Two-Layer Planarization Parameterized by Feedback Edge Set*

Johannes Uhlmann† Mathias Weller‡

December 16, 2013

Abstract

Given an undirected graph G and an integer $k \geq 0$, the NP-hard 2-Layer Planarization problem asks whether G can be transformed into a forest of caterpillar trees by removing at most k edges. 2-Layer Planarization was known to be fixed-parameter tractable with respect to the parameter k. The state of the art is an $O(3.562^k \cdot k + |G|)$-time search tree algorithm and an $O(k)$-size problem kernel. Since transforming G into a forest of caterpillar trees requires breaking every cycle, the size f of a minimum feedback edge set is a natural parameter with $f \leq k$. We improve on previous fixed-parameter tractability results with respect to k by presenting new polynomial-time data reduction rules leading to a problem kernel with $O(f)$ vertices and edges and a new search-tree based algorithm. We expect f to be significantly smaller than k for a wide range of input instances.

1 Introduction

Graphs are an omnipresent concept in all life sciences and it is often important for humans to understand the structure and information that a graph represents. This motivates looking at methods to draw graphs in a human readable form. A possible strategy of drawing graphs in such a way is the "Sugiyama approach" [18, 23] to multilayered graph drawing [5, 11], an important part of which is finding good 2-layered drawings of graphs. In this work, we focus on this problem.

More specifically, we consider an NP-hard graph modification problem that arises in this context, namely 2-Layer Planarization (2LP). In this problem, the task is to make a given graph biplanar by a given (small) number of edge

*A preliminary version appeared in the proceedings of the Annual Conference on Theory and Applications of Models of Computation (TAMC'10), volume 6108 in Lecture Notes in Computer Science, pages 431–442, Springer. Major parts of this work were done while both authors were with Friedrich-Schiller-University of Jena.

†Supported by the DFG, reseach project PABI (NI 369/7).

‡Supported by the DFG, reseach project PABI (NI 369/11).

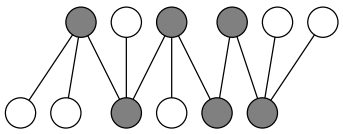

Figure 1: A connected graph that is biplanar and, hence, a caterpillar tree. White vertices are leaves, gray vertices are non-leaves. Note that no vertex has more than two non-leaf neighbors.

deletions. Here, a biplanar graph is a graph that can be drawn in two layers without edge crossings (with edges drawn as straight lines).

It has been shown that biplanar graphs are exactly the graphs that consist of disjoint caterpillar trees (or caterpillars for short) [5, 18]. A caterpillar is a tree such that each of its vertices is adjacent to at most two non-leaf vertices, see Figure 1. This allows us to state the main problem as follows.

> 2-Layer Planarization (2LP):
> **Given:** An undirected graph $G = (V, E)$ and an integer $k \geq 0$.
> **Question:** Is there an edge subset $E' \subseteq E$ with $|E'| \leq k$ such that $(V, E \setminus E')$ is a forest of caterpillars?

Apart from being proposed as an alternative method to minimize crossings [18], solving 2LP is important in DNA mapping [24] and global routing for row-based VLSI layout [17]. It has also been identified as an important special case of the problem to transform a binary matrix into a matrix with the "consecutive ones property" [3] by a minimum number of column removals.

1.1 Previous and Related Work

2-Layer Planarization is NP-hard even in the case that the input graph is bipartite and in one partition each vertex has degree at most two [7] (Note that Eades and Whitesides [7] call the problem Maximum Biplanar Subgraph). Shahrokhi et al. ([21], Section 6) presented a dynamic programming based linear-time algorithm solving the problem on trees. Concerning the parameter k (number of edge deletions), Dujmović et al. [5] showed that 2LP can be solved in $O(k \cdot 6^k + |G|)$ time by devising a search tree algorithm and several polynomial-time data reduction rules leading to a problem kernel with $O(k)$ vertices and edges. Later, Fernau [11] presented a refined search tree for 2LP leading to a running time of $O(k^2 \cdot 5.19276^k + |G|)$. Finally, based on a different branching analysis, Suderman [22] developed an $O(k \cdot 3.562^k + |G|)$-time algorithm.

Dujmović et al. [6] considered a multilayered problem version, where the task is to transform the input graph into a graph that can be drawn in h layers without edge crossings. They present a path decomposition based algorithm that runs in $g(h, k) \cdot |V|$ time (where the function g is not explicitly specified).

2-Layer Planarization is a special case of the problem of transforming a binary matrix into a matrix with so-called "consecutive ones property" by a minimum number of column removals. More specifically, 2LP coincides with this problem for matrices without identical columns that have a maximum of two 1s in each column [3]. Dom et al. [3] showed approximability and fixed-parameter tractability results for these related submatrix problems.

Finally, observe that a caterpillar tree has pathwidth at most one. Thus, 2LP is the problem of transforming a given graph into a graph with pathwidth at most one with edge deletions. Recently, Cygan et al. [2] considered the vertex deletion version of this problem, which they call Pathwidth One Vertex Deletion. They present a quadratic-size problem kernel for the parameter "number of allowed vertex deletions".

1.2 Our Contribution

In this work, we investigate the parameterized complexity of 2LP with respect to the structural parameter "feedback edge set number" f, that is, the minimum number of edges whose removal results in an acyclic graph. Note that the feedback edge set number of a connected n-vertex and m-edge graph G is $f(G) = m - n + 1$ and a minimum feedback edge set can be determined by the computation of a spanning tree in $O(n + m)$ time via depth-first search. We develop efficient preprocessing rules for 2LP that lead to a problem kernel with $O(f)$ vertices and edges. Moreover, we present a new search tree algorithm leading to a total running time of $O(6^f \cdot f^2 + (f + 1) \cdot |G|)$ for solving 2LP.

Our work is motivated as follows.

1. For 2LP the number of necessary edge deletions is at least the feedback edge set number, since one has to destroy all cycles to obtain a forest of caterpillars. In this sense, we improve on the results of Dujmović et al. [5] by providing fixed-parameter algorithms and kernelizations with about the same worst-case bounds for a parameter that we expect to be significantly smaller for a wide range of input instances. For large values of k and small values of f, our search tree algorithm may in fact outperform the state of the art $O^*(3.652^k)$-time algorithm by Suderman [22].

2. Dujmović et al. [5] pointed out that "instances of 2-Layer Planarization for *dense* graphs are of little interest from a practical point of view" since the resulting drawings are unreadable anyway. Thus, they expect the solution size to be small in practice. This is even more plausible for the feedback edge set number of a graph which is directly linked to the number of edges, and, hence, the sparseness of the graph. Also note that the solution size can be arbitrarily large even for trees (the sparsest connected graphs) while the feedback edge set number of trees is zero. Measuring the distance from trees by the feedback edge set number can be seen as a parameterization by "distance from triviality" [14]. In this sense, our results generalize the linear-time algorithm for trees by Shahrokhi et al. [21].

3. The feedback edge set number f is a parameter that can easily be computed in advance and, hence, allows for a meta-algorithm that chooses an algorithm for a given input by computing an estimation on the running time prior to running the algorithm for the problem itself. Since the parameter k ("number of edge deletions") is NP-hard to compute, such an algorithm could not efficiently determine the running time of an algorithm parameterized by k in advance.

4. Fourth, looking for smaller parameters may help to further extend the range of solvable instances. However, this seems only possible if the new algorithms have a modest exponential part of the running time. This should also be seen in the light of the following. One of the most prominent structural parameters in parameterized algorithmics for graph problems is the treewidth of the input graph. When parameterized by treewidth, many graph problems are fixed-parameter tractable, but there are also problems that are W[1]-hard with respect to treewidth or for which the computational complexity for bounded treewidth is open. As proposed by Fellows et al. [8], parameters that are lower-bounded by the treewidth (the vertex cover number of a graph is an example) have recently been considered to establish fixed-parameter tractability for such problems [10, 15, 16]. In the spirit of multivariate algorithmics [9, 20], this work initiates the study of 2-Layer Planarization for new parameterizations.

5. Last but not least, efficient preprocessing or polynomial-time data reduction seems to be essential to obtain good fixed-parameter algorithms. Indeed, kernelization has been recognized as one of the key techniques of parameterized algorithmics [1, 13, 19]. In this context, one of our main contributions is to provide a set of new polynomial-time data reduction rules for 2-Layer Planarization.

1.3 Organization of the Paper

This work is organized as follows. After introducing basic notations and definitions in Section 2, we present in Section 3 several polynomial-time data reduction rules leading to a linear-size problem kernel with respect to f. Based on this kernelization, we present a search-tree algorithm for 2LP that runs in $O(6^f \cdot f^2 + (f+1) \cdot |G|)$ time in Section 4.

2 Preliminaries.

Let G be a graph. We denote the set of all vertices of G by $V(G)$ and the set of all edges of G by $E(G)$. We abbreviate $|V| + |E|$ to $|G|$. For every vertex set $V' \subseteq V(G)$, we denote the subgraph of G that is induced by V' by $G[V']$ and we write $G - V'$ for $G[V(G) \setminus V']$. Equivalently, for every edge-set $S \subseteq E(G)$, consider $G - S$ an abbreviation for $(V(G), E(G) \setminus S)$ and let $V(S)$ denote the set of endpoints of edges in S. We denote the neighborhood of a vertex $v \in V(G)$

in G with $N_G(v)$ and the degree of v in G with $\deg_G(v)$. If clear from the context, we omit the index. Furthermore, let $I(G)$ (isolated vertices) and $L(G)$ (leaves) denote the set of vertices in G with degree zero and one, respectively. For a vertex v in G, let $L(v) := L(G) \cap N(v)$. Following [5], we define the *non-leaf degree* $\widehat{\deg}_G(v) := |N_G(v) \setminus L(G)|$ for every vertex $v \in V(G)$.

