

Two-Layer Planarization: Improving on Parameterized Algorithmics

Henning Fernau

University of Hertfordshire, Computer Science,
College Lane, Hatfield, Herts AL10 9AB, UK

Universität Tübingen, WSI für Informatik, Sand 13,
72076 Tübingen, Germany, fernau@informatik.uni-tuebingen.de

The University of Newcastle, School of Electr. Eng. and Computer Science,
University Drive, Callaghan, NSW 2308, Australia

Abstract

A bipartite graph is *biplanar* if the vertices can be placed on two parallel lines in the plane such that there are no edge crossings when edges are drawn as straight-line segments connecting vertices on one line to vertices on the other line. We study two problems:

- 2-LAYER PLANARIZATION: can k edges be deleted from a given graph G so that the remaining graph is biplanar?
- 1-LAYER PLANARIZATION: same question, but the order of the vertices on one layer is fixed.

Improving on earlier works of Dujmović *et al.* (*Proc. Graph Drawing GD 2001*, pp. 1–15, 2002), we solve the 2-LAYER PLANARIZATION problem in $\mathcal{O}(k^2 \cdot 5.1926^k + |G|)$ time and the 1-LAYER PLANARIZATION problem in $\mathcal{O}(k^3 \cdot 2.5616^k + |G|^2)$ time. Moreover, we derive a small problem kernel for 1-LAYER PLANARIZATION.

Article Type	Communicated by	Submitted	Revised
regular paper	G. Liotta	October 2004	September 2005

1 Introduction

In a *2-layer drawing* of a bipartite graph $G = (A, B; E)$, the vertices in A are positioned on a line in the plane, which is parallel to another line containing the vertices in B , and the edges are drawn as straight-line segments. Such drawings have various applications, not only in graph drawing / visualization [4] but also in bioinformatics [24] and VLSI layouts [16].

A *biplanar graph* is a bipartite graph that admits a 2-layer drawing with no edge crossings; we call such a drawing a *biplanar drawing*. It has been argued that 2-layer drawings in which all the crossings occur in a few edges are more readable than drawings with fewer total crossings [17, 18]—which gives the CROSSING MINIMIZATION problem(s) [8].

This naturally leads to the definition of the 2-LAYER PLANARIZATION problem (2-LP): given a graph G (not necessarily bipartite), and an integer k called the *parameter*, can G be made biplanar by deleting at most k edges? Two-layer drawings are of fundamental importance in the “Sugiyama” approach to multi-layer graph drawing [22]. This method involves (repeatedly) solving the 1-LAYER PLANARIZATION problem (1-LP): given a bipartite graph $G = (A, B; E)$, a permutation π of A , and an integer k , can at most k edges be deleted to permit G to be drawn without crossings with π as the ordering of A ? In this paper, we also present results on this problem.

Fixed parameter tractability. We develop improved algorithms for 2-LP and for 1-LP that are exponential in the parameter k . This has the following justification: when the maximum number k of allowed edge deletions is small, an algorithm for 1- or 2-LP whose running time is exponential in k but polynomial in the size of the graph may be useful. We expect the parameter k to be small in practice. Instances of the 1- and 2-LP problems for dense graphs are of little interest from a practical point of view, as the resulting drawing will be unreadable anyway.

Hence, his analysis fits into the framework of parameterized algorithmics. A parameterized problem P with input size n and parameter size k is *fixed parameter tractable*, or in the class \mathcal{FPT} , if there is an algorithm to solve the problem in $f(k) \cdot n^\alpha$ time, for some function f and constant α (independent of k). There is an alternative characterization of \mathcal{FPT} : P is in \mathcal{FPT} iff there exists a polynomial-time reduction that transforms each instance (I, k) into an equivalent instance (I', k') of P such that the size of I' and k' is bounded by some function $g(k)$ (not depending on the size of the original input I). (I', k') is then called a *problem kernel* of (I, k) . Two basic approaches for producing \mathcal{FPT} algorithms are *kernelization* and *bounded search trees* [5], and these will be applied in this paper, as well. Further details on this notion can be found in [5, 12].

Our results. In this paper, we apply kernelization and search tree methods to obtain algorithms for the 1- and 2-LP problems, this way improving earlier results from [7]. This leads to an $\mathcal{O}(k^2 \cdot 5.1926^k + |G|)$ time algorithm for 2-LP in a graph G . Here $|G| = |V| + |E|$ for a graph $G = (V, E)$. We present a

similar second algorithm to solve the 1-LP problem in $\mathcal{O}(k^3 \cdot 2.5616^k + |G|^2)$ time. By way of contrast, in [7] algorithms were given that solve 2-LP in time $\mathcal{O}(k^2 6^k + |G|)$ and 1-LP in time $\mathcal{O}(3^k |G|)$. Note that we not only improve on the running times of the derived algorithms (which is mainly due to better search tree algorithms), but we also present the first small problem kernel for 1-LP. This is important, since from a practical point of view, reduction rules (that have to be developed to get small kernels) are often the key to efficient solutions of computationally hard problems. Therefore, getting small kernels can be seen as a valid approach rather independent from the problem of getting better exponential running times, see [9] for further (philosophical) comments on this issue. As a further technical contribution, we also present kernelization rules that lead to a cubic kernel for 3-HITTING SET that is vertex-induced; this is essential to get a proper (non-annotated) kernel of cubic size for 1-LP.

Our methodology. To arrive at our results, we draw connections to HITTING SET problems. The top-down analysis technique presented in [10] is applied to obtain the claimed running times in the analysis of the search tree algorithms. We believe that certain insights gained from this approach are more widely applicable to other problems that involve “forbidden structures,” even though often a sort of postprocessing prevents the direct application of HITTING SET algorithms off from the shelf; in the case of our problems, the postprocessing mainly deals with resolving (larger) cycles. Notice that a safe approach under these circumstances is often to enumerate all minimal solutions of the underlying HITTING SET instance (see [3, 12] for details on parameterized enumeration) and then postprocess the solutions in the leaves of the search tree. In a certain sense, this is essentially the approach taken in [7] to solve 2-LP and 1-LP. However, one could try to do better by following the steps listed below.

- Try to “translate” the well-known reduction rules for HITTING SET into the target area of the problem P under consideration. This translation will not always yield sound rules for P . According to our experience, particular problems are raised by the so-called *vertex domination rule*.
- This translation will almost immediately give a polynomial-size kernel.
- Moreover, it is possible to improve on the running time of the search tree algorithm whenever it is possible to find a substitute for the vertex domination rule that allows to cope with at least “some” of the degree-one-vertices (in the corresponding HITTING SET instance), so that one can always branch at vertices of degree at least two.

The search tree algorithms derived by this approach have a quite simple general structure:

1. Exhaustively apply all reduction rules.
2. Decide if a trivial YES- or NO-instance is obtained.

3. If possible, enter the polynomial-time postprocessing phase.
4. Otherwise, select a vertex x (in the corresponding HITTING SET instance) according to so-called heuristic priorities and branch according to the two possibilities if x is in the hitting set or not.

Knowing that the “weak point” of this approach is often the validity of the vertex domination rule also directs the search of useful reduction rules to those who can cope with vertices of degree one.

There is an additional benefit coming from this approach: while it is tedious if not close to impossible to give a formal correctness proof for improved exact algorithms that rely on a scrutinized local analysis and accordingly complicated branching structures (see the 3-HITTING SET algorithm published in [19]), this becomes quite an easy task now, since the proof can be modularized:

1. Prove that all reduction rules are sound.
2. Check if the trivial cases are correctly covered.
3. See if the postprocessing phase is correctly entered.
4. Check if the branching process itself covers all cases.

Based on these observations, a proof of the correctness of the algorithm by induction on the number of vertices (in the HITTING SET terminology) becomes easy. In the case of a HITTING SET algorithm whose design was based on the sketched principles, a formal correctness proof can be found in [10]. The same approach was undertaken for the LINEAR ARRANGEMENT BY DELETING EDGES problem in [13].

Finally, this approach has also benefits for the implementation of such algorithms, since they will correctly work even without implementing the reduction rules and the heuristic priorities; these bits will only affect the running time. So, rapid prototyping is encouraged, as well as experimenting with the reduction rules and priorities; we hope to report on such experiments soon.

The burden is more shifted towards the time analysis of the algorithm, which is no longer trivial. Let us also mention that the recent breakthroughs [15] in the development of exact algorithms for DOMINATING SET are using similar principles, in particular the relations to HITTING SET and the accordingly translated reduction rules, resulting in a nearly trivial algorithm with a far from trivial running time analysis.

There is one last technical bit worth mentioning here: when we try to benefit from the relationships of our problem P to HITTING SET, we often cannot simply remove the vertices that have been decided *not* to form part of a solution of the HITTING SET problem we are after (e.g., in the branch where we do *not* take a certain vertex into the hitting set), since they might be still needed in the postprocessing phase. Rather, we will *mark* them: they will be ignored in the HITTING SET branching hitherto, but they must be (possibly) considered in the postprocessing phase. Hence, formally we will work with an *annotated*

version of our original problem P . Also, the reduction rules will have to work with such annotated instances. This annotation technique has often found to be useful; e.g., in the context of HITTING SET related problems, recent work on FACE COVER gives an example involving rather extensive annotation rules [1].

2 Preliminaries

In this section, we introduce notation, recall a characterization of biplanar graphs and formalize the problem statements. All the mentioned results can be found in [7], although some of them already appear in the older literature.

Basic definitions. In this paper, each graph $G = (V, E)$ is simple and undirected. The subgraph of G induced by a subset E' of edges is denoted by $G[E']$. A vertex with degree one is a *leaf*. If vw is the edge incident to a leaf w , then we say w is a *leaf at v* and vw is a *leaf-edge at v* . The *non-leaf degree* of a vertex v in graph G is the number of non-leaf edges at v in G , and is denoted by $\text{deg}'_G(v)$.

A graph is a *caterpillar* if deleting all the leaves produces a (possibly empty) path. A *2-claw* is a graph consisting of one degree-3 vertex, the *center*, which is adjacent to three degree-2 vertices, each of which is adjacent to the center and one leaf. A graph consisting of a cycle and possibly some leaf-edges attached to the cycle is a *wreath*. Notice that a connected graph that does not have a vertex v with $\text{deg}'(v) \geq 3$ is either a caterpillar or a wreath.

Problem statements. A set T of edges of a (not necessarily bipartite) graph G is called a *biplanarizing set* if $G \setminus T$ is biplanar. The *bipartite planarization number* or *biplanarization number* of a graph G , denoted by $\text{bpr}(G)$, is the size of a minimum biplanarizing set for G . The 2-LP problem is: given a graph G and an integer k , is $\text{bpr}(G) \leq k$?

