Tight Parameterized (In)tractability of Layered Crossing Minimization: Subexponential Algorithms and Kernelization*

Fedor V. Fomin[†] Petr A. Golovach[†] Tanmay Inamdar[‡] Saket Saurabh[†§]
Meirav Zehavi[¶]

Abstract

The starting point of our work is the decade-old open question concerning the subexponential parameterized complexity of the 2-Layer Crossing Minimization problem. In this problem, the input is an n-vertex graph G whose vertices are divided into two independent sets V_1, V_2, and a non-negative integer k. The question is whether G supports a 2-layered drawing with at most k crossings. Here, a 2-layered drawing refers to a drawing of G where each set V_i for $i \in \{1, 2\}$ is placed on a distinct straight line parallel to the x-axis, and all edges are drawn as straight lines connecting vertices. Our first theorem resolves the aforementioned question in the affirmative by providing a fixed-parameter tractable (FPT) subexponential algorithm with running time $2^{\mathcal{O}(\sqrt{k}\log k)} + n \cdot k^{\mathcal{O}(1)}$.

The existence of a subexponential fixed-parameter algorithm for two layers immediately raises the question of whether this phenomenon is specific to two layers or can be extended to more layers. (In this setting, vertices are divided into h independent sets $V_1, \ldots, V_h$, and the question is whether G admits an h-layered drawing with at most k crossings.) Here, we delve into highly technical depths of the topic of layered drawings to answer this question almost completely, by providing a subexponential fixed-parameter algorithm for three layers with running time $2^{\mathcal{O}(k^{0.67})} + n \cdot k^{\mathcal{O}(1)}$, and proving that there does not exist a $2^{\mathcal{O}(k^{1-\epsilon})} \cdot n^{\mathcal{O}(1)}$-time algorithm (for any fixed $\epsilon > 0$) for five or more layers, under the Exponential-Time Hypothesis.

Next to the question of subexponential-time algorithms, lies the question of the existence of polynomial kernels for h-layered crossing minimization. We completely resolve this question as well – while a polynomial kernel was already known for $h = 2$, we derive a new polynomial kernel for $h = 3$. Complementarily, we rule out the existence of a polynomial kernel for any $h \geq 4$, assuming that the polynomial hierarchy does not collapse. Thus, we establish a complete dichotomy regarding polynomial kernelization based on the number of layers h.

Keywords: h-layer drawing, crossing minimization, subexponential algorithms, kernelization

---

*This research was initiated during Dagstuhl Seminar 23162 "New Frontiers of Parameterized Complexity in Graph Drawing".

Fedor V. Fomin acknowledges support from the Research Council of Norway via the BWCA (grant no. 314528) project.

Petr A. Golovach acknowledges support from the Research Council of Norway via the BWCA (grant no. 314528) and Extreme-Algorithms (grant no. 355137) projects.

Tanmay Inamdar acknowledges support from Anusandhan National Research Foundation (ANRF), grant number ANRF/ECRG/2024/004144/PMS, and IIT Jodhpur Research Initiation Grant, grant number I/RIG/TNI/20240072.

Saket Saurabh acknowledges support from European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819416).

Meirav Zehavi acknowledges support from the Israel Science Foundation, grant number 1470/24, and from the European Research Council, grant number 101039913 (PARAPATH).

[†]Department of Informatics, University of Bergen, Norway.

[‡]Department of Computer Science and Engineering, IIT Jodhpur, India

[§]The Institute of Mathematical Sciences, HBNI, Chennai, India

[¶]Ben-Gurion University of the Negev, Israel

Contents

1 Introduction

A popular paradigm in Graph Drawing is that of layered graph drawings [6, 51, 7, 18, 37]. In a layered graph drawing, the graph's vertices are allocated to parallel lines (layers), and the edges are drawn as lines between layers, aiming to minimize the number of crossings. Drawing graphs in layers is a fundamental technique for representing hierarchical graphs. It is integrated into standard graph layout software such as yFiles, Graphviz, OGDF, or TiKz [38, 53]. We recommend [52, Chapter 13] for a comprehensive overview. We remark that in general, during the recent years, there is a significant interest in crossing minimization for various types of drawings [29, 26, 46].

A popular model of a layered graph drawing is known as the h-Layer Crossing Minimization problem. In this problem, for $h \geq 2$, the input is an h-partite n-vertex graph G with h-partition $(V_1, \ldots, V_h)$ such that every edge has its endpoints in sets with consecutive indices, and a non-negative integer $k \in \mathbb{N}_0$. The task is to decide whether G admits an *h-layered drawing* with at most k crossings. Here, an h-layered drawing is a plane drawing such that for every $i \in \{1, 2, \ldots, h\}$, all vertices in V_i belong to a single straight line parallel to the x-axis, and all edges are drawn as straight line segments between consecutive layers (see Figure 1). The h-Layer Crossing Minimization problem remains NP-hard even when $k = 0$ [31]. Furthermore, the question of whether a graph G can be drawn in two layers with at most k crossings, where k is part of the input, is NP-complete [27]. Thus, h-Layer Crossing Minimization is para-NP-hard when parameterized only by k or h (that is, we do not expect an algorithm with running time $f(h) \cdot n^{\mathcal{O}(1)}$ or $g(k) \cdot n^{\mathcal{O}(1)}$, for any function f or g, unless P=NP). Dujmović et al. [14] proved that the problem is FPT parameterized by $k + h$. While the running time of the algorithm of Dujmović et al. is linear in the number of vertices n, its dependence in the parameter is $(k + h)^{\mathcal{O}((k+h)^3)}$.

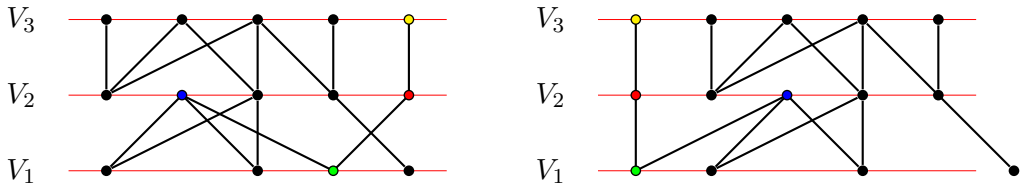

Figure 1: Example of 3-layered drawings of the same graph with five and two crossings.

In this work, we aim to fully delineate the tractability boundaries of h-Layer Crossing Minimization for all values of h, focusing on two fundamental questions in parameterized complexity: subexponential solvability and polynomial kernelization. We present nearly tight positive and negative results regarding subexponential running times in terms of the number of layers, with the case of four layers being the only remaining open case. For polynomial kernelization, we achieve a complete dichotomy: we provide the first polynomial kernel for three layers and, for all $h \geq 4$, we rule out the existence of polynomial kernels unless the polynomial hierarchy collapses. A summary of our results is provided in Table 1. Interestingly, in the context of kernelization complexity, the tractability boundary appears to be a bit surprising – typically the transition from tractability to intractability of problems happens from "2 to 3" (as seen in problems like colorability, matching, SAT); in this case, the transition may be better explained by considering the number of edge layers rather than vertex layers.

The simplest version of h-Layer Crossing Minimization is called One-Sided Crossing Minimization. In this problem, the ordering (embedding) of vertices of V_1 on the line is fixed, and the task is to draw the vertices of V_2 on the line, minimizing the number of crossings. The last year's PACE challenge (PACE 2024) was entirely devoted to this problem.[1] The One-Sided Crossing Minimization problem is known to admit a polynomial kernel with $\mathcal{O}(k^2)$ vertices [15] and is FPT

[1] https://pacechallenge.org/2024/

Number of layers h	Subexponential-time	Kernelization
$h = 2$	$2^{\mathcal{O}(\sqrt{k}\log k)} + n \cdot k^{\mathcal{O}(1)}$ (Theorem 1)	$\mathcal{O}(k^2)$ [43]
$h = 3$	$2^{\mathcal{O}(k^{0.67})} + n \cdot k^{\mathcal{O}(1)}$ (Theorem 2)	$\mathcal{O}(k^8)$ (Theorem 4)
$h = 4$	Open	No poly kernel (Theorem 5)
$h \geq 5$	No $2^{o(k/\log k)} \cdot n^{\mathcal{O}(1)}$ (Theorem 3)	No poly kernel (Theorem 5)

Table 1: Our results

parameterized by k [15, 16, 14]. Moreover, several subexponential-time parameterized algorithms were introduced for this problem [21, 44]. The best running time for this problem is $\mathcal{O}(k2^{\sqrt{2k}} + n)$ [44] and up to the ETH, the problem cannot be solved in time $2^{o(\sqrt{k})} \cdot n^{\mathcal{O}(1)}$. Solving One-Sided Crossing Minimization could be reduced to solving parameterized Feedback Arc Set in Tournament (FAST). For FAST, several subexponential algorithms are known [1, 19, 23, 39]. However, we do not see any reduction from h-Layer Crossing Minimization to FAST or any other known subexponential-FPT problem.