A tree is a *caterpillar tree* (or caterpillar for short) if each of its vertices has non-leaf degree at most two. Equivalently, a caterpillar is a tree that does not contain a 2-claw [7] (see Figure 2*a*)). Thus, unions of caterpillars have the following forbidden-subgraph characterization. A graph is a forest of caterpillars if and only if it is acyclic and does not contain a 2-claw as subgraph. Note that for a caterpillar tree, the non-leaf degree of a vertex is at most 2. A leaf $v \in L(G)$ is called *critical* if its only neighboring vertex has non-leaf degree two. The definition of critical vertices is motivated by the observation that being a caterpillar is invariant with respect to adding neighbors to non-critical vertices. In contrast, adding a neighbor to a critical vertex creates a 2-claw.

Informally speaking, G^* denotes the subgraph of G that contains all edges that are contained in a cycle or that connect cycles. Formally, set $G^0 := G$ and recursively define $G^{i+1} := G^i - (L(G_i) \cup I(G_i))$. Finally, let G^* denote the graph G^i with minimum i such that $G^i = G^{i+1}$. We will call G^* the *cyclic core* of G. Note that G^* is the empty graph if and only if G is acyclic (a forest). Moreover, for G being a forest of caterpillar trees, G^1 is a forest of paths. Furthermore, note that $G - V(G^*)$ is acyclic (a forest). For a vertex $v \in V(G^*)$ let T^v denote the tree of $G - E(G^*)$ that is rooted at v. The tree T^v is called the *pendant tree* of v and v is called its *connection point*. Furthermore, for a rooted tree T and a vertex $x \in T$ let T_x denote the subtree of T rooted at x.

The following special (pendant) trees are of particular interest in this work. A path $p = (\{v, w\}, \{w, x\})$ is called a P_2 with connection point v if $\deg(v) \geq 2$, $\deg(w) = 2$, and $\deg(x) = 1$, see Figure 2*b*) for an example. Vertex w is called the middle point and we refer to it as $m(p)$ and vertex x is called the leaf of p denoted by $l(p)$. For a vertex v let $\mathcal{P}_2(v)$ denote the set of all P_2s that have v as their connection point. A Y-graph with connection point v is a subgraph consisting of the adjacent vertices v and w and two P_2s with connection point w. Optionally, w may additionally be adjacent to a leaf (see Figure 2c)). Herein, w is called the center point of the Y-graph. We refer to w by $c(Y)$. Let $\mathcal{Y}(v)$ denote the set of all Y-graphs that have v as their connection point.

Our results are in the context of parameterized complexity, which is a two-dimensional framework for studying computational complexity [4, 12, 19]. One dimension is the input size n, and the other one is the *parameter* (usually a positive integer). A problem is called *fixed-parameter tractable* (fpt) with respect to a parameter k if it can be solved in $g(k) \cdot n^{O(1)}$ time, where g is a computable function only depending on k. A core tool in the development of fixed-parameter algorithms is polynomial-time preprocessing by *data reduction*. Here, the goal is to transform a given problem instance x with parameter k into an equivalent instance x' with parameter $k' \leq k$ such that the size of x' is

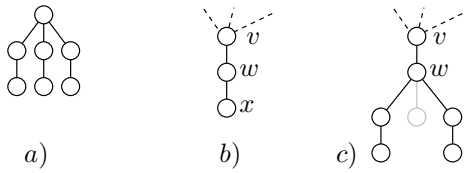

Figure 2: Terminology. *a*) A 2-claw. The degree-three vertex of a 2-claw is called its center vertex. Caterpillars can be characterized as graphs containing neither cycles nor 2-claws. Figure *b*) shows a P_2 pendant with connection point v and Figure *c*) shows a Y-graph with connection point v and center point w. In *c*) the gray leaf may or may not be present in the Y-graph (formally, there are two different Y-graphs, one with and one without the gray leaf).

upper-bounded by some function only depending on k. This is usually achieved by applying data reduction rules. The whole process is called *kernelization*. We call a data reduction rule *correct* if the new instance after an application of this rule is a yes-instance if and only if the original instance is a yes-instance. An instance is called *reduced* with respect to a set of data reduction rules if none of these data reduction rules can be applied to the instance.

3 Kernelizing 2-Layer Planarization

In this section, we present a kernelization for 2LP parameterized by f, denoting the size of a minimum feedback edge set of the input graph. We present a number of polynomial-time executable data reduction rules and show that a graph that is reduced with respect to these rules cannot contain more than $O(f)$ vertices and edges. The kernelization consists of two phases. In the first phase, which we call "tree reduction", roughly speaking, the goal is to reduce the "acyclic part" of the input graph. In the second phase, the goal is to reduce the long non-branching paths in the remaining "cyclic core" G^*, shrinking its size to a linear function in f. We call the second phase "path reduction".

3.1 Tree Reduction

Subsequently, we present data reduction rules for repeatedly replacing a pendant tree T^u for some $u \in V(G^*)$ with a smaller tree, until its size is a constant value. For the explanation of the basic idea, consider a 2-claw C in T^u that has "maximal depth", that is, the sum of distances of the vertices of C to u is maximal. Clearly, all vertices that are "below" C but not contained in C are irrelevant because they are not contained in any 2-claw in T^u. Moreover, since there is no 2-claw with larger depth than C, all 2-claws that intersect C share one of the "highest" edges with C. Hence, it is optimal to destroy C by deleting one of its "highest" edges. Distinguishing all relevant cases, this thought leads to the following five data reduction rules (see Figure 3 for an illustration).

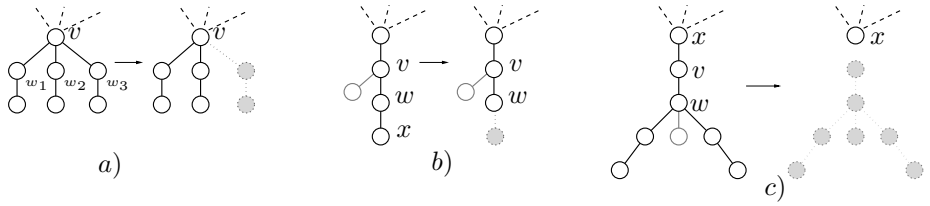

Figure 3: *a*) An example for the application of Tree Reduction Rule 3. Note that we must delete one of the edges $\{v, w_i\}$, $i = 1, 2, 3$ in order to destroy the 2-claw centered at v. By symmetry, there is an optimal solution that contains the edge $\{v, w_3\}$. After deleting $\{v, w_3\}$, we can delete the gray vertices because they form an isolated caterpillar. *b*) An example for the application of Tree Reduction Rule 4. Note that the edge $\{w, x\}$ is not contained in any 2-claw and hence, x can be deleted. *c*) An example for the application of Tree Reduction Rule 5. Since the deletion of the edge $\{x, v\}$ destroys the same 2-claws as the deletion of any other edge in the tree rooted at x, there is an optimal solution that contains $\{x, v\}$.

Tree Reduction Rule 1. *Let v be a vertex with $|L(v)| \geq 2$, Then, delete all but one leaf in $L(v)$.*

Tree Reduction Rule 2. *Let v be a vertex with $|\mathcal{Y}(v)| \geq 1$ and $|\mathcal{Y}(v)| + |L(v)| + |\mathcal{P}_2(v)| \geq 2$. Then, for an arbitrarily chosen Y-graph $Y \in \mathcal{Y}(v)$, delete all vertices of Y except for v and decrease k by one.*

Tree Reduction Rule 3. *Let v be a vertex with $|\mathcal{P}_2(v)| \geq 3$ and let $\mathcal{P}_2(v) = \{p_1, p_2, \ldots, p_q\}$. Then, delete the vertices $l(p_i)$ and $m(p_i)$ for $3 \leq i \leq q$ and decrease k by $q - 2$.*

Tree Reduction Rule 4. *Let v be a vertex with $\widehat{\deg}_G(v) = 2$ and let p be a P_2 with $\mathcal{P}_2(v) = \{p\}$. Then, delete the vertex $l(p)$.*

Tree Reduction Rule 5. *Let v be a vertex with $\deg_G(v) = 2$ and let Y be a Y-graph such that $\mathcal{Y}(v) = \{Y\}$. Then, delete all vertices of Y (including v) and decrease k by one.*

The last data reduction rule for the tree reduction phase is obvious.

Tree Reduction Rule 6. *Let C be a connected component of G that is a caterpillar. Then, delete all vertices of C.*

Lemma 1. *Tree Reduction Rules 1–6 are correct. An instance reduced with respect to Tree Reduction Rules 1–6 can be computed in $O(|G|)$ time.*

Proof. First, we prove the correctness of the tree reduction rules except Tree Reduction Rules 1 and 6 (Tree Reduction Rule 1 is proven in [5], Tree Reduction Rule 6 is obvious), then the claimed running time.

Tree Reduction Rule 2 is correct. First, note that we can assume that, if a solution contains any edge from Y, then it contains the edge $e := \{v, c(Y)\}$ (since its deletion destroys also all the 2-claws that can be destroyed by any other edge in Y). We show that an optimal solution that contains e exists. Let S be an optimal solution with $e \notin S$. Clearly, this solution must contain all edges incident to v except for the edge $\{v, c(Y)\}$. Since $L(v) \neq \varnothing$ or $\mathcal{P}_2(v) \neq \varnothing$ or $|\mathcal{Y}(v)| \geq 2$, solution S must contain an edge e' that is incident to a leaf, to the middle point of a P_2, or to the center point of a Y-graph other than Y. Thus, $S \setminus \{e'\} \cup \{e\}$ clearly is a solution of same size containing e.

