

# Constrained and Ordered Level Planarity Parameterized by the Number of Levels

Václav Blažej  

University of Warwick, Coventry, UK

Boris Klemz  

Institut für Informatik, Universität Würzburg, Germany

Felix Klesen 

Institut für Informatik, Universität Würzburg, Germany

Marie Diana Sieper  

Institut für Informatik, Universität Würzburg, Germany

Alexander Wolff 

Institut für Informatik, Universität Würzburg, Germany

Johannes Zink  

Institut für Informatik, Universität Würzburg, Germany

---

## Abstract

---

The problem LEVEL PLANARITY asks for a crossing-free drawing of a graph in the plane such that vertices are placed at prescribed  $y$ -coordinates (called *levels*) and such that every edge is realized as a  $y$ -monotone curve. In the variant CONSTRAINED LEVEL PLANARITY (CLP), each level  $y$  is equipped with a partial order  $\prec_y$  on its vertices and in the desired drawing the left-to-right order of vertices on level  $y$  has to be a linear extension of  $\prec_y$ . ORDERED LEVEL PLANARITY (OLP) corresponds to the special case of CLP where the given partial orders  $\prec_y$  are total orders. Previous results by Brückner and Rutter [SODA 2017] and Klemz and Rote [ACM Trans. Alg. 2019] state that both CLP and OLP are NP-hard even in severely restricted cases. In particular, they remain NP-hard even when restricted to instances whose *width* (the maximum number of vertices that may share a common level) is at most two. In this paper, we focus on the other dimension: we study the parameterized complexity of CLP and OLP with respect to the *height* (the number of levels).

We show that OLP parameterized by the height is complete with respect to the complexity class XNLP, which was first studied by Elberfeld, Stockhusen, and Tantau [Algorithmica 2015] (under a different name) and recently made more prominent by Bodlaender, Groenland, Nederlof, and Swenenhuis [FOCS 2021]. It contains all parameterized problems that can be solved nondeterministically in time  $f(k) \cdot n^{O(1)}$  and space  $f(k) \cdot \log n$  (where  $f$  is a computable function,  $n$  is the input size, and  $k$  is the parameter). If a problem is XNLP-complete, it lies in XP, but is  $W[t]$ -hard for every  $t$ .

In contrast to the fact that OLP parameterized by the height lies in XP, it turns out that CLP is NP-hard even when restricted to instances of height 4. We complement this result by showing that CLP can be solved in polynomial time for instances of height at most 3.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Design and analysis of algorithms; Theory of computation  $\rightarrow$  Fixed parameter tractability; Human-centered computing  $\rightarrow$  Graph drawings; Theory of computation  $\rightarrow$  Computational geometry

**Keywords and phrases** Parameterized Complexity, Graph Drawing, XNLP, XP,  $W[t]$ -hard, Level Planarity, Planar Poset Diagram, Computational Geometry

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2024.20

**Related Version** *Full Version*: <https://arxiv.org/abs/2403.13702>

**Funding** *Václav Blažej*: Supported by the Engineering and Physical Sciences Research Council [grant number EP/V044621/1].

*Johannes Zink*: Partially supported by DFG project WO 758/11-1.



© Václav Blažej, Boris Klemz, Felix Klesen, Marie Diana Sieper, Alexander Wolff, and Johannes Zink;

licensed under Creative Commons License CC-BY 4.0

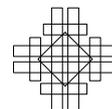
40th International Symposium on Computational Geometry (SoCG 2024).

Editors: Wolfgang Mulzer and Jeff M. Phillips; Article No. 20; pp. 20:1–20:16

Leibniz International Proceedings in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



## 1 Introduction

In an *upward* drawing of a directed graph, every edge  $e = (u, v)$  is realized as a y-monotone curve that goes upwards from  $u$  to  $v$ , i.e., the y-coordinate strictly increases when traversing  $e$  from  $u$  towards  $v$ . Also known as poset diagrams, these drawings provide a natural way to visualize a partial order on a set of items. The classical problem UPWARD PLANARITY asks whether a given directed graph admits a drawing that is both upward and planar (i.e., crossing-free). It is known to be NP-hard [17], but becomes solvable in polynomial time if the y-coordinate of each vertex is prescribed [11, 18, 21]. In contrast, when both the y-coordinate and the x-coordinate of each vertex is prescribed, the problem is yet again NP-hard [23]. The paper at hand is concerned with the parameterized complexity of (a generalization of) the latter variant of UPWARD PLANARITY, the parameter being the number of levels. Next, we define these problems more precisely, adopting the notation and terminology used in [23].

**Level planarity.** A *level graph*  $\mathcal{G} = (G, \gamma)$  is a directed graph  $G = (V, E)$  together with a *level assignment*, which is a surjective map  $\gamma: V \rightarrow \{1, 2, \dots, h\}$  where  $\gamma(u) < \gamma(v)$  for every edge  $(u, v) \in E$ . The vertex set  $V_i = \{v \mid \gamma(v) = i\}$  is called the  $i$ -th *level* of  $\mathcal{G}$ . The *width* of level  $V_i$  is  $|V_i|$ . The *levelwidth* of  $\mathcal{G}$  is the maximum width of any level in  $\mathcal{G}$  and the *height* of  $\mathcal{G}$  is the number  $h$  of levels. A *level planar drawing* of  $\mathcal{G}$  is an upward planar drawing of  $G$  where the y-coordinate of each vertex  $v$  is  $\gamma(v)$ . We use  $L_i$  to denote the horizontal line with y-coordinate  $i$ . Algorithms for computing level planar drawings usually just determine a *level planar embedding* of a level planar drawing, which for each  $i \in \{1, 2, \dots, h\}$  lists the left-to-right sequence of vertices and edges intersected by  $L_i$ . Note that this corresponds to an equivalence class of drawings from which an actual drawing is easily derived. The level graph  $\mathcal{G}$  is called *proper* if  $\gamma(v) = \gamma(u) + 1$  for every edge  $(u, v) \in E$ .

The problem LEVEL PLANARITY asks whether a given level graph admits a level planar drawing. It can be solved in linear time [11, 18, 20, 21]; see [16] for a more detailed discussion on this series of papers. It is easy to see that LEVEL PLANARITY is polynomial time/space equivalent to the variant where  $\gamma$  maps to  $h$  arbitrary distinct real numbers.

**Constrained and ordered level planarity.** In 2017, Brückner and Rutter [8] and Klemz and Rote [23] independently introduced and studied two closely related variants of LEVEL PLANARITY, defined as follows. A *constrained (ordered) level graph*  $\mathcal{G} = (G, \gamma, (\prec_i)_{1 \leq i \leq h})$  is a triplet corresponding to a level graph  $(G, \gamma)$  of height  $h$  equipped with a family containing, for each  $1 \leq i \leq h$ , a partial (total) order on the vertices in  $V_i$ . A *constrained (ordered) level planar drawing* of  $\mathcal{G}$  is a level planar drawing of  $(G, \gamma)$  where, for each  $1 \leq i \leq h$ , the left-to-right order of the vertices in  $V_i$  corresponds to a linear extension of  $\prec_i$  (is  $\prec_i$ ). For a pair of vertices  $u, v \in V_i$  with  $u \prec_i v$ , we refer to  $u \prec_i v$  as a *constraint* on  $u$  and  $v$ .

The problem CONSTRAINED LEVEL PLANARITY (CLP) / ORDERED LEVEL PLANARITY (OLP) asks whether a given constrained / ordered level graph admits a constrained / ordered level planar drawing, in which case the input is called a *constrained / ordered level planar graph*. The special case where the height of all instances is restricted to a given value  $h$  is called  $h$ -level CLP /  $h$ -level OLP. In CLP, each partial order  $\prec_i$  is assumed to be given in form of a directed acyclic graph including all of its transitive edges. In OLP, each total order  $\prec_i$  is encoded by equipping each vertex of level  $V_i$  with an integer that is equal to its rank in the order  $\prec_i$ . Note that OLP is polynomial time/space equivalent to the variant of LEVEL PLANARITY where each vertex is equipped with a prescribed x-coordinate, implying that the only challenge is to draw the edges.

Klemz and Rote [23] showed that OLP (and, thus, CLP) is NP-hard even when restricted to the case where the underlying undirected graph of  $G$  is a disjoint union of paths. Note that such a graph has bounded pathwidth (and treewidth), maximum degree, and feedback vertex set number, ruling out efficient parameterized algorithms with respect to these classical parameters. Additionally, the instances produced by their reduction have a levelwidth of only two. Independently, Brückner and Rutter [8] provided a very different reduction for showing that CLP is NP-hard (in fact, their reduction shows the NP-hardness of PARTIAL LEVEL PLANARITY, which can be seen as a generalization of OLP and a special case of CLP; see below). The instances constructed by their reduction are connected and have bounded maximum degree. They also present a polynomial time algorithm for CLP for the case where the graph has a single sink, which they later sped up [9].