The case $h = 2$ has its own story. 2-Layer Crossing Minimization has a long history, dating back to the 1970s. The problem was introduced by Harary and Schwenk [30] as the bipartite crossing number. Finding a drawing on two lines minimizing the number of crossings is also the key procedure in the Sugiyama algorithm, the well-known layout framework for drawing graphs on layers [51]. The problem is long known to be NP-complete [27]. Kobayashi et al. [43] gave an algorithm of running time $2^{\mathcal{O}(k\log k)} + n^{\mathcal{O}(1)}$. A faster algorithm of running time $2^{\mathcal{O}(k)} + n^{\mathcal{O}(1)}$ was obtained by Kobayashi and Tamaki in [45]. Kobayashi et al. [43] also gave a polynomial kernel with $\mathcal{O}(k)$ edges. The starting point of our work on h-Layer Crossing Minimization is the question initially posed by Kobayashi, Maruta, Nakae, and Tamaki [43]. This question was later reiterated by Kobayashi and Tamaki [45] and Zehavi [54]. They ask whether 2-Layer Crossing Minimization admits a subexponential parameterized algorithm, specifically, an algorithm with a running time of $2^{o(k)} \cdot n^{\mathcal{O}(1)}$. Our first result answers this long-standing open question affirmatively.

Theorem 1. *On an n-vertex graph,* 2-Layer Crossing Minimization *is solvable in time* $2^{\mathcal{O}(\sqrt{k}\log k)} + n \cdot k^{\mathcal{O}(1)}$.

In Section 4, we give a short, elegant, and self-contained proof of this result, which already resolves the open question in the literature as mentioned earlier. However, Theorem 1 naturally raises the question of whether a subexponential $2^{\mathcal{O}(k^{1-\epsilon})} \cdot n^{\mathcal{O}(1)}$-time algorithm can be extended to drawings with more layers. Our next two theorems address this question almost completely. They show that for $h = 3$, the answer is yes, while for any $h \geq 5$, the answer is no (assuming ETH, the Exponential Time Hypothesis of Impagliazzo, Paturi, and Zane [33, 34]). The status of the problem for $h = 4$ remains open. More precisely, we prove the following theorems.

Theorem 2. *On an n-vertex graph,* 3-Layer Crossing Minimization *is solvable in time* $2^{\mathcal{O}(k^{0.67})} + n \cdot k^{\mathcal{O}(1)}$.

Theorem 3. *For any $h \geq 5$, h-*Layer Crossing Minimization *cannot be solved by an algorithm running in* $2^{o(k/\log k)} \cdot n^{\mathcal{O}(1)}$ *time unless ETH fails.*

Essential components of the proofs of Theorems 1 and 2 are linear time algorithms constructing kernels of polynomial sizes. Informally, these are algorithms constructing for an n-vertex h-partite graph G and integer k, an equivalent instance on $k^{\mathcal{O}(1)}$ vertices within $\mathcal{O}(n)$ time. For 2-Layer

Crossing Minimization, such kernelization algorithm was given by Kobayashi et al. [43]. However, for 3-Layer Crossing Minimization, no polynomial kernel was previously known, making the following theorem of independent interest.

Theorem 4. 3-Layer Crossing Minimization *admits a kernel with* $\mathcal{O}(k^8)$ *vertices and edges. Furthermore, the kernelization algorithm can be implemented to run in* $k^{\mathcal{O}(1)} \cdot n$ *time.*

Again, Theorem 4 prompts the question of whether a polynomial kernel can also exist for drawings with more than three layers. The following theorem provides a negative answer to this question.

Theorem 5. *For any* $h \geq 4$, h-Layer Crossing Minimization *does not admit a polynomial kernel unless* NP $\subseteq$ coNP /poly.

Theorems 4 and 5 establish a dichotomy for polynomial kernelization of h-Layer Crossing Minimization based on the number of layers h. We summarize the results of this paper in Table 1. We give a brief summary of our methods in the following subsection, and a detailed technical overview in Section 2.

1.1 Our Methods

At the core of our subexponential-time fixed-parameter algorithms is the divide-and-conquer method. Here, a naive attempt is to base the approach of divide-and-conquer on standard path (or tree) decompositions—indeed, a graph admitting a drawing with k crossings can be proved to admit a path decomposition of width $\mathcal{O}(\sqrt{k})$. However, while we can "guess" some *separator* that is a bag in such a decomposition, it is completely unclear how the *separation* itself can be concluded from this separator in subexponential time. Indeed, we believe that this is the main reason why the problem of the existence of the subexponential-time fixed-parameter algorithms has remained opened for so long.

All the previous approaches for crossing minimization (for general drawings) bound the treewidth of the graph by using the irrelevant vertex technique, and then use dynamic programming or Courcelle's theorem on the tree decomposition [40]. We completely deviate from this approach. While we still use separators, they satisfy additional properties not required by tree decomposition. Indeed, we enrich the standard definition of a separator by including novel additional properties based on the drawing. Intuitively, this separator consists of four edges (or two in the case of two layers) that are crossed few times, and the edges that cross them. We also demand the separator to be "balanced" in some sense (so that the recursion will be efficient), but this, on its own, is far from sufficient. We must enforce the separator, in order for it to be useful, to satisfy additional drawing-based properties so that, from the separator, we can deduce the separation. In the literature, there are a few examples where the existence of tree decompositions or just separators with certain properties is derived from the drawing, e.g., [41, 2]. However, typically, such approaches then completely work on the tree decomposition itself, completely ignoring the extra properties. However, as mentioned above, here the extra properties of the separator are critical for us. In the proof, for the sake of simplicity, we do not explicitly describe the tree structure.

The deduction of the separation itself, particularly in the three-layered case, is a complicated process that requires: *(i)* multiple additional guesses, *(ii)* a propagation process, where, having guessed some vertices in one part of the separation, we propagate the information to derive additional vertices in that part, and *(iii)* handling various components "dangling" from the separator vertices that are not captured by the guesses and the propagation process. Step *(iii)* is particularly challenging for the three-layered case, where we sometimes need to redraw the (unknown) optimal solution our separator is based on, based on novel arguments classifying the aforementioned components into types, and making use of knapsack-based arguments so as ensure that the separator remains a (balanced) separator.

The main idea of the kernelization algorithm (which is a critical component in our subexponential algorithms) is that if (G, V_1, V_2, V_3, k) is a yes-instance of 3-Layer Crossing Minimization and G is sufficiently big with respect to k, then G should have large pieces that have to be drawn without crossings. Moreover, due to the rigidity of drawings without crossing, the drawing of these pieces is essentially unique (up to the reversals of orders and reordering of twins). Thus, our algorithm applies reduction rules that *(i)* find the parts that are (almost) uniquely embeddable without crossings, and *(ii)* reduce their size.

The starting point of our lower bounds is, in fact, the arguments used for our subexponential and kernelization algorithms. Specifically, the cases where these arguments could not be applied revealed the "hard instances" of the problem. These hard instances are encoded using gadgets, and their combination (which yields our hardness reductions) is quite technical on its own.

We believe that the idea of *drawing based separators* is our main conceptual contribution.

1.2 Other Related Work

h-Layer Crossing Minimization belongs to a broader category of graph-level drawing problems, which involve assigning vertices of a (di)graph to lines on a plane according to specific criteria. Historically, level drawings were some of the first styles explored systematically, following the introduction of the Sugiyama framework [51]. These drawings are commonly used to visualize hierarchical data, and extensive research covers every aspect of the process, see e.g., [52, Chapter 13] and [25].

Besides heuristic and parameterized algorithms, the approximation of 2-Layer Crossing Minimization has been studied for several classes of bipartite graphs, as seen in, for example, [50]. However, obtaining non-trivial approximation algorithms with guaranteed performance on (general) bipartite graphs remains challenging.

Closely related to h-Layer Crossing Minimization, is the concept of Level Planarity. In level planarity, each vertex of the graph is assigned to a specific level (one of the parallel lines in the plane) based on predefined criteria, and edges (represented by curves) should not cross internally. A significant amount of work is devoted to efficient algorithms recognizing level planarity and some of its generalizations [11, 35, 5, 42].

Another close relative of h-Layer Crossing Minimization is the Crossing Number problem, which involves drawing a graph in a plane while minimizing the number of crossings. Crossing Number, parameterized by k, was shown to be in FPT by Grohe [28]. Intensive research has focused on approximation algorithms for this problem, leading to the work of Chuzhoy and Tan [8] who developed a randomized algorithm with approximation factor $2^{\mathcal{O}(\log^{\frac{7}{8}} n \log\log n)}\Delta^{\mathcal{O}(1)}$. For further information on Crossing Number and its variants, we recommend exploring the extensive literature starting with [49, 48, 47].

