4}$, $K_{3,5}$, $K_{3,6}$, or $K_{4,4}$.*

Lemma 4 *Let G be a 1-planar graph with a subgraph H of the form $K_{2,i}$ formed by i vertices of degree two, all with the same two neighbors, for any $i > 0$. Then G has a 1-planar drawing in which the induced drawing of H is planar.*

Proof: If two edges of H that share an endpoint cross[1] then we can uncross them, resulting in a drawing with fewer crossings, and if two non-incident edges of H cross each other then we can redraw all of H without crossings near the previous position of these two crossed edges, again reducing the total number of crossings. Therefore, a 1-planar drawing of G that minimizes the total number of crossings has the desired property. □

Lemma 5 *Let a graph G have a known vertex cover C of size $|C| = k$. Then in time $O(n)$ we can transform G into a kernel G_C of size $O(k^2)$ such that G is 1-planar if and only if G_C is 1-planar. A 1-planar drawing of G_C may be transformed into a 1-planar drawing of G in linear time.*

Proof: We begin the construction of the kernel by deleting all vertices in $V(G)\backslash C$ that have degree one in G. This cannot change whether the remaining graph is 1-planar, because deleting a vertex from a 1-planar graph always produces another 1-planar graph, and because if the reduced graph is 1-planar then we

[1]Some definitions of 1-planarity disallow crossings of incident edges but this makes no difference in the class of graphs that have 1-planar drawings.

could add back the deleted degree-one vertices near their neighbors, without introducing new crossings, producing a 1-planar drawing of the original graph.

For each vertex v in $V(G) \setminus C$ that has three or more neighbors in G (all necessarily in C), label the vertex (without changing G) by choosing two of those neighbors arbitrarily, forming a two-edge path through v that starts and ends in C. If G is 1-planar, then the collection of two-edge paths formed in this way can connect at most $5k-10$ distinct pairs of vertices in C, and therefore produces at most $5k-10$ distinct labels. For if this bound did not hold, consider the graph G' whose vertices are the endpoints of these paths and whose edges connect the pairs of endpoints of each path. This graph G' could be drawn by routing its edges along the corresponding pairs of edges in the drawing of G, producing a drawing with k vertices, more than $5k-10$ edges, and at most two crossings per edge, contradicting the bound of [29]: If a graph on n vertices has a drawing with at most 2 crossings per edge, then it has at most $5n-10$ edges. Thus, if the pairs of endpoints of the chosen two-edge paths form more than $5k$ distinct pairs, we can immediately halt the algorithm and return the result that the input is not 1-planar.

Otherwise, we classify the vertices of $G \setminus C$ with degree three or more into at most $5k-10$ groups, according to the identities of their two arbitrarily-chosen neighbors in C. Assuming that G is 1-planar, each of those groups can have at most $6k$ vertices. For, each of the vertices that we grouped in this way has at least a third neighbor in C besides the two already chosen as the endpoints of its path. If one of the groups had more than $6k$ vertices, then at least seven of the vertices in the group would have the same third neighbor, because there are only k neighbors to choose from in C. Therefore, G would contain a $K_{3,7}$ subgraph; however, this is not possible in a 1-planar graph by Lemma 3. It follows that $G \setminus C$ has at most $O(k^2)$ vertices of degree three or more, if G is 1-planar. If this bound is exceeded, then we halt and return that the input is not 1-planar.

The vertices of degree two in $G \setminus C$ can be grouped by radix sort according to the identities of their two neighbors in C, forming a collection of $K_{2,i}$ subgraphs. If G is 1-planar, there are at most $5k-10$ such subgraphs $K_{2,i}$ by the same argument that we used above to bound the number of labels of higher-degree vertices. If one of these $K_{2,i}$ subgraphs has $i > 2k-3$ then we claim that G is 1-planar if and only if the subgraph G' formed by deleting $i-(2k-3)$ vertices within this subgraph to form a smaller $K_{2,2k-3}$ subgraph is also 1-planar. In one direction, if G is 1-planar, then clearly so is G'. In the other direction, suppose G' is 1-planar. Then by Lemma 4 it has a 1-planar drawing in which the given $K_{2,2k-3}$ subgraph is drawn planarly, with $2k-3$ quadrilateral faces. By the pigeonhole principle, two adjacent faces among this set of $2k-3$ must be empty of the $k-2$ vertices of C that are not part of the $K_{2,2k-3}$ subgraph. Therefore, the two edges e and f separating these two faces cannot be crossed by any edge of the 1-planar drawing, for any crossing edge would either have to cross entirely across one of these two faces (violating 1-planarity) or have an endpoint in each of the two faces (violating the assumption that neither of these faces contains a vertex of C). The remaining vertices and edges of G that were deleted to form the $K_{2,2k-3}$ subgraph may be added to the drawing, near path ef, without

violating 1-planarity, showing as desired that G is 1-planar.

Replacing $K_{2,i}$ with $K_{2,\min(i,2k-3)}$ separately for each of the groups of vertices in $G \setminus C$ results in the desired kernel G_C. G_C has $O(k^2)$ vertices of high degree and $O(k)$ groups of $O(k)$ vertices in $K_{2,i}$ subgraphs, for a total of $O(k^2)$ vertices.

If a drawing of G_C is found, a corresponding drawing of G may be found by eliminating crossings between pairs of edges belonging to the same $K_{2,i}$ subgraphs in G_C, finding an uncrossed length-two path with two vertices in C as path endpoints within each $K_{2,2k-3}$ subgraph, expanding each of these $K_{2,2k-3}$ subgraphs to $K_{2,i}$ for the correct value of i from the original graph G (placing the restored vertices near the uncrossed path), and finally adding back any deleted degree-one vertices of G. □

An example of this kernelization is depicted in Figure 1, for a graph with vertex cover number three.

Theorem 1 *We can test the 1-planarity of a given n-vertex graph, parameterized by its vertex cover number k, in time $O(n + 2^{O(k^2)})$.*

Proof: We begin by testing whether the input has more than $4n - 8$ edges. If so, we halt the algorithm and report that the input is not 1-planar. Next, we find a maximal matching M in G. Then $|M| \leq k \leq 2|M|$, where k is the size of the optimal vertex cover. The first of these two inequalities is true because every vertex cover must include at least one endpoint of every matched edge, and the second inequality is true because the set of all endpoints of matched edges is a vertex cover. We apply Lemma 5 to the vertex cover given by the endpoints of edges in M, reducing the input G to a kernel of size $O(k^2)$. Finally, we run the exact algorithm of Lemma 2 on this kernel. □

It would improve the constant factor in the $2^{O(k^2)}$ term of our time bound to use a fixed-parameter-tractable algorithm to find the optimal vertex cover, instead of using the 2-approximate cover given by the endpoints of a maximal matching. However, we omit the details, as the time reduction would not be enough to make this algorithm practical.

