Electronic Colloquium on Computational Complexity, Report No. 78 (2004)

Two-Layer Planarization: Improving on Parameterized Algorithmics

Henning Fernau[1,2]

[1] Universität Tübingen, WSI für Informatik, Sand 13,
72076 Tübingen, Germany
fernau@informatik.uni-tuebingen.de
[2] 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. We study two variants:
- 2-Layer Planarization: can k edges be deleted from a given graph G so that the remaining graph is biplanar?
- 1-Layer Planarization: fix the order of the vertices on one layer.

Improving on earlier works of Dujmović *et al.* [3], 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.

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, see [3]. 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*. Experimental evidence suggests: 2-layer drawings in which all the crossings occur in a few edges are more readable than drawings with fewer total crossings [6]—which gives the crossing minimization problem(s).

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 *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 [8]. 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 vertices in A?

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

ISSN 1433-8092

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 for dense graphs are of little interest from a practical point of view, as the resulting drawing will be unreadable anyway.

This analysis hence fits into the framework of parameterized algorithmics. A parameterized problem 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).

Our results. In this paper we apply so-called kernelization and search tree methods to obtain algorithms for the 1- and 2-LP problems, this way improving earlier results [2, 3]. This leads to an $\mathcal{O}(k^2 \cdot 5.1926^k + |G|)$ time algorithm for 2-LP in a graph G. We present a similar second algorithm to solve the 1-LP problem in $\mathcal{O}(k^3 \cdot 2.5616^k + |G|^2)$ time. To this end, we draw connections to Hitting Set problems. The top-down analysis technique presented in [5] is applied to obtain the claimed running times in the analysis of the search tree algorithms.

2 Preliminaries

In this section we introduce notation, recall a characterization of biplanar graphs and formalize the problem statements. The mentioned results are from [3].

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 $\deg'_G(v)$.

A graph is a *caterpillar* if deleting all the leaves produces a (possibly empty) path. This path is the *spine* of the caterpillar. 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 $\deg'(v) \geq 3$ is either a caterpillar or a wreath.

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

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

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.*

A set T of edges of a graph G is called a *biplanarizing set* if $G \setminus T$ is biplanar. The *bipartite planarization number* of a graph G, denoted by $\mathsf{bpr}(G)$, is the size of a minimum biplanarizing set for G. The 2-LP problem is: given a graph G and an integer k, is $\mathsf{bpr}(G) \leq k$? For a given bipartite graph $G = (A, B; E)$ and permutation π of A, the *1-layer biplanarization number* of G and π, denoted $\mathsf{bpr}(G, \pi)$, 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 $\mathsf{bpr}(G, \pi) \leq k$.

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, $\mathsf{bpr}(G) \geq \frac{1}{2}\Phi(G)$.*

3 2-Layer Planarization: Bounded search tree

The basic approaches for producing $\mathcal{FPT}$ algorithms are *kernelization* and *bounded search trees* [1]. Based on the preceding lemmas, Dujmović *et al.* showed:

Theorem 1. *Given a graph G and integer k, there is an algorithm that determines if $\mathsf{bpr}(G) \leq k$ in $\mathcal{O}(k \cdot 6^k + |G|)$ time.*

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

Can we further improve on the running time of the search tree algorithm? Firstly, observe that whenever $\deg'_{G'_0}(v) \geq \ell$ for any G'_0 obtained from G_0 by edge deletion, then already $\deg'_{G_0}(v) \geq \ell$. 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*, 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. By re-interpreting the edges of G_0 as the vertices of a hypergraph $G = (V, E)$, where the hyperedges correspond to the forbidden structures, a 2-LP instance (G_0, k) is translated into an instance (G, k) of 6-Hitting Set (6HS).

If we delete all those edges in G_0 that correspond to vertices in a hitting set as delivered by the 6HS algorithm, we will arrive at a graph G'_0 which satisfies $\deg'_{G'_0}(v) < 3$ for all vertices v. Hence, $\Phi(G'_0) = 0$, and Lemma 4 applies.

Unfortunately, we cannot simply take the currently best 6HS algorithm, see [5, 7]. Why? The problem is that there may be minimal solutions of the 6HS instance which are "skipped" due to clever branching, since there exists another minimal solution which is "equivalent." This equivalence might not hold any longer if we translate back to the original 2-LP instance, where we still have to resolve the wreath components, and it might be that we "skipped" the only solution that already upon solving the 6HS instance was incidentally also destroying enough wreaths. If we insist on enumerating all minimal hitting sets no larger than the given k, this problem can be circumvented, since we can do the wreath component analysis in the leaves of the search tree, but it would gain nothing in terms of time complexity, since examples of hypergraphs having 6^k minimal hitting sets of size at most k can be easily found: just consider k disjoint hyperedges each of degree six. Notice that in this example, all vertices have a very small degree, namely one.

In the following, we often translate back and forth between the 2-LP instance and the corresponding 6HS instance, always using the terminology which appears to be the most appropriate.

We go a bit more in details into how to solve the 6HS instance. In [5], a recursive search tree algorithm for 6HS is proposed that uses three reduction rules which are always exhaustively applied at the beginning of each recursive call:

1. (hyper)edge domination: A hyperedge e is *dominated* by another hyperedge f if $f \subset e$. Then, delete e, since covering f will automatically also cover e.
2. small edges: Delete all hyperedges of size 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 belongs to e, as well. Then, delete all occurrences of x.

The first two rules are fine when it comes our problem, as well. Only the third rule causes trouble, since the soundness is based on the observation that taking y into the hitting set (instead of x) is never worse; and this may be wrong due to the subsequent "wreath analysis," as previously indicated.

The main purpose of the vertex domination rule is that it guarantees the existent of vertices of "sufficiently" high degree. Our aim is now to provide a more problem-specific analysis which maintains exactly that property. Firstly, observe that one special case of the vertex domination rule is still valid:

3a isolates: If v is a vertex of degree zero, then delete v.

Namely, vertices of degree zero in the 6HS instance correspond to edges in the 2-LP instance that don't participate in any forbidden structure. Surely, we can safely delete them. Can we also rule out vertices of degree one (to a certain extent)? We will discuss this point later on.

In order to be able to do the subsequent wreath analysis, vertices of the 6HS instance will never be deleted but just marked as *virtual*. Hence, we are only dealing with two sorts of hyperedges:

– hyperedges of size six that correspond to 2-claws and
– hyperedges of size five or four that correspond to *injured 2-claws*.