One more problem related to h-Layer Crossing Minimization is called h-Layer Planarization. Here, for a graph G and a non-negative integer $k \in \mathbb{N}_0$, the task is to decide whether there exists a subset of at most k edges that can be removed from G so that it will admit an h-layered drawing with no crossings. In particular, this problem and variants of it have received attention from the perspective of parameterized complexity [14, 13, 20].

2 Technical Overview

2.1 Overview of the Subexponential Algorithm for $h = 2$

We begin with the description of our subexponential fixed-parameter algorithm for two layers, because the one for three layers is based on all of its arguments as well as additional and much more technical ones. In fact, our algorithm for three layers, for some cases, uses the one for two

layers directly as a subroutine. We suppose that the input graph is connected, else each component can be solved independently.

The basic approach that guides the design of our algorithm is that of divide-and-conquer. Specifically, we have a recursive algorithm. Then, at each recursive call, we would like to split the current instance of our problem into two (or more) smaller instances, so that the "combination" of their solutions yields a solution to the current instance. Each of the smaller instances is solved by making one more recursive call. In particular, to obtain the desired subexponential dependency on the parameter in the time complexity, it is crucial to ensure that the parameter of each of the smaller instances is at most a linear fraction of the current parameter—that is, if the current parameter is k and the new parameters are k_1 and k_2, then we must have that $\max(k_1, k_2) \leq \alpha k$ for some fixed constant $\alpha < 1$. It is important to note that we will not produce just a single pair of smaller instances, but a large collection of such pairs, so that if we should find a solution to our current instance, then at least one of them will yield it. Since the number of these pairs will be upper-bounded by $2^{\mathcal{O}(\sqrt{k}\log k)}$, the recurrence for the time complexity will resolve to the one that we claim.

The Separator. Having the above approach in mind, the key player in the proof is that of a *separator* for some (unknown) solution (that is, a 2-layered drawing with at most k crossings). For us, the definition of the separator will be *geometric*, relying on the drawing (or, equivalently, ordering of vertices). Ideally, we would have wanted the separator to be an edge of G (where G is the graph of the current instance) that satisfies two properties: *(i)* the number of edges that cross it at most $c\sqrt{k} = \mathcal{O}(\sqrt{k})$ for some constant c; *(ii)* the numbers of crossings to "its left" and to "its right" are at most αk for a fixed $\alpha < 1$. Clearly, if all edges of G are crossed more than $c\sqrt{k}$ times, then such an "elegant" separator does not exist. However, in this case, we have only $\mathcal{O}(\sqrt{k})$ many vertices in total, and hence can just solve the instance directly by using brute-force. Unfortunately, even if this is not the case and, moreover, we have arbitrarily many edges that are not crossed at all, such an elegant separator might provably not exist—to see a simple situation where this is so, we consider precisely the case where all edges of G are crossed more than $c\sqrt{k}$ times, and then we extend G and the drawing by inserting an arbitrarily large drawing of a graph without any crossings to both the left and the right of the existing drawing (while we can keep the graph connected if so desired).

Thus, we work with a somewhat more complicated separator, consisting of two elements that are each either an edge or one of the two "boundaries of the drawing", lsep and rsep, so that the four following properties are satisfied (see Fig. 2):[2] *(i)* lsep lies to the left of rsep (though they may share an endpoint); *(ii)* the number of edges that cross any of them is upper-bounded by $\mathcal{O}(\sqrt{k})$; *(iii)* the numbers of crossings to the left of lsep and to the right of rsep are at most αk for some fixed $\alpha < 1$ (here, we pick $\alpha = \frac{1}{2}$); (iv) the number of edges that are drawn between lsep and rsep is upper-bounded by $\mathcal{O}(\sqrt{k})$.

Existence of the Separator. First, having some (unknown) solution in mind, it is important to observe that among any set of $\sqrt{k}$ many edges, there will exist at least one that is crossed at most $\sqrt{k}$ many times. Hence, the proof that a separator as defined above *exists* follows from the following logic. First, we consider the leftmost edge (or the right boundary), denoted rsep, that is crossed by at most $\mathcal{O}(\sqrt{k})$ many edges and such that there are at least αk many crossings to its left (for some choice of $0 < \alpha < 1$). Then, neglecting edges that cross rsep, we "keep going" to the left of rsep, until we encounter—for the first time—another edge (or the left boundary), denoted lsep, that is crossed at most $\mathcal{O}(\sqrt{k})$ times. As will not "pass over" more than $\mathcal{O}(\sqrt{k})$ edges while seeking lsep, we can show that this yields a separator as required.

Computation of the Separation. As we do not know a solution (but assume, for now, that one exists), the proof of the existence of a separator is non-algorithmic. Still, we can "guess"

[2]Figures that are relevant to the technical sections later will be repeated in the appropriate locations for convenience.

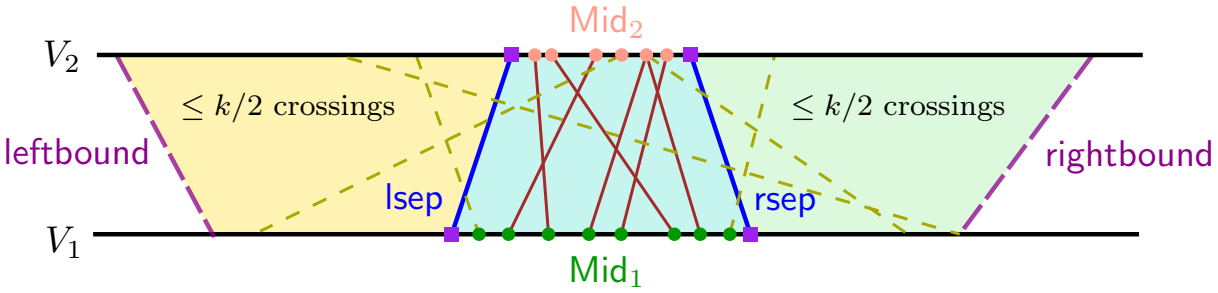

Figure 2: Illustration of a separator. The edges lsep and rsep are shown in blue, and their endpoints are shown in purple. It is possible that the endpoints on V_1 (or V_2) are the same. The edges between lsep and rsep are shown in brown, and the edges that cross lsep or rsep are shown in olive. Vertices between the endpoints of lsep and rsep are shown in dark green (denoted Mid$_1$ for V_1) or light pink (denoted Mid$_2$ for V_2).

all of the required information. For this to be efficient, before the recursive process begins, we call a known kernelization algorithm by Kobayashi et al. [43], so that the given instance can be supposed to have only polynomially in k many vertices and edges. Clearly, we can then guess the separator (consisting of two edges) using just $\binom{m}{2} = k^{\mathcal{O}(1)}$ many guesses, where m is the number of edges in the (already kernelized) graph. The challenge is in guessing the separation (partition of the vertex set of the graph) that the separator induces into left, middle, and right parts. While in the two-layered case, that is not overly technical, the same will not hold for the three-layered case (discussed in the next overview). There, several new ideas and much more technical work are required on top of those for the two-layered case.

First, notice that the middle part of of the separation can be guessed easily, using only $k^{\mathcal{O}(\sqrt{k})}$ many guesses, as it contains (by the definition of a separator) only $\mathcal{O}(\sqrt{k})$ many edges and vertices. Additionally, we can similarly easily guess the edges that cross the two separator edges, and which of their endpoints belongs to which part of the partition. Here, we also guess the *ordering* of these endpoints in the (unknown) solution under consideration. This is necessary in order to count, already here, the crossings involving only the edges that cross the separator edges—they cannot be considered when inspecting each part of the partition independently. In turn, this means that the problem that will correspond to each of the parts will be more general than 2-Layer Crossing Minimization as we will need the sought solutions to respect the guessed ordering. Another issue somewhat of this nature and concerning the crossing edges is that some of them still have to be encoded in the problem that will correspond to each of the parts since other edges of that part may or may not cross them, and we cannot know all information about this at this stage. We briefly remark that, to this end, we replace the crossings edges with some specific weighted (or, equivalently multi-)edges. For the simplicity of this overview, we will not continue to describe the way that these technical issues are handled, but proceed to the rest of the computation of the separation, which we view as more central for intuition.

Having guessed to where the endpoints of the crossing edges belong, we proceed to "propagate" this information using breadth-first search (BFS). Say, to be specific, that we have a vertex determined to belong to the left part. Then, for every edge incident to it that is not a crossing edge (w.r.t. the separator edges), we know that the other endpoint must also belong to the left part (assuming that our guess is "correct"). Afterwards, we can apply the same logic from these neighboring vertices to their own neighbours, and so on. Unfortunately, not all vertices of the graph (even though we suppose it to be connected) are marked left, middle, or right, by all the above guesses as well as the labelling propagation.