**Other related work.** The problem PARTIAL LEVEL PLANARITY (PLP) (introduced and studied by Brückner and Rutter [8]), asks whether a given level planar drawing of a subgraph  $H$  of the input graph  $G$  can be extended to a level planar drawing of  $G$ . This can be seen as a generalization of OLP and, in the proper case, as a specialization of CLP. Several other problems related to the construction of level planar drawings have been studied, including problems with other kinds of ordering constraints (e.g., CLUSTERED LEVEL PLANARITY [15, 1, 23] and T-LEVEL PLANARITY [26, 1, 23]), problems with a more geometric touch (see, e.g., [19, 22]), and variants of LEVEL PLANARITY seeking drawings on surfaces different from the plane (see, e.g., [3, 2, 4]).

**Contribution and organization.** As stated above, Klemz and Rote [23] showed that OLP (and, thus, CLP) is NP-hard even when restricted to instances of levelwidth two. In this paper, we focus on the other “dimension”: we study CLP and OLP parameterized by height.

We show that OLP parameterized by the height is complete with respect to the complexity class XNLP, which was first studied by Elberfeld, Stockhusen, and Tantau [13] (under the name  $N[f \text{ poly}, f \log]$ ) and recently made more prominent by Bodlaender, Groenland, Nederlof, and Swennenhuis [7]. It contains all parameterized problems that can be solved nondeterministically in time  $f(k) \cdot n^{\mathcal{O}(1)}$  and space  $f(k) \cdot \log n$  (where  $f$  is a computable function,  $n$  the input size, and  $k$  the parameter). Elberfeld et al. and Bodlaender et al. study properties of (problems in) this class and provide several problems that are XNLP-complete. In particular, if a problem is XNLP-complete, it lies in XP, but is  $W[t]$ -hard for every  $t$  [7].

► **Theorem 1.** *ORDERED LEVEL PLANARITY parameterized by the height of the input graph is XNLP-complete (and, thus, it lies in XP, but is  $W[t]$ -hard for every  $t$ ). XNLP-hardness holds even when restricted to the case where the input graph is connected. Moreover, there is a constructive XP-time algorithm for ORDERED LEVEL PLANARITY (w.r.t. the height).*

Parameterizing OLP by height captures the “linear” nature of the solution – this is reminiscent of recent results by Bodlaender, Groenland, Jacob, Jaffke, and Lima [6] who established XNLP-completeness for several problems parameterized by linear width measures (e.g., CAPACITATED DOMINATING SET by pathwidth and MAX CUT by linear cliquewidth). However, to the best of our knowledge, this is the first graph drawing (or computational geometry) problem shown to be XNLP-complete. The algorithms are described in Section 2, whereas the hardness is shown in Section 3. In contrast to the fact that OLP parameterized by the height lies in XP, it is not difficult to see that the socket/plug gadget described by Brückner and Rutter [8] can be utilized in the context of a reduction from 3-PARTITION to show that (PLP and, thus) CLP remains NP-hard even when restricted to instances of

constant height. In fact, the unpublished full version of [8] features such a construction with a height of 7 [25]. Here, we present a reduction that is tailor-made for CLP, showing that it is NP-hard even when restricted to instances of height 4. We complement this result by showing that the (surprisingly challenging) case of instances with height at most three can be solved in polynomial time.

► **Theorem 2.** *CONSTRAINED LEVEL PLANARITY is NP-hard even when restricted to height 4, but instances of height at most 3 can be solved constructively in polynomial time.*

We show the hardness in [5], sketch the algorithm in Section 4, and state open problems in Section 5. Proofs of statements marked with a (clickable)  $\star$  are in the full version [5].

**Notation and conventions.** Given an integer  $k > 0$ , we use  $[k]$  as shorthand for  $\{1, 2, \dots, k\}$ . Given a directed or undirected graph  $G$ , let  $V(G)$  denote the vertex set of  $G$ , and let  $E(G)$  denote the edge set of  $G$ . Recall that level graphs are directed with each edge  $(u, v)$  pointing upwards, i.e.,  $\gamma(u) < \gamma(v)$ . As a shorthand and when the direction is not important, we use both  $uv$  and  $vu$  to refer to a directed edge  $(u, v)$ .

## 2 An XP / XNLP Algorithm for Ordered Level Planarity

In this section, we show that ORDERED LEVEL PLANARITY is in XNLP (and thus in XP) when parameterized by the height  $h$  of the input graph. Moreover, we show how to construct an ordered level planar drawing (if it exists) in XP-time. The main idea of our approach is to continuously sweep the plane with an unbounded y-monotone curve  $s$  from left to right in a monotone fashion such that for each edge  $(u, v)$ , there is a point in time where  $u$  and  $v$  are consecutive vertices along  $s$ . When this happens, the edge can be drawn without introducing any crossings due to the fact that  $s$  moves monotonically. To discretize this idea and turn it into an algorithm, we instead determine a sequence of unbounded y-monotone curves  $S = (s_1, s_2, \dots, s_z)$  that is sorted from left to right (i.e., no point of  $s_{i+1}$  is to the left of  $s_i$ ), has a length of  $z \in \mathcal{O}(n)$ , and contains for every edge  $(u, v)$  a curve  $s_i$  along which  $u$  and  $v$  are consecutive vertices. Now, given  $S$ , the desired drawing can be constructed in polynomial time; for an illustration see Figure 1. Moreover, the sequence  $S$  can be obtained in XP-time/space by exhaustively enumerating all possibilities or in XNLP-time/space by nondeterministic guessing. Let us proceed to formalize these ideas.

**Gaps and positions.** Let  $\mathcal{G} = (G = (V, E), \gamma, (\prec_i)_{1 \leq i \leq h})$  be an ordered level graph and consider one of its levels  $V_i$ . Let  $(v_1, v_2, \dots, v_{\lambda_i})$  be the linear order of  $V_i$  corresponding to  $\prec_i$ . In an ordered level planar drawing of  $\mathcal{G}$ , the vertices  $V_i$  divide the line  $L_i$  into a sequence of open line-segments and rays, which we call the *gaps* of  $L_i$ . A *position* on  $L_i$  is a gap of  $L_i$  or a vertex of  $V_i$ . Each position on  $L_i$  is encoded by an index in  $P_i = \{0\} \cup [2\lambda_i]$ : the index 0 represents the gap that precedes  $v_1$ ; an odd index  $p$  represents the vertex  $v_{\lceil p/2 \rceil}$ ; and an even index  $p \neq 0$  represents the gap that succeeds  $v_{\lfloor p/2 \rfloor}$ .

**Separations.** A *separation* for  $\mathcal{G}$  is an element of  $P_1 \times P_2 \times \dots \times P_h$ . Intuitively, a separation  $s = (p_1, p_2, \dots, p_h)$  represents the equivalence class of unbounded y-monotone curves that intersect line  $L_i$  in position  $p_i$  for each  $1 \leq i \leq h$ ; see Figure 1. We say that a vertex  $v \in V$  is *on*  $s$  if  $p_{\gamma(v)}$  represents  $v$ . Moreover, we say that  $v$  is *to the left (right)* of  $s$  if the index corresponding to the position of  $v$  is strictly smaller (larger) than  $p_{\gamma(v)}$ . Consider two vertices  $u, v \in V$  that are on  $s$  and where  $\gamma(u) < \gamma(v)$ . We say that  $u$  and  $v$  are *consecutive* on  $s$  if

all of the indices  $p_{\gamma(u)+1}, p_{\gamma(u)+2}, \dots, p_{\gamma(v)-1}$  represent gaps. In this case,  $v$  is the *successor* of  $u$  on  $s$  and  $u$  is the *predecessor* of  $v$  on  $s$ . We say that  $s$  *uses* an edge  $e = (u, v) \in E$  if  $u$  and  $v$  are consecutive along  $s$ .

**Sweeping sequences.** A *sweeping sequence* for  $\mathcal{G}$  is a sequence  $S = (s_1, s_2, \dots, s_z)$  of separations for  $\mathcal{G}$  such that for each  $j \in [z - 1]$ , we have  $s_j \leq s_{j+1}$ , componentwise. We say that a sweeping sequence *uses* an edge  $e \in E$ , if it contains a separation that uses  $e$ .