Here, an injured 2-claw describes a hyperedge which originally stemmed from a 2-claw, but one or two edges of the 2-claw got "amputated," i.e., marked virtual.

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 $i = 1, 2, 3$. We will call $\{cw_i, w_ix_i\}$ also a *finger* of C, so that the hyperedge corresponding to C is partitioned into three disjoint fingers. Clearly, in an injured 2-claw with five edges, only one of the fingers actually got injured and two fingers are still *pretty*. In an injured 2-claw with four edges, we still have at least one pretty finger left over.

The second ingredient in the approach to hitting set problems described in [5] are so-called *heuristic priorities*. More specifically, we will use the following rules to select hyperedges and vertices to branch at in case of multiple possibilities:

1. Select a hyperedge e of smallest size that if possible corresponds to a short cycle in the 2-LP instance.
2. If $|e| \leq 5$ and $e \cap f \neq \emptyset$ for another hyperedge f of size five or less, modify $e := e \cap f$.
3. Branch at a highest-degree vertex x of e, if possible incident to the center of the 2-claw e, such that x belongs to a pretty finger.

The 6HS algorithm will then basically perform the following two steps on a given instance (G, k) and a preliminary partial solution S:

1. Exhaustively apply all reduction rules, leading to a possibly modified instance (G', k') and partial solution S'.
2. Select a vertex x according to the heuristic priorities and branch:
   (a) Assume: x is in the hitting set. The new partial solution is $S' \cup \{x\}$ and the instance $(G' - E(x), k' - 1)$; $E(x)$ denotes all hyperedges containing x.
   (b) The case that x is not part of the hitting set can be dealt with by recursing on the new instance $(G' - x, k')$ with partial solution S'.

The main point here is that we can actually analyze this algorithm from a parametric view. In the following analysis, assume that we have already branched on all cycles up to length five (see the first priority). Then, the sketched 6HS algorithm is applied to the collection of 2-claws. To further improve on our search tree algorithm, we propose the following reduction rule for (injured) 2-claws:

3b (injured) 2-claws: If y is a vertex of degree one in a hyperedge of size four, five or six corresponding to an (injured) 2-claw, and if y (as an edge in the 2-LP instance) is incident to the center of the corresponding 2-claw, then mark y as virtual.

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

Proposition 1. *During the course of the* Hitting Set *algorithm, there will never occur 2-claws which merely consists of virtual edges due to rule 3b. More precisely, at most one edge per finger will turn virtual.*

Proof. 3b. obviously only affects *one* 2-claw at a time, since only edges of size one are turned virtual. Per 2-claw, the rule triggers at most once per finger.

Proposition 2. *After having successfully run the* Hitting Set *algorithm, we did not create a cycle of length at least five that only consists of virtual edges.*

For the soundness of rule 3b. (Proposition 2), the following observation is crucial.

Property 1. Let $F = \{xy, yz\}$ be one pretty finger of an (injured) 2-claw C with center x such that xy (in the corresponding 6HS instance) has degree one. 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'.

The proposed algorithm is stated in Fig. 1. The subroutine 6HS-all-minimal is working as described above, the interface being described in 2-Layer BST / 6HS-based. It is clear that by choosing an appropriate implementation, we can prevent the overall algorithm from using exponential space: the solutions to the 6HS instance would have to be created one by one.

Now, let us turn to the time analysis of the procedure. Of course, the most important part is the analysis of 6HS-all-minimal. We will follow the ideas explained in [5] for d-Hitting Set. Let $T(k)$ denote the number of leaves in a worst-case search tree for 6HS-all-minimal, 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 hyperedges 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 6HS instance G contains a vertex x of degree 2. One branch is that x is put into the hitting set. If x is not put into the hitting set, then at least two new hyperedges of size five are created.

Some more involved analysis of the T^1- and T^2-branches as well as some algebra for solving the recursions, shows:

Lemma 7. $T^1(k) \leq 2T^0(k-1) + 2T^1(k-1) + T^2(k-1)$.

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

Theorem 2. *Given a graph G and integer k, the algorithm* 2-Layer BST / 6HS-based *determines if* $\mathsf{bpr}(G) \leq k$ *in $\mathcal{O}(k^2 \cdot 5.1926^k + |G|)$ time, when applied to the problem kernel as derived in [3].*

Algorithm 2-Layer BST / 6HS-based
input: graph $G_0 = (V_0, E_0)$;
parameter: a non-negative integer k_0
output: NO if $\mathsf{bpr}(G_0) > k_0$; otherwise, YES.

1. **if** $\Phi(G_0) > 2k_0$ **then** return NO.
2. **else if** $\Phi(G_0) = 0$
   if $k_0 \geq$ # component wreaths of G_0 **then** return YES. (Lemma 4)
   else return NO.
3. **else** // ($\exists\, v \in V_0$ such that $\deg'_{G_0}(v) \geq 3$)
   Let $V' \subseteq V_0$ be all $v \in V_0$ such that $\deg'_{G_0}(v) \geq 3$.
   Find all 2-claws, 3-, 4-cycles or 5-cycles C in G_0 containing some $v \in V'$
   as described in Lemma 3, giving a set $\mathcal{C}$ of edge sets.
   Form the hypergraph $G = (E, \mathcal{C})$.
   Call 6HS-all-minimal on instance $((G, k), \emptyset, \emptyset)$ to find solution set $\mathcal{S}$.
   //$\mathcal{S}$ contains all solutions to the 6HSinstance that are
   //(1) reduced, (2) minimal, and (3) contain at most k elements.
   //Each $S \in \mathcal{S}$ is a triple $S = (S_\sigma, k', S_\nu)$:
   // S_σ is the returned solution set
   // k' is the returned remaining parameter
   // S_ν is the set of virtual edges
   For each solution $S = (S_\sigma, k', S_\nu)$ **do**
   Modify the instance (G_0, k_0) accordingly, yielding (G'_0, k').
   if $k' \geq$ # component wreaths of G'_0 **then** return YES. (Lemma 4)
   return NO. // All 6HS-all-minimal solutions finally fail.

Fig. 1. The algorithm 2-Layer BST / 6HS-based

4 1-Layer Planarization: Kernelization algorithm

The following results from [3] give important properties for π-biplanar graphs.