Specifically, consider the graph from which the endpoints of the separator edges are removed. Then, the unmarked vertices are precisely those that belong to the connected components whose only neighbours are the endpoints of the separator edges themselves, which do not cross the sepa-

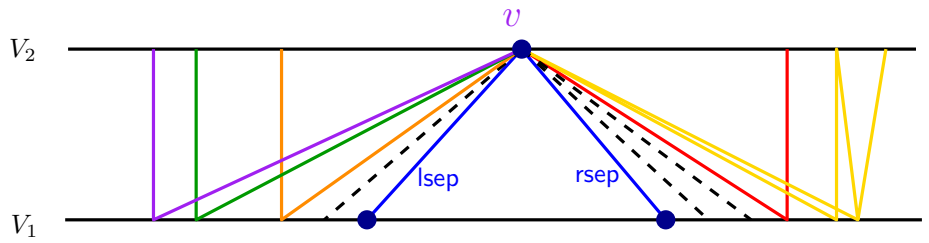

Figure 3: Illustration of the problematic components described in the overview. Here, the endpoint on V_2 of the two separator edges (colored blue) is shared. The edges of the three "leaf components" are denoted by dashed black lines. Each of the other five problematic components is denoted using a different color: four of them (colored purple, green, orange and red) have a single edge each, and the fifth (colored golden) has three edges—the edges connecting the component to the separator vertex on top are not part of the component, but are drawn using the same color for clarity.

rator edges, and which do not belong to the middle part. In case none of the four endpoints of the two separator edges coincides, these unmarked vertices can be directly marked as left or right, as we already know which are all of the middle vertices: so, e.g., if an unmarked vertex belongs to a component having a neighbor that is an endpoint of the left edge of the separator, then we know that it should be marked left. However, if, say, the two endpoints of the two separator edges on V_1 coincide, then for a component whose only neighbor is that vertex in V_1, we do not know yet whether it should belong to the left part or to the right part (see Fig. 3).

Fortunately, we can show that the number of problematic components as described above, and which have at least one edge, is upper-bounded by $\mathcal{O}(\sqrt{k})$. So, for these components, we can directly guess whether they go left or right. This leaves us only with the problematic components that consist just of a single vertex (which are, in the entire graph, simply pendants attached to one endpoint of a separator edge). Ideally, we could have just guessed how many these leaves (for each of at most four possible types) go left or right. However, due to some technicalities, we briefly remark that, here, our argument is slightly more technical as these leaves might be attached using weighted edges. For simplicity, we skip the description of this technicality. Overall, this completes the computation of a separation, which, in turn, completes the intuition guiding the design of our algorithm for two-layered drawings.

2.2 Overview of the Subexponential Algorithm for $h = 3$

Similarly to the algorithm for $h = 2$, we employ divide-and-conquer. Clearly, here, we will need to modify the definition of a separator to a more technical one. However, the main difficulty compared to the case of $h = 2$ will be in the computation of the separation itself, where after the process of label propagation, much more complicated components will remain unmarked, and handling them will require several new ideas.

The Separator. With respect to an (unknown) solution, the separator consists of four elements: two elements between V_1 and V_2 that are each either a "non-boundary" edge or one of the two "bottom boundaries of the drawing" (e.g., $\mathsf{leftbound}_{12}$ or $\mathsf{rightbound}_{23}$), denoted by lsep_{12} and rsep_{12}, and two elements between V_2 and V_3 that are each either an edge or one of the two "top boundaries of the drawing", denoted by lsep_{23} and rsep_{23}. We will describe the case where we "split the number of crossings" between V_1 and V_2, but, symmetrically, we can use the definition for the case where we "split the number of crossings" between V_2 and V_3. For simplicity, let us denote the number of crossings that lie between V_1 and V_2 by k, and suppose that the total number of crossings is at most $2k$. The four elements that compose a separator must satisfy the following properties (see Fig. 4):

1. lsep_{12} (lsep_{23}) lies to the left of rsep_{12} (rsep_{23}), though they may share an endpoint.

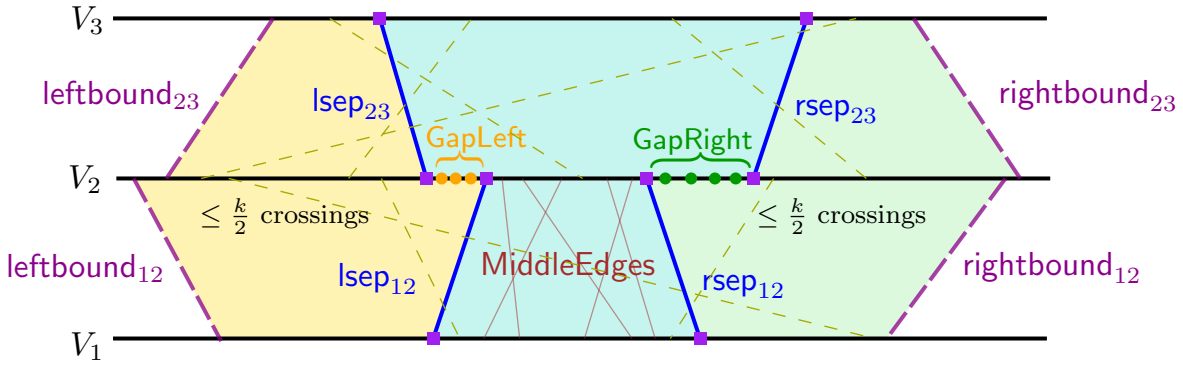

Figure 4: Illustration of a separator. The four edges of the separator are shown in blue, and their endpoints are shown in purple. It is possible that the some of the endpoints of the separator edges are the same. The edges between the left are right parts of the separator whose number is guaranteed to be upper-bounded (specifically, here, those between V_1 and V_2) are shown in brown, and the edges that cross the separator edges are shown in olive. Vertices on V_2 between the endpoints of the two right separator edges and which are incident to edges above them (these edges are not illustrated) are shown in dark green (denoted GapRight), and vertices on V_2 between the endpoints of the two left separator edges and which are incident to edges above them are shown in orange (denoted GapLeft).

2. The number of edges that cross any of the separator edges is upper-bounded by $\mathcal{O}(\sqrt{k})$.
3. The numbers of crossings to the left of lsep_{12} and to the right of rsep_{12} are at most αk for some fixed $\alpha < 1$ (here, we pick $\alpha = \frac{1}{2}$).
4. The number of edges drawn between lsep_{12} and rsep_{12} (and which are, therefore, between V_1 and V_2) is upper-bounded by $\mathcal{O}(\sqrt{k})$.
5. The number of vertices between the endpoints of lsep_{12} and lsep_{23} on V_2 and which have neighbors on V_3 is upper-bounded by $\mathcal{O}(\sqrt{k})$. Symmetrically, the number of vertices between the endpoints of rsep_{12} and rsep_{23} on V_2 and which have neighbors on V_3 is upper-bounded by $\mathcal{O}(\sqrt{k})$.

We remark that the vertices whose number is upper-bounded in the last condition do not include all vertices between the respective endpoints, as some such vertices might have neighbors only on V_1.

Existence of the Separator. Recall (from the two-layered case) that, having some (unknown) solution in mind, among any set of $\sqrt{k}$ many edges, there will exist at least one that is crossed at most $\sqrt{k}$ many times. So, similarly to the two-layered case, to prove the existence of a separator, we start by considering the leftmost edge (or the right boundary) between V_1 and V_2, denoted rsep_{12}, that is crossed by at most $\mathcal{O}(\sqrt{k})$ many edges and such that there are at least αk many crossings to its left (for some choice of $0 < \alpha < 1$). Then, neglecting edges that cross rsep_{12}, we "keep going" to the left of rsep_{12}, until we encounter—for the first time—another edge (or the left boundary), denoted lsep_{12}, that is crossed at most $\mathcal{O}(\sqrt{k})$ times.

To obtain the other two elements as required for the three-layered case, we proceed as follows. First, we consider the endpoint of lsep_{12} that belongs to V_2. Then, we start "going left" from that endpoint until we encounter a vertex incident to an edge between V_2 and V_3 that is crossed at most $\mathcal{O}(\sqrt{k})$ times. That edge is precisely lsep_{23}. In particular, while doing so, we will not "pass over" more than $\mathcal{O}(\sqrt{k})$ many vertices that are neighbors to vertices on V_3. Second, symmetrically, we consider the endpoint of rsep_{12} that belongs to V_2. Then, we start "going right" from that endpoint

until we encounter a vertex incident to an edge between V_2 and V_3 that is crossed at most $\mathcal{O}(\sqrt{k})$ times. That edge is precisely rsep_{23}.

Computation of the Separation: The Separator and the Propagated Vertices. As in the two-layered case, the proof of the existence of a separator is non-algorithmic. Still, we can "guess" all of the required information. Again, for efficiency, as a preprocessing step preceding the divide-and-conquer approach, we need to apply a kernelization algorithm. Prior to our work, such an algorithm (for the three-layered case) was not known, and so we had to design it ourselves. This is a critical component of our algorithm. Since it is a technical algorithm on its own right that is of independent interest, we describe it later in its own overview Section 2.3.