For a given bipartite graph $G = (A, B; E)$ and permutation π of A , the *1-layer biplanarization number* of G and π , denoted $\text{bpr}(G, \pi)$, is the minimum number of edges in G whose deletion produces a graph that admits a biplanar drawing with π as the ordering of the vertices in A . The 1-LP problem asks if $\text{bpr}(G, \pi) \leq k$. When dealing with a bipartite graph $G = (A, B; E)$ and a permutation π of A , we write $<_\pi$ to denote the order induced by π on A .

Useful earlier results. To prove their kernelization result for 2-LP, Dujmović *et al.* introduced the following potential function. For a graph $G = (V, E)$, define

$$\forall v \in V, \Phi_G(v) = \max\{\text{deg}'_G(v) - 2, 0\}, \text{ and } \Phi(G) = \sum_{v \in V} \Phi_G(v) .$$

Intuitively, $\Phi(v)$ approximates the number of edges in the distance-2 neighborhood of v that must be included in any biplanarizing set for G .

Lemma 1 $\Phi(G) = 0$ if and only if G is a collection of caterpillars and wreaths.

Biplanar graphs are easily characterized, and there is a simple linear-time algorithm to recognize biplanar graphs, as the next lemma makes clear.

Lemma 2 *Let G be a graph. The following assertions are equivalent:*

- (a) G is biplanar.
- (b) G is a forest of caterpillars.
- (c) G is acyclic and contains no 2-claw as a subgraph.
- (d) G is acyclic and $\Phi(G) = 0$ (with Lemma 1).

Lemma 2 implies that any biplanarization algorithm must destroy all cycles and 2-claws. The next lemma gives a condition for this situation.

Lemma 3 *If there exists a vertex v in a graph G such that $\deg'_G(v) \geq 3$, then G contains a 2-claw or a 3- or 4-cycle containing v .*

The following two lemmas concerning the potential function $\Phi(G)$ are important for the development of efficient parameterized algorithms; in particular, Lemma 4 exhibits a case when the biplanarization number can be found in polynomial time.

Lemma 4 *For graphs G with $\Phi(G) = 0$, a minimum biplanarizing set of G consists of one cycle edge from each component wreath.*

Lemma 5 *For every graph G , $\text{bpr}(G) \geq \frac{1}{2}\Phi(G)$.*

3 Hitting Set: Setting the scene

As mentioned in the Introduction, the relation of the two problems we are interested in to HITTING SET is crucial for the methodology we are proposing. Therefore, we will recall some terminology and results in connection with HITTING SET.

Some terminology on hypergraphs. A *hypergraph* $G = (V, E)$ is given by its finite set of *vertices* V and its set of (*hyper*)-*edges* E , where a hyperedge is a subset of V . The cardinality of a hyperedge e is also called its *size*. The cardinality of the set of edges which contain a specific vertex v is called the *degree* of v .

Problem definition. d -HITTING SET can be viewed as a “vertex cover problem” on hypergraphs. More formally, this problem can be stated as follows:

Given a hypergraph $G = (V, E)$ with edge size bounded by d and an integer k , the question is if there is a *hitting set* or *cover* of size at most k :

$$\exists C \subseteq V \forall e \in E (C \cap e \neq \emptyset) ?$$

The least upperbound d on the edge size of a hypergraph G is also called the *arity* of G .

Reduction rules. The following simple reduction rules have been obviously rediscovered again and again over the past 40 years, see [10] for a short history.

1. (hyper)edge domination: A hyperedge e is *dominated* by another hyperedge f if $f \subset e$. In that case, delete e .
2. tiny edges: Delete all hyperedges of degree one and place the corresponding vertices into the hitting set.
3. vertex domination: A vertex x is *dominated* by a vertex y if, whenever x belongs to some hyperedge e , then y also belongs to e . Then, we can simply delete x from the vertex set and from all edges it belongs to.

These rules are mainly important to arrive at good estimates for search tree based HITTING SET algorithms, as can be seen in [10]. As they are not mentioning the parameter in any form, they are also valid when solving nonparameterized HITTING SET problems, see [15, 23] for applications.¹ For the purpose of getting better running time estimates, the vertex domination rule is by far the most important. The use of the edge domination rule is usually more subtle. Namely in our analysis, we set up mutual recursions that estimate search tree sizes assuming that a certain number of “small edges” are present. Therefore, it is important to know that, e.g., in the branch that explicitly does not put a vertex x into the cover, a certain number of small edges are created, only depending on the degree of x . Without having applied the edge domination rule, it could be the case that the supposedly new edge is already present in the count of small edges.

For the purpose of getting small kernels, the following large degree rules are important, see [20]. We only formulate them in the case of 3-HITTING SET, but the generalization to HITTING SET problems of higher arity should be clear.

- For any pair of vertices x, y do: if the number of edges that contain both x and y exceeds k , then add an edge $\{x, y\}$.²
- For all vertices x do: if the number of edges that contain x exceeds k^2 , then add an edge $\{x\}$.

Now, if none of the above rules applies to a 3-HITTING SET instance, the following cutting rule applies:

- If the instance has more than k^3 hyperedges, then NO.

¹It should be clear, however, that the tiny edge rule decrements the parameter.

²That is our interpretation of [20, Proposition 1]; actually, it is stated there that “all sets (hyperedges) containing both x and y can be deleted,” but this alone is surely not working, since we must model that either x or y go into the hitting set.

The cutting rule of course implies that there is a $3k^3$ kernel for 3-HITTING SET (measured in terms of vertices). In a similar way, kernels of size $\mathcal{O}(k^d)$ can be derived for other d -HITTING SET problems.

Observe that in fact all mentioned reduction rules are needed to obtain this kernel; e.g., the vertex domination rule is used to get rid of isolated vertices: an isolated vertex is dominated by any other vertex. Notice finally that all but the vertex domination rule are also valid when it comes to the task of enumerating all minimal hitting sets of size at most k ; however, we still have a small kernel (when the arity is bounded by a constant d), since we can easily get rid of isolates by an extra reduction rule:

- 3a. If x is an isolated vertex, then delete x .

Induced problem kernels. There is one drawback of the kernelization from [20]: the kernel is obtained by adding / deleting edges, not only by deleting vertices. In other words, the obtained kernel is not vertex-induced. This is no problem with d -HITTING SET in general, but if we want to “use” this kernelization in situations where the HITTING SET instance is merely used to *model* another problem instance, this tends to become problematic. Namely, often the hyperedges correspond to “forbidden situations” (e.g., forbidden subgraphs) and the vertices are the constituents that form the situations. Deleting a hyperedge would then be meaningless in the “world” that is modeled by the HITTING SET instance: all of a sudden say a certain forbidden subgraph would no longer be forbidden, although all constituents are still there. By way of contrast, vertex deletions will also destroy the forbidden situations, so that a vertex-induced kernel can be meaningfully interpreted in the modeled world. Therefore, the following result is interesting on its own right:³

Theorem 1 3-HITTING SET admits a vertex-induced problem kernel of size $\mathcal{O}(k^3)$, measured both in the number of vertices and in the number of edges.

We do not present a proof of that theorem in what follows, but only describe the reduction rules that we use. We will also use the hyperedge domination rule, tiny edge rule (only needed if the original instance contains such small edges) and isolated vertex rule to obtain the kernel. Whenever we face an instance of HITTING SET, we partition its vertex and edge sets as follows: Let $F = \{x \in V \mid x \text{ shares more than } k \text{ edges with some } y \text{ in } V\}$ and let S be the complement of F in V . Elements of F and S will be referred to in the sequel as *fat* and *slim* vertices, respectively. The set of (hyper-)edges E can be partitioned as follows:

$$E_{j*s} = \{e \in E \mid |e| = 3 \wedge |e \cap S| = j\} \quad \text{for } j = 0, 1, 2, 3; \text{ and}$$

$$E_p = \{e \in E \mid |e| = 2\}.$$

³The work on vertex-induced kernels is common work with F. Abu-Khzam.

Observe that edges that only contain one vertex can be dealt with by using the tiny edge rule, so that in fact we can assume that

$$E = \bigcup_{j=0}^3 E_{j*s} \cup E_p.$$

In the rules listed below, whenever we say that we put x into the hitting set, then this means that we reduce the parameter by one and delete x and all edges containing x from the instance. Later, we will refer to these rules as modified large degree rules.

- If x is a vertex that occurs more than k times in edges from E_p , then put x into the hitting set.
- If x is a slim vertex of degree larger than k^2 , then put x into the hitting set.
- If x is a fat vertex that appears more than k times with more than k different other fat vertices, then put x into the hitting set.
- If x is a fat vertex that belongs to more than k^2 edges of E_{2*s} , then put x into the hitting set.

Let us define the *co-occurrence* of a pair $\{x, y\}$ of vertices to be the number of edges that “contain” the two vertices simultaneously. Denote by $co(x, y)$ the co-occurrence of $\{x, y\}$. Now, let

$$E_{1*s,<} = \{\{x, y, z\} \in E_{1*s} \mid x, y \in F \text{ and } co(x, y) \leq k \text{ in } E\}.$$

Moreover, let $E_{1*s,>} = E_{1*s} \setminus E_{1*s,<}$.

- If x is a vertex that occurs more than k^2 times in edges of $E_{1*s,<}$, then put x into the hitting set.
- If x is a slim vertex that occurs in edges of $E_{1*s,>}$ and does not appear elsewhere in E , then delete x .

4 2-Layer Planarization: Bounded search tree

4.1 The general strategy

The algorithm of Dujmović *et al.* Based on the lemmas collected in Sec. 2, Dujmović *et al.* showed [7] the following result:

Theorem 2 *Given a graph G and integer k , there is an algorithm that determines if $\text{bpr}(G) \leq k$ in $\mathcal{O}(k \cdot 6^k + |G|)$ time.*

That algorithm consists of two parts: a kernelization algorithm and a subsequent search tree algorithm called **2-Layer Bounded Search Tree**. The latter algorithm basically looks for a vertex v with $\deg'(v) \geq 3$: if found, then at most 6 recursive branches are triggered to destroy the forbidden structures described in Lemma 3. After branching, a graph G with $\Phi(G) = 0$ remains that is solvable with the help of Lemma 4.

Linking 2-Layer Planarization and 6-Hitting Set. Can we further improve on the running time of the search tree algorithm? Firstly, observe that whenever $\deg'_{G'}(v) \geq \ell$ for any G' obtained from G by edge deletion, then already $\deg'_G(v) \geq \ell$. This means that we can modify the sketched algorithm by collecting *all* vertices of non-leaf degree at least three and, based on this, all *forbidden structures* F , i.e., 2-claws, 3-cycles, or 4-cycles, according to Lemma 3 (which then might interact). For reasons of improved algorithm analysis, we also regard 5-cycles as forbidden structures in what follows. By re-interpreting the edges of G as the vertices of a hypergraph $H = (E, F)$, where the hyperedges correspond to the forbidden structures, a 2-LP instance (G, k) is translated into an instance (H, k) of 6-HITTING SET (6HS).