Lemma 9. *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. ($\star$)

Let $G = (A, B; E)$ be a bipartite graph with a fixed permutation of A that satisfies condition ($\star$). 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 10. *If $G = (A, B; E)$ is a bipartite graph and π is a permutation of A satisfying condition ($\star$) then* $\mathsf{bpr}(G, \pi) = \sum_{\text{maximal } p\text{-diamonds of } G}(p-1)$.

We are now going to derive a kernelization algorithm for 1-LP. Let us say that an edge e of a bipartite graph G *potentially violates condition* $(\star)$ if, using the notation of condition $(\star)$, $e = e_i$ 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* $(\star)$.

According to Lemma 9 (as well as the proof of Lemma 10 for the last two rules), the following reduction rules are sound, given an instance $(G = (A, B; E), \pi, k)$ of 1-LP. Analogues to the first three rules are well-known from Hitting Set problems, see [5, 7].

1L-RR-edge: If $e \in E$ does not participate in any cycle and does not potentially violate condition $(\star)$, then remove e from the instance (keeping the same parameter k).
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).
1L-RR-large: If $e \in E$ participates in more than k^2 situations that potentially violate condition $(\star)$, then put e into the biplanarization set and modify the instance appropriately (also decreasing the parameter).

Let $E_\star \subseteq E$ be all edges that potentially violate condition $(\star)$. Let $E_\circ \subseteq E$ be all edges that participate in cycles. Let G_{4c} be generated from those edges from $E_0 \setminus E_\star$ that participate in 4-cycles. By construction, G_{4c} satisfies $(\star)$. Lemma 10 shows that the next reduction rule can be applied in polynomial time:

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

Lemma 11. *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 $(\star)$ and participates in cycles of length > 4.*

Theorem 3. *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, $|E| \leq k^3$. The kernel can be found in time $\mathcal{O}(|G|^2)$.*

Proof. Now consider $E_\star$ as vertex set V' of a hypergraph $G' = (V', E')$ and put $\{e_1, e_2, e_3\}$ into E' iff e_1, e_2, e_3 together violate condition $(\star)$. A subset of edges from E whose removal converts $(A, B; E)$ into a bipartite graph which satisfies condition $(\star)$ is in obvious one-to-one correspondence with a hitting set of the hypergraph G'. Niedermeier and Rossmanith have shown [7, Proposition 1] a cubic kernel for 3-Hitting Set, so that at most k^3 edges are in $E_\star$ (else NO). Their reduction rules correspond to our rules 1L-RR-edge and 1L-RR-large.

If $e = xy \in E_\circ \setminus E_\star$ with $y \in B$ does not belong to a 4-cycle, then Lemma 11 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 11, 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 10, and (c) $E_\circ \setminus E_{4c}$: according to our preceding reasoning, there are at most $|E_\star|$ many of these edges.

5 1-Layer Planarization: Bounded search tree

Theorem 4. *(Dujmović* et al. *[3]) Given a bipartite graph $G = (A, B; E)$, a fixed permutation π of A, and integer k, there is an algorithm that determines if $\mathsf{bpr}(G, \pi) \leq k$ 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. 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 $(\star)$. 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 [5, 7], running in time $\mathcal{O}(k \cdot 2.179^k + |G|)$, since destroying triples of edges violating condition $(\star)$ 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 (that correspond to situations violating condition $(\star)$ in the original 1-LP instance) that contain vertices of degree at least two, then we have already destroyed all "large" cycles.

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

Lemma 12. *Let $G = (A, B; E)$ be a bipartite graph and π be a fixed permutation of A. Assume that if $h = \{e_1, e_2, e_3\}$ and $h' = \{e'_1, e'_2, e'_3\}$ are two situations violating $(\star)$, then $h \cap h' = \emptyset$. Let $C = \{ab, bc, cd, da\}$ be a sequence of edges forming a 4-cycle.*

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

Hence, after the indicated branching, for each 4-cycle, at most one hyperedge of size three remains such that the corresponding edge sets have non-empty intersection. Since we have to destroy every 4-cycle, the best we then can obviously do is to take out an edge that takes part in the "accompanying" situation violating $(\star)$. This can be done completely deterministically due to the preceding lemma, where possible nondeterministic situations can be arbitrarily resolved. Finally, the only remaining situations correspond to possibly interacting 4-cycles. These can be solved with Lemma 10.

Theorem 5. *Given a bipartite graph $G = (A, B; E)$, a fixed permutation π of A, and integer k, there is an algorithm that determines if $\mathsf{bpr}(G, \pi) \leq k$ in $\mathcal{O}(k^3 \cdot 2.5616^k + |G|^2)$ time.*

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. In particular, for fixed k, we have polynomial time algorithms to determine if $\mathsf{bpr}(G) \leq k$ and $\mathsf{bpr}(G, \pi) \leq k$. The smaller exponential bases (in comparison with [3]) 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 [3, 4], 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 $\mathcal{O}(k^6)$.
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 [2] only give an $\mathcal{O}(2^{32(2+2k)^3}|G|)$ algorithm. Further research should considerably improve this algorithm.

Acknowledgments We are grateful for discussion of this topic with V. Dujmović.

References

1. R. G. Downey and M. R. Fellows. *Parameterized complexity.* Springer, 1999.
2. V. Dujmović, M. Fellows, M. Hallett, M. Kitching, G. Liotta, C. McCartin, N. Nishimura, P. Ragde, F. Rosamond, M. Suderman, S. Whitesides, and D. R. Wood. On the parameterized complexity of layered graph drawing. In F. Meyer auf der Heide, editor, *9th Annual European Symposium on Algorithms ESA*, volume 2161 of *LNCS*, pages 488–499. Springer, 2001.
3. V. Dujmović, M. Fellows, M. Hallett, M. Kitching, G. Liotta, C. McCartin, N. Nishimura, P. Ragde, F. Rosemand, 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 Symp. on Graph Drawing GD'01*, volume 2265 of *LNCS*, pages 1–15. Springer, 2002.
4. V. Dujmović, H. Fernau, and M. Kaufmann. Fixed parameter algorithms for one-sided crossing minimization revisited. In G. Liotta, editor, *Graph Drawing, 11th International Symposium GD 2003*, volume 2912 of *LNCS*, pages 332–344. Springer, 2004.
5. H. Fernau. A top-down approach to search-trees, 2004. In preparation.
6. P. Mutzel. An alternative method to crossing minimization on hierarchical graphs. *SIAM J. Optimization*, 11(4):1065–1080, 2001.
7. R. Niedermeier and P. Rossmanith. An efficient fixed-parameter algorithm for 3-hitting set. *Journal of Discrete Algorithms*, 1:89–102, 2003.
8. 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.