► **Lemma 3** ( $\star$ ). *Let  $\mathcal{G} = (G = (V, E), \gamma, (\prec_i)_{1 \leq i \leq h})$  be an ordered level graph and let  $S = (s_1, s_2, \dots, s_z)$  be a sweeping sequence for  $\mathcal{G}$  that uses every edge of  $E$ . Given  $\mathcal{G}$  and  $S$ , an ordered level planar drawing of  $\mathcal{G}$  can be constructed in  $\mathcal{O}(zh + n^2)$  time, where  $n$  is the number of vertices.*

Let  $S = (s_1, s_2, \dots, s_z)$  be a sweeping sequence for our ordered level graph  $\mathcal{G}$ . We say that  $S$  is *nice* if for each  $i \in [z - 1]$ , the separation  $s_{i+1}$  is obtained from  $s_i$  by incrementing exactly one component by 1. Moreover, we say that  $S$  is *exhaustive* if it is nice and  $s_1 = (0, 0, \dots, 0)$  and  $s_z = (|P_1| - 1, |P_2| - 1, \dots, |P_h| - 1)$ ; see Figure 1.

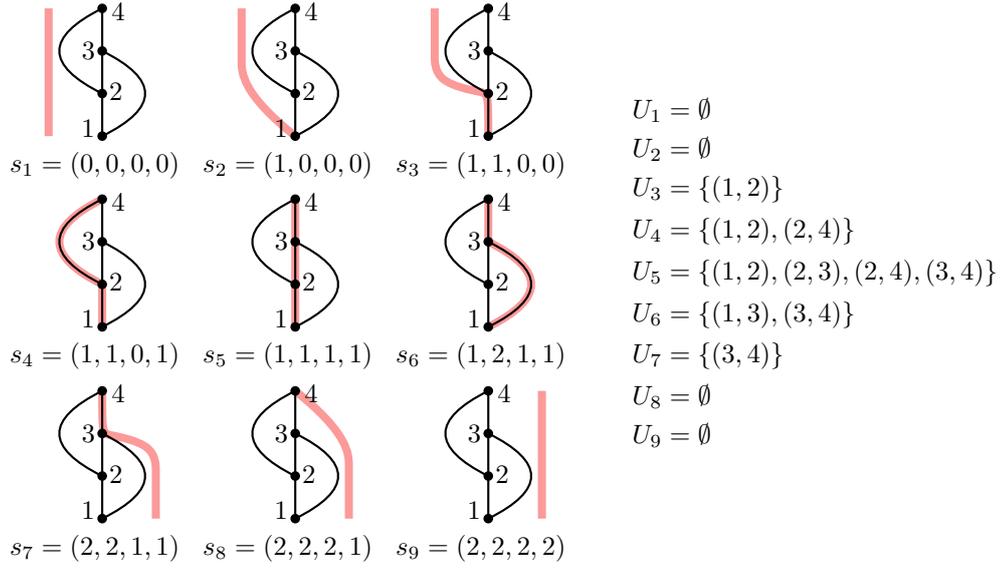
► **Lemma 4** ( $\star$ ). *Let  $\mathcal{G} = (G = (V, E), \gamma, (\prec_i)_{1 \leq i \leq h})$  be an ordered level graph. Then  $\mathcal{G}$  admits an ordered level planar drawing if and only if there is an exhaustive sweeping sequence for  $G$  that uses every edge in  $E$ .*

**Proof (sketch).** The “if”-direction follows from Lemma 3. For the “only if”-direction, we show that the drawing can be internally triangulated by inserting y-monotone edges (allowing the use of parallel edges). For each y-monotone path  $P$  that spans all levels, there is a nice sweeping sequence from  $s_1 = (0, \dots, 0)$  to the separation corresponding to  $P$  that uses all edges left of  $P$ . This is shown by arguing inductively on the number of triangles left of  $P$ . ◀

We remark that an exhaustive sweeping sequence using all edges corresponds directly to a particularly well-structured path decomposition. Thus, Lemma 4 implies that every (ordered) level planar drawing of height  $h$  represents a graph of pathwidth at most  $h - 1$ ; a statement that was independently proven in [12]. However, the path decompositions constructed in the proof of [12, Lemma 1] do not exhibit the same properties that are inherent to exhaustive sweeping sequences and on which our algorithms heavily rely. In particular, these path decompositions may contain bags with multiple vertices of a given level (unless the drawing is proper). Moreover, the existence of a path decomposition of width at most  $h - 1$  for an ordered level graph  $\mathcal{G}$  of height  $h$  does not characterize the fact that  $\mathcal{G}$  is ordered level planar (recall that, in fact, OLP is NP-hard even when restricted to instances of pathwidth 1 [23]).

► **Lemma 5** ( $\star$ ). *There is an algorithm that determines whether a given ordered level graph admits an exhaustive sweeping sequence using all of its edges. It can be implemented deterministically using  $\mathcal{O}^*(2^{\binom{h}{2}} \prod_{j \in [h]} (2\lambda_j + 1)) \subseteq \mathcal{O}^*(2^{\binom{h}{2}} (2\lambda + 1)^h) \subseteq \mathcal{O}^*(2^{\binom{h}{2}} (2n + 1)^h)$  time and space, or nondeterministically using polynomial time and  $\mathcal{O}(h^2 + h \log I)$  space, where  $\lambda$  and  $h$  denote the width and height of the input graph, respectively,  $n$  denotes the number of vertices,  $\lambda_j$  denotes the width of level  $j \in [h]$ , and  $I$  denotes the input size. Further, the deterministic version can report the sequence (if it exists).*

**Proof (sketch).** Let  $\mathcal{G} = (G = (V, E), \gamma, (\prec_i)_{1 \leq i \leq h})$  be an ordered level graph and let  $s = (p_1, p_2, \dots, p_h)$  be a separation for  $\mathcal{G}$ . Further, let  $U \subseteq E$  be a subset of the edges that are joining pairs of (not necessarily consecutive) vertices on  $s$ . We define  $T[s, U] = \mathbf{true}$  if there exists a nice sweeping sequence for  $\mathcal{G}$  that starts with  $s_1 = (0, 0, \dots, 0)$ , ends with  $s$ ,



■ **Figure 1** An exhaustive sweeping sequence using all edges of the depicted graph and the corresponding (cf. Lemma 3) ordered level planar drawing, as well as the corresponding sequence of **true** dynamic programming table entries  $T[s, U]$  (cf. Lemma 5). Note that in any ordered level planar drawing of the graph, exactly one of the edges  $(2, 4)$ ,  $(1, 3)$  is located to the left of the path  $(1, 2, 3, 4)$ , while the other is located to the right. Similarly, in any exhaustive sweeping sequence containing separation  $s_5$ , exactly one of the edges  $(2, 4)$ ,  $(1, 3)$  is used by a separation preceding  $s_5$ , while the other is used by a separation succeeding  $s_5$ . Hence, when iteratively building an exhaustive sweeping sequence, it is key to remember which edges between vertices of the current separation have already been used – this is exactly the purpose of the sets  $U$ . E.g., the fact that  $(2, 4) \in U_5$  corresponds to  $(2, 4)$  being used before  $s_5$  (in  $s_4$ ), from which one can infer how to proceed.

and uses all edges in  $U$ , as well as all edges in  $E$  incident to at least one vertex to the left of  $s$ . Otherwise,  $T[s, U] = \text{false}$ . Additionally, we allow  $U = \perp$ , in which case  $T[s, U] = \text{false}$ . Figure 1 illustrates several **true** table entries  $T[s_i, U_i]$  along with the corresponding sweeping sequences  $s_1, s_2, \dots, s_i$ . Our goal is to determine  $T[(|P_1| - 1, |P_2| - 1, \dots, |P_h| - 1), \emptyset]$ , which is **true** if and only if there exists an exhaustive sweeping sequence for  $\mathcal{G}$  using all edges in  $E$ .

We will determine the entries  $T[s, U]$  by means of a dynamic programming recurrence. For the base case, we simply set  $T[(0, 0, \dots, 0), \emptyset] = \text{true}$ . Now assume that  $s \neq (0, 0, \dots, 0)$ . For each index  $1 \leq j \leq h$  where  $p_j \geq 1$ , we define a separation

$$s'_j = (p_1, p_2, \dots, p_{j-1}, p_j - 1, p_{j+1}, \dots, p_h).$$

Further, we define an edge set  $U'_j \subseteq E$  as follows:

- If  $p_j$  represents a vertex  $v$ , then
  - $U'_j = \perp$  if  $U$  contains edges joining  $v$  with vertices on  $s$  that are not its predecessor or successor on  $s$ ;
  - otherwise  $U'_j$  is created from  $U$  by removing the (up to two) edges incident to  $v$ .
- If  $p_j$  represents a gap and, thus,  $p_{j-1}$  represents a vertex  $v$ , then
  - $U'_j = \perp$  if  $v$  is adjacent to a vertex to the right of  $s$ ;
  - otherwise  $U'_j$  is created from  $U$  by
    - \* removing the edge between the predecessor and successor of  $v$  along  $s'_j$  (if it exists and is contained in  $U$ ); and
    - \* adding all edges in  $E$  that join  $v$  with some vertex on  $s'_j$ .