If we delete all those edges in G that are elements in a hitting set as delivered by a 6HS algorithm, we arrive at a graph G' which satisfies $\deg'_{G'}(v) < 3$ for all vertices v . Hence, $\Phi(G') = 0$, and Lemma 4 applies.

Unfortunately, we cannot simply take any 6HS algorithm, as described, e.g., in [19]. Why? The problem is that the 6HS algorithm would be completely oblivious of large wreath components. Therefore, clever branching in 6HS algorithms might miss optimal solutions to the original 2-LAYER PLANARIZATION instance. Hence, instead of solving the decision problem 6HS we should enumerate all minimal hitting sets [3, 12]. This throws us back to the $\mathcal{O}^*(6^k)$ algorithm complexity found in [7].

Let us therefore see how the methodology sketched in the Introduction works out in this case.

4.2 The basic ingredients of our algorithm

In order to maintain the structure of the original graph for the final wreath analysis, we will *mark* edges that will not be put into a solution during the recursive branching process, but we will not delete them. Hence, during the course of the algorithm we present, a set M of marked edges will be maintained. A *forbidden structure* f is a set of edges of the graph instance $G = (V, E)$ such that

- f describes a cycle of length up to five or a 2-claw, and
- $f \setminus M \neq \emptyset$.

$c(f) = f \setminus M$ is the *core* of f ; $s(f) = |c(f)|$ is the *size* of f . The number of unmarked forbidden structures to which a specific edge e belongs is also called the *degree* of e .

Translating reduction rules. We will use the following reduction rules that can be obtained by translating the according reduction rules for HITTING SET as described in the preceding section:

1. (structure domination: A forbidden structure f is *dominated* by another structure f' if $c(f') \subset c(f)$.) Then, mark f .
2. tiny structure: If $s(f) = 1$, put the only unmarked edge into the solution that is constructed.
- 3a. isolate: If e is an edge of degree zero, then mark e .

These rules are valid, since they are valid rules for the task of enumerating minimal hitting sets. In fact, as mentioned in the previous section, the benefits of what is now the structure domination rule will only show up within a much finer analysis than we are after here. Hence, we will disregard that rule in what follows, since it creates additional complications when describing the overall algorithm, due to the otherwise unnecessary marking of (some) forbidden structures.

In order to facilitate the following arguments, we shall actually work with a variant of the isolates rule:

3a.' isolate': If e is an edge of degree zero, then do:

- Mark e if this would not create any cycle that only consists of marked edges.
- Otherwise, delete e and decrement the parameter.

We could have also incorporated a translation of the modified large degree rules, which are valid as well. This way, it is possible to derive a kernel for 2-LAYER PLANARIZATION in a rather canonical fashion (as we will show in the case of 1-LAYER PLANARIZATION below). However, the corresponding kernel size is worse than what has been obtained in [7], so that we do not follow this trail.

The rules we got so far do not suffice to improve on the running time. To achieve this goal, we must be able to handle edges of degree one (to a certain extent) by reduction rules. As we already commented on, we cannot simply translate the vertex domination rule; this would not be a valid rule for 2-LP, since larger cycles are neglected. We will discuss this point later on.

To formulate the new rule, we need to fix some notions to talk about 2-claws. Let $C = \{c, w_1, w_2, w_3, x_1, x_2, x_3\}$ be a 2-claw centered at c , such that w_i is adjacent (at least) to c and to x_i for $i = 1, 2, 3$. We will call $F_i = \{cw_i, w_ix_i\}$ a *finger* of C , so that the forbidden structure f_C corresponding to C is partitioned into three disjoint fingers. A 2-claw where one or more edges are marked is called *injured*. Clearly, in an injured 2-claw with five edges, only one of the fingers actually got injured and two fingers are still *healthy*. In an injured 2-claw with four edges, we still have at least one healthy finger left over.

In the following analysis, assume that we have already branched on all cycles up to length five (see the first heuristic priority below). Then, we can apply the following reduction rule for (injured) 2-claws:

- 3b. (injured) 2-claws: If e is an edge of degree one in a forbidden structure of size four, five or six corresponding to an (injured) 2-claw, and if e is incident to the center of the corresponding 2-claw, then mark e .

This rule is a weak substitute of the vertex domination rule of HITTING SET. It finally allows us to bound the running time as claimed.

Heuristic priorities. The second ingredient in the approach to hitting set problems described in the Introduction are so-called *heuristic priorities*. These describe “rules of thumb” according to which our algorithm will select an unmarked edge e to branch at. Here, *branch at e* means the following in the context of Alg. 1 that describes the procedure TLP:

```

if TLP( $G - e, k - 1, M$ ) then
    return YES
else if  $G[M \cup \{e\}]$  is acyclic then
    return TLP( $G, k, M \cup \{e\}$ )
else
    return NO
end if
    
```

Hence, either we put e into the biplanarization set, or we (possibly) mark e . If $G[M \cup \{e\}]$ contains a cycle but not $G[M]$, then e is the last unmarked edge on a long cycle and hence must be put into the biplanarization set (basically following the tiny structure rule); this is covered by the TLP($G - e, k - 1, M$)-branch.

The heuristic priorities are the following ones.

1. If possible, select a forbidden structure c that corresponds to a cycle of length at most five and continue at point 5.
2. Choose a forbidden structure f of minimum size $s(f)$. Let $c = c(f)$.
(Since the previous priority did not apply, f must correspond to a 2-claw.)
3. If there is another forbidden structure f' with $s(f') = s(f)$ and with $c(f) \cap c(f') \neq \emptyset$, then restrict $c = c(f) \cap c(f')$.
4. If the condition in the previous priority does not apply, then c corresponds to a (possibly injured) 2-claw.
 - (a) If $4 \leq s(f) \leq 5$, then restrict c further to collect only those unmarked edges that belong to healthy fingers.
 - (b) If $3 = s(f)$, then restrict c further to collect only those unmarked edges that do not belong to other injured 2-claws; if this means that c would become empty, restore $c = c(f)$.
5. Select $e \in c$ of maximal degree.

4.3 The algorithm and its correctness

Algorithm 1 A search tree algorithm for 2-LP, called TLP

Require: a graph $G = (V, E)$, a positive integer k , a set of marked edges M

Ensure: YES if there is a biplanarization set $B \subseteq E$, $|B| \leq k$ (and it will implicitly produce such a small biplanarization set then) or
NO if no such set exists.

Exhaustively apply the reduction rules 2. and 3a.’ (as well as 3b. if G contains no cycles of length at most five); the resulting instance is also called (G, k, M) .

if $k < 0$ **then**

 return NO

else if $E = M$ **then**

 return YES

else

 Select e according to heuristic priorities.

 Branch at e .

end if

The correctness of Alg. 1. To prove the soundness of rule 3b., we have to show that we will never miss out cycles this way. We therefore show the following assertions:

Proposition 1 *At most one edge per finger will be marked due to rule 3b.*

Proof: 3b. obviously only affects *one* 2-claw at a time, since only edges of degree one are marked. Per 2-claw, the rule triggers at most once per finger. \square

Proposition 2 *Cycles of length at least six that only consists of marked edges can never be created by running Alg. 1.*

To prove Proposition 2, the following observation is crucial.

Property 1 *Let $F = \{xy, yz\}$ be one finger of a (injured) 2-claw C with center x such that xy occurs only in one forbidden structure. Then, y has degree two.*

Proof: If the conclusion were false, there must be an edge yv in the given 2-LP instance. Hence, there is an (injured) 2-claw C' with center x which is like C , only having z replaced by v . This contradicts that xy has degree one, since xy participates both in C and in C' . \square

Proof of Proposition 2. As can be seen in the pseudo-code for “branching at e ”, during branching edges are only marked if marking them would not introduce cycles exclusively consisting of marked edges. Similarly, the application of rule 3a.’ is protected.

Therefore, the only situation that might miss out long cycles is introduced by applying reduction rule 3b. Let C be a (possibly injured) 2-claw that contains an edge $e = xy$ that is incident to the center x of C , belongs to a healthy finger $F = \{e, e'\}$ and has degree one (with respect to the collection of forbidden structures), so that 3b. triggers. If e belongs to some long cycle, then e' does so because of Property 1. Therefore, e can be marked without losing the capability of destroying all long cycles. \square

Based on the validity of the lemmas listed so far, it is now relatively straightforward to prove the correctness of Alg. 1 by induction on the number of unmarked edges of a graph. As a technical notice, we remark that reduction rule 3a.⁷ is essential for the correctness of the algorithm (by way of contrast to our general remarks on the role of reduction rules in HITTING SET algorithms in the preceding section). Namely, it completely deals with wreath components that were subject to postprocessing in the previously published algorithm in [7].

Theorem 3 *Given a graph G and an integer parameter k , Alg. 1 when called with $TLP(G, k, \emptyset)$, returns YES iff $\text{bpr}(G) \leq k$, and it returns NO otherwise.*

4.4 The time analysis of Algorithm 1

Putting up recurrences. Now, let us turn to the time analysis of the procedure. We will follow the ideas explained in [10] for 3-HITTING SET. Let $T(k)$ denote the number of leaves in a worst-case search tree for Alg. 1, which incidentally also is the worst-case for the number of solutions returned by the routine. More distinctly, let $T^\ell(k)$ denote the situation of a search tree assuming that at least ℓ forbidden structures in the given instance (with parameter k) have size five. Of course, $T(k) \leq T^0(k)$. We analyze the recurrences for T^0 , T^1 and T^2 .

Lemma 6 $T^0(k) \leq T^0(k-1) + T^2(k)$.

Proof: Due to the reduction rule 3b., the 2-LP instance G contains an edge e of degree 2 in a forbidden structure f of size 6, since f represents a healthy 2-claw. Hence, there is another healthy 2-claw corresponding to a forbidden structure f' with $e \in f \cap f'$. One branch is that e is put into the biplanarization set. The size of the corresponding subtree can be estimated by $T^0(k-1)$. If e is not put into the biplanarization set, then e is marked and hence at least two forbidden structures of size five are created with cores $f \setminus \{e\}$ and $f' \setminus \{e\}$. Therefore, the size of that subtree is bounded above by $T^2(k)$. \square

Let us first do a simplified analysis to see that our venue is worth pursuing at all; here, we basically ignore reduction rules in the derivation of the following simple branching lemmas.

Lemma 7 $T^1(k) \leq 5T^0(k-1)$.