Corollary 1 *We can test 1-planarity for split graphs in time $O(n)$.*

Proof: If a given split graph has a clique of size seven, it is not 1-planar, and otherwise, it has a vertex cover of size six and we use the above algorithm. □

3 Tree-depth

As we now show, 1-planarity parameterized by tree-depth may be tested by a fixed-parameter tractable algorithm. The *tree-depth* of a graph G is the smallest depth of a forest F on the same vertex set as G such that every edge of G connects an ancestor-descendant pair in F, where we measure the depth of a tree as the maximum number of vertices on a root-leaf path [28]. This ancestor-descendant property is always true of depth-first search trees, but (unlike depth-first search

trees) the forest F is not required to be a subgraph of G. Equivalently, the tree-depth is the size of a maximum clique in a trivially perfect supergraph of G chosen to minimize this clique size. This follows because one way to define a trivially perfect graph is that it is the graph of ancestor-descendant pairs in a forest.

The tree-depth can also be related to the treewidth and vertex cover number. Since the trivially perfect graphs are a special case of the chordal graphs, and the treewidth of a graph is (one less than) the maximum size of a clique in a chordal supergraph chosen to minimize this clique size, it follows that tree-depth is always at least one plus treewidth. A graph G with vertex cover number k has tree-depth at most $k+1$, for we may find a tree T of depth $k+1$ that has the k vertices of the cover on a path, from which all other vertices descend as leaves; for this tree, all edges of G connect ancestor-descendant pairs in the tree. Because the tree-depth can be bounded in this way by the vertex cover number, in some sense the result of this section is stronger than that of Theorem 1, although the dependence on the parameter is worse.

An n-vertex path has tree-depth $\lceil \log_2(n+1) \rceil$. It follows that, for any graph G of tree-depth d, an arbitrary depth-first search tree for G will necessarily have a depth in the range $[d, 2^d - 1]$, because if it were deeper than this range it would contain a path that is too long for the given depth and at least the tree-depth d. This provides a crude but easy to compute approximation for tree-depth. Additionally, based on this observation, one can derive a fixed-parameter tractable algorithm for computing the tree-depth, by finding a depth-first search tree, using it to construct a tree decomposition, and applying standard dynamic programming techniques to this decomposition [28].

Lemma 6 *Let G be a graph with tree-depth at most d, as witnessed by a forest F of depth d for which all edges of G connect ancestor-descendant pairs. Then in linear time it is possible to replace G by an equivalent kernel for 1-planarity consisting of a collection of disconnected subgraphs with $O(2^{2d^2+O(d)})$ vertices each.*

Proof: If G is not biconnected we may test 1-planarity on each biconnected component of G separately; therefore, we can assume without loss of generality that the given graph G is biconnected, and that we have a tree T of depth d such that every edge of G connects an ancestor-descendant pair in T. We can also assume without loss of generality that each node of T is adjacent to at least one node in each of its child subtrees, because otherwise we could move those children up to be siblings of the node, which does not increase the depth. Additionally, we can assume that each child subtree induces a connected subgraph of G, because otherwise we could split it into two separate children. Since the tree-depth is d, the longest path in G has length less than 2^d.

Now consider how many children a node v in T can have. For each child subtree T_i, consider the set S_i of v and ancestors of v that are connected to nodes in T_i. We will classify the subtrees T_i by these sets of upward connections; for each subset S of v and its ancestors, let $C(S)$ be the set of child subtrees T_i

of v with the same classification, the ones for which $S_i = S$. There are at most 2^d different subsets S of v and its ancestors, and we want to show that for each of them, $C(S)$ has bounded size.

As a first step towards this goal, we observe that when $|S| = 1$, we must have that $|C(S)| = 0$. For, if $|S| = 1$ and $S \neq \{v\}$, there can be no subtree T_i at v that has $S_i = S$, as such a subtree would violate the assumption that v is adjacent to at least one node in its child subtree T_i. And if $S = \{v\}$ then again there can be no subtree T_i at v that has $S_i = S$, for if there were then v would be an articulation point, violating the assumption of biconnectivity.

Next, consider the case that $|S| \geq 3$. That is, we have a set S consisting of v and two or more of its ancestors, and a set $C(S)$ of child subtrees of v that are each connected to all of the nodes in S. Choose exactly three nodes of S and, for each child subtree T_i in $C(S)$, let X_i be a smallest subgraph connecting the three chosen nodes in the subgraph of G induced by $T_i \cup S$. By the bound on the length of paths in G, $|X_i| = O(2^d)$. Note that, among any three of these trees X_i, X_j, and X_k (all for members of $C(S)$) there must be at least one crossing between two of the trees, because contracting each tree to a single node produces a $K_{3,3}$ subgraph. There are $\Omega(|C(S)|^3)$ triples of trees, and at least one crossing per triple. Any single crossing of this type determines two of the three trees X_i, X_j, and X_k (the two that contain the crossing edges), and there are $|C(S)|$ choices for the third tree, so any single crossing can be produced by at most $|C(S)|$ triples. Thus, multiplying the number of triples by the number of crossings per triple and dividing by the number of triples per crossing shows that there are $\Omega(|C(S)|^2)$ crossings altogether, among a set of only $O(|C(S)|2^d)$ edges. In order to prevent the pigeonhole principle from forcing some edge to be crossed twice, we must have $|C(S)| = O(2^d)$.

Finally, consider the case that $|S| = 2$. In this case, $|C(S)|$ can be unbounded (e.g. consider the graph $K_{2,n}$, which has tree-depth three). But, if it is greater than 2^d, then it does not matter how much greater it is: no cycle in any 1-planar drawing of G can separate the two vertices in S, because the minimal such cycle would have to have length at most 2^d but would have to cross each of the subgraphs T_i, a contradiction. So in this case we can split the graph into subgraphs formed from each child T_i together with an uncrossable edge between the two nodes in S, and test 1-planarity separately for each of these subgraphs. When $C(S)$ is small enough that no such split is possible, $|C(S)| = O(2^d)$.

After performing any splits from the $|S| = 2$ case, the remaining graph has its nodes arranged into a tree of height d in which each node has $O(2^{2d})$ children. Therefore, the total number of nodes in the tree is $O(2^{2d^2+O(d)})$. □

By combining this kernelization with the known fixed-parameter tractable algorithm for computing tree-depth and with Lemma 2 for testing the 1-planarity of the kernel, we obtain

Theorem 2 *The 1-planarity of a given graph with tree-depth d may be computed in time*

$$O(n2^{2^{2d^2+O(d)}}).$$

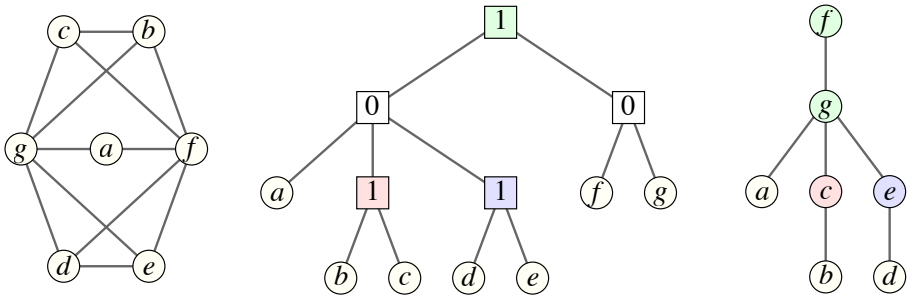

Figure 2: Finding a low-tree-depth representation of a cograph by forming a path for each 1-labeled cotree node, consisting of the cotree leaves that descend from it but are not in its heaviest child. Left: a cograph. Center: its cotree. Right: the tree formed by connecting together the paths L_x. Each cotree node has the same color as its corresponding path.