For illustrations, refer to Figure 1 (where for each  $i \in [9]$  and  $s = s_i$  and  $U = U_i$ , we have  $s'_j = s_{i-1}$  and  $U'_j = U_{i-1}$  for some  $j \in [4]$ ). The following technical claim describes our recurrence relation.

► **Claim 6** ( $\star$ ).  $T[s, U] = \text{true}$  if and only if there exists an index  $1 \leq j \leq h$  where  $p_j \geq 1$  and such that  $T[s'_j, U'_j] = \text{true}$ .

Our table  $T$  has  $2^{\binom{h}{2}} \prod_{j \in [h]} (2\lambda_j + 1)$  entries. In view of Claim 6, we can compute the value of a table entry  $T[s, U]$  in polynomial time (disregarding the time spent for the recursive calls) by simply constructing the tuples  $(s'_1, U'_1), \dots, (s'_h, U'_h)$  and then setting  $T[s, U] = \bigvee_{j \in [h]} T[s'_j, U'_j]$  (only taking into account the defined quantities). Hence, by employing memoization and the usual back-linked strategy, we obtain the claimed deterministic algorithm. For the nondeterministic algorithm, we perform  $2n$  steps. In each step, we nondeterministically guess the next separation  $s$  and its set of prescribed edges  $U$  and check whether for some  $j \in [h]$ , the previous separation  $s'$  is equal to  $s'_j$  and the previous set of prescribed edges  $U'$  is equal to  $U'_j$  (starting with  $s' = (0, 0, \dots, 0)$  and  $U' = \emptyset$ ). At each point in time, we only need to keep two separations, two edge sets, and  $\mathcal{O}(1)$  pointers in memory, which can be done using  $\mathcal{O}(h^2 + h \log I)$  space. ◀

As a corollary of Lemmas 3–5, we obtain the algorithmic statements in Theorem 1.

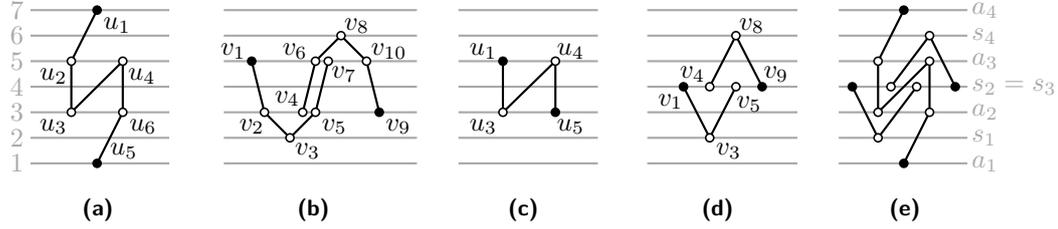
### 3 XNLP-Hardness of Ordered Level Planarity

For the ease of presentation, we first show that the OLP problem is  $W[1]$ -hard. To this end, we use a *parameterized reduction*<sup>1</sup> from MULTICOLORED INDEPENDENT SET (defined below) with  $k$  colors to ORDERED LEVEL PLANARITY with  $\mathcal{O}(k)$  levels. We then describe how to extend this reduction to obtain XNLP-hardness of OLP. We start by introducing the building blocks of our reductions. Recall that, normally, the level assignment  $\gamma$  of a level graph surjectively maps to a set  $[h]$  of consecutive numbers. In this section, to facilitate the description of our gadgets, we relax this condition by temporarily allowing level graphs in which not every level is occupied – nevertheless, the final outcome of our reduction will be an ordered level graph  $\mathcal{G}$  in the original sense.

**Basic building blocks of our reduction.** Our construction of  $\mathcal{G}$  is heavily based on two gadgets that we call *plugs* and *sockets* (a very basic version of these gadgets was already used earlier, in an NP-hardness proof for CLP [8]; here we introduce generalized versions). We define both in terms of the list of levels their vertices occupy. Their  $\prec_i$  orderings are according to the indices of their vertices. The vertices are deliberately given in an unintuitive order to allow for the ordering in  $\prec_i$  by indices and to make the (degenerate) cases behave nicely later. Let  $\ell_1, \ell_2, \ell_3, \ell_4 \in [h]$  such that  $\ell_1 < \ell_2 < \ell_3 < \ell_4$ . A (non-degenerate)  $(\ell_1, \ell_2, \ell_3, \ell_4)$ -*plug*, see Figure 2a, contains vertices  $u_1, u_2, u_3, u_4, u_5$ , and  $u_6$ , where  $\gamma(u_5) = \ell_1$ ,  $\gamma(u_3) = \gamma(u_6) = \ell_2$ ,  $\gamma(u_2) = \gamma(u_4) = \ell_3$ , and  $\gamma(u_1) = \ell_4$ . It contains the edges  $u_1u_2, u_2u_3, u_3u_4, u_4u_6$ , and  $u_6u_5$ ; i.e., a plug is a path that traverses its four levels in the order  $\ell_4, \ell_3, \ell_2, \ell_3, \ell_2, \ell_1$ . Similarly, an  $(\ell_1, \ell_2, \ell_3, \ell_4)$ -*socket*, see Figure 2b, consists of vertices  $v_1, v_2, \dots, v_{10}$  such that  $\gamma(v_3) = \ell_1$ , vertices  $v_2, v_4, v_5, v_9$  occupy level  $\ell_2$ , vertices  $v_1, v_6, v_7, v_{10}$  occupy level  $\ell_3$ , and  $\gamma(v_8) = \ell_4$ . It contains the edges  $v_1v_2, v_2v_3, v_3v_5, v_5v_7, v_4v_6, v_6v_8, v_8v_{10}$ , and  $v_{10}v_9$ . Observe that a socket consists of two disconnected paths whose vertices interleave on levels  $\ell_2$  and  $\ell_3$ . Let vertices  $v_1$

<sup>1</sup> For an overview of this standard technique, refer to a standard textbook [10].

and  $v_9$  of plugs and vertices  $u_1$  and  $u_5$  of sockets be *connecting vertices*. Connecting vertices of sockets will be identified with other vertices of the construction and connecting vertices of some plugs may be connected to other plugs via additional edges.



■ **Figure 2** Plugs and sockets; (a) a  $(1, 3, 5, 7)$ -plug, (b) a  $(2, 3, 5, 6)$ -socket, (c) a degenerate  $(3, 3, 5, 5)$ -plug, (d) a degenerate  $(2, 4, 4, 6)$ -socket, (e) a  $(1, 3, 5, 7)$ -plug that is linked to a degenerate  $(2, 4, 4, 6)$ -socket. Connecting vertices are filled in black. Note how in degenerate gadgets ((c) and (d)), the edges and vertices of the repeated levels are contracted.

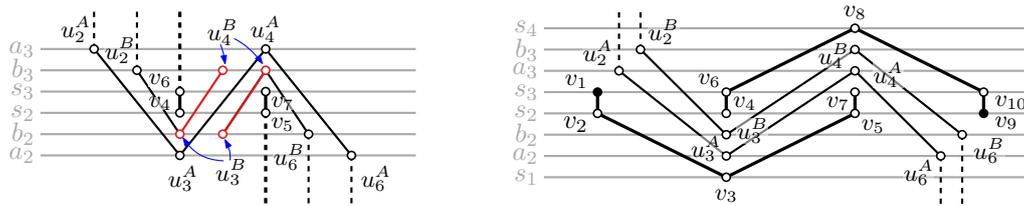
Now we lift the strict inequality restriction on the levels of our gadgets. For plugs we only require  $\ell_1 \leq \ell_2 < \ell_3 \leq \ell_4$  and for sockets we require  $\ell_1 < \ell_2 \leq \ell_3 < \ell_4$  and we call a plug or a socket *degenerate* if it has at least one pair of repeated levels, i.e., for some  $i \in [3]$ ,  $\ell_i = \ell_{i+1}$ . We create the degenerate plugs and sockets by contracting the edges between vertices of the repeated levels while keeping the vertex with the lower index; see Figure 2c and 2d.

Let  $P$  be a plug, and let  $\text{com}(P)$  be the connected component of  $P$  in  $G$ . A plug *fits* into a socket if the gadgets could be “weaved” as illustrated in Figure 2e. Formally, we say an  $(a_1, a_2, a_3, a_4)$ -plug  $P$  fits into a  $(s_1, s_2, s_3, s_4)$ -socket  $S$  when  $\min_{v \in \text{com}(P)} \{\gamma(v)\} \leq s_1$ ,  $\max_{v \in \text{com}(P)} \{\gamma(v)\} \geq s_4$ , and  $s_1 < a_2 < s_2 \leq s_3 < a_3 < s_4$ . In an ordered level planar drawing of  $\mathcal{G}$ , we say a plug  $P$  *links* to an  $(s_1, s_2, s_3, s_4)$ -socket  $S$  when  $P$  fits into  $S$ , and  $P$  is drawn between the connecting vertices of  $S$  (that is, the edges of  $P$  traversing level  $s_3$  are to the right of  $v_1$  and the edges of  $P$  traversing level  $s_2$  are to the left of  $v_9$ ); see Figure 2e.