Proof: This is a consequence of trivial branching at the five edges collected in a forbidden structure of size five. \square

Lemma 8 $T^2(k) \leq \max\{25T^0(k - 2), T^0(k - 1) + 16T^0(k - 2), 2T^0(k - 1) + 9T^0(k - 2), 3T^0(k - 1) + 4T^0(k - 2), 4T^0(k - 1) + T^0(k - 2)\}$.

Proof: Let f_1 and f_2 be the two forbidden structures of size five. We have to consider some sub-cases:

1. If $c(f_1) \cap c(f_2) = \emptyset$, then we get by the analysis of Lemma 7, keeping in mind that by branching on say f_1 we still keep the low-size forbidden structure f_2 on which we would then continue branching:

$$T^2(k) \leq 25T^0(k - 2).$$

2. If $|c(f_1) \cap c(f_2)| = 1$, our heuristic priorities let us branch at $e \in c(f_1) \cap c(f_2)$. If we take e into the biplanarization set, we get a $T^0(k - 1)$ -branch. If we don't take e into the biplanarization set, we are left with two forbidden structures of size four, namely $c(f'_1) = c(f_1) \setminus \{e\}$ and $c(f'_2) = c(f_2) \setminus \{e\}$. Trivial branching on those forbidden structures (which will be done according to the heuristic priorities) gives sixteen $T^0(k - 2)$ -branches. Hence,

$$T^2(k) \leq T^0(k - 1) + 16T^0(k - 2).$$

3. More generally, if $|c(f_1) \cap c(f_2)| = \ell$, we would first have ℓ many $T^0(k - 1)$ -branches according to priority 3. Taking none of the edges into the biplanarization set leaves us with two structures each of size $5 - \ell$; trivial branching (according to priority 2.) gives $(5 - \ell)^2$ many $T^0(k - 2)$ -branches.

Solving recurrences. We now look for an estimate $T^0(k) \leq c^k$, where we always assume $T^\ell(0) = 1$ as an anchor of the recursion:

Assuming $T^2(k) \leq 25T^0(k - 2)$.

$$T^0(k) \leq T^0(k - 1) + T^2(k) \leq T^0(k - 1) + 25T^0(k - 2)$$

The ansatz $T^0(k) = c^k$ then shows that we have to find the largest real zeros of the following polynomial: $c^2 - c - 25$. Hence, $c \leq \boxed{5.5250}$.⁴

Assuming $T^2(k) \leq T^0(k - 1) + 16T^0(k - 2)$.

$$T^0(k) \leq T^0(k - 1) + T^2(k) \leq 2T^0(k - 1) + 16T^0(k - 2)$$

$T^0(k) = c^k$ yields $c = 1 + \sqrt{17} \leq 5.1232$.

⁴We include some mathematical explanation for this approach in the appendix.

Assuming $T^2(k) \leq 2T^0(k - 1) + 9T^0(k - 2)$.

$$T^0(k) \leq T^0(k - 1) + T^2(k) \leq 3T^0(k - 1) + 9T^0(k - 2)$$

$$T^0(k) = c^k \text{ yields } c \leq 4.8542.$$

Assuming $T^2(k) \leq 3T^0(k - 1) + 4T^0(k - 2)$.

$$T^0(k) \leq T^0(k - 1) + T^2(k) \leq 4T^0(k - 1) + 4T^0(k - 2)$$

$$T^0(k) = c^k \text{ again yields } c \leq 4.8285.$$

Assuming $T^2(k) \leq 4T^0(k - 1) + T^0(k - 2)$.

$$T^0(k) = T^0(k - 1) + T^2(k) \leq 5T^0(k - 1) + T^0(k - 2)$$

$$T^0(k) = c^k \text{ yields } c = 2.5 + \sqrt{7.25} \leq 5.1926.$$

After this preliminary analysis, we can draw the following conclusions:

- The approach we used looks promising, since we can rather easily improve on the earlier claimed search tree size of $\mathcal{O}(6^k)$.
- By far the worst case is encountered in the only situation when we actually use the (trivial) estimate for $T^1(k)$ as derived in Lemma 7. More specifically, that case (marked by a box in the previous derivations) yields a search tree estimate worse than $\mathcal{O}(5.5^k)$, while in all other cases, the estimates are better than $\mathcal{O}(5.2^k)$.

This gives us an excellent hint on how to further improve on the estimates of the search tree sizes: we “only” have to analyze the T^1 -branchings more thoroughly.

Refined time analysis of Alg. 1. A more involved analysis of the T^1 - and T^2 -branches as well as some algebra for solving the recursions (included in the appendix), shows:

Lemma 9 $T^1(k) \leq 2T^0(k - 1) + 2T^1(k - 1) + T^2(k - 1)$.

This readily improves our estimate for $T^2(k)$ as follows:

Lemma 10 $T^2(k) \leq \max\{2T^1(k - 1) + 3T^2(k - 1), T^0(k - 1) + 16T^0(k - 2), 2T^0(k - 1) + 9T^0(k - 2), 3T^0(k - 1) + 4T^0(k - 2), 4T^0(k - 1) + T^0(k - 2)\}$.

We finally notice that we can implement the branching in Alg. 1 by actually looking for vertices on non-leaf-degree at least three instead of looking for forbidden structures of size at most six (in particular, also referring to cycles of length five) based on the following observations:

- We only need the fact that small cycles are already worked on “locally” in the analysis included in the appendix.

- Also, the heuristic priorities are only analyzed based on a local scheme, i.e., following up the way they trigger further (local) situations.

Hence, we can maintain all necessary local information, as it is indicated within [21] when describing an implementation of the algorithm from [7].

Theorem 4 *Given a graph G and an integer k , Alg. 1 can be implemented such that it determines if $\text{bpr}(G) \leq k$ in $\mathcal{O}(k^2 \cdot 5.1926^k + |G|)$ time, when applied to the problem kernel derived in [7].*

To prove these results, the following lemma is important, which is also interesting from a structural point of view on its own account; this also explains why we considered 5-cycles as forbidden structures.

Lemma 11 *In a graph without cycles up to length five, each 2-claw is vertex-induced. In particular, the center is uniquely determined.*

By changing the heuristic priorities in one case and by using a generalization of rule 3b., we can improve the base further to 5.1844. Details on this marginal improvement can be found in the report version [11].

5 1-Layer Planarization: Kernelization and branching algorithms

5.1 Combinatorial properties

Preliminary results. The next two results from [7] give important properties for π -biplanar graphs.

Lemma 12 *A bipartite graph $G = (A, B; E)$ with a fixed permutation π of A is π -biplanar if and only if G is acyclic and the following condition (*) holds.*

For every path (x, v, y) of G with $x, y \in A$, and for every vertex $u \in A$ between x and y in π , the only edge incident to u (if any) is uv .

Let us say that an edge e of a bipartite graph G *potentially violates condition (*)* if, using the notation of condition (*), $e = e_i$ for $i = 1, 2, 3$, where $e_1 = xv$ or $e_2 = vy$ or $e_3 = uz$ for some u strictly between x and y in π such that $z \neq v$. We will also say that e_1, e_2, e_3 (together) *violate condition (*)*.

Let $G = (A, B; E)$ be a bipartite graph with a fixed permutation of A that satisfies condition (*). Let $H = K_{2,p}$ be a complete bipartite subgraph of G with $H \cap A = \{x, y\}$, and $H \cap B = \{v \in B : vx \in E, vy \in E, \deg_G(v) = 2\}$, and $|H \cap B| = p$. Then H is called a p -diamond. Every cycle of G is in some p -diamond with $p \geq 2$.

Lemma 13 *If $G = (A, B; E)$ is a bipartite graph and π is a permutation of A satisfying condition (*), then $\text{bpr}(G, \pi) = \sum_{\text{maximal } p\text{-diamonds of } G} (p - 1)$.*

Basically, we are going to further analyze the following lemma from [7]:

Lemma 14 *If $G = (A, B; E)$ is a bipartite graph and π is a permutation of A that satisfies condition (*), then all the cycles of G are 4-cycles and any two non-edge-disjoint cycles share exactly two edges. Moreover, the degree of any vertex in B that appears in a cycle is exactly two.*

More precisely, we are going to prove:

Lemma 15 *Let $G = (A, B; E)$ be a bipartite graph and let π be a permutation of A . Then, every cycle C of length $2\ell > 4$ has at most two edges not violating condition (*).*

More precisely, let u be the leftmost vertex of C on A and v be the rightmost vertex of C on A . Then, there is at most one edge incident to u and at most one edge incident to v that potentially violate condition ().*

Proof: Let u be the leftmost vertex of C on A and v be the rightmost vertex of C on A . C contains two disjoint paths P and P' between u and v .

Claim 1: Any edge on C not incident with u or v potentially violates (*).

Namely, assume that $e = \{x, y\}$ belongs to P , $x \in A \setminus \{u, v\}$. Then, consider the vertex $x' <_{\pi} x$ of A that is closest to x among all vertices of P' . Vertex x' exists, since $x \neq u$. By the choice of x' , there is a vertex $z' >_{\pi} x$ on P' at distance two from x' on P' , since $x \neq v$. Let y' be the vertex on B that interconnects x' and z' on P' . Then, (x', y', z') and e violate condition (*). This shows the claim. \diamond

Consider now the case that $e = \{x, y\}$ belongs to P with $x = v$. Let x' be defined as above. Vertex y has a neighbor z on P , $z \neq u$, since $2\ell > 4$. If $z <_{\pi} x'$, (x, y, z) and the edge $\{x', y'\}$ violate (*) for any neighbor y' of x' on P' . Otherwise, $x' <_{\pi} z$. Consider the path P' as starting at v : (v, y', z', \dots) . By definition of x' , $z' \leq_{\pi} x' <_{\pi} z <_{\pi} x = v$. Hence, (v, y', z') together with $\{z, y\}$ violate (*). We can conclude:

Claim 2: There is at most one edge incident to v that does not potentially violate (*). \diamond

By symmetry, an analogous statement is true for u . \square

5.2 Kernelization algorithm

Kernelization rules. We are now going to derive a kernelization algorithm for 1-LAYER PLANARIZATION. As announced, we will do this by first translating the HITTING SET reduction rules that were useful for kernelization, which were in particular the modified large degree rules, as well as the isolated vertex rule.

According to Lemma 12 (as well as the proof of Lemma 13 for the last two rules), the following reduction rules are sound, given an instance $(G = (A, B; E), \pi, k)$ of 1-LP.

1L-RR-edge: If $e \in E$ does not participate in any cycle and does not potentially violate condition (*), then remove e from the instance (keeping the same parameter k).

1L-RR-large: Notice that we can translate the modified large degree rules described after the formulation of Theorem 1, since that reduction is only dealing with vertex-induced hypergraphs. We omit the further details here.

1L-RR-isolate: If $v \in A \cup B$ has degree zero, then remove v from the instance and modify π appropriately (keeping the same parameter k).