Tree Reduction Rule 3 is correct. To destroy all the 2-claws in the subgraph induced by the vertices of the P_2s in $\mathcal{P}_2(v)$ we must delete $q-2$ edges. Since, for a $p \in \mathcal{P}_2(v)$, it is at least as good to delete edge $\{v, m(p)\}$ as to delete edge $\{m(p), l(p)\}$, by symmetry one can choose the edges $\{v, m(p_i)\}$, $3 \leq i \leq l$.

Tree Reduction Rule 4 is correct. Let G' denote the instance that is reduced with respect to Tree Reduction Rule 4. Since $\widehat{\deg}_{G'}(v) = 1$, we know that for every solution S' for G', the vertex $m(p)$ is non-critical in $G' - S'$. Consequently, all solutions for G' are also solutions for G.

Tree Reduction Rule 5 is correct. Let x denote the single vertex in $N(v) - c(Y)$. Note that the Y-graph Y together with x forms a graph containing a single 2-claw. Clearly, this 2-claw is best destroyed by deleting the edge $\{x, v\}$ since this also destroys all the 2-claws that can be destroyed by deleting any other edge in Y.

To apply the tree reduction rules efficiently, we proceed as follows. First, in a depth-first traversal of the graph, we determine the vertices in G^*. Moreover, for every vertex, we remember its parent in the spanning tree corresponding to the depth-first traversal. Then, we consider the vertices in a postorder. Doing so, for every vertex, we maintain three lists, $L(v)$, $\mathcal{P}_2(v)$, and $\mathcal{Y}(v)$. If we consider a leaf x, we add x to $L(p)$, where p is the parent of x. If we consider an inner vertex x, we can assume that all children have already been considered because we traverse the instance bottom-up. Thus, the lists $L(x)$, $\mathcal{P}_2(x)$, and $\mathcal{Y}(x)$ have already been determined and we have all information at hand to decide whether one of the tree reduction rules can be applied to x. After applying the tree reduction rules to x, we have the following situation. If we decide that the edge from x to its parent is deleted (Tree Reduction Rules 4 and 5), then x does not have any effect on its parent. Otherwise, if neither Tree Reduction Rule 4 nor Tree Reduction Rule 5 applies to x, then we have to consider the following cases. If x is a leaf, we add x to the list L of its parent. Otherwise, if x has a single child that is a leaf, we add x to the list $\mathcal{P}_2$ of the parent of x. Note that one of these cases must apply to x. Thus, for a vertex v, all changes can be made in $O(d(v))$ time, where $d(v)$ denotes the number of neighbors of v in $V \setminus V(G^*)$. Hence, the presented tree reduction rules can be exhaustively applied in $O(|G|) + O(\sum_{v \in V} d(v)) = O(|G|)$ time. □

The structure of an instance reduced with respect to these reduction rules is described by the following lemma, which concludes the presentation of the "tree reduction".

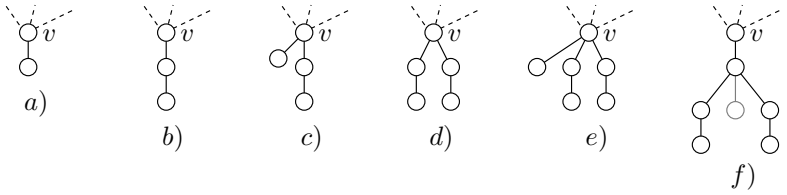

Figure 4: In a reduced instance, the pendant tree T^v of a vertex $v \in V(G^*)$ is isomorphic to one of the trees shown in a) to f). Note that the tree shown in f) is exactly the Y-graph as defined in Figure 2b).

Lemma 2. *In an instance that is reduced with respect to the tree reduction rules, for each vertex $v \in V(G^*)$, its pendant tree T^v is either a singleton or isomorphic to one of the trees shown in Figure 4.*

Proof. To prove the lemma, we consider the height of the pendant tree T^v.

If T^v has height one, then all children of v are leaves and since the instance is reduced with respect to Tree Reduction Rule 1, v has exactly one child. That is, T^v is isomorphic to the tree shown in Figure 4a).

If T^v has height two, then, since the instance is reduced with respect to Tree Reduction Rule 1, every non-leaf child of v has degree two and v has at most one leaf child. Moreover, v has at most two non-leaf children since otherwise $|\mathcal{P}_2(v)| \geq 3$ and we could apply Tree Reduction Rule 3. Moreover, since $|L(v)| \leq 1$, the pendant tree T^v is isomorphic to one of the trees shown in Figure 4b–e).

To prove the remaining case, we need the following observation. Let $v' \in V(T^v) - \{v\}$ be a vertex such that the height of $T^v_{v'}$ (the subtree of T^v rooted at v') is two. Since the height of T^v is neither one nor two, v' exists. We show that v' is the center of a Y-graph with connection point v. Since the height of $T^v_{v'}$ is two, it is isomorphic to either d) or e). First, note that if v' had more than two non-leaf children, then, with the same argument as in the second case above, we could apply Tree Reduction Rule 3. Moreover, if v' had exactly one non-leaf child, then $\widehat{\deg}_G(v') = 1$ and we could apply Tree Reduction Rule 4. Thus, v' has exactly two non-leaf children and at most one leaf child.

If T^v has height three, then let W denote the set of children of v such that the tree T^v_w has height two for every $w \in W$. By the above observation, we know that for every $w \in W$ the tree T^v_w together with v forms a Y-graph. Thus, $|W| = 1$, since otherwise we could apply Tree Reduction Rule 2. Moreover, we know that v cannot have other children, since otherwise we could apply Tree Reduction Rule 2. Hence, T^v is a Y-graph.

Finally, we show by contradiction that the height of T^v is at most three. Assume that the height of tree T^v is at least four and let w be some vertex in T^v such that T^v_w has height three. With the same argument as in the previous case, we can assume that T^v_w is a Y-graph and w has exactly one child. Then, however, the degree of w is two and we can apply Tree Reduction Rule 5. □

3.2 Path Replacement

The tree reduction rules presented in the previous subsection are not sufficient to yield a problem kernel for our parameterization. For example, if the input graph G is a simple cycle, then none of the above data reduction rules applies. Recall the notions of G^* (also called the "cyclic core" of G) and pendant trees. The purpose of the data reduction rules presented in this subsection is to reduce non-branching paths of G^* (hence, they are called "path reduction rules"). The first two reduction rules take care of paths containing Y-graphs as pendant trees. Then, we introduce the notion of "tokens" which allows us to handle the remaining cases in a unified manner.

In the following, we assume the input to be reduced with respect to all tree reduction rules (see previous subsection). Consider vertices $u, w \in V(G^*)$ that may be identical. We denote a path $P_{u,w} = (u = v_0, v_1, \ldots, v_\ell, v_{\ell+1} = w)$ between u and w as *degree-2 path* if $\deg_{G^*}(v_i) = 2$ for all $1 \leq i \leq l$. Its *length* is $\ell + 2$. We refer to the vertices v_i, $1 \leq i \leq l$, as *inner path vertices*. Furthermore, denote the edges $\{v_{i-1}, v_i\}$ by e_i for all $1 \leq i \leq \ell+1$. Throughout this subsection, let P be some degree-2 path in the given graph G and let v_i be some inner path vertex of P. In this context, let $T_P := G[\bigcup_{i=1}^{l} V(T^{v_i}) \cup \{u, w\}]$ and for $1 \leq i \leq j \leq l$, let $T_{i,j}$ denote the subtree of T_P containing all vertices reachable from v_i in $T_P - \{e_i, e_{j+1}\}$. Note that $T_{i,i} = T^{v_i}$.

The first two path reduction rules handle all Y-graphs that have a vertex on a degree-2 path as their connection point. For their presentation, we need two additional tools. First, observe that for any Y-graph Y with connection point v, deleting the edge of Y that is incident to v is at least as good as deleting any combination of edges of Y. This allows us to assume optimal solutions to contain either this edge or all other edges incident to v. Furthermore, since pendant trees do not overlap, we can assume this for all vertices v.

Observation 1. *There is an optimal solution S for G such that for each Y-graph Y with connection point v, it holds that $S \cap E(Y) \subseteq \{\{v, c(Y)\}\}$.*

Second, observe that we can assume certain vertices to be non-critical in the target caterpillar-forest corresponding to an optimal solution.

Lemma 3. *Let v be a degree-2 vertex in G^* such that T^v is neither a singleton nor a Y-graph. Then, there is an optimal solution S for G such that v is non-critical in $G - S$.*

Proof. Let S denote an optimal solution for G such that v is critical in $G - S$. We show that there is a solution S' with $|S'| = |S|$ such that v is non-critical in $G - S'$. Since v is critical in $G - S$, by definition, v is a leaf in $G - S$ and adjacent to a vertex x with non-leaf degree two in $G - S$. Since T^v is not a Y-graph, Lemma 2 implies that every vertex in $V(T^v)$ (except for v) has non-leaf degree at most one in G and, hence, also in $G - S$. This implies that $x \notin V(T^v)$. Let $F := E(G) \cap \{\{v, z\} \mid z \in V(T^v)\}$. Since v is a leaf adjacent to x in $G - S$, we have $F \subseteq S$ and since T^v is not a singleton, we know that $|F| \geq 1$. Observe that deleting $\{v, x\}$ from $G - S$ renders v isolated.