A defining feature of our constructions is a division into vertical strips, which is accommodated by the following notion. Given a set  $L$  of levels, a *wall* of  $L$  is a path that starts in a vertex on the bottommost level in  $L$ , goes through a vertex on each intermediate level of  $L$ , and ends at a vertex on the topmost level of  $L$ . Note that each gadget type essentially has a unique drawing in the sense that it corresponds to an ordered level planar graph for which all of its ordered level planar drawings have the same level planar embedding. This and further gadget properties are detailed in the full version [5]. We use Lemma 7 to design specific plugs that can or cannot link to specific sockets in the same ordered level planar drawing.

► **Lemma 7** ( $\star$ ). *Consider an ordered level graph  $\mathcal{G}$  that contains an  $(a_1, a_2, a_3, a_4)$ -plug  $A$ , a  $(b_1, b_2, b_3, b_4)$ -plug  $B$ , and an  $(s_1, s_2, s_3, s_4)$ -socket  $S$  that occupy three disjoint sets of levels. There is an ordered level planar drawing of the subgraph of  $\mathcal{G}$  spanned by  $A$ ,  $B$ , and  $S$  where  $A$  and  $B$  link to  $S$  if and only if both  $A$  and  $B$  fit into  $S$  and  $a_2 < b_2$  and  $a_3 < b_3$ , or, both vice versa,  $a_2 > b_2$  and  $a_3 > b_3$ .*

**Proof (sketch).** To distinguish the vertices of the plugs  $A$  and  $B$ , we use the name of the plug as superscript, so, e.g., we refer to  $u_3^A$  and  $u_3^B$ . If the conditions are satisfied, we can draw both plugs as depicted in Figure 3b. Towards a contradiction, assuming the conditions are not satisfied, we observe that the drawing must locally “nest” plug  $B$  into plug  $A$  while drawing them around the edge  $v_4v_6$  of  $S$ , at the same time  $B$  “nests” into  $A$  while going around the edge  $v_7v_5$  of  $S$ ; see Figure 3a. The first nesting implies that the edge  $u_3^B u_4^B$  is to the left of  $u_3^A u_4^A$  and the second nesting implies the opposite order – a contradiction. ◀



(a) If the order of the layers follows the pattern  $a_2, b_2, b_3, a_3$ , then there is no crossing-free drawing that has both  $A$  and  $B$  linked to  $S$ .  
 (b) If the order of the layers follows the pattern  $a_2, b_2, a_3, b_3$ , then there is a crossing-free drawing that has both  $A$  and  $B$  linked to  $S$ .

■ **Figure 3** Sketches for the proof of Lemma 7. The edges of the socket  $S$  are bold.

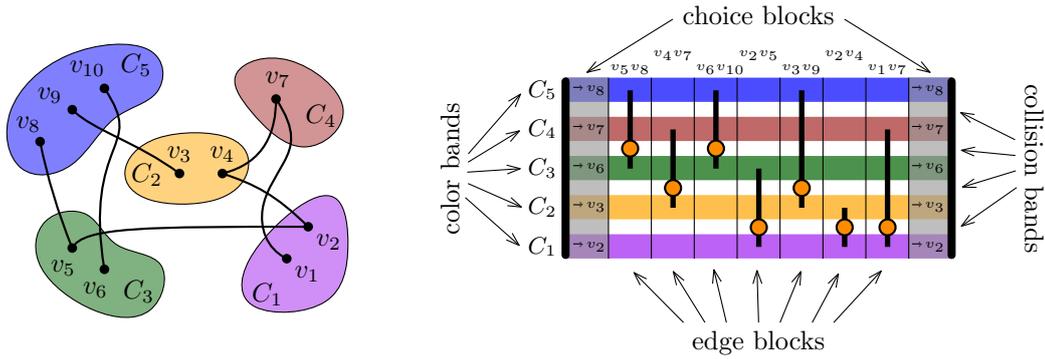
**$W[1]$ -hardness.** MULTICOLORED INDEPENDENT SET (MCIS) is a well-known  $W[1]$ -hard problem [14, 24]. In this problem, we are given a graph  $H$ , an integer  $k$  (the parameter), and a  $k$ -partition  $C_1, C_2, \dots, C_k$  of  $V(H)$ , and the task is to decide whether  $H$  has an independent set  $X \subseteq V(H)$  that contains, for every  $j \in [k]$ , exactly one vertex of  $C_j$ . We may assume without loss of generality that in  $H$ , there is no edge whose endpoints have the same color  $C_j$ .

► **Theorem 8** ( $\star$ ). ORDERED LEVEL PLANARITY is  $W[1]$ -hard with respect to the height.

**Proof (sketch).** We give an overview of a parameterized reduction from MCIS to OLP; refer to Figure 4 for an illustration. The ordered level graph constructed in this reduction admits a grid-cell structure where the vertical strips are separated by walls and the horizontal strips represent related groups of levels. More precisely, we allocate a constant number of consecutive levels per color of the MCIS instance, to which we refer as a *color band*. Between each pair of neighboring color bands, we have a *collision band*. We group several consecutive vertical strips to obtain *blocks*. We call the leftmost and rightmost block *choice blocks* and the other blocks *edge blocks*. Edge blocks correspond bijectively to edges of the MCIS instance.

In the grid cells of the color bands, we insert sockets and we add as many plugs as we have sockets there. No two plugs can link to the same socket, so all available sockets are occupied by one plug. Each plug is a separate connected component, which allows that plug to freely “choose” a socket within its color band where it fits. However, we have different types of plugs and sockets – the first type of plug, called *high plug*, fits only into the sockets of the choice blocks. In a drawing of our ordered level graph, the number of high plugs that link to the left (and not to the right) choice block within a color band forces the second type of plugs, called *color plugs*, to be *shifted* further to the right. In the MCIS instance, a shift by  $i$  corresponds to selecting the  $i$ -th vertex of the respective color to be in the solution.

To obtain an independent set, we need to prevent that the shifts select both endpoints of an edge. To that end, we model an edge  $uv$  as follows. The color plugs of one color band all occupy the same set of levels so they form a sequence specified in the ordered level graph. Because of this, we can identify, for a given shift, what socket a particular color plug links to. Consider the edge block of  $uv$  and the color plugs that necessarily link to its sockets (a block has circa twice as many sockets than the maximal shift). For an illustration of the construction, assume that we have a shift in the color band of  $u$  that selects  $u$  and a shift in the color band of  $v$  that selects  $v$ . We select a specific color plug of the color band of  $u$  and one in the color band of  $v$  to be extended so that they “reach” towards each other and end up in the same cell of a collision band. To this cell, we add a *collision socket*. If at most one extended plug ends up in a collision socket, then it can be drawn planarly. However, if both extended plugs end up in the same collision socket, then this cannot be drawn without crossings. Hence, there is an ordered level planar drawing of our ordered level graph if and only if there is a solution (i.e., an independent set) to the given MCIS instance. ◀



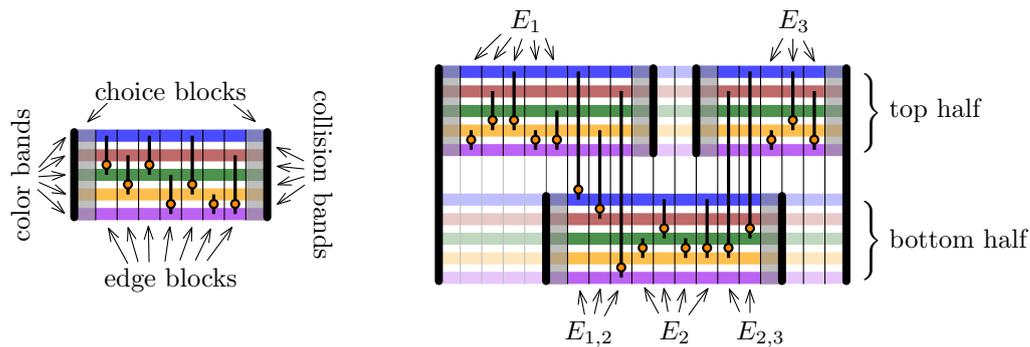
■ **Figure 4** Example of our parameterized reduction from MULTICOLORED INDEPENDENT SET (MCIS) to ORDERED LEVEL PLANARITY. On the left side, there is an instance of MCIS with  $k = 5$  colors. On the right side, there is the schematized grid structure of the ordered level graph constructed from the MCIS instance. The orange disks in the edge blocks represent the places where the collision sockets are placed. Here, the solution  $\{v_2, v_3, v_6, v_7, v_8\}$  for the MCIS instances is found.