Let $E_\star \subseteq E$ be all edges that potentially violate condition (*). Let $E_\circ \subseteq E$ be all edges that participate in cycles. Let G_{4c} be generated by those edges from $E_\circ \setminus E_\star$ that participate in 4-cycles. By construction, G_{4c} satisfies condition (*). Lemma 13 shows that the next reduction rule can be applied in polynomial time:

1L-RR-4C: If $\text{bpr}(G_{4c}, \pi) > k$, then NO.

Lemma 16 *Let $G = (A, B; E)$ be a bipartite graph and let π be a permutation of A . Let $v \in B$. Then, there is at most one edge e incident to v that does not potentially violate condition (*) and participates in cycles of length > 4 .*

Proof: Let $C = (v_1, v_2, \dots, v_{2\ell})$ and $C' = (v'_1, v'_2, \dots, v'_{2\ell'})$ be “long cycles”, such that (w.l.o.g.) $e = v_2v_3$ and $e' = v'_2v'_3$, with $e \neq e'$, and $v = v_2 = v'_2$. Assume that e and e' both do not potentially violate condition (*). W.l.o.g., we can further assume that $v_1 <_\pi v_3$. We now consider all possible positions of v'_1 with respect to $v_1 <_\pi v_3$.

- If $v_1 <_\pi v'_1 <_\pi v_3$, then (v_1, v_2, v_3) together with $v'_1v'_{2\ell'}$ do not satisfy condition (*), so that e potentially violates (*), contradicting our assumptions. Similarly, $v'_1 <_\pi v_1 <_\pi v_3$ can be ruled out.
- If $v'_1 \leq_\pi v_1 <_\pi v_3$, then both cases $v_3 <_\pi v'_3$ and $v'_3 <_\pi v_3$ lead to contradictions (symmetric to the previous situation). But if $v_3 = v'_3$, then $e = e'$.
- $v'_1 = v_3$ is ruled out by Lemma 15, since then e and e' would be two subsequent edges on C' which do not potentially violate (*).
- Hence, $v_1 <_\pi v_3 <_\pi v'_1$ remains as the only possibility. Interchanging the roles of C and C' in the argument, $v'_1 <_\pi v'_3$ is ruled out. Hence, $v_1 <_\pi v_3, v'_3 <_\pi v'_1$ with $v_3 \neq v'_3$ since $e \neq e'$. Consider (a) $v_1 <_\pi v_3 <_\pi v'_3 <_\pi v'_1$. Then, (v_3, v, v'_1) together with $v'_3v'_4$ violate (*), contradicting our assumption on e . If (b) $v_1 <_\pi v'_3 <_\pi v_3 <_\pi v'_1$, we get a similar violation of our assumption on e' . □

Theorem 5 *Let $G = (A, B; E)$ be a bipartite graph, π be a permutation of A and $k \geq 0$. Assume that none of the reduction rules applies to the 1-LP instance (G, π, k) . Then, $|E| = \mathcal{O}(k^3)$. The kernel can be found in time $\mathcal{O}(|G|^2)$.*

Proof: Since no reduction rules apply to (G, π, k) , $E = E_\circ \cup E_\star$. Our rules 1L-RR-edge and 1L-RR-large correspond to the rules from Theorem 1. Hence, $|E_\star| = \mathcal{O}(k^3)$.

If $e = xy \in E_\circ \setminus E_\star$ with $y \in B$ does not belong to a 4-cycle, then Lemma 16 shows that there is no other edge $zy \in E_\circ \setminus E_\star$. But since $xy \in E_\circ$, there must

be some “continuing edge” zy on the long circle xy belongs to, so that $zy \in E_\star$ follows. We can take zy as a *witness* for xy . By Lemma 16, zy can witness for at most one edge from $E_\circ \setminus E_\star$ incident to y and not participating in a 4-cycle.

This allows us to partition E_\circ into three disjoint subsets: (a) $E_\circ \cap E_\star$, (b) $E_{4c} = \{e \in E_\circ \setminus E_\star \mid e \text{ participates in a 4-cycle}\}$: there can be at most $4k$ such edges according to 1L-RR-4C and Lemma 13, and (c) $E_\circ \setminus E_{4c}$: according to our preceding reasoning, there are at most $|E_\star|$ many of these edges. \square

Remark 1 (annotated kernel) *If we are willing to accept annotated instances as the result of a kernelization, we can get a slightly better kernel bound of $|E| \leq k^3$ with the same argument, see Sec. 3. Then, besides the graph $G = (A, B; E)$, we would have a set \mathcal{E} of pairs of edges such that, for all $\{e_1, e_2\} \in \mathcal{E}$, e_1 or e_2 must be removed (and this will be done by the subsequent branching).*

5.3 Bounded search tree

Dujmović *et al.* obtained the following result in [7]:

Theorem 6 *Given a bipartite graph $G = (A, B; E)$, a fixed permutation π of A , and integer k , there is an algorithm that determines if $\text{bpr}(G, \pi) \leq k$ in $\mathcal{O}(3^k \cdot |G|)$ time.*

Can we further improve on this algorithm? Firstly, it is clear that we can combine the search tree algorithm with the kernelization algorithm described above. Here, the annotated kernel version would have the advantage that we could start branching at the edge pairs in the annotation set \mathcal{E} ; this would give a nice 2^k -branching.

But furthermore, observe that the search tree algorithm basically branches on all members of E_\star , trying to destroy the corresponding triples of edges violating condition (*). This means that we again take ideas stemming from solutions of the naturally corresponding instance of 3-HITTING SET. Unfortunately again, we cannot simply “copy” the currently best search tree algorithm for 3-HITTING SET [10, 19], running in time $\mathcal{O}(k \cdot 2.179^k + |G|)$, since destroying triples of edges violating condition (*) might incidentally also destroy more or less of the 4-cycles. As explained in the 2-LP case, the problem is again the vertex domination rule. In order to gain anything against the previously sketched algorithm 1-Layer Bounded Search Tree, we must somehow at least *avoid branching on vertices of degree one contained in hyperedges of size three*.

Firstly, we can prove a lemma that shows that, whenever we have branched on all hyperedges of size three in the 3-HITTING SET instance (This corresponds to situations violating condition (*) in the original 1-LP instance) that contain vertices of degree at least two, then we have already destroyed all “large” cycles. More precisely, we show:

Lemma 17 *Let $G = (A, B; E)$ be a bipartite graph and π be a fixed permutation of A . If C is a cycle of length six or more, then C contains two vertex-disjoint paths (x, y, z) and (b, a, c) such that $x, a, z \in A$ and $b, y, c \in B$ and $x_\pi < a <_\pi z$.*

Proof: Let x be the leftmost vertex of C in A . On C , the neighbors of x be $\{x, a, z\}$. Assume $x <_{\pi} z$. By definition, $x <_{\pi} a$. Call the common neighbor of x and z on C y , and the two neighbors of a on C b, c . This shows the claim. \square

Then, we investigate the possible interaction between a cycle of length four and a structure violating $(*)$, after having “destroyed” all “mutually interacting” structures violating $(*)$.

Lemma 18 *Let $G = (A, B; E)$ be a bipartite graph and π be a fixed permutation of A . Let $C = \{ab, bc, cd, da\}$ be a sequence of edges forming a 4-cycle.*

Assume that the following condition $(+)$ is true: if $h = \{e_1, e_2, e_3\}$ and $h' = \{e'_1, e'_2, e'_3\}$ are two situations violating $()$, then $h \cap h' = \emptyset$.*

Then, there is at most one hyperedge h_C —among the hyperedges modeling situations violating $()$ —such that $C \cap h_C \neq \emptyset$.*

The rather technical proof is contained in the Appendix.

Our Alg. 2 is worked out again according to the HITTING SET methodology. We therefore inherit some terminology from the previous section: A *forbidden structure* is either a set of three edges $f = \{e_1, e_2, e_3\}$ violating $(*)$ or a 4-cycle (this viewpoint makes the presentation of the algorithm easier). The *core* $c(f)$ of a forbidden structure f is $c(f) = f \setminus M$. Accordingly, the notions of $s(f) = |c(f)|$ and the degree of an edge e are understood. Let

$$F_d = \{f \mid f \text{ is a forbidden structure} \wedge s(f) = d\} \quad \text{for } d = 2, 3, 4;$$

$$d_{\min} = \min_{d=2,3,4} \{d \mid F_d \neq \emptyset\} \text{ and}$$

$$F_d^* = \{f \in F_d \mid \exists f' \in F_{d_{\min}} \setminus \{f\} : c(f) \cap c(f') \neq \emptyset\}.$$

The list of heuristic priorities according to which we select the edge e to branch at is as follows (in addition, we describe a forbidden structure f that contains e):

1. Determine $d = d_{\min}$. If $d = 3$ and $F_d^* = \emptyset$, then let $d = 4$.
2. If $F_d^* \neq \emptyset$, then choose $f \in F_d$ and $f' \in F_{d_{\min}} \setminus \{f\}$ and let $c = c(f) \cap c(f')$. Otherwise, choose $f \in F_d$ and let $c = c(f)$.
3. Select $e \in c$ of maximal degree.

Furthermore, we make use of the following rules that can be seen as translating the tiny edge rule and the vertex domination rule, respectively; observe that (already upon branching) we now also mark certain edges. In fact, we require that the translated tiny edge rule will be always applied prior to applying any other rules.

1. If e is the last unmarked edge in a forbidden structure, then delete e (and decrement the parameter).
2. If e is an edge from a forbidden structure such that e is not part of another forbidden structure, then mark e .

Algorithm 2 A search tree algorithm for 1-LP, called OLP

Require: a bipartite graph $G = (A, B; E)$, a permutation π of A , a positive integer k , a set of marked edges M

Ensure: NO if $\text{bpr}(G, \pi) > k$; otherwise, YES (and it will implicitly produce such a small biplanarization set then)

Exhaustively apply the reduction rules; the resulting instance is also called (G, π, k, M) .

if $k < 0$ **then**

 return NO

else if $E = M$ **then**

 return YES

else

 Select e in forbidden structure f according to heuristic priorities.

if $\text{OLP}(G - e, \pi, k - 1, M)$ **then**

 return YES

else if f is not a 4-cycle **then**

 return $\text{OLP}(G, \pi, k, M \cup \{e\})$

else

 return NO

end if

end if

For the validity of these rules, Lemma 14 and our interpretation of forbidden structures (including 4-cycles) is essential.

In the algorithm depicted in Alg. 2, we again use a set of *marked edges* M to mark edges which (according to our previous branching) we shall *not* put into the biplanarization set. This part of the input is therefore initialized with \emptyset at the very beginning.

Theorem 7 1-LP can be solved in $\mathcal{O}(k^3 \cdot 2.5616^k + |G|^2)$ time.