*** By changing the heuristic priorities in one case and using a generalization of rule 3b., we can improve the base to 5.1844.

Appendix

Definition of Hitting Set

For the reader's convenience, we include the following definitions.
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:
Problem name: d-Hitting Set
Given: A hypergraph $G = (V, E)$ with edge size bounded by d
Parameter: k
Output: Is there a *hitting set* (or *cover*) of size at most k:

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

Regarding Proposition 2

Proof of Proposition 2. Let C be a 2-claw (or an injured 2-claw) that contains an edge e that is incident with the center of C, belongs to a pretty finger F and has (in the 6HS instance) degree one, so that 3b. triggers.

Sub-Claim: A cycle c that contains e will also contain the other edge e' of F. This sub-claim is true because of Property 1.

Hence, if we remove e from the Hitting Set instance by marking it as virtual, we still keep the edge e' as non-virtual which we might choose to take into the biplanarization set to destroy any cycle in which e participated. Hence, in each cycle, there will never occur the situation that the "last" edge of a finger is turned into a virtual edge. This shows the claim.

The subroutine 6HS-all-minimal

The algorithm in Fig. 2 outlines how to find all minimal 6-Hitting Sets with the restriction that we don't have to branch on small-degree vertices as explained in the main text.

Remark 1. The related problem of enumerating all minimum solutions say of vertex cover problems has been also studied in our COCOON 2002 contribution entitled *On Parameterized Enumeration*. A specific application of this enumeration approach has been detailed by P. Damaschke in his IWPEC 2004 paper.

A couple of remarks are in order here to comment on the working of the algorithm 6HS-all-minimal.

Algorithm 6HS-all-minimal
inputs: hypergraph $G = (V, E)$;
a partial solution S_σ and
a list of virtual edges S_ν
parameter: a non-negative integer k
output: all minimal solutions, annotated as described in Fig. 1

Exhaustively apply the reduction rules 1., 2., and 3a.
if $E = \emptyset$ **then** return $\{(S_\sigma, k, S_\nu)\}$
else if $k \leq 0$ AND $E \neq \emptyset$ **then** return $\emptyset$ // No solution
else
Find a hyperedge h of minimal size.
if $|h| > 4$ **then**
// only (injured) 2-claws are left over
Exhaustively apply the reduction rules 1., 2., 3a., 3b.
//The obtained hypergraph is also called $G = (V, E)$ and the parameter k.
//Likewise, we keep the notations S_σ and S_ν.
Find a hyperedge h of minimal size.
//Notice that h could have "changed" by the reduction rules.
if $|h| \leq 5$ and there is another hyperedge h' with $h \cap h' \neq \emptyset$ and $|h'| \leq 5$
then $h := h \cap h'$.
Choose an e of maximal degree in h.
return the union of
6HS-all-minimal called with $G - E(e)$ and parameter $k - 1$ and
6HS-all-minimal called with $G - e$ and parameter k
//Likewise, S_σ and S_ν are transferred.

Fig. 2. The algorithm 6HS-all-minimal

- The correctness of the procedure can be easily proved by induction. Note that the heuristic priorities are not influencing the correctness of the algorithm itself, but only the order in which branches occur. This has been detailed for d-Hitting Set in [5].
- The algorithm ideally runs in two phases:
  1. All cycles of length up to five are dealt with.
  2. All 2-claws are branched on.

  Of course, as stated, there could be 2-claws handled before all small cycles have been examined, since 2-claws may get injured by branching on cycles. This does not, however, influence the correctness of the running time of the algorithm, since we can, more generally, formulate these two phases as follows:
  1. Branch on all hyperedges of size up to five.
  2. Branch on all remaining 2-claws.

  Now, it is clear that the size of the search tree belonging to phase one is bounded by $\mathcal{O}(5^k)$. This will be a sort of lower bound on our approach.

Conversely, a naive bound on the size of the search tree belonging to second phase is $\mathcal{O}(6^k)$. In the following, we work on getting this upperbound closer to the one of the first phase.

Proof of the analysis of the branches—trivial branching

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

Proof. This is a consequence of trivial branching at the five vertices collected in a hyperedge of size five.

Lemma 14. $T^2(k) \leq \max\{25T^0(k-2), T^0(k-1) + 16T^0(k-2), 2T^0(k-1) + 9T^0(k-2), 3T^0(k-1) + 4T^0(k-2), 4T^0(k-1) + T^0(k-2)\}$.

Proof. Let e_1 and e_2 be the two hyperedges of size five. We have to consider some sub-cases:

1. If $e_1 \cap e_2 = \emptyset$, then we get by the analysis of Lemma 13, keeping in mind that by branching on say e_1 we still keep the low-size hyperedge e_2:
$$T^2(k) \leq 25T^0(k-2).$$
2. If $|e_1 \cap e_2| = 1$, our heuristic priorities let us branch at $x \in e_1 \cap e_2$. If we take x into the hitting set, we get a $T^0(k-1)$-branch. If we don't take x into the hitting set, we are left with two hyperedges of size four, namely $e_1' = e_1 \setminus \{x\}$ and $e_2' = e_2 \setminus \{x\}$. Trivial branching on those edges (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. If $|e_1 \cap e_2| = 2$, our heuristic priorities let us branch at $x \in e_1 \cap e_2$ and then at $y \in e_1 \cap e_2$. This gives two $T^0(k-1)$-branches. If we don't take neither x nor y into the hitting set, we are left with two hyperedges of size three, namely $e_1' = e_1 \setminus \{x, y\}$ and $e_2' = e_2 \setminus \{x, y\}$. Trivial branching on those edges (which will be done according to the heuristic priorities) gives nine $T^0(k-2)$-branches. Hence,
$$T^2(k) \leq 2T^0(k-1) + 9T^0(k-2).$$
4. If $|e_1 \cap e_2| = 3$, our heuristic priorities let us branch at $x \in e_1 \cap e_2$ and then at $y \in e_1 \cap e_2$ and at $z \in e_1 \cap e_2$. Further trivial branching on the remaining edges $e_1' = e_1 \setminus \{x, y, z\}$ and $e_2' = e_2 \setminus \{x, y, z\}$ gives four $T^0(k-2)$-branches. Hence,
$$T^2(k) \leq 3T^0(k-1) + 4T^0(k-2).$$
5. If $|e_1 \cap e_2| = 4$, our heuristic priorities let us branch at all four elements in the intersection of e_1 and e_2. A trivial analysis then gives:
$$T^2(k) \leq 4T^0(k-1) + T^0(k-2).$$

Some algebra for 2-LP

We now look for an estimate $T^0(k) \leq c^k$. Since we can assume that the worst case is attained when all inequalities derived for T^ℓ are met by the corresponding equalities, we have to solve the following recurrences:

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

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

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, $c \leq \boxed{5.5250}$.