Now, having kernelized the instance, we can then directly guess the separator (consisting of four edges) using just $\binom{m}{4} = k^{\mathcal{O}(1)}$ many guesses, where m is the number of edges in the (already kernelized) graph. Similarly to the two-layered case, we guess all of the edges drawn between lsep_{12} and rsep_{12}. Additionally, similarly to the two-layered case, we guess all the edges that cross the four separator edges, and which of their endpoints belongs to which part of the partition. Here, for the same purposes discussed for the two-layered case, we also guess the *ordering* of these endpoints in the (unknown) solution under consideration. This completes the description of our "obvious" set of guesses.

Here, unlike the two-layered case, the obvious set of guesses does not immediately yield the entire middle part, as the middle part contains edges between lsep_{23} and rsep_{23} as well. Thus, we further guess the vertices in GapLeft and GapRight, and the ordering between them (so to enforce that our solutions to the left and middle instances, as well as to our right and middle instances, will be consistent). From here, it is somewhat easy to see that by employing a propagation process similar to the one described for the two-layered case—particularly since we already know all edges in the bottom of the middle part—we can derive the entire middle part. To be more precise, we will not derive those vertices on V_2 between the endpoints of the two left separator edges (as well as those between the endpoints of the two right separator edges) that only have neighbors on V_1, but these are "irrelevant" for the middle part anyway, and can be "inserted" when recursing into the left and right parts. In particular, the internal ordering between the aforementioned "derived" vertices and the "underived" vertices of the middle part in either side is immaterial to the middle part. Since these considerations are easy when compared to the substantially more complicated upcoming issues, we do not delve into further details here.

Still, before we proceed to analyze the left and right parts, one last important comment concerning the middle part is in place. The middle part might host a huge number of crossings on both its top and bottom, unlike the left and right parts, which can host only some αk crossings each on the bottom (by the definition of a separator). Thus, we cannot simply recursively solve the middle instance, as the parameter might not drop sufficiently to attain our desired time complexity overall. However, since the middle part has just $\mathcal{O}(\sqrt{k})$ many vertices on V_1 as well as only $\mathcal{O}(\sqrt{k})$ many edges between V_1 and V_2, we can guess some ordering concerning these $\mathcal{O}(\sqrt{k})$ many elements so as to reduce the middle instance to the two-layered case. Then, the middle instance is solved using our algorithm for the two-layered case instead of direct recursion.

Next, let us proceed with the analysis of the left and right parts. By employing a propagation process similar to the two-layered case, we can conclude which vertices belong to the left or right parts with the exception of the following. Consider the graph from which we remove the endpoints of the separator edges (and the edges incident to them). Now, further consider the connected components that do not have edges crossing the separator edges as well as do not have vertices belonging the guessed vertices on GapLeft and GapRight or to the middle part. Then, the vertices of these components are precisely those that we have not yet classified. Still, by using arguments similar to the two-layered case (in particular, to handle components having vertices on V_1 or not having vertices on V_3), we can further classify some of these vertices, so that eventually we are left with a particular situation (or a situation symmetric to it), described in the next paragraph as "the problematic situation".

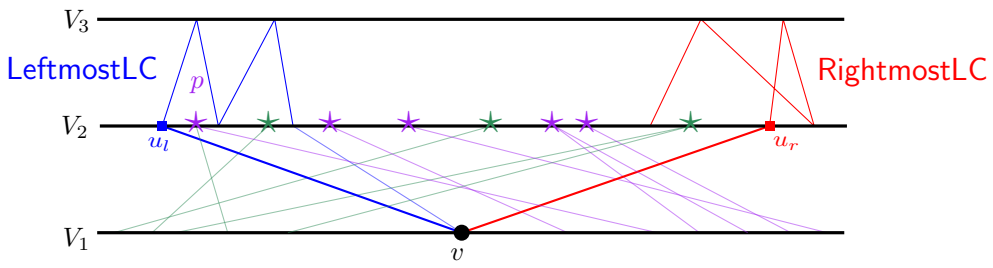

Figure 5: Illustration of components LeftmostLC, RightmostLC and star vertices. Left-star vertices (shown in green) have at least one incident edge crossing vu_l, and right-star (purple) have at least one incident edge crossing vu_r. Note that a vertex can be both left- and right-star vertex, e.g., p.

Computation of the Separation: Handling the Non-Propagated Vertices. The "problematic situation" occurs when lsep_{12} and rsep_{12} share the endpoint on V_1, denoted by v. Then, the components whose vertices are not classified as left or right by any of the arguments considered above (or other arguments along the same lines that we omit from this overview) are the following. Consider the graph from which we remove the endpoints of the separator edges. Now, further consider the connected components within it, such that the only neighbor from the endpoints of the separator edges that they have is v—they can be thought of as "pendant components hanging from v" (see, e.g., the red and blue components illustrated in Fig. 6). Among these, the unclassified components further satisfy the following properties:

- They do not contain any edge crossing the separator edges.
- They neither belong to the middle part, nor have vertices on GapLeft or GapRight (that are adjacent to vertices on V_3).
- They do not contain vertices on V_1 (recall that v cannot belong to the components themselves).
- They contain at least one vertex on V_3.

Now, we further guess the leftmost and rightmost components among the problematic ones (see Fig. 5), denoted respectively by LeftmostLC and RightmostLC, that do not contain an edge crossed more than $\sqrt{k}$ many times. Having these components at hand, we further guess (using just $k^{\mathcal{O}(\sqrt{k})}$ many guesses) which of the problematic components: *(i)* crosses at least one among LeftmostLC or RightmostLC; *(ii)* lies to the left of LeftmostLC, or *(iii)* lies to the right of RightmostLC. We classify these components as left or right, and henceforth, exclude them from the set of problematic components under consideration. Furthermore, from now on, when we consider any element (e.g., vertex or edge), we suppose it to lie between LeftmostLC or RightmostLC; intuitively, think about the area between LeftmostLC or RightmostLC as our "workspace" until the end of this overview.

To handle the (remaining) problematic components, we define a notion of so-called *star vertices* (see Fig. 5). Roughly speaking, a star-vertex is a vertex on V_2 that is incident to an edge crossing a bottom edge of LeftmostLC (then it is a left-star), or a bottom edge of RightmostLC (then it is a right-star), or both. A crucial observation in this context is that all bottom edges (of the problematic components) whose endpoints on V_2 lie between the same star vertices are crossed the same number of times, and, in fact, by exactly the same edges. Further, we note that there can exist at most $\mathcal{O}(\sqrt{k})$ star-vertices.

We first handle the problematic components that "surround" at least one star-vertex. We are able to bound their number by $\mathcal{O}(k^{\frac{2}{3}})$ based on two observations. First, all problematic components surrounding the same star-vertex cross each other (see Fig. 6). Second, as we "go" further left starting from the rightmost left-star and towards the leftmost left-star (and, symmetrically, w.r.t. right-stars), the crossings of the components surrounding them (or even just lying in-between them) with the edges incident with these star-vertices that cross the bottom left boundary "accumulate". Having bounded the number of these components from above, we simply guess whether

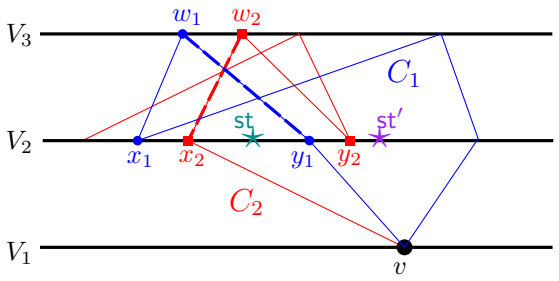

Figure 6: Problematic components C_1 (blue) and C_2 (red) surround the same star vertex st (C_1 happens to surround st′ as well). In particular, their edges y_1w_1 and x_2w_2 cross.

they should be marked left or right.

Finally, we are left with the most difficult problematic components: each of them lies entirely strictly in-between some two consecutive star-vertices (where different such components can lie between different consecutive pairs of star-vertices). We classify these components into $\mathcal{O}(k^{2/3})$ "types", where each type, essentially, encodes *(i)* the number of edges between v and the component, and *(ii)* the number of edges in the component. We briefly remark that we do not consider all possible such types—for example, the components with "large" number of either of edges in *(i)* or *(ii)* above are bundled together into a single type (the number of such components can be directly upper-bounded). For the types that we do consider, we are able to prove that the (unknown) solution itself can be altered so that components of the same type can be redrawn in any order one next to another between the same two star-vertices.