Because the kernel may contain multiple connected components, and we are bounding the component size but not the number of components, the dependence of the time bound for d is multiplied by n rather than (as in Theorem 1) added to it. Alternatively, it would be possible to remove isomorphic components from the kernel, and get a bound of the form $O(n + f(d))$, but with a larger dependence on d.

As an example of the power of this approach, we show how to use it to recognize 1-planar cographs. The cographs are the graphs G that can be represented by cotrees. A cotree has the vertices of G as its leaves; every internal node is labeled either 0 or 1, and two vertices of G are adjacent if and only if their lowest common ancestor is labeled 1. We assume that this tree is in canonical form meaning that no two adjacent internal nodes have the same label as each other and that each internal node has at least two children.

Cographs are well-quasi-ordered by induced subgraphs [12], from which it follows that there is an algorithm for testing 1-planarity by checking for the existence of a finite set of forbidden induced subgraphs; however, we do not know how to explicitly list these forbidden subgraphs nor do we know how to turn a recognition algorithm along such lines into an algorithm for finding a 1-planar drawing. In contrast, the algorithm outlined below for recognizing 1-planar cographs is explicit (albeit with impractically large constants) and constructs a drawing of the graph.

Lemma 7 *Let $\mathcal{C}_{a,b}$ denote the class of cographs that do not contain K_a nor $K_{b,b}$ as subgraphs. Then, for any integers a and b, the graphs in $\mathcal{C}_{a,b}$ have tree-depth at most $1 + (a-1)(b-1)$.*

Proof: For any graph G in this class, we use a cotree in canonical form representing G, and use it to guide the construction of a forest F on the nodes of G.

For each node x labeled 1 in the cotree, let H_x denote the subtree descending from a child of x that contains the largest number of leaves (breaking ties arbitrarily) and let L_x denote the set of leaf descendants of x that are not in H_x. For each maximal set L_x (not contained in L_y for some 1-labeled node y), we form a path, which will form a subgraph of F. If the closest 1-labeled ancestor of cotree node x is node y, we set the parent of the top node of path L_x to be the bottom node of path L_y. In addition, if any vertex v of G does not belong to a set L_x, we make it a leaf of the forest F, and we set the parent of v to be the bottom node of the path for the lowest 1-labeled ancestor of v in the cotree. The forest constructed in this way (shown in Figure 2) will necessarily have the defining property of tree-depth that every edge in G connects an ancestor-descendant pair in F.

If L_x has at least $2b-1$ leaves, then the leaf descendants of x contain a $K_{b,b}$ subgraph. For this reason, every path L_x has at most $2(b-1)$ vertices of G in it. Additionally, on any path from the root to a leaf in the cotree, at most one of the 1-labeled cotree nodes can have more than $b-1$ nodes in L_x, for if one such node does, then each of its ancestors must have at most $b-1$ nodes in L_x, or else we would again have a $K_{b,b}$ subtree. Finally, observe that a path from the root to a leaf in the cotree that has $a-1$ 1-labeled nodes would give rise to a K_a subgraph; therefore, every such path has at most $a-2$ 1-labeled nodes. By this analysis, the longest path from leaf to root that could exist in the forest F consists of one vertex of G that does not belong to a set L_x, one set L_x of size $2(b-1)$, and $a-3$ sets L_x of size $b-1$, matching the depth given in the statement of the lemma. □

Corollary 2 *We can recognize 1-planar cographs, and find 1-planar drawings of them, in $O(n)$ time.*

Proof: We first test whether the given cograph contains K_7 or $K_{5,5}$ as a subgraph. If it does, it is not 1-planar. If it does not, we may apply Lemma 7 and Theorem 2. □

4 Cyclomatic number

We say that a graph G has *cyclomatic number* k if k is the smallest number of edges that must be removed from G to yield a forest; equivalently $k = m - n + c$, where c is the number of connected components in G. By a *maximal degree two path* we shall mean a path between two vertices each of degree greater than two such that all vertices in the interior of the path have degree two. For technical reasons, an edge between vertices each having degree greater than two will also be considered a maximal degree two path. Gurevich et al. define a *k-almost-tree* to be a graph G such that given a spanning tree T of G every biconnected component of G has at most k edges not in T [22]. The choice of T is arbitrary; every spanning tree gives the same number of additional non-tree edges. Equivalently, a k-almost tree can be defined as a graph in which each biconnected component has cyclomatic number k.

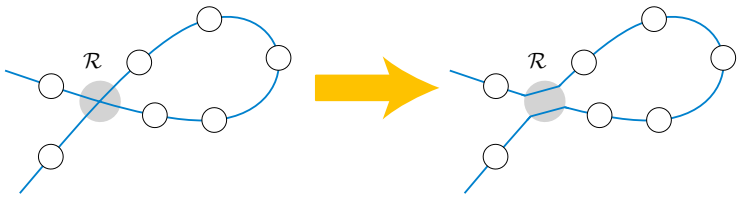

Figure 3: Removing a crossing in a degree two path

The cyclomatic number and k-almost-tree parameter have previously been used as parameters in fixed parameter algorithms. For example, in biology, gene expression can be represented as a Boolean network in which individual genes are represented as vertices and edges represent correlations between pairs of genes. Fixed parameter tractable algorithms have been designed for the control problem, which involves finding sequences of valid labelings of genes as being active or inactive [1]. In operations research, road networks may be modeled by graphs with weighted edges, and fixed-parameter algorithms for continuous facility location on such models have been constructed [22]. Intraprogram communication networks in distributed systems use vertices to represent modules of a program to be computed in parallel and edges to represent communicating pairs of modules; they also have structure yielding fixed-parameter algorithms [17] with respect to this parameter.

Lemma 8 *If G is a graph with cyclomatic number k and no degree one vertices, then G has at most $2k-2$ vertices of degree greater than two. Furthermore, this bound is tight. Also, the number of maximal degree two paths is at most $3k-3$.*

Proof: Double counting edges yields $2(n-c+k) \geq 2a+3b$, where a is the number of degree two vertices and b is the number of vertices of degree greater than two. Using $n = a+b$ and $c \geq 1$ we obtain $b \leq 2k-2$, establishing the upper bound. For the upper bound consider any biconnected cubic graph with $2k-2$ vertices, such as a cubic Halin graph whose underlying tree has k leaves.