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

$$\begin{aligned} T^0(k) &= T^0(k-1) + T^2(k) \\ T^2(k) &= T^0(k-1) + 16T^0(k-2) \end{aligned}$$

Hence, we have to solve $0 = T^0(k) - 2T^0(k-1) - 16T^0(k-2)$ for $T^0(k) = c^k$ which yields $c \leq 5.1232$.

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

$$\begin{aligned} T^0(k) &= T^0(k-1) + T^2(k) \\ T^2(k) &= 2T^0(k-1) + 9T^0(k-2) \end{aligned}$$

Hence, we have to solve $0 = T^0(k) - 3T^0(k-1) - 9T^0(k-2)$ for $T^0(k) = c^k$ which yields $c \leq 4.8542$.

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

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

Hence, we have to solve $0 = T^0(k) - 4T^0(k-1) - 4T^0(k-2)$ for $T^0(k) = c^k$ which yields $c \leq 4.8285$.

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

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

Hence, we have to solve $0 = T^0(k) - 5T^0(k-1) - T^0(k-2)$ for $T^0(k) = c^k$ which yields $c \leq 5.1926$.

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 13. 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.

Proof of the analysis of the branches—more details

In one of the proofs, we need the following result on 2-claws:

Lemma 15. *Let $G = (V, E)$ be a graph without 3- or 4- or 5-cycles. If $C \subseteq V$ is a set of seven vertices, then there is only at most one way in which C can define a 2-claw (besides renaming the fingers).*

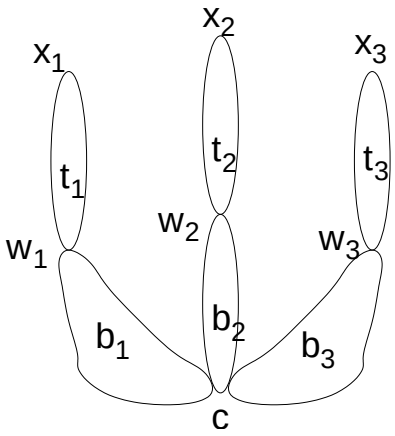

Fig. 3. A typical 2-claw.

Proof. 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. 3.

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.

Similar reasonings rule out w_2, w_3, x_2 and x_3.

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.

Hence, any vertex set C with seven vertices uniquely defines a 2-claw (if any).

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

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

Proof. The fact that there are still two pretty fingers is trivial, since we have still five edges left over in the injured 2-claw, so only one finger got amputated. If the proposition were false, reduction rule 3b. would trigger, contradicting our assumption that the instance is reduced.

Proof of Lemma 7. 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 $f_i = \{b_i, t_i\}$, see Fig. 3.

Assume that the 2-claw is injured, i.e., either b_1 or t_1 are marked virtual. 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 non-virtual edges in common.
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 non-virtual edges in common. Again, we would face a $T^{\geq 2}$ situation.

3. Both b_2 and b_3 (in the corresponding 6HS instance) have degree at least two. 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. If b_2 were not the tip of another 2-claw, it would be the base of another 2-claw. Hence, either w_2 is the center of that 2-claw, which contradicts Claim 1, or c is the center of that 2-claw, contradicting Claim 2. If the base of that 2-claw would be another edge adjacent to c but not contained in C, then a reasoning along the lines of Claim 1 leads to a contradiction. If finally t_2 is the base, then x_2 is the center (as claimed), due to Claim 1. A similar reasoning is valid for the third finger of C. Since C is a 2-claw, $x_2 \neq x_3$ by definition. Therefore, $C_2 \neq C_3$: due to Lemma 15, C_2 and C_3 actually define different 2-claws.
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.
5. Similarly: $w_3 \notin C_2$ and $w_2 \notin C_3$.

Now we are ready to analyze the branching for $T^1(k)$. C is the injured 2-cycle we are going to branch on. Since we now switch to hitting set terminology, we consider C as a set of edges; this is sound according to Lemma 15.

Due to the last heuristic priority, we would select either b_2 or b_3 for branching, whatever has larger degree. If b_2 goes into the hitting set, we create a $T^0(k-1)$-branch. In the case that we do not take $b_2 \in C$ into the hitting set, b_2 will get marked. Hence, at least one new hyperedge 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 hyperedge. 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 hitting 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 hitting set, then we will create a new small hyperedge, namely $C_3' = C_3 \setminus \{b_3\}$. In that case, we *can* proceed branching at the injured finger $\{b_1, t_1\}$.[†] So, let z denote the still unmarked part of that finger. According to Claim 4, if we put z into the hitting 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.

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

If the assumption that only one hyperedge 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 hyperedges e_1 and e_2 of size five intersect, such that $3 \leq |e_1 \cap e_2| \leq 4$. To cover these cases, we may wish

[†] Strictly speaking, this branching has to be included as a special case into the heuristic priorities.

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 14.

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 7, since it is not really dealing with a T^1-situation.

Proof of Lemma 8. Let e_1 and e_2 be the two hyperedges of size five. We have to consider only one further sub-case, the other ones we already dealt with in the proof of Lemma 14.

If $e_1 \cap e_2 = \emptyset$, then we get by the analysis of Lemma 7, keeping in mind that by branching on say e_1 we still keep the low-size hyperedge e_2:

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

More algebra for 2-LP

We have to analyze the following sets of recurrences:

1. The resulting T^1-case is "pure:"

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

   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):

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

   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, $T^0(k) \leq 5.1844^k$.