Unfortunately, when redrawing the problematic components as described above, a number of technicalities arise. Most importantly, we need to make sure that our separator remains a *balanced* separator for the new drawing—in particular, in this context, we need to ensure that both the left and right parts still each have only some αk many crossings for a fixed $\alpha < 1$. To handle this, while performing the "redrawing" mentioned in the previous paragraph, for each type, we need to draw a certain subset of components on the left side of the separator, and the remaining number of components on the right side. Note that, it is infeasible to guess all possible left-right partition for each type. Instead, we use knapsack-based arguments—based on the two numbers encoding the type, along with the number of "optimal number of crossings" to draw an individual component on its own—in order to limit the number of guesses to be subexponential. In order to keep the overview readable, we will not delve into further details in this context. Overall, this completes the entire classification process and thereby the description of our algorithm for three layers.

Lastly, we remind that for a five (or more) layers, we cannot obtain an algorithm with running time $2^{\mathcal{O}(k^{1-\epsilon})} \cdot n^{\mathcal{O}(1)}$ for any fixed $\epsilon > 0$ under the ETH. For intuition on why the approach described in this overview does not extend to yield this (presumably impossible) result, let us now address the two main components of this overview that are specific to three layers, and do not extend to four (or more) layers. First and foremost is the necessity of having a kernel to make all our guesses efficient—recall that even for $h = 4$, we prove that a polynomial kernel is unlikely to exist. Second is our arguments for deducing the separation itself—in the case of four layers, much more complicated unclassified (as left or right) components arise, and it is unclear how to extend our arguments (while maintaining efficiency) to encompass them.

2.3 Kernelization Overview

The main idea of the kernelization algorithm in Theorem 4 is that if (G, V_1, V_2, V_3, k) is a yes-instance of 3-Layer Crossing Minimization and G is sufficiently big with respect to k then G should have large pieces that have to be drawn without crossings. Moreover, due to the rigidity of drawings without crossing, the drawing of these pieces is essentially unique (up to the reversals of orders and reordering of twins). Thus, our algorithm applies reduction rules that (i) find the parts

that are (almost) uniquely embeddable without crossings and (ii) reduce their size.

After the initial preprocessing allowing us to simplify the input instance, we apply crucial reduction rules to deal with pieces of the graph that admit a drawing without crossings that are connected to the remaining graph via (not necessarily minimal) separators of size 4, 5, and 6. To explain these rules, we consider in more detail the case when we can cut off a subgraph by a separator of size 4.

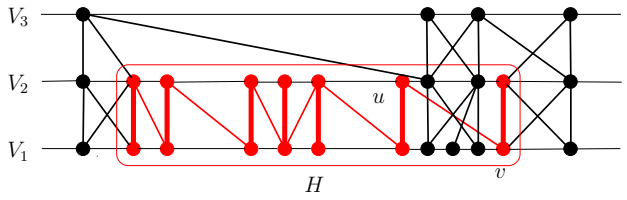

Figure 7: Reducing H attached by 4 vertices. The graph H is shown in red and the part outside H is shown in black; the edges of M are highlighted by thick lines.

Suppose that there is a connected induced subgraph H of G (see Figure 7) such that (i) $V(H) \subseteq V_1 \cup V_2$, (ii) H has a matching M of size $4k+3$, and (iii) H admits a 2-layer drawing without crossings respecting $(V_1 \cap V(H), V_2 \cap V(H))$ such that the endpoints of two distinct edges of M separate H from the remaining graph. Then because of the big matching, the ordering of the vertices in the middle part of the 2-layer drawing of H is rigid in the sense that for any 3-layer drawing of G with at most k crossings such that the number of crossing is minimum, the vertices of the middle part of H are ordered in the same way as in the 2-layer drawing without crossings. Still, we may have crossings between the edges of the middle part of H and some other edges of the part of G outside of H. However, because H admits a 2-layer drawing without crossings, H is a caterpillar. Then it could be assumed that all such crossings are confined to a specific single edge uv of H as shown in Figure 7. This allows us to modify the part of H between the three middle edges of M and reduce its size. Notice that by symmetry, we can do the same for H with $V(H) \subseteq V_2 \cup V_3$.

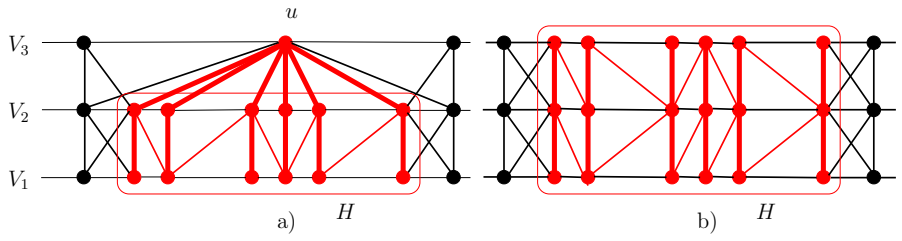

Figure 8: Reducing H attached by 5 (a) and 6 (b) vertices. The graph H is shown in red and the part outside H is shown in black; the paths of length 2 in H are highlighted by thick lines.

For separators of sizes 5 and 6, the idea is the same. Suppose that there is a vertex $u \in V_3$ and a connected induced subgraph H of G (see Figure 8(a)) such that (i) $V(H) \subseteq V_1 \cup V_2$, (ii) H has a matching M of size $4k+3$ such that endpoints of the edges of M in V_2 are adjacent to u, and (iii) H admits a 2-layer drawing without crossings respecting $(V_1 \cap V(H), V_2 \cap V(H))$ such that the endpoints of two distinct edges of M together with u separate H from the remaining graph. Then the paths formed by u and M together with connectivity of H ensure that the middle part of H should be drawn without crossings in any 3-layer drawing of G. This makes it possible to reduce the middle part. The same arguments work for the symmetric case when $u \in V_1$. For the case of separators of size 6, we are looking for connected induced subgraphs H of G (see Figure 8(b)) such that (i) H has a family $\mathcal{P}$ of $4k+3$ vertex disjoint paths of length two between V_1 and V_3 and (ii) H admits a 3-layer drawing such that the vertices of two distinct paths of $\mathcal{P}$ separate H from

the remaining graph, and repeat the trick with the middle part.

If after the exhaustive application of the reduction rules the graph remains sufficiently big, we can correctly report that (G, V_1, V_2, V_3, k) is a no-instance of 3-Layer Crossing Minimization. Otherwise, we have that the number of vertices and edges of the obtained graph is $\mathcal{O}(k^8)$ and the kernelization algorithm returns the graph.

With some extra work, we prove that the kernelization algorithm could be implemented to run $k^{\mathcal{O}(1)} \cdot n$ time. This way, we ensure that the running time of the subexponential algorithm in Theorem 2 is linear in the number of vertices. Finally, we remark that our reduction rules are robust in the sense that, given a solution for the instance obtained by the kernelization algorithm, we can lift it to the original instance.

2.4 Lower Bounds

We show that the complexity h-Layer Crossing Minimization is drastically different for $h \leq 3$ and $h > 3$. To see the reason, return to the reduction rule of the kernelization algorithm where we consider H attached via 4 vertices. We observed that edges of the middle part of H may cross some edges that are not edges of H as shown in Figure 7. Still, we can confine all such crossings to a unique edge of H. However, our arguments are particular to the case when H admits a 2-layer drawing without crossings, and could not be scaled for three or more layers. We use this in our reductions showing lower bounds. In particular, we construct two pairs of complementary gadgets Z and $\widehat{Z}$, and U and $\widehat{U}$ depicted in Figure 28 and Figure 29, respectively. The key property of these gadgets is that if they are forced to cross (and for this, we need additional layers) then they should cross in a very specific way shown in Figure 30 to minimize the number of crossings. Using these gadgets, we are able to encode the symbols of an alphabet and reduce the Disjoint Factors problem.

Given a string s over an alphabet Σ, a *factor* is a nontrivial substring of s whose first and last symbols are the same. The task of Disjoint Factors is to decide whether a string s over an alphabet Σ with k symbols has k disjoint factors with distinct first (and last) symbols. Bodlaender, Thomassé, and Yeo [4] proved that Disjoint Factors parameterized by k does not admit a polynomial kernel unless NP $\subseteq$ coNP /poly. We show our kernelization lower bound in Theorem 5 by providing a polynomial parameter transformation from Disjoint Factors. Theorem 3 is proved in similar way. The difference is that in our reduction for four layers, the number of crossings is $\mathcal{O}(k^2 \log k)$ and this is not good enough to exclude subexponential algorithms. Thus, we introduce an additional fifth layer and modify the reduction to decrease the number of crossings to $\mathcal{O}(k \log k)$. Because the NP-hardness proof for Disjoint Factors [4] implies that the existence of an algorithm running in $2^{o(k)}|s|^{\mathcal{O}(1)}$ time would violate ETH, we obtain that for any $h \geq 5$, h-Layer Crossing Minimization cannot be solved by an algorithm running in $2^{o(k/\log k)} \cdot n^{\mathcal{O}(1)}$ time unless ETH fails.

3 Preliminaries and Problem Statement

This section introduces the basic notation used throughout the paper and provides some auxiliary results.