For the bound on the maximal degree two paths consider the graph G' where each maximal degree two path is reduced to a single edge. The graph G' has cyclomatic number k and at most $2k-2$ vertices. This implies that G' has at most $3k-3$ edges, establishing the bound. □

Lemma 9 *If G is 1-planar, then there is a 1-planar drawing of G such that no maximal degree two path crosses itself.*

Proof: It suffices to show that a self crossing in a maximal degree two path can be removed without increasing the number of crossings on any edges. We can locally uncross a self intersection changing the drawing within a circular region $\mathcal{R}$ around the intersection that is not crossed by other edges. See Figure 3 for an example of this operation. □

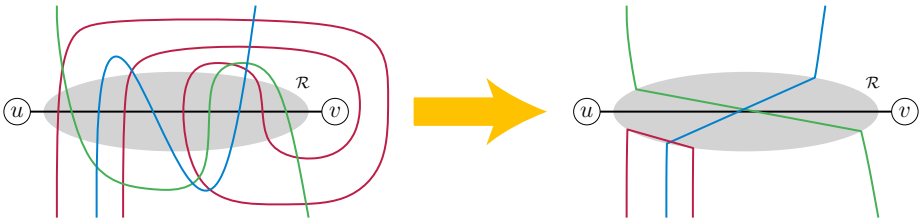

Figure 4: Left crossing sequence `rgbrbrgbrg`; Right crossing sequence `bg`

Lemma 10 *Every word on $n > 1$ symbols, without consecutive equal symbols, of length greater than $2n! - 1$ has a subword on $k > 1$ symbols, for some $k \leq n$, such that each symbol appears at least k times in the subword. Furthermore, this bound is tight, i.e., there exists a word w of length $2n! - 1$ on n symbols such that for every $1 < k \leq n$, w has no subword on k symbols in which each symbol appears at least k times.*

Proof: Let w be a word on n symbols of length at least $2(n!) - 1$, and let σ be the symbol appearing least often in w. If σ occurs more than n times in w, then we are done. So assume that σ occurs at most $n - 1$ times. Removing σ from w leaves us with at least $2(n!) - n$ symbols split into at most n subwords. Thus, the longest of these subwords has length at least

$$\frac{2(n!) - n}{n} = 2(n-1)! - 1.$$

Call this long subword u. Since u contains at most $n - 1$ unique symbols we are done by induction on n.

To construct a word on n symbols of length $2(n!) - 1$ with no reducible subword, let $\sigma_0, \sigma_1, \ldots, \sigma_n$ be our n symbols. Now recursively define the words by

$$w_k = (w_{k-1}\sigma_k)^{k-1} w_{k-1}$$

and $w_2 = \sigma_0\sigma_1\sigma_0$. A simple induction argument shows that the length of w_k is $2(k!) - 1$. □

Lemma 11 *If G is a 1-planar graph with p maximal degree two paths, then G has a 1-planar drawing such that every maximal degree two path is crossed at most $2(p!) - 1$ times.*

Proof: We need only to show that given a maximal degree two path from u to v with more than $2(p!) - 1$ crossings, we can reduce the number of times that it is crossed without increasing the crossing count on other degree two paths.

First, we continuously deform the plane such that the path from u to v is a straight line. This is possible since we may assume that maximal degree two paths do not self intersect by Lemma 9. Now we consider the sequence of crossings between the path from u to v and the other maximal degree two paths.

In this sequence there are at most p symbols. So if the number of crossings on the path from u to v is greater than $2(p!) - 1$, Lemma 10 implies that there is a subword on p' symbols such that every symbol appears at least p' times.

Now, we construct a strictly convex region $\mathcal{R}$ around the crossings represented by this word such that only paths represented in the word intersect the region, and such that a path does not reintersect the path from u to v without first leaving $\mathcal{R}$. For every path we shortcut it from the first time it intersects $\mathcal{R}$ to the last time it intersects $\mathcal{R}$, in path order, with a straight line. So now each path in $\mathcal{R}$ is a straight line, and therefore they can only intersect each other at most once. So, we have reduced the number of crossings on the path from u to v, without increasing the crossings on the other paths. □

Lemma 12 *Let G be a graph with cyclomatic number k. Then in linear time we can transform G into a kernel G_C of size $O((3k-3)(3k-3)!)$ such that G is 1-planar if and only if G_C is 1-planar. In addition, a 1-planar drawing of G_C may be transformed into a 1-planar drawing of G in linear time.*

Proof: We remove degree one vertices from G until no more are left, producing the 2-*core* of G [32]. This process can be done in linear time by maintaining a queue of degree one vertices. A degree one vertex may be added to any drawing without introducing crossings, so a graph has a 1-planar drawing if and only if its 2-core has a 1-planar drawing.

Lemma 8 implies that we have at most $p = 3k - 3$ maximal degree two paths. For each of these maximal degree two paths we reduce the number of degree two vertices to $2p! + 1$ if they exceed this amount. Since Lemma 11 guarantees that, if G is 1-planar, then it has a drawing such that no maximal degree two path is crossed more than $2p! - 1$ times, this reduction does not change the 1-planarity of the graph. Thus, we have a kernel G_C of size $O((3k-3)(3k-3)!)$ such that G is 1-planar if and only if G_C is 1-planar. □

Theorem 3 *We can test the 1-planarity of a graph with cyclomatic number k in time $O\left(n + 2^{O((3k)!)}\right)$.*

Since a graph can be decomposed into its biconnected components in linear time and edges in separate biconnected components need not cross we have the following corollary to Theorem 3.

Corollary 3 *We can test the 1-planarity of a k-almost tree in time $O(n2^{O((3k)!)})$.*

5 Bandwidth

If the vertices of a graph G are arranged on the real line with distinct integer coordinates, the *bandwidth* of the arrangement is the maximum length of an edge of G. The bandwidth of the graph G itself is the minimum, over all possible linear arrangements of G, of the length of the longest edge in the arrangement. The bandwidth may also be defined as one less than the minimum clique number

of any proper interval graph having G as a subgraph, a formulation that makes clear the relation between bandwidth, pathwidth (the same notion with interval graphs in place of proper interval graphs), and treewidth (the same notion with chordal graphs in place of interval graphs) [24].

In this section we show that 1-planarity remains NP-complete even when restricted to graphs of bounded bandwidth. Graphs of bounded bandwidth also have bounded pathwidth, treewidth, and clique-width, so 1-planarity is also hard for those parameters.

5.1 Overview

Our proof of NP-completeness of 1-planarity for graphs of bounded bandwidth is based on a standard gadget-based reduction from 3-satisfiability, but because of the complexity of the gadgets we break the proof up into several steps. In this subsection we outline the general idea of the reduction.

The overall structure is a graph G of bounded bandwidth with three parts: one part for the variables of the 3-satisfiability instance (blue in Figure 5), one part for the clauses of the instance (red in the figure), and one part that (despite its low bandwidth) forms a grid-like structure that holds the other two parts in their places (black in the figure). The variable and clause gadgets will form subgraphs that are only attached to the grid gadget at one end (with the points of attachment not all lying within a single face of the grid); the points of attachment are shown as small green circles in the upper left of the figure.

Although not attached graph-theoretically to the rest of the grid, the variable and clause gadgets will still interact with the rest of the grid by the crossings that are allowed between their edges. Because of these allowed crossings, the variable part of graph G will be forced to zigzag horizontally back and forth across the grid; every horizontal stretch of this part will correspond to a single variable from the 3-satisfiability instance. Again, because of its interactions with the grid, the clause part of graph G will be forced to zigzag vertically up and down across the grid; every vertical stretch of this part will correspond to a single clause from the 3-satisfiability instance.