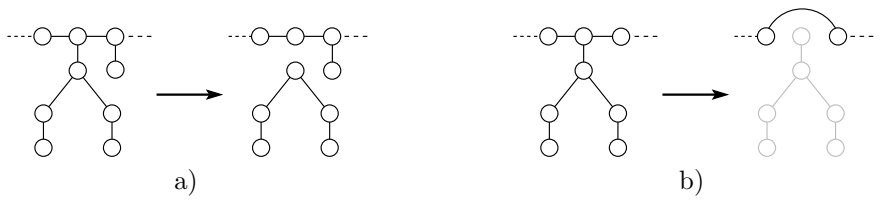

Figure 5: Path Reduction Rules 1 and 2: a) shows an application of case 1 of Path Reduction Rule 1 while b) shows an application of case 2 of Path Reduction Rule 2. Gray subgraphs are deleted by the reduction rules.

Thus, subsequently undoing the edge deletions in F does not create a cycle or 2-claw. Hence, $S' := (S \setminus F) \cup \{\{v, x\}\}$ is also a solution for G. Finally, note that $G - S'$ contains T^v as a connected component. Since T^v is not a Y-graph, v is non-critical in $G - S'$. □

In the proof of Lemma 3, the only pendent tree affected by the modifications in S is T^v, which, by assumption, is not a Y-graph. Hence, we can apply Lemma 3 and Observation 1 independently. We use such an optimal solution to prove the correctness of the next reduction rule.

The first path reduction rule describes Y-graphs Y with connection point v_i for which it is optimal to delete the edge $\{v_i, c(Y)\}$ (see also Figure 5).

Path Reduction Rule 1. *Let P denote a degree-2 path and let both v_i and v_{i+1} be inner path vertices of P such that $Y := T^{v_i}$ is a Y-graph. If*

1. *$T^{v_{i+1}}$ is neither a singleton nor a Y-graph, or*
2. *v_{i+2} is an inner path vertex, $\deg_G(v_{i+1}) = 2$, and $T^{v_{i+2}}$ is either a singleton or a Y-graph,*

then delete $\{v_i, c(Y)\}$ and decrease k by one.

Lemma 4. *Path Reduction Rule 1 is correct.*

Proof. For the correctness, we show that if Path Reduction Rule 1 applies to v_i, then there is an optimal solution for G that contains $\{v_i, c(Y)\}$ and hence, it is safe to remove edge $\{v_i, c(Y)\}$. In the following, let $e_Y := \{v_i, c(Y)\}$ and let S denote an optimal solution for G as described by Observation 1 and Lemma 3. That is, S is an optimal solution such that

1. for each Y-graph Y' with connection point v', $S \cap E(Y') \subseteq \{\{v', c(Y')\}\}$, and
2. if $T^{v_{i+1}}$ is neither a singleton nor a Y-graph, then v_{i+1} is non-critical in $G - S$.

As discussed above, such a solution exists.

If S contains e_Y, then we are done, hence, in the following, we assume that $e_Y \notin S$. Then, by Property 1, S does not contain any edge of Y, implying $e_i, e_{i+1} \in S$. If v_{i+1} is non-critical in $G - S$, then $S \setminus \{e_{i+1}\} \cup \{e_Y\}$ is an optimal solution for G containing e_Y. To show that v_{i+1} is non-critical in $G-S$, assume that v_{i+1} is critical in $G - S$. In Case 1 of Path Reduction Rule 1, this contradicts Property 2. In Case 2 of Path Reduction Rule 1, $\deg_G(v_{i+1}) = 2$ and $T^{v_{i+2}}$ is either a singleton or a Y-graph. Since $e_{i+1} \in S$, the fact that v_{i+1} is critical in $G - S$ implies that v_{i+2} is adjacent to v_{i+1} in $G - S$ and $\widehat{\deg}_{G-S}(v_{i+2}) \geq 2$. If $T^{v_{i+2}}$ is a Y-graph, then, since S does not contain $\{v_{i+1}, v_{i+2}\}$, Property 1 implies that S contains the edge $\{v_{i+2}, c(T^{v_{i+2}})\}$. Hence, v_{i+2} has at most two neighbors in $G - S$, regardless of whether $T^{v_{i+2}}$ is a Y-graph or a singleton. However, since one of these neighbors is v_{i+1} (a leaf), v_{i+2} has non-leaf degree at most one in $G - S$, a contradiction to v_{i+1} being critical in $G - S$. □

The second data reduction rule handles almost all remaining cases where a Y-graph occurs as the pendant tree of some inner path vertex v_i of P by "bypassing" v_i in P.

Path Reduction Rule 2. *Let P denote a degree-2 path and let both v_i and v_{i+1} be inner path vertices of P such that $Y := T^{v_i}$ is a Y-graph. If*

1. *$T^{v_{i+1}}$ is a Y-graph, or*
2. *v_{i-1} and v_{i+1} have degree two, or*
3. *v_{i+2} is an inner path vertex, $\deg(v_{i+1}) = 2$, and $T^{v_{i+2}}$ is neither a singleton nor a Y-graph,*

then remove all vertices of Y from G, insert the edge $e = \{v_{i-1}, v_{i+1}\}$, and decrease k by one.

Lemma 5. *Path Reduction Rule 2 is correct.*

Proof. Let G' denote the graph that results from applying Path Reduction Rule 2 to some v_i in G and let $e_Y := \{v_i, c(Y)\}$. Let $\widehat{G}$ denote the result of contracting e_{i+1} in $G - \{e_Y\}$ and observe that $\widehat{G}$ is identical to G' with the exception of one connected component (containing all vertices of Y but v_i) which is a caterpillar. Consequently, $\widehat{G}$ and G' are equivalent in the sense that a solution for one is also a solution for the other (considering that e_i in $\widehat{G}$ plays the role of e in G'). In the following, we show that G has a solution S of size at most k if and only if G' has a solution S' of size at most $k - 1$.

"⇒:" By Observation 1, either e_Y or both e_i and e_{i+1} are in S. If $e_Y \in S$, then, since contracting an edge does not create 2-claws or cycles, $S' := S \setminus \{e_Y\}$ is a solution of size at most $k - 1$ for $\widehat{G}$ and, thus, for G'. Otherwise, $e_i, e_{i+1} \in S$. However, by construction, $G' - \{e\}$ is a subgraph of $G - \{e_i, e_{i+1}\}$ and thus, $S' := S \setminus \{e_i, e_{i+1}\} \cup \{e\}$ is a solution for G' of size at most $k - 1$.

"⇐:" If a solution S' for G' of size at most $k - 1$ contains e, then the equivalent solution $\widehat{S}$ for $\widehat{G}$ contains e_i, and clearly, $S := \widehat{S} \cup \{e_{i+1}\}$ is a solution

of size at most k for G. Thus, in the following, we assume that there is no solution for G' of size at most $k-1$ that contains e (in particular, $e \notin S'$ and, thus, v_{i-1} and v_{i+1} are neighbors in $G' - S'$).

Observe that the subdivision of an edge e' of a caterpillar can only create a 2-claw if e' is incident to a critical vertex. Hence, if v_{i-1} and v_{i+1} are both non-critical in $G' - S'$, then we can subdivide e without affecting the solution and thus, $S := S' \cup \{e_Y\}$ is a solution of size at most k for G. Hence, in the following, we assume that v_{i-1} or v_{i+1} is critical in $G' - S'$. Since v_{i-1} and v_{i+1} are adjacent in $G' - S'$, this implies that one of them has degree at least three in $G' - S'$ while the other is a leaf in $G' - S'$. In the following, we consider the three cases of Path Reduction Rule 2 separately.

Case 1: The first condition of Path Reduction Rule 2 applies.
Then, $Y' := T^{v_{i+1}}$ is a Y-graph. Let $e_{Y'} := \{v_{i+1}, c(Y')\}$. By Observation 1, we can assume that S' contains either $e_{Y'}$ or both e and e_{i+2} (if S' contains $e_{Y'}$ and e but not e_{i+2}, then we can exchange $e_{Y'}$ for e_{i+2} in S'). However, by the assumption that $e \notin S'$, it follows that $e_{i+2} \notin S'$ and $e_{Y'} \in S'$, implying $\deg_{G'-S'}(v_{i+1}) = 2$. Thus, neither v_{i+1}, nor v_{i-1} is critical in $G' - S'$, a contradiction.

Case 2: The second condition of Path Reduction Rule 2 applies.
Then, $\deg_G(v_{i-1}) = \deg_G(v_{i+1}) = 2$, implying that neither v_{i-1} nor v_{i+1} has degree at least three in $G' - S'$, contradicting the assumption that v_{i-1} or v_{i+1} is critical.

Case 3: The third condition of Path Reduction Rule 2 applies.
By Lemma 3 we can assume that v_{i+2} is non-critical in $G' - S'$. Clearly, v_{i-1} is non-critical in $G' - S'$ since $\deg_G(v_{i+1}) = 2$. Hence, v_{i+1} is critical in $G' - S'$ and thus, $S' \cup \{e\}$ isolates v_{i+1}. Since v_{i+2} is non-critical in $G' - S'$, it follows that $(S' \cup \{e\}) \setminus \{e_{i+2}\}$ is a solution of size at most k for G' containing e, a contradiction. □

The two path reduction rules presented so far eliminate Y-graphs in all long degree-2 paths. In the following, consider P to be *Y-graph-free*, that is, P does not contain an inner path vertex whose pendant tree is a Y-graph. Consider a graph that is reduced with respect to Path Reduction Rules 1 and 2. All degree-2 paths P' that are not Y-graph-free contain at most two inner path vertices, whose pendant trees are a singleton and a Y-graph, respectively. Thus, it is clear that $|V(T_{P'})| \leq 10$. In the following, we focus on Y-graph-free degree-2 paths. In this case, we can show that we do not need to consider deleting edges in pendant trees, thus allowing us to restrict our attention to deleting edges on the degree-2 path.