**XNLP-hardness.** We now extend our construction for  $W[1]$ -hardness to obtain XNLP-hardness as well. We design a *parameterized tractable log-space reduction* [7, Section V.B.] – a parameterized reduction that runs in  $\mathcal{O}(g(k) \cdot n^c)$  time and uses only  $\mathcal{O}(f(k) \cdot \log n)$  space for internal computation, where  $g$  and  $f$  are computable functions,  $k$  is the parameter, and  $n$  is the input size. It suffices to design the reduction such that it returns a single bit of the output on demand. Observe that with such a reduction, we can retrieve the entire output by requesting one bit at a time. We reduce from CHAINED MULTICOLORED INDEPENDENT SET (CMCIS) – an XNLP-complete problem defined by Bodlaender et al. [7], which is a sequential version of MCIS. As in MCIS, we are given a  $k$ -colored graph  $H$  with color classes  $C_1, \dots, C_k$  and, additionally, there is an  $r$ -partition  $V_1, \dots, V_r$  of  $V(H)$  such that for every  $vw \in E(H)$ , if  $v \in V_i$  and  $w \in V_j$ , then  $|i - j| \leq 1$ . The task is to select an independent set  $X \subseteq V(G)$  such that, for each  $i \in [r]$  and for each color  $j \in [k]$ ,  $|X \cap V_i \cap C_j| = 1$ .

► **Theorem 9** (★). *ORDERED LEVEL PLANARITY is XNLP-hard with respect to the height.*

**Proof (sketch).** The main idea in our reduction from CMCIS to OLP is to use  $r$  instances of the  $W[1]$ -hardness construction from Theorem 8. We have one instance for each subgraph induced by  $V_i$  (for  $i \in [r]$ ) where additionally the edges connecting  $V_i$  with  $V_{i-1}$  and  $V_{i+1}$  (if existent) are represented by edge blocks shared by the corresponding two instances; see Figure 5. To this end, we arrange all instances in a zigzag pattern in the top and the bottom half of the combined construction. Now to make it a parameterized tractable log-space reduction, which outputs bit by bit, we essentially traverse the grid structure with a vertical sweep-line from left to right (i.e., column by column) and save as the status only the  $\mathcal{O}(k)$  information belonging to the current column and the  $\mathcal{O}(\log n)$  information telling us in which column we are currently globally. We describe this in more detail in the full version [5]. ◀

The construction in the proof of Theorem 9 can be altered to make the graph connected as we show in the full version [5]. The main idea is to add a new vertex  $q$  on top of the construction and connect all connected components to it. All sockets are already connected to walls and from walls it suffices to create edges to  $q$  from their topmost vertex. For the plugs, we duplicate every column into a left and a right part such that each left part behaves as before, while each right part hosts paths that connect the “old” plugs to  $q$ . These paths are made of a sequence of a new kind of plugs. This comes at the expense of using more levels, namely  $\Theta(k^2)$  instead of  $\Theta(k)$ . This yields the XNLP-hardness statement in Theorem 1.



■ **Figure 5** Comparison of the  $W[1]$ -hardness construction (left) and the XNLP-hardness construction (right) with  $r = 3$  and  $k = 5$ . The thick walls terminating the choice blocks have vertices on all levels. Vertical bars and orange disks between color bands represent the extended plugs and the collision sockets, respectively, which are used for the collision mechanism.

#### 4 Tractability of 3-Level Constrained Level Planarity

In this section, we sketch our polynomial-time algorithm for 3-level CLP (see Theorem 2). For simplicity, we assume that the given level graph  $\mathcal{G}$  is constrained level planar and show how to compute a constrained level planar drawing of  $\mathcal{G}$ . We ask the reader to consult the figures to which we refer in order to get an intuitive understanding of the notions that we cannot define formally due to space constraints. For details and proofs, see the full version [5].

Without loss of generality, we make the following assumptions: (i)  $\mathcal{G}$  is proper; otherwise we subdivide each edge that crosses level 2. (ii)  $\mathcal{G}$  has no isolated vertices; they can easily be inserted in a postprocessing step. (iii) The *component-constraint graph*  $H$  is strongly connected. This directed graph has a node for each connected component (for short, component) of  $\mathcal{G}$  and an arc for each pair  $(C, C')$  of components such that there is a constraint from a vertex in  $C$  to a vertex in  $C'$ .

We often refer to level 1 as *bottom* level, to level 2 as *middle* level, and to level 3 as *top* level. Similarly, we call the pair of bottom and middle level *lower band* and the pair of middle and top level *upper band*. Our algorithm successively adds new constraints to the given constrained level graph  $\mathcal{G}$  that do not violate planarity. We deduce these new constraints from the structure of  $(G, \gamma)$  and from the current set of constraints. In the end, this yields a total order of the vertices for each of the three levels and, hence, a constrained level planar drawing of  $\mathcal{G}$ . In the very beginning and whenever we add new constraints, we exhaustively add the following *implicit* constraints:

- transitivity:  $\forall a, b, c \in V(G), \forall i \in [3]: a \prec_i b \wedge b \prec_i c \Rightarrow a \prec_i c$
- planarity:  $\forall ab, cd \in E(G)$  with  $i := \gamma(a) = \gamma(c)$  and  $j := \gamma(b) = \gamma(d)$ :  
 $a \prec_i c \Rightarrow (b \prec_j d \vee b = d)$

The former ensures that the orderings  $(\prec_i)_{i \in [3]}$  remain transitive while the latter can be added without violating realizability, as they need to be respected in every constrained level planar drawing. The propagation of these constraints is quite useful as it can dictate the relative positions of vertices that are initially unrelated, see Figure 6.

In a constrained level planar drawing of  $\mathcal{G}$ , a component  $C'$  *hooks into* a component  $C$  if there are vertices  $u \neq v$  of  $C$  and vertices  $u' \neq v'$  of  $C'$  such that  $u, u', v, v'$  occur in this order on the middle level,  $u' \prec_2 v$ , and  $u$  and  $v'$  are the leftmost and rightmost middle-level vertices of  $C \cup C'$ , respectively (as  $C_7$  hooks into  $C_3$  in Figure 7). We guess a pair  $(C, C')$  of components in order to find, with the help of  $H$ , a unique sequence  $\langle C = C_1, C_2, \dots, C_k = C' \rangle$

20:12 Constrained and Ordered Level Planarity Parameterized by the Number of Levels

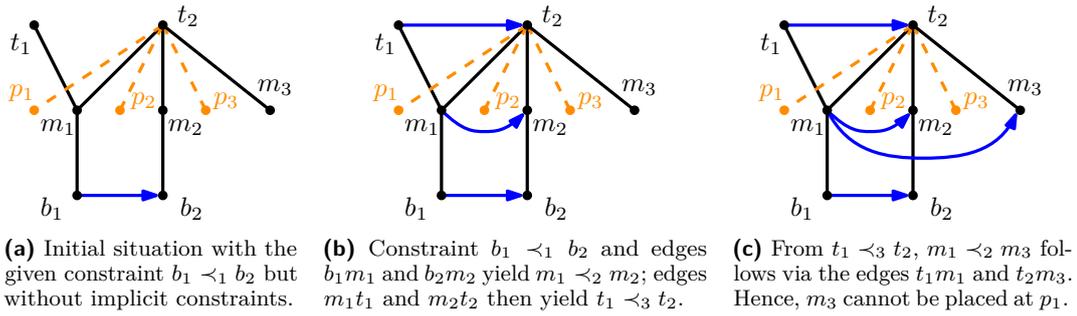


Figure 6 Given a constrained level graph with only one constraint ( $b_1 \prec_1 b_2$ ). Among the three possible positions ( $p_1$ ,  $p_2$ , and  $p_3$ ) to place the middle-level leaf  $m_3$  relative to the neighbors ( $m_1$  and  $m_2$ ) of its parent  $t_2$ , position  $p_1$  is excluded due to the implicit constraints ensuring planarity.