Graphs. We use the standard graph-theoretic terminology compatible with the textbook of Diestel [12]. We consider only finite undirected graphs. For a graph G, $V(G)$ and $E(G)$ denote its vertex and edge sets. Throughout the paper, we use n to denote the number of vertices if it does not create confusion. For a graph G and a subset $X \subseteq V(G)$ of vertices, we write $G[X]$ to denote the subgraph of G induced by X. We denote by $G - X$ the graph obtained from G by deleting every vertex of X (together with incident edges); we write $G - v$ instead of $G - \{v\}$ for a single vertex set. For $v \in V(G)$, we use $N_G(v)$ to denote the *(open) neighborhood* of v, that is, the set of

vertices of G that are adjacent to v; $N_G[v] = N_G(v) \cup \{v\}$ is the *closed neighborhood* of v. We use $d_G(v) = |N_G(v)|$ to denote the *degree* of v. We say that v is a *pendent* vertex if $d_G(v) = 1$. For $X \subseteq V(G)$, $N_G(X) = \left(\bigcup_{v \in X} N_G(v)\right) \setminus X$. A *separation* of G is a pair of vertex subsets (A, B) such that G has no edges with endpoints in both $A \setminus B$ and $B \setminus A$; $A \cap B$ is a *separator*. A vertex v is a *cut vertex* if $G - v$ has more connected components than G. We use $P = v_0 v_1 \cdots v_\ell$ to denote the path with the vertices $\{v_0, \ldots, v_\ell\}$ and the edges $v_{i-1}v_i$ for $i \in \{1, \ldots, \ell\}$; v_0 and v_ℓ are *end-vertices* and we say that P is an (v_0, v_ℓ)-path.

h-Layer Crossing Minimization. Let G be an h-partite graph with an h-partition $(V_1, \ldots, V_h)$ of the vertex set for $h \geq 2$ with the property that for any edge $e \in E(G)$, there is $i \in \{2, \ldots, h\}$ such that e has one endpoint in V_{i-1} and the other in V_i. Throughout the paper, whenever we consider an h-partite graph with an h-partition $(V_1, \ldots, V_h)$, we assume the property that the edges have their endpoints in consecutive sets. An *h-layer drawing of G respecting* $(V_1, \ldots, V_h)$ is an embedding of G in the plane such that the vertices of G are drawn on h consecutive parallel lines $L_1, \ldots, L_h$, with the vertices of V_i drawn on L_i for each $i \in \{1, \ldots, h\}$, and the edges are drawn as straight line segments. It can be assumed that L_i is defined by the equation $y = i$ in the Euclidean plane for each $i \in \{1, \ldots, h\}$. Formally, an h-layer drawing is an h-tuple $\sigma = (\sigma_1, \ldots, \sigma_h)$ of linear orders of $V_1, \ldots, V_h$, respectively. Given such an h-tuple, the vertices of each V_i are drawn on L_i according to σ_i with strictly increasing x-coordinates. Two edges u_1v_1 and u_2v_2 with $u_1, u_2 \in V_{i-1}$ and $v_1, v_2 \in V_i$ for some $i \in \{2, \ldots, h\}$ are *crossing* if the segments representing these edges on the plane are intersecting or, equivalently if either $\sigma_{i-1}(u_1) < \sigma_{i-1}(u_2)$ and $\sigma_i(v_1) > \sigma_i(v_2)$ or, symmetrically, $\sigma_{i-1}(u_1) > \sigma_{i-1}(u_2)$ and $\sigma_i(v_1) < \sigma_i(v_2)$. Then, the number of crossings is the number of crossing pairs of edges. We consider the following problem.

h-Layer Crossing Minimization

Input:	An h-partite graph G with an h-partition $(V_1, \ldots, V_h)$ for $h \geq 2$ and a positive integer k.
Task:	Decide whether there is an h-layer drawing $\sigma = (\sigma_1, \ldots, \sigma_h)$ of G respecting $(V_1, \ldots, V_h)$ with at most k crossings.

Parameterized Complexity. We refer to the books of Cygan et al. [10] and Fomin et al. [22] for an introduction to the area. We remind that a *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$ where Σ^* is a set of strings over a finite alphabet Σ. An input of a parameterized problem is a pair (x, k) where x is a string over Σ and $k \in \mathbb{N}$ is a *parameter*. A parameterized problem is *fixed-parameter tractable* (or FPT) if it can be solved in time $f(k) \cdot |x|^{\mathcal{O}(1)}$ for some computable function f. The complexity class FPT contains all fixed-parameter tractable parameterized problems. A *kernelization algorithm* or *kernel* for a parameterized problem L is a polynomial-time algorithm that takes as its input an instance (x, k) of L and returns an instance (x', k') of the same problem such that (i) $(x, k) \in L$ if and only if $(x', k') \in L$ and (ii) $|x'| + k' \leq f(k)$ for some computable function $f\colon \mathbb{N} \to \mathbb{N}$. The function f is the *size* of the kernel; a kernel is *polynomial* if f is a polynomial. It is common to wright down a kernelization algorithm as a series of *reduction rules*, that is, polynomial-time algorithms that take as the input an instance of a parameterized problem and output a "smaller" instance of the problem. A rule is *safe* if it outputs an equivalent instance.

While every decidable parameterized problem is FPT if and only if the problem admits a kernel, all FPT problems are unlikely to have polynomial kernels. In particular, to rule out a polynomial kernel, one can use a *polynomial parameter transformation*. Given two parameterized problems L and L', a polynomial parameter transformation is a polynomial-time algorithm that transforms an instance (x, k) of L into an instance (x', k') of L' such that (i) $(x, k) \in L$ if and only if $(x', k') \in L'$ and (ii) $k' \leq p(k)$ for a polynomial p. Then if L does not admit a polynomial kernel (up to some complexity assumptions), the same holds for L'.

The *Exponential Time Hypothesis* (ETH) of Impagliazzo, Paturi, and Zane [33, 34] is an important tool for obtaining fine-grained complexity lower bounds for parameterized problems. We remind that ETH is the assumption that $\delta_k > 0$ for $k \geq 3$, where δ_k is the infimum of the values δ such that k-SAT can be solved in $\mathcal{O}(2^{\delta n})$ time on formulas with n variables. In particular, this means that 3-SAT cannot be solved in $2^{o(n)}$-time. Moreover, by the Sparsification Lemma [34], 3-SAT cannot be solved in $2^{o(m)} \cdot (n+m)^{\mathcal{O}(1)}$ time on formulas with n variables and m clauses.

For $h = 2$, testing whether a graph could be drawn without crossing is accomplished by using the following observation, instead of using the heavy machinery of the linear-time FPT algorithm of [14]. Recall that a *caterpillar* is a tree such that every vertex is adjacent to a vertex of a *central* path (or a *backbone*).

Observation 1 ([17])**.** A bipartite graph G with a bipartition (V_1, V_2) of the vertex set admits a 2-layer drawing respecting (V_1, V_2) without any crossings if and only if each connected component of G is a caterpillar. Moreover, a 2-layer drawing of a caterpillar is unique in that it reversals the orders and permutations of pendent vertices in the orders, and the vertices of a backbone are ordered in the path order.

Also, to verify the existence of an h-layered drawing without crossings, we can use the algorithm of Jünger, Leipert, and Mutzel [36]. We are using this result in the following special form.

Proposition 1 ([36])**.** *It can be decided in $\mathcal{O}(n)$ time for a connected h-partite graph G with an h-partition $(V_1, \ldots, V_h)$ and h pairs of distinct vertices $s_i, t_i \in V_i$ for $i \in \{1, \ldots, h\}$, whether G admits an h-layered drawing $\sigma = (\sigma_1, \ldots, \sigma_h)$ without crossings such that for each $i \in \{1, \ldots, h\}$, $\sigma_i(s_i) \leq \sigma_i(v) \leq \sigma_i(t_i)$ for all $v \in V_i$. Furthermore, if such a drawing exists, it can be found in linear time.*

Proof. By the results of [36], it can be decided in $\mathcal{O}(n)$ time whether G admits an h-layered drawing $\sigma = (\sigma_1, \ldots, \sigma_h)$ respecting $(V_1, \ldots, V_h)$ that has no crossings. It remains to show that we can check the existence of a drawing with the additional constraints on the first and last vertices in each layer.

Consider a connected h-partite graph G with an h-partition $(V_1, \ldots, V_h)$ given together with h pairs of distinct vertices $s_i, t_i \in V_i$ for $i \in \{1, \ldots, h\}$. Notice that if there is $i \in \{2, \ldots, h\}$ such that s_{i-1} has a neighbor in V_i distinct from s_i and s_i has a neighbor in V_{i-1} distinct from s_{i-1} then G has no h-layered drawing without crossings such that s_{i-1} and s_i are the first vertices in the corresponding orders. Symmetrically, G has no required h-layered drawing if t_{i-1} has a neighbor distinct from t_i in V_i and t_i is adjacent to a vertex of V_{i-1} distinct from t_{i-1}. We can check whether G has such a structure in linear time. Then we stop and return a no-answer. From now on, we assume that this is not the case.