Lemma 6. *Let P be a Y-graph-free degree-2 path. Then, there is an optimal solution for G that does not contain any edge of $E(T_P) \setminus E(P)$.*

Proof. Consider a solution S for G that contains an edge $e \in E(T_P) \setminus E(P)$. Clearly, we can assume that e is incident to $v_i \in P$. If any two edges e and e' that are incident to v_i are in S, then we can exchange e and e' for e_i and e_{i+1} in S. Otherwise, we show that deleting one of e_i, e_{i+1} destroys all 2-claws

that are destroyed by the deletion of e. Assume that this is false. Then, the deletion of e destroys a 2-claw that does not contain e_i and a different 2-claw that does not contain e_{i+1}. Since P is Y-graph-free, the pendant tree $T_{i,i}$ with connection point v_i contains two P_2s. However, only one of these P_2s can be destroyed by deleting e and, hence, we can exchange e for either e_i or e_{i+1}, a contradiction. □

In order to handle the remaining degree-2 paths (recall the definition on page 10) in a unified manner, we introduce the notion of "tokens" as sets of consecutive edges on the degree-2 paths. Consider a vertex v on a degree-2 path P that is the center of a 2-claw. Then, Lemma 6 tells us that this 2-claw must be destroyed by deleting an edge of P. We model this fact by letting v generate a token containing all edges that may be deleted in order to destroy this 2-claw. The introduction of this notion is split into three parts. First, we define tokens, second, we point out how they are generated, and third, we specify what it means to destroy a token.

In the following, a vertex v in P is called *crossable* if $\deg_G(v) = 2$. A *token* K of P is a set of at most four consecutive edges of P. Let v_i be a vertex in P. If $i \leq \ell - 1$, then the *upper token* $K^{\mathrm{up}}(v_i)$ of v_i is $\{e_{i+1}, e_{i+2}\}$ if v_{i+1} is crossable and it is $\{e_{i+1}\}$, otherwise. Equivalently, if $i \geq 2$, then the *lower token* $K^{\mathrm{low}}(v_i)$ of v_i is $\{e_{i-1}, e_i\}$ if v_{i-1} is crossable and it is $\{e_i\}$ otherwise. In this sense, we say that tokens can only "span" over crossable vertices. For the vertices u, v_1, v_ℓ, and w, we need the following auxiliary tokens: $K^{\mathrm{low}}(u) := K^{\mathrm{up}}(w) := \{\diamond\}$, $K^{\mathrm{low}}(v_1) := \{\diamond, e_1\}$, and $K^{\mathrm{up}}(v_\ell) := \{e_{\ell+1}, \diamond\}$.

Each inner path vertex v_i of P for which $\mathcal{P}_2(v_i) \neq \varnothing$ *generates* tokens in the following way: If $|\mathcal{P}_2(v_i)| = 1$ (in this case T^{v_i} is isomorphic to one of the trees shown in Figure 4 *b*) and *c*)), then v_i generates one token $K^{\mathrm{up}}(v_i) \cup K^{\mathrm{low}}(v_i)$. If $|\mathcal{P}_2(v_i)| = 2$ (in this case T^{v_i} is isomorphic to one of the trees shown in Figure 4 *d*) and *e*)), then v_i generates two tokens, $K^{\mathrm{up}}(v_i)$ and $K^{\mathrm{low}}(v_i)$. We define $\mathcal{K}(v_i)$ as the set of tokens generated by v_i and $\mathcal{K}(P)$ as the set of all tokens generated by inner path vertices of P. See Figure 6 for an illustration. We can observe that only vertices that are centers of 2-claws generate tokens. Moreover, all 2-claws centered at inner path vertices are represented in the tokens of $\mathcal{K}(P)$, implying that all 2-claws on P can be destroyed by removing an edge of each token in $\mathcal{K}(P)$.

Observation 2. *For each $K \in \mathcal{K}(P)$, there is a 2-claw C centered at the vertex that generates K such that $E(C) \cap E(P) = K$ Furthermore, for each 2-claw C that is centered at an inner path vertex of P, there is a token $K \in \mathcal{K}(P)$ such that K is a subset of all edges that are in both C and P.*

A non-auxiliary token (token that does not contain $\diamond$) $K \in \mathcal{K}(P)$ is *destroyed* by an edge set E' if $K \cap E' \neq \varnothing$. Observe that if K contains $\diamond$, then it either contains e_1 or e_ℓ. We say that a token containing e_1 and $\diamond$ is destroyed by E' if either $K \cap E' \neq \varnothing$ or E' contains all edges incident to v_0 except for e_1. An auxiliary token containing $e_{\ell+1}$ is destroyed analogously. Informally speaking, a

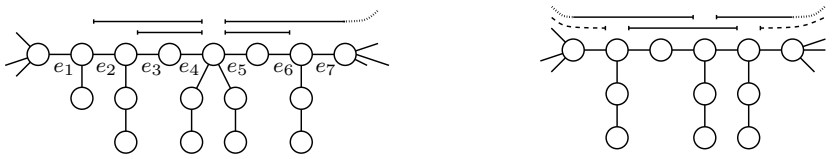

Figure 6: *a*) Illustration of the tokens generated by a degree-2 path. The tokens are depicted by horizontal bars. That is, the tokens are $\{e_2, e_3, e_4\}$, $\{e_3, e_4\}$, $\{e_5, e_6\}$, and $\{e_5, e_6, e_7, \diamond\}$. *b*) An example of a chained degree-2 path. The tokens are depicted by horizontal bars. Furthermore, dashed lines represent auxiliary tokens.

token represents the need to delete an edge. By Lemma 6 it suffices to consider edges in P. Thus, the task is to destroy all tokens by deleting only few edges.

The following path reduction rules are designed to shrink degree-2 paths and rely heavily on the notion of tokens. The first data reduction rule reduces degree-2 paths that do not contain any tokens.

Path Reduction Rule 3. *If there is a degree-2 path P with $|V(T_P)| > 7$ and $\mathcal{K}(P) = \varnothing$, then contract $T_{2,\ell-1}$ to a single vertex.*

Lemma 7. *Path Reduction Rule 3 is correct.*

Proof. Let G denote the input graph and G' the graph that results from applying Path Reduction Rule 3 to a degree-2 path P in G. We show that for each solution S for G there is also a solution for G' of the same size, and vice versa.

By Observation 2, $\mathcal{K}(P) = \varnothing$ implies that there is no 2-claw centered at an inner path vertex of P. However, no other 2-claw can intersect $T_{2,\ell-1}$ and, thus, the set of edges that are in 2-claws in G and in G' are equal, implying that $G - S$ contains 2-claws if and only if $G' - S$ does.

Observe that $T_{2,\ell-1}$ is a tree and edge contraction cannot create cycles. Hence, if S does not contain any edge from $T_{2,\ell-1}$, then S also breaks all cycles of G'. Otherwise, there is an edge e in S and in $T_{2,\ell-1}$ and we can exchange e for e_0 in S, thereby obtaining a solution S' for G'.

Finally, observe that $|V(T_P)| > 7$ and $\mathcal{K}(P) \neq \varnothing$ implies $|V(T_{2,\ell-1})| > 1$. □

Next, we focus on degree-2 paths generating tokens. Note that the end vertices u and w of P could be centers of 2-claws containing inner path vertices of P. To account for this possibility, we define $\mathcal{K}'(P) := \mathcal{K}(P) \cup \{K^{\text{up}}(u), K^{\text{low}}(w)\}$. Note also that tokens are basically edge sets, which may overlap. This behavior is exploited in the following reduction rules requiring a more formal definition. We call an inner path vertex v_i of P a *token separator* if there is no token in $\mathcal{K}'(P)$ containing both e_i and e_{i+1}. If P does not have a token-separator, then P is called *chained* (see Figure 6*b*)).

In the following, we present a reduction rule that works on degree-2 paths P containing a token separator v. In particular, this reduction rule splits P at v,

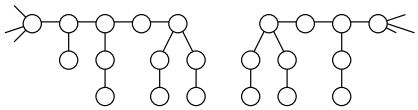

Figure 7: The graph that results from applying Path Reduction Rule 4 to the graph shown in Figure 6*a*).

duplicating T^v. This operation leaves all tokens intact, indicating the correctness of the reduction rule (see Figure 7 for an illustration).

Path Reduction Rule 4. *Let P be a degree-2 path with $\mathcal{K}(P) \neq \varnothing$ and P is not chained. Let v_i be a token separator of P. Then, replace its pendant tree T^{v_i} by two copies of T^{v_i}, connect one to v_{i-1} (by inserting an edge between its connection point and v_{i-1}), and connect the other to v_{i+1}.*

Lemma 8. *Path Reduction Rule 4 is correct.*

Proof. Let G' denote the graph that results from applying Path Reduction Rule 4 to P. In the following, we show for all k that (G, k) is a yes-instance if and only if (G', k) is a yes-instance.

"$\Rightarrow$": By Lemma 6, we know that there is a solution S of size at most k for G that does not delete edges in T^{v_i}. For the sake of contradiction, assume that S is not a solution for G'. Since all cycles in G' are cycles in G, there is a 2-claw C in $G' - S$ that is not in $G - S$. If C contains both copies of v_i, then there is a cycle in G that consists entirely of edges of C. Since S is a solution for G, it is clear that S breaks this cycle, contradicting C being a 2-claw in $G' - S$. If C contains only one of the copies of v_i, then we can delete the other copy from $G' - S$ without destroying C. The resulting graph is a subgraph of $G - S$ that contains C, contradicting S being a solution for G.