2. The involved T^1-case is actually a T^2-case, with two small hyperedges whose intersection contains three elements:

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

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):

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

An argument shift in the first of these equations plus multiplying with two yields:

$$6T^0(k-2) - 2T^0(k-3) = 2T^1(k-1) - 4T^1(k-2).$$

Adding with the previously derived equation gives:

$$T^0(k) - T^0(k-1) + 6T^0(k-2) - 2T^0(k-3) = 5T^1(k-1).$$

We can use this equation to derive from Eq. (2):

$$\begin{aligned} 5T^0(k) - 5T^0(k-1) = {} & 3(T^0(k) - T^0(k-1) + 6T^0(k-2) - 2T^0(k-3)) \\ & +4(T^0(k-1) - T^0(k-2) + 6T^0(k-3) - 2T^0(k-4)) \end{aligned}$$

Hence, $0 = 2T^0(k) - 6T^0(k-1) - 14T^0(k-2) - 18T^0(k-3) + 8T^0(k-4)$. This gives $T^0(k) \leq 4.8089^k$.

3. The involved T^1-case is actually a T^2-case, with two small hyperedges whose intersection contains four elements:

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

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):

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

An argument shift in the first of these equations and multiplying the second one with two yields:

$$3T^0(k-2) - T^0(k-3) = T^1(k-1) - 2T^1(k-2),$$
$$2T^0(k) - 2T^0(k-1) = 8T^1(k-1) + 2T^1(k-2).$$

Adding these equations gives:

$$2T^0(k) - 2T^0(k-1) + 3T^0(k-2) - T^0(k-3) = 9T^1(k-1).$$

We can use this equation to derive:

$$\begin{aligned} 9T^0(k) - 9T^0(k-1) = {} & 4(2T^0(k) - 2T^0(k-1) + 3T^0(k-2) - T^0(k-3)) \\ & +(2T^0(k-1) - 2T^0(k-2) + 3T^0(k-3) - T^0(k-4)) \end{aligned}$$

Hence, $0 = T^0(k) - 3T^0(k-1) - 10T^0(k-2) + T^0(k-3) + T^0(k-4)$. This gives $T^0(k) \leq 4.9653^k$.

So, as predicted, only the "proper" T^1-branch gives a branching behavior which is now actually the second-worst one, only "surpassed" by the scenario when two hyperedges intersect with four elements in their intersection. This is now the natural point of attack for further improvements.

Further improvements for 2-Layer Planarization

With the same reasoning, we can strengthen reduction rule 3b. to a more general, vertex domination like rule.

3b general general (injured) 2-claws: If y is a vertex of degree one in a hyperedge of size four, five or six corresponding to an (injured) 2-claw, and if y (as an edge in the 2-LP instance) is incident to the center of the corresponding 2-claw and participates in a pretty finger $\{y, z\}$ such that (in the 6HS instance) any hyperedge in which y takes part also contains z, then mark y as virtual.

For the soundness of rule 3b general, the following observation is crucial.

Property 2. Let $F = \{cw, wx\}$ be one finger of an (injured) 2-claw C with center c such that cw (in the corresponding 6HS instance) is only contained in hyperedges in which also wx in contained. Then, w has degree two.

The proof of Property 1 nearly literally transfers to this case.

Now, with a somewhat modified "special case" heuristic priority (which will become clear in the following argument), we can further improve the analysis in one case:

If $|e_1 \cap e_2| = 4$, then by pigeon-hole principle, (at least) one of the $x \in e_1 \cap e_2$ will be situated incident to the center of the claw C_1 corresponding to e_1. Since the general rule 3b. did not fire, x participates in another hyperedge (2-claw) besides e_1 and e_2. Hence, not taking x into the hitting set produces a new small hyperedge e' in which neither all vertices from e_1 nor all from e_2 participate (by edge domination). Hence, there are two sub-cases:

– There is an element $y \in e_1 \cap e_2 \setminus \{x\}$ which does not belong to e'. Then, branching at y we get a $T^1(k-1)$-branch. Then, trivial branching at the elements in $e_1 \cap e_2 \setminus \{x, y\}$ give altogether:

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

– Otherwise, $(e_1 \cap e_2) \setminus \{x\} \subseteq e'$. Then, the elements $\{y_1, y_2\}$ from the symmetric difference of e_1 and e_2 do not belong to e', since else the edge domination rule would have destroyed e'.
The heuristic priorities would then first branch at all elements in $e_1 \cap e_2 \setminus \{x\}$ and finally the case that both y_1 and y_2 go into the hitting set remains. In that case, we would continue with a trivial branch at the two elements from $(e' \setminus ((e_1 \cap e_2) \setminus \{x\}))$. Hence:

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

Observe that the underlined branch would not be done according to the present list of heuristic priorities, which would have to be accordingly modified to cover this special situation.

How does this analysis improve the constants?

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

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

The last equation can be reordered to give an expression for $T^1(k)$ (after an argument shift) that can be plugged into the second equation for $T^1(k)$ (together with the first equation that gives an expression for T^2). This yields:

$$\begin{aligned}
T^1(k) &= T^2(k+1) - 3T^0(k) - T^0(k-1) \\
&= \underline{T^0(k+1) - 4T^0(k) - T^0(k-1)} \\
&= 2T^0(k-1) + 2T^1(k-1) + T^2(k-1) \\
&= 2T^0(k-1) + 2(T^2(k) - 3T^0(k-1) - T^0(k-2)) + T^2(k-1) \\
&= 2T^0(k-1) + 2(T^0(k) - 4T^0(k-1) - T^0(k-2)) + T^0(k-1) - T^0(k-2) \\
&= \underline{2T^0(k) - 5T^0(k-1) - 3T^0(k-2)}
\end{aligned}$$

Subtracting the underlined expressions yields:

$$0 = T^0(k+1) - 6T^0(k) + 4T^0(k-1) + 3T^0(k-2),$$

so that we infer $T^0(k) \leq 5.1005^k$.

We stop our hunt for improvements here, since further improvements (along the direction given by the top-down approach to search trees) would involve either a deeper analysis of T^1-branches or a first analysis of T^3-branches. Both alternatives appear to be very tedious.

Assuming $T^2(k) \leq 4T^0(k-1) + 2T^0(k-3)$ This can be directly plugged into

$$\begin{aligned} T^0(k) &= T^0(k-1) + T^2(k) \\ &= 5T^0(k-1) + 2T^0(k-3), \end{aligned}$$

so that we infer $T^0(k) \leq 5.0776^k$.