Within the variable and clause gadgets of this structure, we will incorporate smaller gadgets that allow each variable or clause gadget to twist relative to itself, so that it has two different possible orientations (twisted or untwisted) as it passes across the grid. In particular, each individual variable clause (a single horizontal stretch of the zigzag pattern formed by the variable part of the graph) will have twist gadgets at either of its two ends, allowing it to take either of these two possible orientations freely. We will use those orientations to encode the truth value of the variable. Each clause gadget will be forced (by its interaction with the grid at either end of its vertical stretch) to have at least one twist, so that the top and the bottom ends of the clause gadget have opposite orientations to each other. We will place twist gadgets within each clause gadget that allow it to have such a twist only when one of the variables has a truth value that satisfies it.

In short, then, we have a grid component of the graph which exists in order

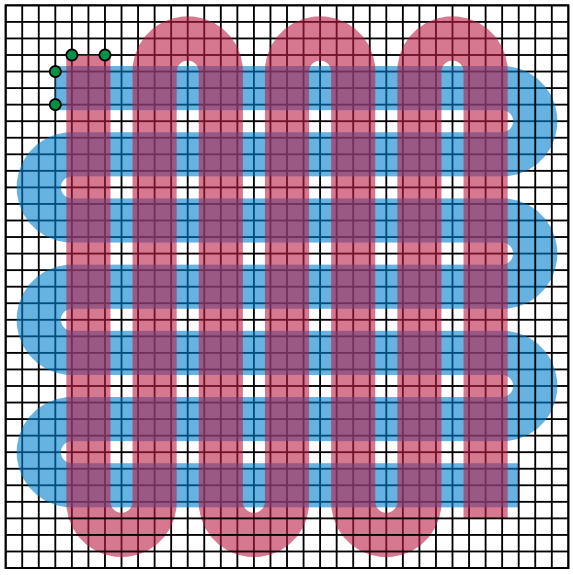

Figure 5: Global structure of the graph produced by the NP-completeness proof for graphs of bounded bandwidth: a grid structure (black) holding in place two paths that zigzag across each other, with each straight segment of these paths either representing a variable (blue) or a clause (red) of a 3-satisfiability instance. The small green circles near the top left indicate the points at which the variable and clause gadgets are attached to the grid; the remaining shape of the variable and clause gadgets is controlled by their allowed crossings with the grid.

to guide the layout of the other parts and make them cross each other in the correct locations. We have a variable gadget for each variable of the 3-sat instance that may take on one of two different orientations according to the 1-planar embedding of the twist gadgets at either of its ends. And, we have a clause gadget for each clause of the 3-sat instance that must twist at least once, and can only twist at the point where it crosses a variable gadget of a variable that belongs to the clause and that has an orientation corresponding to a truth assignment to that variable that would satisfy that clause. As we will describe, it is possible to find a graph of bounded bandwidth that contains gadgets of all these types, and that allows no other 1-planar embeddings than the ones intended to exist as part of the construction. As a result, the graph constructed in this way will have a 1-planar embedding if and only if the given 3-satisfiability instance is satisfiable. This reduction, when complete, will prove the NP-completeness of 1-planarity for graphs of bounded bandwidth.

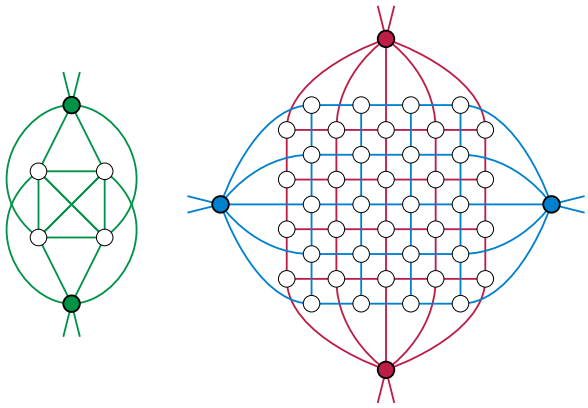

Figure 6: Gadgets for reducing colored 1-planarity to uncolored 1-planarity. Left: an uncrossable edge gadget. Right: two crossing grids. In each case the colored vertices indicate the endpoints of the original edge while the uncolored vertices are part of the gadget added to replace it.

5.2 Crossing control

Our NP-completeness proof involves crossings between several different types of gadgets, and it will be helpful to have some fine-level control of how these crossings may occur. To do so, we introduce a variant of 1-planarity, which we call *colored 1-planarity*, in which the input instance is augmented with edge colors that describe which crossings are allowed. Specifically, in the colored 1-planarity problem, we assume that the edges are labeled from a finite set of colors. One designated color (black, say) is not allowed to participate in any edge crossings; otherwise, an edge may only cross another edge of the same color. The task is to determine whether the given graph has a 1-planar embedding satisfying these color constraints.

Lemma 13 *An instance of colored 1-planarity may be reduced to an instance of 1-planarity without colors, preserving the existence or nonexistence of a valid 1-planar drawing, in such a way that the bandwidth of the uncolored instance is* $O(1)$ *times the bandwidth of the colored instance.*

Proof: We replace each edge of the uncrossable color by a gadget whose unique 1-planar embedding does not allow it to be crossed by any other part of a drawing (Figure 6, left). For each other color, we choose an integer i and replace each edge of that color by a grid with i rows and $i+1$ columns, with the two endpoints of the edge connected to the grid points in the two extreme columns of the grid (for instance the red grid in Figure 6, right). Two grids of the same size may cross each other, as shown in the figure, but it is not possible for grids of different sizes to cross. □

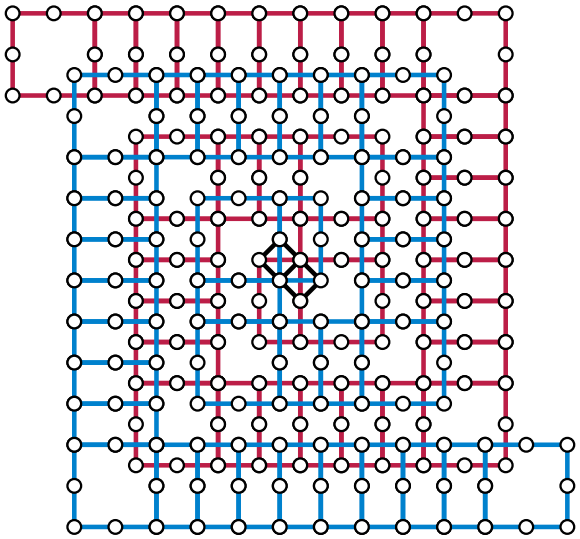

Figure 7: Double spiral grid structure. Colors are used to show the two arms of the spiral, but they are unrelated to color 1-planarity.