"$\Leftarrow$": Let S denote a solution for G' containing at most k edges. If S contains any edge e of the copy of T^{v_i} connected to v_{i-1} (as described in Path Reduction Rule 4), then it is easy to see that $S \setminus \{e\} \cup \{e_i\}$ is also a solution for G' that does not contain such an edge. Analogously, we can assume that S does not contain an edge of the copy of T^{v_i} connected to v_{i+1}. In the following, we show that S is also a solution for G. For the sake of contradiction, assume that $G - S$ is not a forest of caterpillars.

First, assume that there is a 2-claw C in $G - S$. Note that both e_i and e_{i+1} are contained in C, since otherwise, C is entirely contained in G' implying that C is also a 2-claw in $G' - S$.

Case 1: The center of C is v_i.
Then, $|\mathcal{P}_2(v_i)| \geq 1$ and thus, $\mathcal{K}(v_i) \neq \varnothing$. Since v_i is a token separator, $|\mathcal{K}(v_i)| = 2$ and thus, $|\mathcal{P}_2(v_i)| = 2$. However, since both e_i and e_{i+1} are in C, we know that C only uses one of the two P_2s in $\mathcal{P}_2(v_i)$. Since S does not contain any edge of any copy of T^{v_i}, we know that there is another 2-claw C' in $G - S$ that contains both P_2s of $\mathcal{P}_2(v_i)$ and only one of e_i and e_{i+1}. As observed above, C' is entirely contained in $G' - S$, contradicting S being a solution for G'.

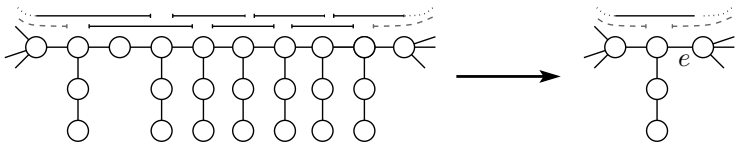

Figure 8: An application of Path Reduction Rule 5 with $|\mathcal{K}(P)| = 7$. In the example, $M_p = \{1, 3, 4, 5, 6, 7, 8\}$ and, hence, $p = 1$, $q = 8$, and $e = \{v_1, v_9\}$.

Case 2: The center of C is not v_i.
Then, since e_i and e_{i+1} are in C, we know that C is centered either at v_{i-1} or v_{i+1}. Without loss of generality, let v_{i+1} be the center of C. If v_i is crossable in P, then v_{i+1} generates a token that spans over v_i, contradicting v_i being a token-separator of P. Hence, v_i is not crossable and, consequently, its degree in G is at least three, implying that T^{v_i} is not a singleton. Thus, there is also a 2-claw C' in G that uses an edge of T^{v_i} instead of e_i. Since C' is entirely contained in G', the solution S contains an edge of C' and by the assumption that S does not contain an edge of any copy of T^{v_i}, we know that S contains an edge of C' that is also in C, contradicting the existence of C in $G - S$.

Second, assume that there is a cycle in $G - S$. Then, it contains v_i, since otherwise, it is also a cycle in $G' - S$. By definition, P is a subpath of this cycle. However, since $\mathcal{K}(P) \neq \varnothing$, Observation 2 implies that there is a 2-claw C in G that is centered at a vertex of P. By the argument above, S contains an edge e' of C. If e' is not in P, then for one of the endpoints of P, the solution S contains all but one of the edges incident to it (this is represented by the $\diamond$-symbol). Otherwise, e' is in P. Both cases contradict the existence of a cycle with subpath P. □

Having dealt with degree-2 paths that contain token separators or no tokens at all, only chained degree-2 paths remain untouched. In the following, we show that, if P is such a chained degree-2 path, then P can be reduced. To this end, we first observe that each edge deletion can only destroy two tokens. This easily follows from the fact that being crossable and generating tokens is mutually exclusive.

Observation 3. *Let P be a degree-2 path. Then, each edge of P is contained in at most two tokens.*

Next, we use this observation to "contract" P such that the outermost tokens remain the same (see Figure 8). To this end, recall that $T_{i,j}$ is the subtree of T_P containing all vertices reachable from v_i in $T_P - \{e_i, e_{j+1}\}$. Furthermore, find an example of a chained degree-2 path in Figure 6*b*).

Path Reduction Rule 5. *Let P be a chained degree-2 path with $|\mathcal{K}(P)| \geq 3$. Let $M_P := \{i \in \mathbb{N} \mid \mathcal{K}(v_i) \neq \varnothing\}$, let $p := \min M_P$ and $q := \max M_P$. If $|\mathcal{K}(P)|$ is even, then delete $T_{p+1,q-1}$, insert the edge $e := \{v_p, v_q\}$, and reduce k*

by $(|\mathcal{K}(P)| - 2)/2$*; otherwise delete* $T_{p+1,q}$*, insert the edge* $e := \{v_p, v_{q+1}\}$*, and reduce* k *by* $(|\mathcal{K}(P)| - 1)/2$*.*

Lemma 9. *Path Reduction Rule 5 is correct.*

Proof. First, consider the case that $|\mathcal{K}(P)|$ is even. Let G' denote the graph that results from applying Path Reduction Rule 5 to P, let $k' := k - (|\mathcal{K}(P)| - 2)/2$, and let $E_T := \{e_{p+1}, e_{p+2}, \ldots, e_q\}$. Since neither v_p nor v_q is crossable, we know that $|\mathcal{K}(P)| - 2$ out of all tokens of P are subsets of E_T. We show that (G, k) is a yes-instance if and only if (G', k') is a yes-instance.

"⇒": Let S be a solution of size at most k for G, let $S_T := S \cap E_T$, and let $S' := S \setminus E_T$. Since S destroys all tokens in $\mathcal{K}(P)$ and, by Observation 3, each edge in S can destroy at most two tokens of $\mathcal{K}(P)$, we know that $|S_T| \geq (|\mathcal{K}(P)| - 2)/2$. If S' is a solution for G', then we are done; otherwise, $G' - S'$ contains a cycle or a 2-claw. Assume that $G' - S'$ contains a cycle. Then, since $G - S$ is acyclic, this cycle contains all edges of E_T. If $|S_T| > (|\mathcal{K}(P)| - 2)/2$, then, by definition, $|S'| < k'$ and, hence, $S' \cup \{e\}$ is a solution of size at most k' for G' (note that $G' - \{e\}$ is a subgraph of G). If $|S_T| = (|\mathcal{K}(P)| - 2)/2$, then S_T cannot destroy all tokens in $\mathcal{K}(P)$. Hence, there are some edges in S' that destroy $\mathcal{K}(v_p)$ or $\mathcal{K}(v_q)$. It is easy to see that at least one of these edges is on the cycle in $G' - S'$, which is a contradiction.

In the following, we assume that $G' - S'$ is acyclic but contains a 2-claw C. Observe that C contains e, since otherwise, C is also in $G - S$. If C is not centered at an inner path vertex of P, then, since v_p and v_q both generate tokens (and therefore, have degree at least three in G), Lemma 6 implies that S' contains an edge of C. Hence, C is centered at v_p or v_q. Without loss of generality, assume that C is centered at v_p. Let e_i denote an edge in $S \cap K(v_p)$ (which, by Observation 2, exists). Since $e_i \notin S'$, we know that $e_i \in S_T \cap K(v_p)$. Then, however, S_T destroys at least $|\mathcal{K}(P)| - 1$ tokens of $\mathcal{K}(P)$. Since $|\mathcal{K}(P)|$ is even, $|S_T| \geq |\mathcal{K}(P)|/2$ and, thus, $|S'| < k'$. As a consequence, $S' \cup \{e\}$ is a solution for G' containing at most k' edges.

"⇐": Let S' be a solution of size at most k' for G' and recall that, by Observation 2, destroying all tokens in $\mathcal{K}(P)$ destroys all 2-claws centered at inner path vertices. If $e \notin S'$, then it suffices to show that we can extend S' to a solution S for G by adding $k - k' = (|\mathcal{K}(P)| - 2)/2$ edges of E_T that destroy $|\mathcal{K}(P)| - 2$ tokens. Since P is chained, this is possible. If $e \in S'$, then we extend $S' - e$ to a solution S for G by adding $|\mathcal{K}(P)|/2$ edges of E_T that destroy all tokens in $\mathcal{K}(P)$. Again, this is possible because P is chained.

Next, consider the case that $|\mathcal{K}(P)|$ is odd. Let G' denote the graph that results from applying Path Reduction Rule 5 to P, let $k' := k - (|\mathcal{K}(P)| - 1)/2$, and let $E_T := \{e_{p+1}, e_{p+2}, \ldots, e_{q+1}\}$. Furthermore, observe that at least $|\mathcal{K}(P)| - 2$ of the tokens of P are subsets of E_T. We show that (G, k) is a yes-instace if and only if (G', k') is a yes-instance.

"⇒": Let S denote a solution of size at most k for G, let $S_T := S \cap E_T$, and let $S' := S \setminus E_T$. By the same arguments as in the case that $|\mathcal{K}(P)|$ is even, it follows that $|S_T| \geq (|\mathcal{K}(P)| - 2)/2$ and that $G' - S'$ is acyclic, but contains a 2-claw C that contains e. Clearly, by definition of p and q, if C is not centered at v_p,

then the center of C is not in P. Then, however, S_T destroys $|\mathcal{K}(P)|+1$ tokens and, by Observation 3, S_T contains at least $(|\mathcal{K}(P)|+1)/2$ edges, implying $|S'| < k'$. Hence, $S' \cup \{e\}$ is a solution of size at most k' for G'. In the following, we assume that C is centered at v_p and consider the token $\mathcal{K}(v_q)$ that is destroyed by S.