Proof of lemmas for kernelization algorithm 1-LP

Basically, we are going to further analyze the following lemma due to Dujmović *et al.*:

Lemma 16. *If $G = (A, B; E)$ is a bipartite graph and π is a permutation of A that satisfies condition $(\star)$, 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 17. *Let $G = (A, B; E)$ be a bipartite graph and let π be a permutation of A. If e is an edge in some cycle of length $2\ell > 4$, then e potentially violates condition $(\star)$ or both its neighboring edges (on the cycle) do so. Moreover, every cycle of length $2\ell > 4$ has at most two edges not violating condition $(\star)$.*

Proof of Lemma 17. Consider the cycle $v_1, v_2, \ldots, v_{2\ell}$, $2\ell > 4$, where the vertices with odd indices belong to A. Without loss of generality, we discuss the edge $e = v_2v_3$. Furthermore, by symmetry, we can assume that $v_1 < v_3$ according to the ordering induced by π on A. Now, assume that v_2v_3 and v_1v_2 do *not* potentially violate condition $(\star)$. As in the proof of Lemma 16 (in the long version of [3]), we can conclude that then v_5 cannot be to the left of v_3; otherwise, either v_2v_3 or (v_1v_2 and v_3v_4) potentially violate(s) condition $(\star)$. Since $2\ell > 4$, v_5 is to the right of v_3.

Let $i > 2$ be maximal such that, for all $2 < j \leq i$, v_{2j-1} is to the right of v_3. Hence, $v = v_{2i \bmod (2\ell)+1}$ is to the left of v_3. We discuss three sub-cases:

- If $v_1 < v < v_3$, then (v_1, v_2, v_3) and the edge vv_{2i} violate condition $(\star)$.
- If $v = v_1$, then (v_1, v_{2i}, v_{2i-1}) and the edge e together don't satisfy condition $(\star)$, since $i > 2$.
- If $v < v_1 < v_3$, which is only possible if $i > 3$, then (v, v_{2i}, v_{2i-1}) and the edge e together don't satisfy condition $(\star)$.

Hence, the only situation when $e = v_2v_3$ does not potentially violate condition $(\star)$ arises when v_5 comes to the left of v_1. Pictorially, this means that the embedding of the cycle $v_1v_2 \ldots v_{2\ell}$ (as indicated by π) "makes a turn" at e. A similar turn might happen at the other end (the "left-hand side" of the

embedding). This then gives the possibly two edges on the circle which do not potentially violate ($\star$). If there are more of these turns, then only the "outermost" turns can yield edges which do not potentially violate condition ($\star$).

Proof of Lemma 11. Let $C = (v_1, v_2, \ldots, v_{2\ell})$ and $C' = (v'_1, v'_2, \ldots, v'_{2\ell'})$ 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 ($\star$). W.l.o.g., we can further assume that $v_1 < v_3$.

- If $v_1 < v'_1 < v_3$, then (v_1, v_2, v_3) together with $v'_1v'_{2\ell'}$ do not satisfy condition ($\star$), so that e potentially violates ($\star$), contradicting our assumptions. Similarly, $v'_1 < v_1 < v_3$ can be ruled out.
- If $v'_1 \leq v_1 < v_3$, then both cases $v_3 < v'_3$ and $v'_3 < v_3$ 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 17, since then e and e' would be two subsequent edges on C' which do not potentially violate ($\star$).
- Hence, $v_1 < v_3 < v'_1$ remains as the only possibility. Interchanging the roles of C and C' in the argument, $v'_1 < v'_3$ is ruled out. Hence, $v_1 < v_3, v'_3 < v'_3$. We discuss (w.l.o.g.) $v_1 < v_3 < v'_3 < v'_3$. Then, (v_3, v, v'_1) together with $v'_3v'_4$ violate ($\star$), contradicting our assumption on e.

Remark 2. (following Theorem 3) Rule 1L-RR-isolate enables us to also conclude that a reduced instance $G = (A, B; E)$ does not contain more than $k^3/2$ vertices in $A \cup B$, since $|E| \leq k^3$.

Proofs for the bounded search tree of 1-LP

Lemma 18. *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 three different vertices x, y, z with $x, z \in A$ such that the path (x, y, z) is a part of C, and C contains vertices a, b, c with b, c, y being pairwisely different elements of B such that the edges ab and ac are parts of C and $x < a < z$ according to π.*

Proof of Lemma 18. Let C have length 2ℓ. We first show the following claim:

There is a sub-path (x, y, z), $x, z \in A$, of C such that there is some vertex $a \in A$ on C with $x < a < z$ according to π.

The claim is obviously true for $\ell = 3$ by inspection. Assume it is true up to length-$2\ell'$-cycles. Consider a cycle C' of length $2\ell' + 2$. If the claim were false on C', for sure we can find a sub-path (x, y, z), $x, z \in A$, of C' such that there is no vertex $a \in A$ on C' with $x < a < z$ according to π'. Cutting out y and

identifying x and z (and appropriately updating the permutation π' to π'') gives a cycle C'' of length $2\ell'$ which would also violate the claim, contradicting our induction hypothesis.

Now, take the sub-path (x, y, z) and the vertex $a \in A$ as given by the claim. a has two neighbors b and c on C. Since the two neighbors x and z of y are different from a, neither b nor c equals y. This shows the lemma.

Proof of Lemma 12. So, let $C = \{ab, bc, cd, da\}$ be a 4-cycle with $a \in A$ and $a < c$.

We first investigate how a single additional edge can create one (but not two) situation(s) violating $(\star)$.

1. If, w.l.o.g., (a, b, c) together with some edge e violates $(\star)$, then either also (a, d, c) together with e violates $(\star)$ or d is incident with $e = xd$. In the first case, there is a hyperedge (namely $\{ad, dc, e\}$) that has non-empty intersection with the hyperedge $\{ab, bc, e\}$. This cannot happen after the assumed exhaustive branching.
2. Consider now the case that some path (x, y, z) together with ab violates $(\star)$. Firstly assume that $y \neq d$. Then, (x, y, z) does not satisfy $(\star)$ both together with ab and with ad, which is impossible after exhaustive branching. 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 $a < c$ and if e is an edge that—together with two edges from C—forms a hyperedge reflecting a situation violating $(\star)$, then

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

More specifically, the situation $e = x_d d$ and $a < x_d < c$ showed up in the first part of our case analysis, while the situation $e = x_d d$ and $x_d < a$ 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 hyperedge reflecting situations violating $(\star)$.