Proof: Let us firstly comment on the correctness of the algorithm. We deviate from the standard scheme of HITTING SET algorithms only when we select an edge e from a 4-cycle for branching, because we suppress the otherwise necessary case not taking e into the planarization set. This is justified as follows. By the heuristic priorities, a 4-cycle will be selected for branching only if $F_2 = \emptyset$ and if either $F_3 = \emptyset$ or if $F_3 \neq \emptyset$ but $F_3^* = \emptyset$. Since we are always dealing with reduced instances, isolated forbidden structures (in particular, isolated 4-cycles) will not show up when the algorithm branches. Hence, $F_4^* = F_4$ if $F_2 = \emptyset$. Therefore, the case $F_3 = \emptyset$ implies $F_4 = \emptyset$ and hence will not trigger branching; in fact, if $F_2 = F_3 = F_4 = \emptyset$, then the algorithm will have already terminated, since all edges will then be marked by the reduction rules (and possibly, the parameter value would be below zero); therefore, this case cannot show up in the branching stage of the algorithm. When $F_3 \neq \emptyset$ but $F_3^* = \emptyset$, Lemma 18 assures that we have to take out the edge that is selected according to the heuristic priorities in

order to deal with the remaining forbidden structures of size three.

Therefore, branching only actually takes place when we “solve” the corresponding 3-HITTING SET instance. During these recursions, we can always assume that, whenever we branch at forbidden structures of size three, there is some element contained in that forbidden structure which actually participates in at least two forbidden structures. Namely, a forbidden structure that does not interact with any other forbidden structure will be dealt with by the reduction rules (more specifically, by the translations of the vertex domination and the tiny edge rule). A forbidden structure of size three that only interacts with 4-cycles will be dealt with as described in the preceding paragraph.

Let us now turn to the time analysis. To this end, let $T^\ell(k)$ denote the situation of a search tree assuming that at least ℓ forbidden structures in the given instance (with parameter k) have a size of (at most) 2. Of course, $T^0(k)$ is the worst case tree size. We again analyze the recurrences for T^0 , T^1 and T^2 , and we use the notions of core and size of a forbidden structure similarly to the 2-LP case.

The reasoning of Lemma 6 basically transfers to 1-LP, yielding

$$T^0(k) \leq T^0(k - 1) + T^2(k).$$

Namely, we face the situation that we have no forbidden structures of size two, i.e., $F_2 = \emptyset$. According to the argument given in the first paragraph of this proof, we then must have $F_3 \neq \emptyset$ and $F_3^* \neq \emptyset$ in order to actually trigger the binary branching process. (More formally, if the binary branching is not triggered, we get the estimate $T^0(k) \leq T^0(k - 1)$ which is clearly better than what we claim and can therefore be neglected). Since $F_3^* \neq \emptyset$ and $F_2 = \emptyset$, the heuristic priorities will select an edge e to branch at that is contained in two forbidden structures of size three. Hence, if e is not taken into the biplanarization set, then we encounter at least two forbidden structures of size two (as claimed).

For T^1 , we cannot claim to “gain” any new forbidden structures of size two. Therefore, a trivial branching gives:

$$T^1(k) \leq 2T^0(k - 1).$$

For T^2 , we distinguish two sub-cases, considering two forbidden structures f_1, f_2 of size two:

1. $c(f_1) \cap c(f_2) = \emptyset$. Then, trivial branching gives:

$$T^2(k) \leq 4T^0(k - 2).$$

2. $\exists e \in c(f_1) \cap c(f_2)$. Branching at e (which our algorithm will do) then yields:

$$T^2(k) \leq T^0(k - 1) + T^0(k - 2).$$

The first sub-case leads to:

$$T^0(k) \leq T^0(k - 1) + 4T^0(k - 2) \leq 2.5616^k.$$

The second sub-case gives:

$$T^0(k) \leq T^0(k-1) + (T^0(k-1) + T^0(k-2)) \leq 2.4143^k.$$

So, the first sub-case yields the worst case. \square

6 Conclusion

In this paper we have presented two methods for producing \mathcal{FPT} algorithms in the context of 2-layer and 1-layer planarization. The smaller exponential bases (in comparison with [7]) are due to the tight relations with HITTING SET, as we exhibited. For small values of k , our algorithms provide a feasible method for the solution of these \mathcal{NP} -complete problems.

With the results in [7, 8], we have now good kernelization and search tree algorithms for three types of “layered planarization” problems:

1. For 2-LP, we got an $\mathcal{O}(k^2 \cdot 5.1926^k + |G|)$ algorithm and a kernel size $\mathcal{O}(k)$.
2. For 1-LP, we found an $\mathcal{O}(k^3 \cdot 2.5616^k + |G|^2)$ algorithm and a kernel size $\mathcal{O}(k^3)$.
3. For 1-LAYER CROSSING MINIMIZATION, we obtained an $\mathcal{O}(1.4656^k + k|G|^2)$ algorithm and a kernel size $\mathcal{O}(k^2)$, where k is now the number of crossings.

For 2-LAYER CROSSING MINIMIZATION, the (more general) results of [6] only give an $\mathcal{O}(2^{32(2+2k)^3} |G|)$ algorithm, which should be further improvable.

In [18], also the importance of *weighted variants* of 2-LAYER PLANARIZATION and 1-LAYER PLANARIZATION is mentioned. If one likes to attack such problems from a parameterized point of view, it is customary to assume that all weights are at least one; the parameter is then an upperbound on the weight of the edges taken out to planarize the given instance. The $\mathcal{O}^*(3^k)$ and $\mathcal{O}^*(6^k)$ search tree algorithms (for 1-LP and 2-LP, respectively) that are essentially based on enumerating minimal solutions of the underlying HITTING SET problem will also work in the weighted case, especially with the same time analysis. However, the task of improving these running times becomes even more challenging, since already in the simplest *weighted* HITTING SET case, i.e., WEIGHTED VERTEX COVER, the degree-one case becomes non-trivial, see [20].

Finally, it would be interesting to see if the \mathcal{FPT} route to exact algorithms for 1-LP and 2-LP can be combined with other approaches, as [2, 18].

Acknowledgments

We are grateful for discussion of this topic with V. Dujmović. Moreover, the numerous comments of the referees of JGAA were very helpful to improve on the presentation of the paper.

References

- [1] F. Abu-Khzam, H. Fernau, and M. A. Langston. Asymptotically faster algorithms for parameterized face cover. In H. Broersma, M. Johnson, and S. Szeider, editors, *Algorithms and Complexity in Durham ACiD 2005*, volume 4 of *Texts in Algorithmics*, pages 43–58. King’s College Publications, 2005.
- [2] L. Carmel, D. Harel, and Y. Koren. Combining hierarchy and energy for drawing directed graphs. *IEEE Transactions on Visualization and Computer Graphics*, 10(1):46–57, 2004.
- [3] P. Damaschke. Parameterized enumeration, transversals, and imperfect phylogeny reconstruction. In R. Downey, M. Fellows, and F. Dehne, editors, *International Workshop on Parameterized and Exact Computation IWPEC 2004*, volume 3162 of *LNCS*, pages 1–12. Springer, 2004.
- [4] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.
- [5] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
- [6] V. Dujmović, M. R. Fellows, M. Hallett, M. Kitching, G. Liotta, C. McCartin, N. Nishimura, P. Ragde, F. A. Rosamond, M. Suderman, S. Whitesides, and D. R. Wood. On the parameterized complexity of layered graph drawing. In F. M. auf der Heide, editor, *9th Annual European Symposium on Algorithms ESA*, volume 2161 of *LNCS*, pages 488–499. Springer, 2001.
- [7] V. Dujmović, M. R. Fellows, M. Hallett, M. Kitching, G. Liotta, C. McCartin, N. Nishimura, P. Ragde, F. A. Rosamond, M. Suderman, S. Whitesides, and D. R. Wood. A fixed-parameter approach to two-layer planarization. In P. Mutzel, M. Jünger, and S. Leipert, editors, *9th International Symposium on Graph Drawing GD 2001*, volume 2265 of *LNCS*, pages 1–15. Springer, 2002.
- [8] V. Dujmović, H. Fernau, and M. Kaufmann. Fixed parameter algorithms for one-sided crossing minimization revisited. In G. Liotta, editor, *11th International Symposium on Graph Drawing GD 2003*, volume 2912 of *LNCS*, pages 332–344. Springer, 2004.
- [9] V. Estivill-Castro, M. R. Fellows, M. A. Langston, and F. A. Rosamond. FPT is P-time extremal structure I. In H. Broersma, M. Johnson, and S. Szeider, editors, *Algorithms and Complexity in Durham ACiD 2005*, volume 4 of *Texts in Algorithmics*, pages 1–41. King’s College Publications, 2005.
- [10] H. Fernau. A top-down approach to search-trees: Improved algorithmics for 3-Hitting Set. Technical Report TR04-073, Electronic Colloquium on Computational Complexity ECCC, 2004.

- [11] H. Fernau. Two-layer planarization: Improving on parameterized algorithmics. Technical Report TR04-078, Electronic Colloquium on Computational Complexity ECCC, 2004.
- [12] H. Fernau. *Parameterized Algorithmics: A Graph-Theoretic Approach*. Habilitationsschrift, Universität Tübingen, Germany, 2005. Submitted.
- [13] H. Fernau. Parameterized algorithmics for linear arrangement problems. Submitted for publication, Aug. 2005.
- [14] H. Fernau. Two-layer planarization: Improving on parameterized algorithmics. In P. Vojtáš, M. Bieliková, B. Charron-Bost, and O. Sýkora, editors, *SOFSEM*, volume 3381 of *LNCS*, pages 137–146. Springer, 2005.
- [15] F. V. Fomin, F. Grandoni, and D. Kratsch. Measure and conquer: domination – a case study. Technical Report 294, Department of Informatics, University of Bergen (Norway), Apr. 2005. To appear in *Proc. ICALP 2005*.
- [16] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. John Wiley, 1990.
- [17] P. Mutzel. An alternative method to crossing minimization on hierarchical graphs. *SIAM J. Optimization*, 11(4):1065–1080, 2001.
- [18] P. Mutzel and R. Weiskircher. Two-layer planarization in graph drawing. In K.-Y. Chwa and O. H. Ibarra, editors, *Algorithms and Computation — 9th International Symposium ISAAC’98*, volume 1533 of *LNCS*, pages 69–78, 1998.
- [19] R. Niedermeier and P. Rossmanith. An efficient fixed-parameter algorithm for 3-Hitting Set. *Journal of Discrete Algorithms*, 1:89–102, 2003.
- [20] R. Niedermeier and P. Rossmanith. On efficient fixed parameter algorithms for weighted vertex cover. *Journal of Algorithms*, 47:63–77, 2003.
- [21] M. Suderman and S. Whitesides. Experiments with the fixed-parameter approach for two-layer planarization. In G. Liotta, editor, *11th International Symposium on Graph Drawing GD 2003*, volume 2912 of *LNCS*, pages 345–356. Springer, 2003.
- [22] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Systems Man Cybernet.*, 11(2):109–125, 1981.
- [23] M. Wahlström. Exact algorithms for finding minimum transversals in rank-3 hypergraphs. *Journal of Algorithms*, 51:107–121, 2004.
- [24] M. S. Waterman and J. R. Griggs. Interval graphs and maps of DNA. *Bull. Math. Biol.*, 48(2):189–195, 1986.