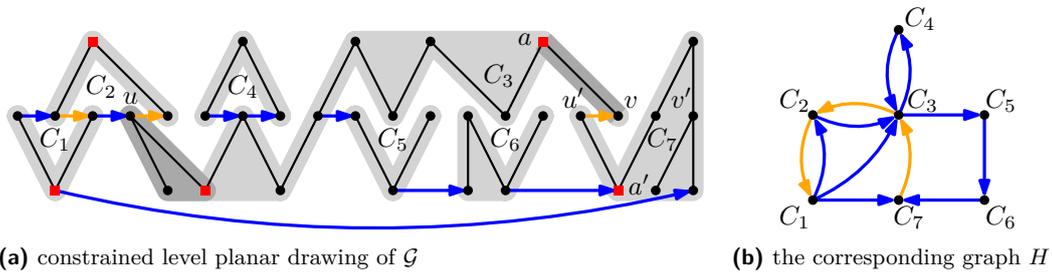


Figure 7 Finding the hook chain  $\langle C_1, C_2, C_3, C_7 \rangle$  (corresponding arcs marked in orange).

of components such that, for  $i \in [k-1]$ ,  $C_{i+1}$  hooks into  $C_i$ . We call  $\langle C_1, C_2, \dots, C_k \rangle$  the *hook chain* of  $\mathcal{G}$ . Let  $\mathcal{G}_1$  be a copy of  $\mathcal{G}$  where we add, for each pair of consecutive components in the hook chain, two edges that connect these components; see the green edges in Figure 8. We show that  $\mathcal{G}_1$  is constrained level planar. Note that  $\mathcal{G}_1$  consists of a single *main component* and *enclosed components* that are forced (by constraints) to “nest” within the main component. Enclosed components (as  $C_4$ ,  $C_5$ , or  $C_6$  in Figure 7a) do not hook.

Next we guess a pair  $(s, t)$  of vertices of the main component such that removing all vertices that lie on simple  $s-t$  paths yields components that lie either on the lower or on the upper band (and hence, have simple structure). Let the *backbone* be the set of all vertices that lie on any  $s-t$  path. In Figure 9, the backbone is marked in red. Each component that hangs off the backbone (including its *anchor* vertex on the backbone) is called a *piece* (orange in Figure 9). E.g., in Figure 9,  $m_{14}$  is part of two pieces. Let  $\mathcal{G}_2$  to be the constrained level graph based on  $\mathcal{G}_1$  and our choice of  $s$  and  $t$ , with the additional constraint  $u \prec_i v$  for every pair  $(u, v)$  of vertices such that  $i = \gamma(u) = \gamma(v)$  and  $u$  comes before  $v$  on a simple  $s-t$  path. (This implies that  $s$  is the first and  $t$  is the last middle-level vertex of the backbone.) We show that there is a pair  $(s, t)$  such that  $\mathcal{G}_2$  is constrained level planar.

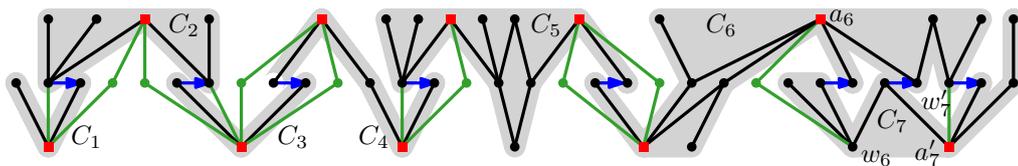
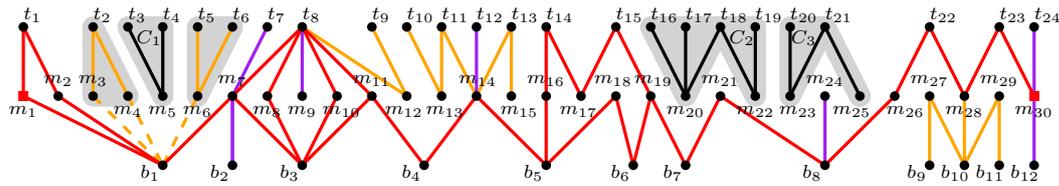


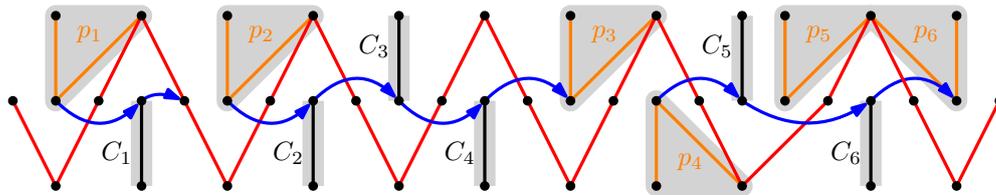
Figure 8 Constructing the main component: original components are colored in gray, hooking constraints in blue, and component-connecting edges in green.



■ **Figure 9** Drawing of the main component of a level graph with special vertices  $s = m_1$  and  $t = m_{30}$ : backbone edges are colored in red, enclosed components in gray, pieces in orange (edges connecting fingers to their anchors dashed) and edges incident to leaves in purple.

Let a *finger* be a piece that lies on the upper (lower) band whereas its anchor lies on the bottom (top) level; see the orange subgraphs with gray shading in Figure 9. Let  $\mathcal{G}_3$  be a copy of  $\mathcal{G}_2$  where we remove all edges (dashed-orange in Figure 9) that connect fingers to their anchors and add constraints that keep each finger in the interval between the surrounding middle-level vertices on the backbone. We show that  $\mathcal{G}_3$  is constrained level planar. Note that the remainder of each finger is an enclosed component.

Then we set up a 2-SAT formula  $\Phi$  to decide whether to place the remaining *flat* pieces (orange-gray in Figure 10) to the left or to the right of their anchor. This depends on chains of enclosed components that alternate between upper and lower band, are separated by the

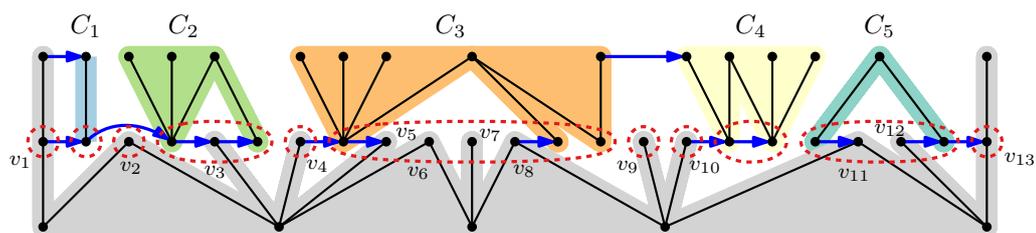


■ **Figure 10** We set up a 2-SAT formula to restrict the positions of the flat pieces (orange-gray) with respect to their anchors; e.g.,  $p_1$  must be left,  $p_5$  and  $p_6$  must be on different sides,  $p_2$  cannot be right while  $p_3$  is left,  $p_4$  and  $p_6$  force each other to the left and right respectively. (The main component is red, the black components are enclosed, and the blue arrows are constraints.)

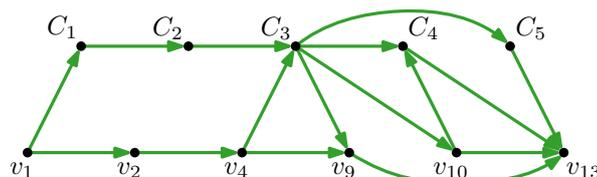
backbone, and connected by constraints. The chains are indicated by the blue arrows in Figure 10. If  $\Phi$  is satisfiable, we add constraints to fix the flat pieces to the respective sides of their anchors, which at the same time fixes their orientation. Let  $\mathcal{G}_4$  be the result. Then  $\mathcal{G}_4$  is constrained level planar. Next we restrict and then place the enclosed components.

A *separator vertex* is a backbone vertex on the middle level that has a neighbor on the top level and a neighbor on the bottom level. Let a *gap* be a pair  $(a, c)$  of separator vertices such that  $a \prec_2 c$  and there is no separator vertex  $b$  with  $a \prec_2 b \prec_2 c$ . An *upper (lower) gap* can accommodate enclosed components only from the upper (lower) band. E.g., the gap  $(v_1, v_{13})$  in Figure 11a is upper. Now we go through the enclosed components in topological order (of a kind of component–constraint graph) and fix each enclosed component from the upper (lower) band into an upper (lower) gap as far left in  $\mathcal{G}_4$  as possible, respecting the constraints from components placed before. Then the result  $\mathcal{G}_5$  is constrained level planar.

Finally, for each gap  $g$ , we set up a directed graph  $H_g$  that has a node (dashed in Figure 11) for each enclosed component and for some of the middle-level vertices inside the gap (those not “covered” by an enclosed component). The arcs of  $H_g$  correspond to constraints. Topologically sorting  $H_g$  yields a total order of the enclosed components and determines their orientation (if it is not arbitrary). Let  $\mathcal{G}_6$  be the result of adding the constraints that fix these placements. Then  $\mathcal{G}_6$  is constrained level planar. The remaining



(a) Constrained level planar drawing of  $g$  with middle-level vertices  $v_1, \dots, v_{13}$  and its enclosed components. Vertices of  $H_g$  are marked by (red) dashed ellipses. Component  $C_2$  covers  $v_3$ , whereas  $v_4$  is not covered.