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

- $x_d < x'_d < a$: Then, (x_d, d, c) together with ab violates $(\star)$, and (x'_d, d, c) together with ab violates $(\star)$.
- $x_d < a < x'_d$: Then, (x_d, d, c) together with ab violates $(\star)$, and (a, b, c) together with $x'_d d$ violates $(\star)$.
- $a < x_d < x'_d$: Then, (a, b, c) together with $x_d d$ as well as with $x'_d d$ violate $(\star)$.

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 hyperedge reflecting situations violating $(\star)$.

So, the only possibility that C could interfere with more than one hyperedge reflecting situations violating $(\star)$ is when both $e_b = x_b b$ and $e_d = x_d d$ exist. This is rejected with the following case analysis.

- $x_d < a < c < x_b$: Then, (x_d, d, c) together with ab violates $(\star)$, and (a, b, x_b) together with cd violates $(\star)$. Hence, ab shows up in both violating hyperedges.
- $a < x_d < c < x_b$: Then, (x_d, d, c) together with ab violates $(\star)$, and (a, b, x_b) together with $x_d d$ violates $(\star)$. Hence, $x_d d$ shows up in both violating hyperedges.
- $x_d < a < x_b < c$: Symmetrical to the previous case.
- $a < x_d < x_b < c$: Then, (x_d, d, c) together with $x_b b$ violates $(\star)$, and (a, d, c) together with $x_d d$ violates $(\star)$. Hence, $x_d d$ shows up in both violating hyperedges.
- $a < x_b < x_d < c$: Symmetrical to the previous case.

The pseudo-code of the bounded search tree algorithm for 1-LP

In that algorithm, we again use a set of *virtual edges* to mark edges which (according to our previous branching) we shall *not* put into the bipartization set. This part of the input is therefore initialized with $\emptyset$ at the very beginning. The notation $\mathsf{bpr}(G_0, \pi_0, E)$ is accordingly understood.

Proof of Theorem 5

Proof. We only have to analyze the running time in the following. As said before, branching only actually takes place when we "solve" the corresponding 3-Hitting Set instance. During these recursions, we always assume that, whenever we branch at hyperedges of size three, there is some vertex contained in that hyperedge which has size at least two.

More distinctly, let $T^\ell(k)$ denote the situation of a search tree assuming that at least ℓ hyperedges in the given instance (with parameter k) have a size of (at most) 2. Of course, $T(k) \leq T^0(k)$. We again analyze the recurrences for T^0, T^1 and T^2.

Lemma 6 literally transfers, yielding

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

For T^1, we cannot claim to "gain" any new hyperedges 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 hyperedges e_1, e_2 of size two:

1. $e_1 \cap e_2 = \emptyset$. Then, trivial branching gives:

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

2. $\exists x \in e_1 \cap e_2$. Branching at x (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.

Remarks on approximability

In the long version of [3], also approximation algorithms for 2-LP and 1-LP have been discussed. Note that the approximation algorithm for 1-LP found by Dujmović *et al.* could be also read as a translation of the well-known factor-3-approximation algorithm for 3-Hitting Set, see [7] for further hints.

Further conclusions

Observe that recently D. Juedes and P. Shaw (personal communication) have derived an $\mathcal{O}(k^2)$ kernel for 3-Hitting Set, which would entail an $\mathcal{O}(k^2)$ kernel for 1-LP. In their IWPEC 2004 contribution, Nishimura, Ragde and Thilikos obtained a similar result for 3-Hitting Set.

Algorithm 1-Layer BST / 3HS-based
input: graph $G_0 = (A_0, B_0, E_0)$; permutation π_0 of A_0; virtual edge set E
parameter: non-negative integer k_0
output: NO if $\mathsf{bpr}(G_0, \pi_0, E) > k$; otherwise, YES.

1. Exhaustively apply the reduction rules.
   The reduced instance is also denoted (G_0, π_0, k_0, E).
2. **if** $(\star)$ fails for some path (x, v, y) and edge ab with $x < a < y$
   (a) **then if** $k_0 = 0$ **then** return NO.
   (b) **if possible** choose (x, v, y) and ab such that $\{xv, vy, ab\} \cap E \neq \emptyset$
      i. **then if possible** choose (x', v', y') and $a'b'$ such that
         $\{x'v', v'y', a'b'\} \cap E \neq \emptyset$ and
         $\{e\} = \{xv, vy, ab\} \cap \{x'v', v'y', a'b'\} \cap (E_0 \setminus E)$
         // This is a T^2-branch
         – **if** 1-Layer BST / 3HS-based $(G_0 \setminus \{e\}, \pi_0, k_0 - 1, E)$='YES'
         – **then** return YES.
         – **else**
           • Let $\{e, f\} = \{xv, vy, ab\} \cap (E_0 \setminus E)$.
           • Let $\{e, f'\} = \{x'v', v'y', a'b'\} \cap (E_0 \setminus E)$.
           • return 1-Layer BST / 3HS-based $(G_0 \setminus \{f, f'\}, \pi_0, k_0 - 2, E \cup \{e\})$
      ii. **else** // only "isolated" hyperedges of size two (also T^1)
         – Let $\{e, f\} = \{xv, vy, ab\} \cap (E_0 \setminus E)$.
         – **if** 1-Layer BST / 3HS-based $(G_0 \setminus \{e\}, \pi_0, k_0 - 1, E)$='YES'
         – **then** return YES.
         – **else** return 1-Layer BST / 3HS-based $(G_0 \setminus \{f\}, \pi_0, k_0 - 1, E)$
   (c) **else** // T^0-branch
   (d) **if possible** choose (x, v, y) and ab such that
      there are (x', v', y') and $a'b'$ such that
      $e \in \{xv, vy, ab\} \cap \{x'v', v'y', a'b'\}$
      – **if** 1-Layer BST / 3HS-based $(G_0 \setminus \{e\}, \pi_0, k_0 - 1, E)$ = 'YES'
      – **then** return YES.
      – **else** return 1-Layer BST / 3HS-based $(G_0, \pi_0, k_0, E \cup \{e\})$.
   (e) **else** // only "isolated conflicts"
      – **if** $h = \{xv, vy, ab\}$ intersects with some 4-cycle C.
      – **then** Let e be a common edge of h and C.
      – **else** Choose some $e \in h$.
   (f) return 1-Layer BST / 3HS-based $(G_0 \setminus \{e\}, \pi_0, k_0 - 1, E)$.
3. **else** return $k_0 \geq \sum_{\text{maximal } p\text{-diamonds of } G_0} (p - 1)$

Fig. 4. The algorithm 1-Layer BST / 3HS-based

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'