5.3 Double spiral grid

To form a grid-like substrate on which to build our reduction from 3-satisfiability, we use the double spiral pattern shown in Figure 7, in which two arms spiral from the center of the pattern. We may color the edges of this pattern with a constant number of colors (with color 0, denoting uncrossable edges, used for edges that do not cross any others) and apply Lemma 13 in order to find a graph with the same spiral structure as the figure in which only the crossings shown by the figure are allowed. With this edge coloring, the red and blue spiral arms of the figure (both of which have bounded bandwidth) are forced to continue crossing each other as they spiral around the center, with the leading edge of one arm crossing through the trailing edge of the other arm.

The two arms together generate a grid-like pattern of unbounded size. It is not itself a grid graph (which would have unbounded bandwidth) but its planarization (the planar graph formed by replacing each crossing by a vertex) contains a subdivision of a grid whose size is proportional to the number of windings of the spiral. By additional subdivisions of edges into paths and by using colored 1-planarity, we may allow edge crossings between edges in the double spiral that do not participate in this grid subdivision and edges from other components of the reduction, without allowing unplanned crossings between pairs of edges that both belong to the double spiral. In this way, we may make the parts of the spiral that do not participate in this grid be transparent to the remaining components of the reduction, making the spiral pattern function as a perfect grid for the purposes of describing its interactions with these other

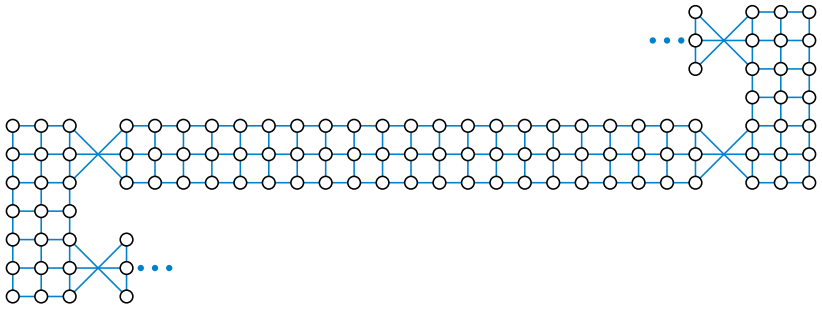

Figure 8: Schematic view of the gadget for a single 3SAT variable, consisting of a rigid $3 \times i$ grid with crossover gadgets at either end.

components.

In the full reduction, we will also replace some of the edges of this grid subdivision by paths in a colored 1-planarity instance, using a color for each path edge that does not appear anywhere else within the nearby faces of the pattern, in order to allow this grid structure to be crossed in a controlled way by the variable and clause gadgets of the reduction. By using Lemma 13 this replacement can be done in a way that does not affect the set of valid 1-planar embeddings of the grid itself.

In order to control the bandwidth of the graph formed by combining this grid gadget with the variable and clause gadgets of the reduction, we require that the attachment points where the variable and clause gadgets attach to the grid gadget (the green circles of Figure 5) lie near each other within the same spiral arm of the grid gadget. This constraint will not cause any theoretical difficulties in the construction.

5.4 Variable and crossover gadgets

As discussed in Subsection 5.1, we will represent the variables of a 3SAT instance by a gadget that is forced (by its set of allowed crossings with the grid gadget) to zigzag horizontally back and forth across the grid of Subsection 5.3. Each horizontal stretch of this zigzagging pattern will be formed by a gadget for a single variable, in the form of a $3 \times i$ grid of vertices (for some value i chosen sufficiently large to allow this variable gadget to be crossed by each of the clause gadgets). At either end of this grid, we will have crossover gadgets, shown schematically in Figure 8, that allow the grid to have one of two possible orientations: either of its two rows of squares may form the top row of squares in the drawing, with the other row of squares below it. These two orientations will correspond to the two truth values of the variable. Although Figure 8 depicts this gadget as three mutually-crossing edges, it actually consists of three longer paths, and is depicted in more detail in Figure 9.

Within the variable gadget, there are three parallel and disjoint paths of

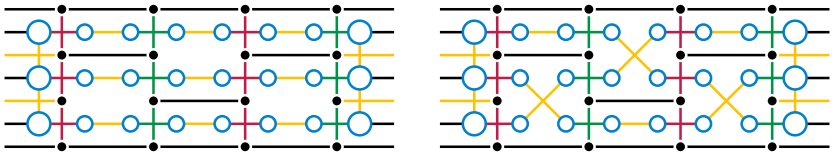

Figure 9: The crossover gadget for the two ends of a variable gadget, in its uncrossed (left) and crossed (right) states.

length i, extending horizontally across the gadget. To allow the gadget to have its two different orientations, these three paths must be allowed to cross each other within the crossover gadget. However, it would not work to allow the crossings between these three paths to lie close enough to each other that they belong to a single face of the underlying grid (as might be suggested by the drawing of Figure 8, which shows all three paths crossing at a single point). The reason is that, if all three paths of the gadget were allowed to pass through a single face of the grid, it would then be possible to continue drawing the rest of the variable gadget, and the sequence of variable gadgets connected to it, all within this same face, thwarting our intended zigzag pattern. In order to control the global shape formed by the variable gadgets as they extend across the grid, we must make sure that only two of the three paths ever lie in a single face of the grid.

A crossover gadget that allows the variable gadget to take either of its two orientations, but does not allow it to escape into a single face of the grid, is shown in Figure 9. The colored edges of the figure are used to describe pairs of edges that may cross in a colored 1-planarity problem, as described in Subsection 5.2. Note that the gadget consists of two disconnected subgraphs: the blue vertices within a variable gadget and the black vertices within the grid. As long as the three leftmost vertices of the variable gadget are constrained (by earlier parts of the construction) to lie in the three grid faces that they are shown in, the rest of the crossover gadget must extend rightwards across the grid as shown, in order to reach another set of faces from which the three right vertices can be connected to each other. The crossings in the middle of the gadget would allow the three right vertices to be arranged in any of the six possible permutations of the three left vertices, but only the original permutation and its reversal allow a consistent placement of the yellow edges at the right of the figure.

5.5 Clause and gated crossover gadgets

As discussed in Subsection 5.1, we will represent the clauses of a 3SAT instance by a gadget that is forced (by its set of allowed crossings with the grid gadget) to zigzag vertically up and down across the grid of Subsection 5.3. Each vertical stretch of this zigzagging pattern will be formed by a gadget for a single clause, and (like the variable gadgets) will consist of a $3 \times i$ grid of vertices, for some appropriate choice of i, interspersed with *gated crossover* gadgets, one for each