Appendix: Explanation of the analysis approach

Our approach to analyze search trees based on multiple recurrences is justified on the following grounds:

Generally speaking, the technical approach to recurrences is via generating functions. In the simple first example of a recurrence analyzed after Lemma 8, we are looking for a closed form of the function $G(z)$ given by $G(z) = \sum_k T^0(k)z^k$, i.e.,

$$\begin{aligned} G(z) &= \sum_k T^0(k)z^k \\ &= \sum_k T^0(k-1)z^k + 25 \sum_k T^0(k-2)z^k + 1 \\ &= zG(z) + 25z^2G(z) + 1 \\ \rightsquigarrow G(z) &= \frac{1}{-25z^2 - z + 1} \end{aligned}$$

It is known that then $T^0(k)$ can be expressed in a closed form as $T^0(k) = a_1\rho_1^k + a_2\rho_2^k$, where ρ_i are the roots of the reflected polynomial $z^2 - z - 25$ (which is also sometimes called the *characteristic polynomial* of that linear recurrence), i.e., $\rho_1 = .5 + \sqrt{25.25}$ and $\rho_2 = .5 - \sqrt{25.25}$. Obviously, ρ_1 is the largest positive root, so that $T^0(k) = \mathcal{O}(\rho_1^k) \leq \mathcal{O}(5.5250^k)$ as claimed.

This justifies our approach taken above, where this largest root has been simply computed using a numerical approximation program.

When facing mutually recurrent linear inequalities of a more general form, i.e., in our case,

$$\begin{aligned} T^0(k) &\leq \max\{f_1^0(k), f_2^0(k), \dots, f_{n_0}^0(k)\}, \\ T^1(k) &\leq \max\{f_1^1(k), f_2^1(k), \dots, f_{n_1}^1(k)\}, \text{ and} \\ T^2(k) &\leq \max\{f_1^2(k), f_2^2(k), \dots, f_{n_2}^2(k)\}, \end{aligned}$$

where $f_i^\ell(k)$ are positive linear combinations of some $T^j(k')$, we might be happy that we can transform this into a single inequality of the form

$$T^0(k) \leq \max\{g_1^0(k), g_2^0(k), \dots, g_{n_0}^0(k)\},$$

where $g_i^0(k)$ are positive linear combinations of some $T^0(k')$. Then, we can find the roots of the characteristic polynomials of each of the recurrences $T^0(k) \leq g_i^0(k)$; the largest c of all these roots will satisfy all inequalities. In fact, this is possible with all the algorithmic analysis from the main text body.

In the general case, we can first transform the mutually recurrent linear inequalities into a set of mutually recurrent linear inequalities without the maximum operator. In the case sketched above, this gives $n_0n_1n_2$ inequalities, from

$$T^0(k) \leq f_1^0(k), T^1(k) \leq f_1^1(k), T^2(k) \leq f_1^2(k) \text{ to}$$

$$T^0(k) \leq f_{n_0}^0(k), T^1(k) \leq f_{n_1}^1(k), T^2(k) \leq f_{n_2}^2(k).$$

Each such mutually recurrent linear inequality system can be solved by the approach that we get the “best estimates” by assuming equalities in all inequalities. This should give an estimate $T^0(k) \leq c_j^k$ for each of the mutually recurrent linear inequalities. Now, we can pursue this approach by setting

$$c = \max\{c_j \mid 1 \leq j \leq n_0 n_1 n_2\}.$$

The claim would be then that $T^0(k) = c^k$ allows to satisfy all given inequalities. In the more general situation, we should however also find expressions for T^1 and for T^2 to validate this claim.

Here, it is helpful to know that the ansatz $T^0(k) = c^k$ can be supplemented by $T^1(k) = \alpha_1 c^k$ and $T^2(k) = \alpha_2 c^k$, for suitable $0 < \alpha_1 \leq \alpha_2 \leq 1$ depending on c . Notice that there is quite a practical interpretation of the α_i : $(1 - \alpha_i)$ gives the fraction of running time that is saved (at least) when facing the T^i -situation instead of the T^0 -situation, since the search tree size shrinks down by this factor.

Appendix: the branches of Alg. 1—more details

Proof of the analysis of the branches

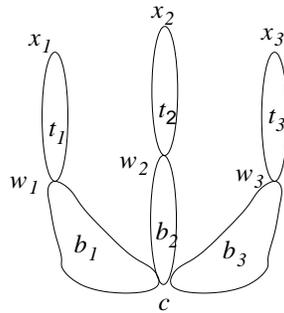


Figure 1: A typical 2-claw.

Proof of Lemma 11. Assume that $C = \{c, w_1, w_2, w_3, x_1, x_2, x_3\}$ is a 2-claw centered at c with fingers $f_i = \{b_i, t_i\}$: the edges forming the fingers are called bases $b_i = cw_i$ and tips $t_i = w_i x_i$, see Fig. 1.

Now, assume that alternatively w_1 were the center of a 2-claw with vertex set C . This means that there must be an edge in G that connects w_1 with w_i or x_i for some $i \neq 1$ in order to form another finger. But this would introduce short cycles in G .

If x_1 were the center of a 2-claw, then again there must be edges connecting x_1 with c , w_i or x_i for some $i \neq 1$ in order to form another finger. But this would again introduce short cycles in G .

By symmetry, w_2 , w_3 , x_2 and x_3 cannot be centers of another 2-claw with vertex set C .

Therefore, c must be also the center of the assumed alternative 2-claw with vertex set C . Now, if one of the x_i would be a finger articulation in this alternative 2-claw interpretation, this means that there is an edge connecting x_i and c , so that (c, x_i, w_i) would form a 3-cycle. Hence, only the w_i could be finger articulations. Therefore, only the x_i could be finger tips, since an edge between w_i and t_j would introduce short cycles if $i \neq j$. \square

In these proofs, we need the following easy corollary from reduction rule 3b.:

Proposition 3 *In a reduced instance, injured 2-claws corresponding to forbidden structures of size five in the 6HS instance always still have two healthy fingers, and the edges incident to the center that belong to these fingers have degree of at least two in the 6HS instance.*

Proof of Lemma 9. Let us first analyze the situation that we actually only have one injured 2-claw with five edges within our reduced problem instance. Let $G = (V, E)$ be a reduced instance such that G contains no 3-, 4-, or 5-cycles (they have been already branched at).

Let $C = \{c, w_1, w_2, w_3, x_1, x_2, x_3\}$ be a 2-claw centered at c , such that w_i is neighbored (at least) to c and x_i for $i = 1, 2, 3$, with fingers $f_i = \{b_i, t_i\}$, as depicted in Fig. 1.

Assume that the 2-claw is injured, so that w.l.o.g., either b_1 or t_1 are marked. Now, observe that we can show a couple of claims for this situation:

1. The degree of w_2 (and of w_3) in G is two.
 If this were false, we would face a $T^{\geq 2}$ -situation, since there would be another injured 2-claw C' centered at c ; in fact, C' and C would have four unmarked edges in common, including the injured finger $\{b_1, t_1\}$. \diamond
2. c is not the center of another 2-claw.
 Avoiding the previous case, this means that the assumed other 2-claw \hat{C} with center c has at least one finger $F = \{b, t\}$ disjoint from the ones of C . But then, F together with the injured finger of C and one non-injured finger of C would produce another injured 2-claw C' centered at c , different from C . Here, C' and C would have three unmarked edges in common. Again, we would face a $T^{\geq 2}$ -situation. \diamond
3. Both b_2 and b_3 interact with other 2-claws. Hence, b_2 must be the tip of another 2-claw C_2 with center x_2 and b_3 must be the tip of another 2-claw

C_3 with center x_3 , where $C_2 \neq C_3$.

b_2 and b_3 have degree at least two, since our instance is assumed to be reduced. Let us discuss possible interactions between C and another 2-claw C_2 that contains b_2 and has center c_2 . If b_2 were not the tip of another 2-claw, it would be the base of another 2-claw. Hence, either $c_2 = w_2$, which contradicts Claim 1, or $c_2 = c$, contradicting Claim 2.

Consider the case that the base of C_2 would be another edge adjacent to c but not contained in C . Then, C_2 would have two other fingers g_1 and g_2 (besides $\{c_2c, b_2\}$) that do not interfere with C (otherwise, short cycles would be introduced). Now, e.g., f_1, f_2 and a finger formed by cc_2 and say the base of g_1 would be injured, so that we face a $T^{\geq 2}$ -situation, in contrast to our assumptions.

If finally t_2 is the base of C_2 , then x_2 is the center (as claimed), due to Claim 1.

A similar reasoning is valid for the third finger of C and the corresponding 2-claw C_3 that contains b_3 .

Since C is a 2-claw, $x_2 \neq x_3$ by definition. Therefore, $C_2 \neq C_3$: due to Lemma 11, C_2 and C_3 actually define different 2-claws. \diamond

4. $w_1 \notin C_2 \cup C_3$.
 Assume $w_1 \in C_2$. Then, there must be a path P of length at most two between w_1 and x_2 , which together with the path (w_1, c, w_2, x_2) forms a 5-cycle. \diamond
5. Similarly: $w_3 \notin C_2$ and $w_2 \notin C_3$. \diamond

Now we are ready to analyze the branching for $T^1(k)$. C is the injured 2-claw we are going to branch on. According to Lemma 11, C can be uniquely described alternatively as a set of edges or as a set of vertices. We adopt the first notation in the following.

Due to the heuristic priorities 4a. and 5., we would select either b_2 or b_3 for branching, whatever has larger degree. Assume that the degree of b_2 is larger than the degree of b_3 . If b_2 goes into the biplanarization set, we create a $T^0(k-1)$ -branch. In the case that we do not take $b_2 \in C$ into the biplanarization set, b_2 will get marked. Hence, at least one new forbidden structure of size five is created, namely (following the terminology introduced in the claims) $C'_2 = C_2 \setminus \{b_2\}$.

Nonetheless, $C' = C \setminus \{b_2\}$ is now the smallest forbidden structure with $s(C') = 4$. Due to the last heuristic priority, we would select b_3 for branching. According to Claim 5, b_3 is not part of C_2 . Hence, if we take b_3 into the biplanarization set, we won't destroy C_2 , so that this is a $T^1(k-1)$ -branch.