Case 1: $(K^{\text{low}}(v_q) \cup \{e_{q+1}\}) \cap S \neq \varnothing$.
Since $K^{\text{low}}(v_q) \cup \{e_{q+1}\} \subseteq E_T$, this means that S_T destroys *all* tokens in $\mathcal{K}(P)$. By Observation 3, $|S_T| \geq |\mathcal{K}(P)|/2$. Since $\mathcal{K}(P)$ is odd, $|S_T| \geq (|\mathcal{K}(P)|+1)/2$ and, thus, $S' \cup \{e\}$ is a solution for G' containing at most k' edges.

Case 2: $(K^{\text{low}}(v_q) \cup \{e_{q+1}\}) \cap S = \varnothing$.
Since v_{q+1} is not a token-separator, v_{q+1} is either the end-vertex of P or a degree-two vertex. Then, however, S contains all edges incident to v_{q+1} except for e_q. Since $S \setminus E_T$ also contains all these edges, it is clear that S' contains them and thus, C is destroyed by S', a contradiction to the assumption above.

"$\Leftarrow$": Let S' denote a solution of size at most k' for G'. If $e \notin S'$, then it suffices to show that we can extend S' to a solution S for G by adding $k - k' = (|\mathcal{K}(P)| - 1)/2$ edges of E_T that destroy $|\mathcal{K}(P)| - 1$ tokens. Since P is chained, this is possible. If $e \in S'$, then we can analogously extend $S' \setminus \{e\} \cup \{e_{q+1}\}$ to a solution S for G. □

With the presented tree- and path reduction rules, we can now upper-bound the size of the graph that remains after all reduction rules have been applied exhaustively by a function of the size of a minimum feedback edge set. To this end, we first show that the number of vertices and edges in each degree-2 path and its pendant trees is small.

Lemma 10. *Let G be reduced with respect to all path reduction rules and let V_3 denote the set of vertices with degree at least 3 in G^*. If two vertices $u, w \in V_3$ are connected in G by a degree-2 path P, then $|V(T_P) \setminus \{u, w\}| \leq 10$ and $|E(T_P)| \leq 11$.*

Proof. Let P denote the degree-2 path between u and w.

Case 1: There is some vertex v_i in P such that its pendant tree T^{v_i} is a Y-graph.
Then, since G is reduced with respect to Path Reduction Rules 1 and 2, P contains at most two inner path vertices, one of which has degree 2. All in all, $|V(T_P) \setminus \{u, w\}| \leq 8$ and $|E(T_P)| \leq 9$.

Case 2: $\mathcal{K}(P) = \varnothing$.
Since G is reduced with respect to Path Reduction Rule 3, we know that $|V(T_P) \setminus \{u, w\}| \leq 5$ and, hence, $|E(T_P)| \leq 6$.

Case 3: $\mathcal{K}(P) \neq \varnothing$.
Since G is reduced with respect to Path Reduction Rule 4, P is chained. But G is also reduced with respect to Path Reduction Rule 5, and, thus, P contains at most six inner path vertices. At most two of these inner path vertices generate tokens, all others are crossable (and therefore, have degree two). Let v_p and v_q denote these two vertices in P. Then, their pendant trees T^{v_p} and T^{v_q} are P_2s. Thus, $T_P \setminus \{u, w\}$ contains at most 10 vertices and at most 11 edges. □

With the help of Lemma 10, we can now upper-bound the total number of vertices and edges in graphs that are reduced with respect to all presented data reduction rules.

Theorem 3.1. 2-Layer Planarization *admits a problem kernel containing at most* $44(f-1)$ *vertices and at most* $45(f-1)$ *edges, with* $f > 0$ *being the size of a minimum feedback edge set of* G*. The problem kernel can be constructed in* $O((f+1) \cdot |G|)$ *time.*

Proof. Assume that the input G is reduced with respect to all presented data reduction rules. For the analysis of the kernel size, we need the following notation. Let V_3 denote the set of vertices with degree at least 3 in G^* and let $G_3^* := (V_3, E_3)$ denote the multigraph on V_3 that contains an edge for every maximal degree-2 path in G. More specifically, E_3 contains an edge $\{u, w\}$ for every edge $\{u, w\} \in E$ with $u, w \in V_3$, and, in addition, E_3 contains an edge $\{u, w\}$ for every maximal degree-2 path of length at least three between two (not necessarily different) vertices $u, w \in V_3$. Thus, G_3^* may contain loops. Furthermore, let F with $|F| = f$ be a minimum feedback edge set of G and let F_3 be a minimum feedback edge set of G_3^* (we require that a feedback edge set of G_3^* contains all loops and all but at most one edge between any two vertices). Clearly, $|F_3| \leq f$ and $G_3^* - F_3$ is a forest and, thus, $|E_3| \leq |V_3| + f - 1$. Since the minimum degree[1] of a vertex in G_3^* is 3, we know that $\sum_{v \in V_3} \deg_{G_3^*}(v) \geq 3|V_3|$, and since the sum on the left hand side equals $2|E_3|$, we know that $2(|V_3| + f - 1) \geq 3|V_3|$, implying $|V_3| \leq 2(f-1)$ and $|E_3| \leq 3(f-1)$.

With G_3^* bounded, we can use Lemma 10 to bound G. Each edge in G_3^* corresponds to a degree-2 path in G^*. Each vertex in V_3 may additionally be incident to a pendant tree (see Figure 4). Thus, we can bound the number of vertices in G by $|V(G)| \leq |E_3| \cdot 10 + |V_3| \cdot 6 + |V_3| \leq 44(f-1)$ and the number of edges in G by $|E(G)| \leq 45(f-1)$.

It remains to show the running time of the kernelization. In the following, we prove that the path reduction rules can be applied to G in $O((f+1) \cdot |G|)$ time. Recall that, in the prove of Lemma 1, $L(v)$, $\mathcal{P}_2(v)$ and $\mathcal{Y}(v)$ are constructed in linear time for all vertices of G. Since both Path Reduction Rule 1 and 2 reduce the number of edges, they can only apply $|E|$ times in total. Each such application can be performed in constant time. It is easy to see that no degree-2 path is subject to more than one of the Path Reduction Rules 3–5. In the worst case, the application of such a path reduction rule requires reapplying the tree reduction rules, which may take $O(|G|)$ time. Since $|E(G_3^*)| \leq O(f)$, we can bound the running time by $O((f+1) \cdot |G|)$. □

4 Search Tree Algorithm

In this section, we provide an algorithm that solves the 2-Layer Planarization problem in $O(6^f \cdot f^2 + (f+1) \cdot |G|)$ time. It employs a search tree based

[1] A loop at vertex v contributes 2 to the degree of v.

strategy, branching on different structures in different phases. Herein, each time the algorithm branches, the feedback edge set shrinks, allowing us to bound the running time as stated above. The algorithm runs in three phases. First, we apply a search tree enumerating partial solutions by branching on a certain type of 2-claw. Second, we branch on small cycles in the remaining graph. In the third phase, we branch on the tokens (see Section 3.2, page 14) that remain in the graph. The input graph G considered in each phase is assumed not to be subject to any previous phase, that is, G does not contain a structure that is branched on in a previous phase. Furthermore, the input graph of each phase is assumed to be reduced with respect to the data reduction rules presented in the previous section. We show that the total number of edge deletions done by branching in all three phases does not exceed the feedback edge set number of the input graph G. In the following, we present the three phases of our algorithm.

First Phase In this paragraph, we describe the phase of the algorithm that branches on short cycles and 2-claws whose edges are all contained in cycles. More specifically, we first compute a set E_C of all edges of G that are contained in cycles. This can be done in $O(|G|)$ time. Then, we consider the subgraph $G_C := (V, E_C)$ of G. By finding a vertex v with $\deg_{G_C}(v) \geq 3$ (which can be done in $O(|G_C|)$ time), we always obtain either a 2-claw all of whose edges are part of some cycle, or a cycle of length at most four (see Lemma 2 in [5]). In both cases, we can branch into the at most six possibilities to destroy this object by edge deletions. Note that in each branch, the minimum feedback edge set number decreases by one, since we delete an edge without disconnecting the graph. Clearly, every solution must delete one of these edges and, thus, one of these cases leads to a partial solution that can be extended to an optimal solution for G.

The branching is performed as long as G_C contains a vertex of degree at least three. If all vertices in G_C have degree at most two, then all cycles in G are vertex disjoint.

Observation 4. *If $G = (V, E)$ is not subject to Phase 1, then all cycles in G are vertex-disjoint.*

Second Phase With the first branching done, the next step is to eliminate all small cycles in the remaining graph. In this context, "small" means at most five vertices. We do this to assure that there is a remaining cycle that contains a token. Since all cycles in G are vertex-disjoint, we can find a cycle of length at most five in G in $O(|G|)$ time. Then, we can branch on the at most five possibilities to destroy this cycle by edge deletion. Again, we remove an edge without disconnecting the graph, therefore decreasing the minimum feedback edge set number by one in each branch. Again, we know that every solution must contain one of these edge deletions and, thus, one of these cases leads to a partial solution that can be extended to an optimal solution for G.