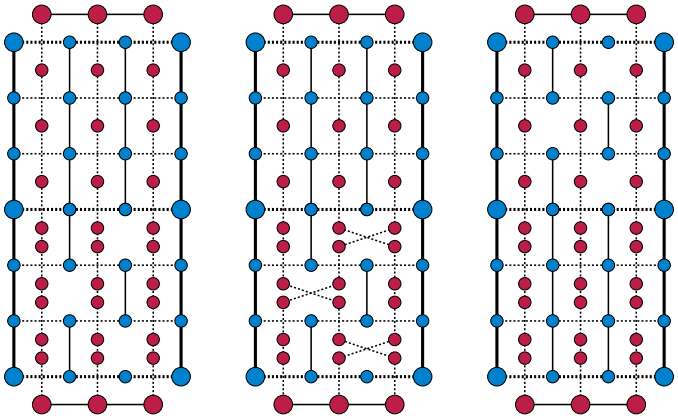

Figure 10: Three states of a gated crossover gadget. Left: the variable gadget's orientation corresponds to a truth value that satisfies the clause, but the clause gadget is uncrossed. Center: the variable satisfies the clause, and the clause gadget is crossed. Right: the variable does not satisfy the clause; no crossing is possible. The edges of the figure are colored to form a colored 1-planarity problem that prevents crossings other than the ones shown between crossing pairs of dashed edges. The underlying grid and its crossings within the gadget are not depicted.

term in the clause. The top and bottom connections of this gadget with the adjacent clauses will form more grid graphs that are rigid in their layout with respect to the grid gadget, forcing the clause to twist an odd number of times within its gated crossover gadgets. A twist will only be allowed within a gated crossover gadget when the variable gadget corresponding to one of the terms in the clause has the correct orientation (corresponding to a truth assignment for which that variable satisfies the clause). In this way, it will only be possible to find a 1-planar layout for all of the clause gadgets if the variable gadgets are oriented in a way that corresponds to a satisfying assignment for the whole 3SAT instance.

To form a gated crossover gadget, we replace two vertically-adjacent squares of the grid of a variable gadget, as shown in Figure 10. One of the two squares (the one that is the bottommost of the two squares for truth assignments that cause the variable to satisfy the clause) is replaced by the grid part of a crossover gadget, rotated by 90° from the ones in Figure 9, while the other square is replaced by a modified crossover gadget with extra edges that prevents its crossed state from being a valid layout. The clause gadget forms three paths that pass through both of these crossover gadgets, with their edges colored to form a colored 1-planarity instance in such a way that the parts of the paths within the top square are not allowed to cross each other (they can only make the required crossings with the gadget) while the paths may cross each other

within the bottom square. Thus, when the gadget is aligned so that the bottom square is the one containing the unmodified crossover gadget, the three clause gadget paths may cross each other or not, but when it is aligned so that the top square is the one containing the unmodified crossover gadget, no crossing is possible.

The parts of the gated crossover gadget that come from the variable and clause gadgets, shown in the figure, are overlaid by additional parts coming from the grid gadget, which force the left and right sides of the variable gadget to spread across multiple grid faces, and also force the top and bottom sides of the clause gadget to spread across multiple grid faces, preventing additional unwanted 1-planar layouts where part of a variable or clause gadget and all the rest of the gadgets connected in sequence to it lie within a single face; we omit the details, as they complicate the gated crossover gadget without significantly affecting the mechanism by which it works.

5.6 NP-completeness

Theorem 4 *It is* NP*-complete to test 1-planarity for graphs of bounded bandwidth.*

Proof: We use a reduction from 3SAT as described above, by forming a graph from the grid gadgets, variable gadgets, and clause gadgets described in the previous subsections. The subgraph formed by the sequence of variable gadgets, and the subgraph formed by the sequence of clause gadgets, are connected at an appropriate place (near one corner of the grid) to the subgraph formed by the grid gadget; otherwise these three graphs are disconnected from each other, and interact only through their crossings. The variable and clause gadgets are forced by their crossings with the grid gadget to take zigzagging paths across the grid, so that each variable gadget crosses each clause gadget in a controlled area within the grid.

Each variable gadget may have one of two orientations (given by the crossover gadgets at either of its ends), one of which is associated with a true value of the variable and the other of which is associated with a false value. When a variable gadget crosses a clause gadget that has a term containing that variable, the gated crossover gadget associated with that crossing (consisting of subconfigurations within both the variable and clause gadget) allows the clause gadget to twist at that point, if and only if the variable's truth assignment would satisfy that clause. For variables with the wrong truth assignment for the given clause, or that do not participate in the clause, no such twist is possible. The parts of the graph that connect consecutive clause gadgets are arranged in such a way that they can be embedded in a 1-planar way only if the two ends of the clause gadget have the correct orientations, with a single twist relative to each other.

If the input 3SAT instance has a satisfying assignment, it can be used to choose twists for the crossover and gated crossover gadgets of this reduction in such a way that the entire graph is embedded in a 1-planar way. On the other hand, any embedding of the graph must come from a satisfying truth assignment

in this way. The translation described by this reduction can be performed by a polynomial time algorithm, as it involves only putting the correct gadgets together in the correct sequence. Therefore, this translation is a valid many-one reduction from 3SAT to 1-planarity. The graph that results from the reduction consists of a bounded number of bounded-bandwidth pieces (the grid gadget, sequence of variable gadgets, and sequence of clause gadgets), whose bandwidth is expanded by a constant factor due to the reduction from colored 1-planarity to uncolored 1-planarity used to control the pairs of edges that are allowed to cross. Moreover the pieces can be joined keeping the bandwidth bounded: take an arrangement placing the variable zigzag first, with the green vertices of attachment to the grid at the end, followed with an arrangement of the grid with the green vertices of attachment to the variable zigzag at the beginning and the green vertices of attachment to the clause zigzag at the end, and finish with an arrangement of the clause zigzag. Therefore, the whole graph has bounded bandwidth overall. □

6 Conclusions

We have shown that 1-planarity is fixed-parameter tractable when parameterized by vertex cover number, tree-depth, and cyclomatic number, but that it is NP-complete when parameterized by bandwidth, pathwidth, or treewidth. It is likely that the same approach will also work to show the fixed-parameter tractability of k-planarity for vertex cover number, tree-depth, and cyclomatic number, but we have not worked out the details of such an extension.

One weakness of our algorithms is that their dependence on the parameter is impractically large. It would be of interest to develop parameterized algorithms for this problem whose dependence on the parameter is low enough that they can be run on nontrivial instances.

Acknowledgements

The research of Bannister and Eppstein was supported in part by the National Science Foundation under grants 0830403, 1217322, 1618301, and 1616248, and by the Office of Naval Research under MURI grant N00014-08-1-1015. The research of Cabello was supported in part by the Slovenian Research Agency, program P1-0297, projects J1-4106 and J1-8130, and within the EUROCORES Programme EUROGIGA (project GReGAS) of the European Science Foundation. We also gratefully acknowledge the Slovenian Research Agency for travel funds allowing the authors to meet and perform this research. A preliminary version of this paper was presented at the 13th International Symposium on Algorithms and Data Structures (WADS 2013) and appears in *Lecture Notes in Computer Science* 8037, pp. 97–108.

References

[1] T. Akutsu, M. Hayashida, W.-K. Ching, and M. K. Ng. Control of Boolean networks: Hardness results and algorithms for tree structured networks. *Journal of Theoretical Biology*, 244(4):670–679, 2007. doi:10.1016/j.jtbi.2006.09.023.

[2] C. Auer, F. J. Brandenburg, A. Gleißner, and J. Reislhuber. 1-planarity of graphs with a rotation system. *Journal of Graph Algorithms and Applications*, 19(1):67–86, 2015. doi:10.7155/jgaa.00347.