If we don't put b_3 into the biplanarization set, then we will create a new small forbidden structure, namely $C'_3 = C_3 \setminus \{b_3\}$. In that case, we proceed branching at the injured finger $\{b_1, t_1\}$ according to heuristic priority 4b. So, let z denote the still unmarked part of that finger. According to Claim 4, if

we put z into the biplanarization set, neither C'_2 nor C'_3 get destroyed. Hence, this is a $T^2(k-1)$ -branch. Finally, branching at say t_2 would only destroy C'_2 but not C'_3 , so this is a $T^1(k-1)$ -branch, and then branching at t_3 gives a $T^0(k-1)$ -branch.

Altogether, this shows the claimed relation if we actually had encountered a T^1 -situation and not a $T^{\geq 2}$ -situation in disguise.

$$\text{Hence, } T^1(k) \leq 2T^0(k-1) + 2T^1(k-1) + T^2(k-1).$$

If the assumption that only one forbidden structure of size five is present was wrong, our branching analysis could only get better. More precisely, our earlier reasoning shows that we only encountered the two cases that two forbidden structures f_1 and f_2 of size five intersect, such that $3 \leq |c(f_1) \cap c(f_2)| \leq 4$. To cover these cases, we may wish to include the estimates

$$T^1(k) \leq \max\{3T^0(k-1) + 4T^0(k-2), 4T^0(k-1) + T^0(k-2)\}$$

that have been derived when proving Lemma 8. □

We will include the two inequalities in the algebraic analysis that follows but refrain from putting it into the (set of) inequalities listed in the formulation of Lemma 9, since it is not really dealing with a T^1 -situation.

Proof of Lemma 10. Let f_1 and f_2 be the two forbidden structures of size five. We have to consider only one further sub-case, the other ones we already dealt with in the proof of Lemma 8.

If $c(f_1) \cap c(f_2) = \emptyset$, then we get by the analysis of Lemma 9, keeping in mind that by branching on say $c(f_1)$ we still keep the low-size forbidden structure $c(f_2)$:

$$T^2(k) \leq 2T^1(k-1) + 2T^2(k-1) + T^3(k-1) \leq 2T^1(k-1) + 3T^2(k-1). \quad \square$$

More algebra for 2-LP

We only include the case when the resulting T^1 -case describes a situation with exactly one forbidden structure of size five, i.e., an injured 2-claw.

$$\begin{aligned} T^0(k) &= T^0(k-1) + T^2(k) \\ T^1(k) &= 2T^0(k-1) + 2T^1(k-1) + T^2(k-1) \\ T^2(k) &= 2T^1(k-1) + 3T^2(k-1) \end{aligned}$$

The first equation allows to replace all occurrences of T^2 by according occurrences of T^1 , leading to two equations (derived from the last two):

$$\begin{aligned} T^1(k) &= 2T^0(k-1) + 2T^1(k-1) + T^0(k-1) - T^0(k-2) \\ &= 3T^0(k-1) + 2T^1(k-1) - T^0(k-2) \\ T^0(k) - T^0(k-1) &= 2T^1(k-1) + 3T^0(k-1) - 3T^0(k-2) \\ \rightsquigarrow 2T^1(k-1) &= T^0(k) - 4T^0(k-1) + 3T^0(k-2) \end{aligned} \tag{1}$$

The last equation can be now plugged into the first one, leading to:

$$T^1(k) = T^0(k) - T^0(k-1) + 2T^0(k-2)$$

This implies—by applying an argument shift to Eq. (1) to get the second line:

$$\begin{aligned} 2T^1(k) &= 2T^0(k) - 2T^0(k-1) + 4T^0(k-2) \\ &= T^0(k+1) - 4T^0(k) + 3T^0(k-1) \end{aligned}$$

which yields:

$$0 = T^0(k+1) - 6T^0(k) + 5T^0(k-1) - 4T^0(k-2)$$

Hence, $T^0(k) \leq 5.1844^k$, i.e., $c \leq 5.1844$ for the ansatz $T^0(k) = c^k$.

If the involved T^1 -case is actually a T^2 -case, the corresponding recurrences get only better. Details are contained in the ECCC report, see [10].

So, for the whole set of mutual recurrences that have to be satisfied by T^0 , T^1 and T^2 , we get $T^0(k) \leq c^k$ with $c = 2.5 + \sqrt{7.25} \leq 5.1926$.

To actually get expressions for the other two functions involved, we follow the approach setting $T^1(k) = \alpha_1 c^k$ and $T^2(k) = \alpha_2 c^k$. Hence, from $T^0(k) \leq T^0(k-1) + T^2(k)$ we derive $\alpha_2 \geq 1 - 1/c$. From $T^2(k) \leq 4T^0(k-1) + T^0(k-2)$, i.e., the equation yielding the worst c -estimate, we get $\alpha_2 \leq 4 + 1/c$. This implies that we have to take

$$\alpha_2 = \frac{1.5 + \sqrt{7.25}}{2.5 + \sqrt{7.25}} \approx 0.8074.$$

Hence, we make the choice $T^2(k) = \frac{1.5 + \sqrt{7.25}}{2.5 + \sqrt{7.25}} (2.5 + \sqrt{7.25})^k$ alongside with $T^0(k) = (2.5 + \sqrt{7.25})^k$. It is easy to check that now also $T^2(k) \leq \max\{T^0(k-1) + 16T^0(k-2), 2T^0(k-1) + 9T^0(k-2), 3T^0(k-1) + 4T^0(k-2), 4T^0(k-1) + T^0(k-2)\}$ is true with this choice (it suffices to check that $4T^0(k-1) + T^0(k-2)$ is the largest among the entities that are maximized for $k = 2$). The only tricky bit is the inequality $T^2(k) \leq 2T^1(k-1) + 3T^2(k-1)$, since this one actually involves T^1 . Since we are now actually treating the whole set of mutually recursive relations and since T^1 was not involved in creating the worst-case recursion, we should choose T^1 in a way that trivially satisfies $T^2(k) \leq \max\{2T^1(k-1) + 3T^2(k-1), T^0(k-1) + 16T^0(k-2), 2T^0(k-1) + 9T^0(k-2), 3T^0(k-1) + 4T^0(k-2), 4T^0(k-1) + T^0(k-2)\}$, i.e., T^1 should

be as small as possible. Since $T^2 \leq T^1$, the easiest choice would be $T^1 = T^2$, i.e., $\alpha_1 = \alpha_2$. Surely, the equations for T^2 are now satisfied. What about the equation for T^1 itself? $T^1(k) \leq 2T^0(k-1) + 2T^1(k-1) + T^2(k-1)$ now becomes $T^2(k) \leq 2T^0(k-1) + 3T^2(k-1)$. This is true if $c\alpha_2 = 1.5 + \sqrt{7.25} \leq 2 + 3\alpha_2$, which is easy to check.

Appendix: A rather technical proof of Lemma 18

Proof: So, let $C = \{ab, bc, cd, da\}$ be a 4-cycle with $a \in A$ and $a <_\pi c$.

We first investigate how a single additional edge can create one (but not two) situation(s) violating (*). More specifically, we discuss the ways in which (w.l.o.g.) the edge ab can be engaged in situations violating (*).

1. If, w.l.o.g., (a, b, c) together with some edge e violates (*), then either also (a, d, c) together with e violates (*) or d is incident with $e = xd$. In the first case, there is a forbidden structure (namely $\{ad, dc, e\}$) that has non-empty intersection with the forbidden structure $\{ab, bc, e\}$, violating (+).
2. Consider now the case that some path (x, y, z) together with ab violates (*). Firstly assume that $y \neq d$. Then, (x, y, z) does not satisfy (*) both together with ab and with ad , which violates (+). Consider now $y = d$. A simple case analysis yields that the only non-prohibited situation is $y = d$ and $c \in \{x, z\}$.

We can summarize our observations as follows: if $C = \{ab, bc, cd, da\}$ is a 4-cycle with $a \in A$ and $a <_\pi c$ and if e is an edge that—together with two edges from C —forms a forbidden structure reflecting a situation violating (*), then

- either $e = x_b b$ (in that case, $x_b >_\pi a$ and $x_b \neq c$)
- or $e = x_d d$ (then, $x_d <_\pi c$ and $x_d \neq a$).

More specifically, the situation $e = x_d d$ and $a <_\pi x_d <_\pi c$ showed up in the first part of our case analysis, while the situation $e = x_d d$ and $x_d <_\pi a$ appeared in the second part of the case analysis. The other situations reflect symmetries.

Claim: If there are two edges $e = x_d d$ and $e' = x'_d d$, then the edge ab appears in more than one forbidden structure reflecting situations violating (*).

Proof: The following case distinctions cover all possibilities, assuming, w.l.o.g., $x_d <_\pi x'_d$:

- $x_d <_\pi x'_d <_\pi a$: Then, (x_d, d, c) together with ab violates (*), and (x'_d, d, c) together with ab violates (*).
- $x_d <_\pi a <_\pi x'_d$: Then, (x_d, d, c) together with ab violates (*), and (a, b, c) together with $x'_d d$ violates (*).

- $a <_{\pi} x_d <_{\pi} x'_d$: Then, (a, b, c) together with $x_d d$ as well as with $x'_d d$ violate (*). \diamond

Analogously, one can show:

Claim: If there are two edges $e = x_b b$ and $e' = x'_b b$, then the edge cd appears in more than one forbidden structure reflecting situations violating (*). \diamond

So, the only possibility that C could interfere with more than one forbidden structure reflecting situations violating (*) is when both $f_b = x_b b$ and $f_d = x_d d$ exist. This is rejected with the following case analysis.

- $x_d <_{\pi} a <_{\pi} c <_{\pi} x_b$: Then, (x_d, d, c) together with ab violates (*), and (a, b, x_b) together with cd violates (*). Hence, ab shows up in both violating forbidden structures.
- $a <_{\pi} x_d <_{\pi} c <_{\pi} x_b$: Then, (x_d, d, c) together with ab violates (*), and (a, b, x_b) together with $x_d d$ violates (*). Hence, $x_d d$ shows up in both violating forbidden structures.
- $x_d <_{\pi} a <_{\pi} x_b <_{\pi} c$: Symmetrical to the previous case.
- $a <_{\pi} x_d <_{\pi} x_b <_{\pi} c$: Then, (x_d, d, c) together with $x_b b$ violates (*), and (a, d, c) together with $x_d d$ violates (*). Hence, $x_d d$ shows up in both violating forbidden structures.
- $a <_{\pi} x_b <_{\pi} x_d <_{\pi} c$: Symmetrical to the previous case. \square