(b) The directed graph  $H_g$  has a node for every enclosed component and non-covered middle-level vertex.

■ **Figure 11** Arranging the enclosed components  $C_1, \dots, C_5$  in a gap  $g$  of the main component. Constraints between vertices of  $G_5$  are indicated by blue arrows and arcs between nodes of  $H_g$  are indicated by green arrows. For clarity, implicit constraints, constraints along the backbone, and transitive arcs are omitted.

missing decisions involve leaves and pairs of middle-level vertices on the backbone that do not lie on the same  $s$ - $t$  path. By topologically sorting the vertices on the three levels independently, we get a constrained level planar drawing of  $G_6$ . Finally, we insert the isolated vertices and the edges incident to the fingers, and we remove the dummy edges between the hooked components. This yields a constrained level planar drawing of  $G$ .

## 5 Open Problems

1. We have shown that 3-level CLP can be solved in polynomial time, without optimizing the runtime of our algorithm. Can 3-level CLP be solved in, say, quadratic time?
2. In the problem variant CONNECTED CLP, we insist that the given constrained level graph is connected. It is easy to make the graph in our NP-hardness proof for 4-level CLP connected by adding two new levels, so 6-level CONNECTED CLP is NP-hard. What is the complexity of 5-level CONNECTED CLP?

---

## References

- 1 Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, and Vincenzo Roselli. The importance of being proper: (in clustered-level planarity and  $T$ -level planarity). *Theor. Comput. Sci.*, 571:1–9, 2015. Conference version in Proc. GD 2014 (doi:10.1007/978-3-662-45803-7\_21). doi:10.1016/j.tcs.2014.12.019.
- 2 Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Ignaz Rutter. Beyond level planarity: Cyclic, torus, and simultaneous level planarity. *Theor. Comput. Sci.*, 804:161–170, 2020. Conference version in Proc. GD 2016 (doi:10.1007/978-3-319-50106-2\_37). doi:10.1016/J.TCS.2019.11.024.
- 3 Christian Bachmaier, Franz-Josef Brandenburg, and Michael Forster. Radial level planarity testing and embedding in linear time. *J. Graph Algorithms Appl.*, 9(1):53–97, 2005. doi:10.7155/jgaa.00100.

- 4 Christian Bachmaier and Wolfgang Brunner. Linear time planarity testing and embedding of strongly connected cyclic level graphs. In Dan Halperin and Kurt Mehlhorn, editors, *Proc. 16th Ann. European Sympos. Algorithms (ESA)*, volume 5193 of *LNCS*, pages 136–147. Springer, 2008. doi:10.1007/978-3-540-87744-8\_12.
- 5 Vacláv Blažej, Boris Klemz, Felix Klesen, Marie Diana Sieper, Alexander Wolff, and Johannes Zink. Constrained and ordered level planarity parameterized by the number of levels. Arxiv report, 2024. arXiv:2403.13702.
- 6 Hans L. Bodlaender, Carla Groenland, Hugo Jacob, Lars Jaffke, and Paloma T. Lima. XNLP-completeness for parameterized problems on graphs with a linear structure. In Holger Dell and Jesper Nederlof, editors, *Proc. 17th Int. Symp. Param. Exact Comput. (IPEC)*, volume 249 of *LIPICs*, pages 8:1–8:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.IPEC.2022.8.
- 7 Hans L. Bodlaender, Carla Groenland, Jesper Nederlof, and Céline M. F. Swennenhuis. Parameterized problems complete for nondeterministic FPT time and logarithmic space. In *Proc. 62nd Ann. IEEE Symp. Foundat. Comput. Sci. (FOCS)*, pages 193–204, 2022. doi:10.1109/FOCS52979.2021.00027.
- 8 Guido Brückner and Ignaz Rutter. Partial and constrained level planarity. In Philip N. Klein, editor, *Proc. 28th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA)*, pages 2000–2011. SIAM, 2017. doi:10.1137/1.9781611974782.130.
- 9 Guido Brückner and Ignaz Rutter. An SPQR-tree-like embedding representation for level planarity. In Yixin Cao, Siu-Wing Cheng, and Minming Li, editors, *Proc. 31st Int. Symp. Algorithms Comput. (ISAAC)*, volume 181 of *LIPICs*, pages 8:1–8:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICs.ISAAC.2020.8.
- 10 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshantov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, Cham, 2015. doi:10.1007/978-3-319-21275-3.
- 11 Giuseppe Di Battista and Enrico Nardelli. Hierarchies and planarity theory. *IEEE Trans. Systems, Man, and Cybernetics*, 18(6):1035–1046, 1988. doi:10.1109/21.23105.
- 12 Vida Dujmović, Michael R. Fellows, Matthew Kitching, Giuseppe Liotta, Catherine McCartin, Naomi Nishimura, Prabhakar Ragde, Frances Rosamond, Sue Whitesides, and David R. Wood. On the parameterized complexity of layered graph drawing. *Algorithmica*, 52:267–292, 2008. doi:10.1007/s00453-007-9151-1.
- 13 Michael Elberfeld, Christoph Stockhusen, and Till Tantau. On the space and circuit complexity of parameterized problems: Classes and completeness. *Algorithmica*, 71(3):661–701, 2015. doi:10.1007/s00453-014-9944-y.
- 14 Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.*, 410(1):53–61, 2009. doi:10.1016/j.tcs.2008.09.065.
- 15 Michael Forster and Christian Bachmaier. Clustered level planarity. In Peter van Emde Boas, Jaroslav Pokorný, Mária Bielíková, and Julius Stuller, editors, *Proc. 30th Conf. Curr. Trends Theory & Practice Comput. Sci. (SOFSEM)*, volume 2932 of *LNCS*, pages 218–228. Springer, 2004. doi:10.1007/978-3-540-24618-3\_18.
- 16 Radoslav Fulek, Michael J. Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. Hanani–Tutte, monotone drawings, and level-planarity. In *Thirty Essays on Geometric Graph Theory*, pages 263–287. Springer, 2013. doi:10.1007/978-1-4614-0110-0\_14.
- 17 Ashim Garg and Roberto Tamassia. On the computational complexity of upward and rectilinear planarity testing. *SIAM J. Comput.*, 31(2):601–625, 2001. Conference version in Proc. GD 1994 (doi:10.1007/3-540-58950-3\_384). doi:10.1137/S0097539794277123.
- 18 Lenwood S. Heath and Sriram V. Pemmaraju. Recognizing leveled-planar dags in linear time. In Franz-Josef Brandenburg, editor, *Proc. Int. Symp. Graph Drawing (GD)*, volume 1027 of *LNCS*, pages 300–311. Springer, 1995. doi:10.1007/BFb0021813.

- 19 Seok-Hee Hong and Hiroshi Nagamochi. Convex drawings of hierarchical planar graphs and clustered planar graphs. *J. Discrete Algorithms*, 8(3):282–295, 2010. doi:10.1016/j.jda.2009.05.003.
- 20 Michael Jünger, Sebastian Leipert, and Petra Mutzel. Pitfalls of using PQ-trees in automatic graph drawing. In Giuseppe Di Battista, editor, *Proc. 5th Int. Symp. Graph Drawing (GD)*, volume 1353 of *LNCS*, pages 193–204. Springer, 1997. doi:10.1007/3-540-63938-1\_62.
- 21 Michael Jünger, Sebastian Leipert, and Petra Mutzel. Level planarity testing in linear time. In Sue Whitesides, editor, *Proc. 6th Int. Symp. Graph Drawing (GD)*, volume 1547 of *LNCS*, pages 224–237. Springer, 1998. doi:10.1007/3-540-37623-2\_17.
- 22 Boris Klemz. Convex drawings of hierarchical graphs in linear time, with applications to planar graph morphing. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *Proc. 29th Ann. Europ. Symp. Algorithms (ESA)*, volume 204 of *LIPICs*, pages 57:1–57:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ESA.2021.57.
- 23 Boris Klemz and Günter Rote. Ordered level planarity and its relationship to geodesic planarity, bi-monotonicity, and variations of level planarity. *ACM Trans. Algorithms*, 15(4):53:1–53:25, 2019. Conference version in Proc. GD 2017 (doi:10.1007/978-3-319-73915-1\_34). doi:10.1145/3359587.
- 24 Krzysztof Pietrzak. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *J. Comput. Syst. Sci.*, 67(4):757–771, 2003. doi:10.1016/S0022-0000(03)00078-3.
- 25 Ignaz Rutter. Personal communication, 2022.
- 26 Andreas Wotzlaw, Ewald Speckenmeyer, and Stefan Porschen. Generalized  $k$ -ary tanglegrams on level graphs: A satisfiability-based approach and its evaluation. *Discrete Appl. Math.*, 160(16-17):2349–2363, 2012. doi:10.1016/j.dam.2012.05.028.