Since, in this phase, G is not subject to Phase 1, it is basically a tree of cycles and singletons. If G contains a cycle, then arbitrarily rooting this tree enables us to consider a cycle of G that has maximum distance from this root. Clearly, this cycle contains a vertex u such that $P_{u,u}$ is a degree-2 path (see Section 3.2, page 10 for the definition of degree-2 paths). If G is also not subject to Phase 2, then $P_{u,u}$ contains at least five inner path vertices. However, since G is reduced with respect to Path Reduction Rules 1 and 2, we know that $P_{u,u}$ cannot contain a vertex whose pendant tree is a Y-graph. Furthermore, since G is reduced with respect to Path Reduction Rule 3, we also know that $\mathcal{K}(P_{u,u}) \neq \varnothing$ (see Section 3.2, page 14 for the definition of tokens). Finally, if $\mathcal{K}(P_{u,u})$ contained only two tokens and both contained $\diamond$, then both tokens together can only contain six edges of $P_{u,u}$. However, since $|E(P_{u,u})| \geq 6$, the two tokens do not overlap, implying that there is a token-separator in $P_{u,u}$, which contradicts G being reduced with respect to Path Reduction Rule 4. Hence, there is a token in $\mathcal{K}(P_{u,u})$ that does not contain $\diamond$.

Observation 5. *If G contains a cycle and is neither subject to Phase 1 nor to Phase 2, then there is a cycle of G that contains a vertex that generates a token not containing $\diamond$.*

Third Phase In this paragraph, we describe the final part of the algorithm that branches on the tokens in the remaining cycles. Recall that, if the input graph is acyclic, then Lemma 2 implies that its size is constant. Hence, we assume G to contain a cycle. Then, by Observation 5, some vertex of G generates a token K on which we can branch. Since the token of each vertex can be computed in $O(1)$ time, we can find v in linear time. By definition, K contains at most four edges and, thus, we can branch on the four possibilities to destroy this token by deleting one of its edges. Since K does not contain $\diamond$, all of the edges of K are edges of $P_{u,u}$ and hence, they are edges of a cycle, implying that the minimum feedback edge set number decreases by one in each branch. Observation 2 and Lemma 6 imply that there is an optimal solution containing one of these edge deletions and, thus, one of these cases leads to a partial solution that can be extended to an optimal solution for G.

If we cannot find any further cycles in G, then it is a tree. However, since G is reduced with respect to the data reduction rules, we know that in this case, G has constant size (by Lemma 2) and, thus, can be solved in constant time.

Running Time The algorithm presented in the previous paragraphs consists of three consecutive branching algorithms. In each branching step, the minimum feedback edge set number of the graph decreases by one. In the three phases, the algorithm branches into at most six, five, and four cases, respectively. The running time in each branching step is dominated by the application of the data reduction rules. All in all, the algorithm runs in $O(6^f \cdot (f+1) \cdot |G|)$ time

Theorem 4.1. 2-Layer Planarization *can be solved in $O(6^f \cdot (f+1) \cdot |G|)$ time.*

By initially kernelizing the input instance, we can assume $|E|$ to be linear in f for the branching algorithm.

Corollary 4.2. 2-Layer Planarization *can be solved in* $O(6^f \cdot f^2 + (f+1) \cdot |G|)$ *time. Moreover, if* $|E| \leq |V| + O(\log |V|)$*, then* 2-Layer Planarization *can be solved in polynomial time.*

Note that, in the spirit of multivariate algorithmics [9, 20], it is possible to combine both our algorithm and the $O(3.562^k \cdot k + |G|)$-time algorithm by Suderman [22] to solve the problem in time that is asymptotically equal to the minimum of both running times.

5 Conclusion

In this work, we presented a linear-size problem kernel and a search tree algorithm for 2-Layer Planarization parameterized by the "feedback edge set number", an alternative parameterization upper-bounded by the size of an optimal solution.

We conclude with some open questions for future work. First, since 2LP is an interesting problem in practice, we plan to implement our kernelization approach and test it on real-world data, especially in comparison with previous data reduction techniques. Second, in the light of the linear-time algorithm for edge-weighted 2LP on trees presented by Shahrokhi et al. [21], it is interesting to investigate whether our kernelization approach also holds for the edge-weighted case. Moreover, it may be worthwhile to investigate whether the branching analysis suggested by Suderman [22] can be used to obtain a better search tree algorithm for the parameter feedback edge set number. Providing efficient fixed-parameter algorithms (in particular polynomial-size problem kernels) for parameters upper-bounded by the feedback edge set number is a natural next step to extend the range of solvable instances. The feedback vertex set number would be a canonical candidate. Additionally, it may be interesting to investigate the parameter $k' = (k - f)$ that represents an "above guarantee" parameter for the problem. Finally, it would be interesting to extend our results to the multilayered problem versions [6].

Acknowledgments We are grateful to Nadja Betzler, Jiong Guo, and Rolf Niedermeier for fruitful discussions and helpful comments. We also thank an anonymous reviewer of *Theoretical Computer Science* for providing helpful comments.

References

[1] H. L. Bodlaender. Kernelization: New upper and lower bound techniques. In *Proceedings of the 4th International Workshop on Parameterized and Exact Computation (IWPEC '09)*, volume 5917 of *Lecture Notes in Computer Science*, pages 17–37. Springer, 2009.

[2] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk. An improved FPT algorithm and quadratic kernel for pathwidth one vertex deletion. In *Proceedings of the 5th International Symposium on Parameterized and Exact Computation (IPEC'10)*, volume 6478 of *Lecture Notes in Computer Science*, pages 95–106. Springer, 2010. ISBN 978-3-642-17492-6.

[3] M. Dom, J. Guo, and R. Niedermeier. Approximation and fixed-parameter algorithms for consecutive ones submatrix problems. *Journal of Computer and System Sciences*, 76 (3-4):204–221, 2010.

[4] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[5] V. Dujmović, M. Fellows, M. Hallett, M. Kitching, G. L. C. McCartin, N. Nishimura, P. Ragde, F. Rosamond, M. Suderman, S. Whitesides, and D. R. Wood. A fixed-parameter approach to 2-layer planarization. *Algorithmica*, 45(2):159–182, 2006.

[6] V. Dujmović, M. Fellows, M. Hallett, M. Kitching, G. L. C. McCartin, N. Nishimura, P. Ragde, F. Rosamond, M. Suderman, S. Whitesides, and D. R. Wood. On the parameterized complexity of layered graph drawing. *Algorithmica*, 52(2):267–292, 2008.

[7] P. Eades and S. Whitesides. Drawing graphs in two layers. *Theoretical Computer Science*, 131(2):361–374, 1994.

[8] M. Fellows, D. Lokshtanov, N. Misra, M. Mnich, F. Rosamond, and S. Saurabh. The complexity ecology of parameters: An illustration using bounded max leaf number. *Theory of Computing Systems*, 45(4):822–848, 2009.

[9] M. R. Fellows. Towards fully multivariate algorithmics: Some new results and directions in parameter ecology. In *Proceedings of the The 20th International Workshop on Combinatorial Algorithms (IWOCA'09)*, volume 5874 of *Lecture Notes in Computer Science*, pages 2–10. Springer, 2009.

[10] M. R. Fellows, D. Lokshtanov, N. Misra, F. A. Rosamond, and S. Saurabh. Graph layout problems parameterized by vertex cover. In *Proceedings of the 19th International Symposium on Algorithms and Computation (ISAAC'08)*, volume 5369 of *Lecture Notes in Computer Science*, pages 294–305. Springer, 2008.

[11] H. Fernau. Two-layer planarization: Improving on parameterized algorithmics. *Journal of Graph Algorithms and Applications*, 9(2):205–238, 2005.

[12] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[13] J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(1):31–45, 2007.

[14] J. Guo, F. Hüffner, and R. Niedermeier. A structural view on parameterizing problems: Distance from triviality. In *Proceedings of the 1st International Workshop on Parameterized and Exact Computation (IWPEC'04)*, volume 3162 of *Lecture Notes in Computer Science*, pages 162–173. Springer, 2004.

[15] B. M. P. Jansen and H. L. Bodlaender. Vertex cover kernelization revisited: Upper and lower bounds for a refined parameter. In *Proccedings of the 28th International Symposium on Theoretical Aspects of Computer Science (STACS'11)*, volume 9 of *LIPIcs*, pages 177–188. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011. ISBN 978-3-939897-25-5.

[16] S. Kratsch and P. Schweitzer. Isomorphism for graphs of bounded feedback vertex set number. In *Proceedings of the 12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT'10)*, volume 6139 of *Lecture Notes in Computer Science*, pages 81–92. Springer, 2010.

[17] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. John Wiley & Sons, Inc., New York, NY, USA, 1990. ISBN 0-471-92838-0.

[18] P. Mutzel. An alternative method to crossing minimization on hierarchical graphs. *SIAM Journal on Optimization*, 11(4):1065–1080, 2001.

[19] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Number 31 in Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2006.

[20] R. Niedermeier. Reflections on multivariate algorithmics and problem parameterization. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS '10)*, volume 5 of *LIPIcs*, pages 17–32. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.

[21] F. Shahrokhi, O. Sýkora, L. A. Székely, and I. Vrto. On bipartite drawings and the linear arrangement problem. *SIAM Journal on Computing*, 30(6):1773–1789, 2001.

[22] M. Suderman. *Layered Graph Drawing*. PhD thesis, School of Computer Science, McGill University Montréal, 2005.

[23] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man and Cybernetics*, 11(2):109–125, 1981.

[24] M. S. Waterman and J. R. Griggs. Interval graphs and maps of DNA. *Bulletin of Mathematical Biology*, 48(2):189–195, 1986.