[3] J. Barát and G. Tóth. Improvements on the density of maximal 1-planar graphs. *Journal of Graph Theory*. In press. doi:10.1002/jgt.22187.

[4] M. A. Bekos, T. Bruckdorfer, M. Kaufmann, and C. Raftopoulou. 1-planar-graphs have constant book thickness. In N. Bansal and I. Finocchi, editors, *Algorithms—ESA 2015, 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 130–141. Springer, 2015. doi:10.1007/978-3-662-48350-3_12.

[5] O. V. Borodin. Solution of the Ringel problem on vertex-face coloring of planar graphs and coloring of 1-planar graphs. *Metody Diskretnogo Analiza*, 41:12–26, 108, 1984.

[6] F. J. Brandenburg, D. Eppstein, A. Gleißner, M. T. Goodrich, K. Hanauer, and J. Reislhuber. On the density of maximal 1-planar graphs. In W. Didimo and M. Patrignani, editors, *Graph Drawing: 20th International Symposium, GD 2012, Redmond, WA, USA, September 19-21, 2012, Revised Selected Papers*, 2013. doi:10.1007/978-3-642-36763-2_29.

[7] S. Cabello and B. Mohar. Adding one edge to planar graphs makes crossing number and 1-planarity hard. *SIAM Journal on Computing*, 42(5):1803–1829, 2013. doi:10.1137/120872310.

[8] J. Chen, I. A. Kanj, and G. Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40-42):3736–3756, 2010. doi:10.1016/j.tcs.2010.06.026.

[9] Z.-Z. Chen and M. Kouno. A linear-time algorithm for 7-coloring 1-plane graphs. *Algorithmica*, 43(3):147–177, 2005. doi:10.1007/s00453-004-1134-x.

[10] D. G. Corneil, H. Lerchs, and L. Stewart Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3(3):163–174, 1981. doi:10.1016/0166-218X(81)90013-5.

[11] J. Czap and D. Hudák. 1-planarity of complete multipartite graphs. *Discrete Applied Mathematics*, 160(4-5):505–512, 2012. doi:10.1016/j.dam.2011.11.014.

[12] P. Damaschke. Induced subgraphs and well-quasi-ordering. *Journal of Graph Theory*, 14(4):427–435, 1990. doi:10.1002/jgt.3190140406.

[13] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999. doi:10.1007/978-1-4612-0515-9.

[14] V. Dujmović, D. Eppstein, and D. R. Wood. Structure of graphs with locally restricted crossings. *SIAM Journal on Discrete Mathematics*, 31(2):805–824, 2017. doi:10.1137/16M1062879.

[15] P. Eades, S.-H. Hong, N. Katoh, G. Liotta, P. Schweitzer, and Y. Suzuki. A linear time algorithm for testing maximal 1-planarity of graphs with a rotation system. *Theoretical Computer Science*, 513:65–76, 2013. doi:10.1016/j.tcs.2013.09.029.

[16] P. Eades and G. Liotta. Right angle crossing graphs and 1-planarity. *Discrete Applied Mathematics*, 161(7-8):961–969, 2013. doi:10.1016/j.dam.2012.11.019.

[17] D. Fernandez-Baca. Allocating modules to processors in a distributed system. *IEEE Transactions on Software Engineering*, 15(11):1427–1436, 1989. doi:10.1109/32.41334.

[18] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. Springer, 2006. doi:10.1007/3-540-29953-X.

[19] S. Földes and P. L. Hammer. Split graphs. In *Proceedings of the Eighth Southeastern Conference on Combinatorics, Graph Theory and Computing (Louisiana State Univ., Baton Rouge, La., 1977)*, volume XIX of *Congressus Numerantium*, pages 311–315, Winnipeg, 1977. Utilitas Math.

[20] A. Grigoriev and H. L. Bodlaender. Algorithms for graphs embeddable with few crossings per edge. *Algorithmica*, 49(1):1–11, 2007. doi:10.1007/s00453-007-0010-x.

[21] J. Guo, R. Niedermeier, and S. Wernicke. Parameterized complexity of generalized vertex cover problems. In F. Dehne, A. López-Ortiz, and J.-R. Sack, editors, *Algorithms and Data Structures: 9th International Workshop, WADS 2005, Waterloo, Canada, August 15-17, 2005, Proceedings*, volume 3608 of *Lecture Notes in Computer Science*, pages 36–48. Springer, 2005. doi:10.1007/11534273_5.

[22] Y. Gurevich, L. Stockmeyer, and U. Vishkin. Solving NP-hard problems on graphs that are almost trees and an application to facility location problems. *Journal of the ACM*, 31(3):459–473, June 1984. doi:10.1145/828.322439.

[23] S.-H. Hong, P. Eades, G. Liotta, and S.-H. Poon. Fáry's theorem for 1-planar graphs. In J. Gudmundsson, J. Mestre, and T. Viglas, editors, *Computing and Combinatorics: 18th Annual International Conference, COCOON 2012, Sydney, Australia, August 20-22, 2012, Proceedings*, volume

7434 of *Lecture Notes in Computer Science*, pages 335–346. Springer, 2012. doi:10.1007/978-3-642-32241-9_29.

[24] H. Kaplan and R. Shamir. Pathwidth, bandwidth, and completion problems to proper interval graphs with small cliques. *SIAM Journal on Computing*, 25(3):540–561, 1996. doi:10.1137/S0097539793258143.

[25] V. P. Korzhik. Minimal non-1-planar graphs. *Discrete Mathematics*, 308(7):1319–1327, 2008. doi:10.1016/j.disc.2007.04.009.

[26] V. P. Korzhik and B. Mohar. Minimal Obstructions for 1-Immersions and Hardness of 1-Planarity Testing. *Journal of Graph Theory*, 72(1):30–71, 2013. doi:10.1002/jgt.21630.

[27] G. L. Miller. Finding small simple cycle separators for 2-connected planar graphs. *Journal of Computer and System Sciences*, 32(3):265–279, 1986. doi:10.1016/0022-0000(86)90030-9.

[28] J. Nešetřil and P. Ossona de Mendez. *Sparsity: Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and Combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.

[29] J. Pach and G. Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17(3):427–439, 1997. doi:10.1007/BF01215922.

[30] G. Ringel. Ein Sechsfarbenproblem auf der Kugel. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 29:107–117, 1965. doi:10.1007/BF02996313.

[31] H. Schumacher. Zur Struktur 1-planarer Graphen. *Mathematische Nachrichten*, 125:291–300, 1986.

[32] S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5(3):269–287, 1983. doi:10.1016/0378-8733(83)90028-X.

[33] Y. Suzuki. Optimal 1-planar graphs which triangulate other surfaces. *Discrete Mathematics*, 310(1):6–11, 2010. doi:10.1016/j.disc.2009.07.016.

[34] R. I. Tyshkevich and A. A. Chernyak. Canonical partition of a graph defined by the degrees of its vertices. *Vestsī Akadèmīī Navuk BSSR, Seryya Fīzīka-Matèmatychnykh Navuk*, 5:14–26, 1979.