We construct the graph G' from G as follows.

- For each $i \in \{2, \ldots, h\}$, make s_{i-1} adjacent to s_i and make t_{i-1} adjacent to t_i (unless these edges are already in G).
- Construct h vertices $s'_1, \ldots, s'_h$ and create the path $s'_1 \cdots s'_h$. Symmetrically, construct h vertices $t'_1, \cdots, t'_h$ and construct the path $t'_1 \cdots t'_h$. Put s'_i and t'_i in V_i for all $i \in \{1, \ldots, h\}$.
- For every $i \in \{2, \ldots, h\}$, make s'_{i-1} adjacent to s_i if s_{i-1} has a neighbor in V_i distinct from s_i and make s'_i adjacent to s_{i-1}, otherwise.
- For every $i \in \{2, \ldots, h\}$, make t'_{i-1} adjacent to t_i if t_{i-1} has a neighbor in V_i distinct from t_i and make t'_i adjacent to t_{i-1}, otherwise.

Observe that if G has an h-layered drawing $\sigma = (\sigma_1, \ldots, \sigma_h)$ without crossings such that for each $i \in \{1, \ldots, h\}$, $\sigma_i(s_i) \leq \sigma_i(v) \leq \sigma_i(t_i)$ for all $v \in V_i$ then G' admits an h-layered drawing that has no crossings. Such a drawing $\sigma' = (\sigma'_1, \ldots, \sigma'_h)$ can be obtained by constructing σ'_i from σ_i by

adding s'_i in the beginning of the order and making t'_i the last vertex for all $i \in \{1, \ldots, h\}$. We claim that if G' has an h-layered drawing $\sigma' = (\sigma'_1, \ldots, \sigma'_h)$ without crossings then G has an h-layered drawing $\sigma = (\sigma_1, \ldots, \sigma_h)$ without crossings such that for each $i \in \{1, \ldots, h\}$, $\sigma_i(s_i) \leq \sigma_i(v) \leq \sigma_i(t_i)$ for all $v \in V_i$. Moreover, such a drawing is induced by σ'.

To see this, notice that G' has paths $P = s_1 \cdots s_h$, $P' = s'_1 \cdots s'_h$, $Q = t_1 \cdots t_h$, and $Q' = t'_1 \cdots t'_h$. Because σ' has no crossings, we have that P and P' do not cross and either $\sigma'_i(s_i) < \sigma'_i(t_i)$ for every $i \in \{1, \ldots, h\}$ or $\sigma'_i(t_i) < \sigma'_i(s_i)$ for every $i \in \{1, \ldots, h\}$. By symmetry, we assume without loss of generality that $\sigma'_i(s_i) < \sigma'_i(t_i)$ for every $i \in \{1, \ldots, h\}$. Then because H is connected, we have that for every $i \in \{1, \ldots, h\}$, $\sigma'_i(s'_i) < \sigma'_i(s_i) < \sigma'_i(t_i) < \sigma'_i(t'_i)$ as P, P', Q, and Q' have no crossings and P' and Q' do not cross the edges of G. Consider $i \in \{2, \ldots, h\}$ and let $v \in V_i$ be a neighbor of s_{i-1} in G distinct from s_i. Then $\sigma_i(v) > \sigma_i(s_i)$ because, otherwise, $s_{i-1}v$ would cross the edge $s'_{i-1}s_i$ of G'. Similarly, if $v \in V_{i-1}$ is a neighbor of s_i in G distinct from s_{i-1} then $\sigma_{i-1}(v) > \sigma_{i-1}(s_{i-1})$. We use the same argument for the vertices $t_1 \ldots, t_h$ and obtain that for each $i \in \{1, \ldots, h\}$, $\sigma'_i(s'_i) < \sigma'_i(s_i) \leq \sigma'_i(v) \leq \sigma'_i(t_i) < \sigma'_i(t'_i)$ for all $v \in V_i$. We define σ_i to be the order obtained from σ'_i by the deletion of s'_i and t'_i for all $i \in \{1, \ldots, h\}$ and conclude that $\sigma = (\sigma_1, \ldots, \sigma_h)$ is an h-layered drawing of G without crossings such that for each $i \in \{1, \ldots, h\}$, $\sigma_i(s_i) \leq \sigma_i(v) \leq \sigma_i(t_i)$ for all $v \in V_i$.

Observe that G' can be constructed from G in linear time. Then in linear time, we can verify whether G' admits an h-layered drawing without crossings and find such a drawing $\sigma' = (\sigma'_1, \ldots, \sigma'_h)$ if exists using the results of [36]. Finally, we construct the drawing of G without crossings in linear time. This concludes the proof. □

We will also require the following proposition due to Blažej et al. [3].

Proposition 2 ([3]). *For $h \leq 3$, there exists a polynomial-time algorithm, that takes as an input an h-layered graph $G = (V, E)$, where $V = \bigcup_{i \in [h]} V_i$ and partial orders π_i for $i \in [h]$, and outputs an h-layered planar drawing of G that also respects the partial orders π_i for each $i \in [h]$; or correctly outputs that there exists no such drawing.*

We conclude this section by discussing the kernelization result of Kobayashi et al. [43] for 2-Layer Crossing Minimization showing that the problem admits a kernel with $\mathcal{O}(k^2)$ vertices and edges for connected graphs. We note that the kernelization can be done in linear time.

Proposition 3 ([43]). 2-Layer Crossing Minimization *admits a kernel with $\mathcal{O}(k^2)$ vertices and edges for instances restricted to connected graphs. Moreover, the kernelization algorithm can be implemented to run in $\mathcal{O}(n)$ time.*

Proof of Proposition 3: Sketch of the running time evaluation. Notice that if $|E(G)| \geq |V(G)| + k$ then (G, V_1, V_2, k) is a trivial no-instance of 2-Layer Crossing Minimization by Observation 1. Also, if $|V(G)| \leq k^2$, we can return the original instance and stop. Thus, we can assume that $|E(G)| \leq n + k$ and $n > k$. In particular, this means that any linear in the input size algorithm runs in $\mathcal{O}(n)$ time. By its first step, the kernelization algorithm of Kobayashi et al. finds the blocks of G, and this can be done in linear time using standard tools [32] that simultaneously produce the block-cut tree of G. Given the set $\mathcal{B}$ of nontrivial blocks, it is shown that if $\ell = \frac{1}{3}\sum_{B \in \mathcal{B}}(E(B) - 1) > k$ then (G, V_1, V_2, k) is a no-instance. Therefore, it can be assumed that $\ell \leq k$. Then, the algorithm boils down to contracting certain bridges.

A bridge e is called *order-inducing* if each of the two connected components of $G - e$ has more than $k - \ell$ edges. Further, an order-inducing bridge e is *contractable* if each endpoint of e (i) is incident to an order-inducing bridge e' distinct from e and (ii) is not adjacent to any other non-leaf edge $e'' \neq e, e'$ (an edge is a *leaf-edge* if it is incident to a pendent vertex). Then, the algorithm contracts all contractible edges. To see that the algorithm can be implemented to run in linear time, it is sufficient to observe that given the block-cut tree, one can compute all order-inducing bridges and all contractible bridges in time $\mathcal{O}(n)$. Finally, we contract these bridges simultaneously. □

4 Subexponential Algorithm for 2-Layer Crossing Minimization

Recall that an instance of 2-Layer Crossing Minimization is given by $(H, U_1, U_2, k^\star)$, where $H = (U_1 \uplus U_2, E)$ is a two-layered (i.e., bipartite) graph. Throughout the algorithm, we will refer to $k^\star$ as the *original* parameter. We will use n and m to denote the number of vertices and edges in the original instance, respectively.[3] Before delving into our algorithm, in this subsection we set up some notation and define some notions that will be used to design our subexponential algorithm. We start with the following definition.

Definition 1. Consider a two-layered graph (G, V_1, V_2). For an edge e and $i \in [2]$, we refer its endpoint in V_i by $\mathsf{endpt}(e, i)$. The set of both endpoints of e is denoted by $\mathsf{Endpoints}(e)$. We also extend the notations $\mathsf{endpt}(\cdot, i)$ and $\mathsf{Endpoints}(\cdot)$ to subsets of edges.

Next, we define the notion of extended instance. Here, we will define two kinds of extended instances – (i) *normal* extended instances, which are the simpler kind of instances encountered while designing a recursive algorithm 2-Layer Crossing Minimization; and (ii) *elaborate* extended instances, which have additional constraints that arise when we wish to use 2-Layer Crossing Minimization as a subroutine in the recursive algorithm for 3-Layer Crossing Minimization. For the most part, normal extended instances are simpler to handle, as compared to the elaborate extended instances, due to the additional constraints on the latter. Next, we define these two instances. Here, the properties that are only applicable for *normal* extended instances are marked with ♦, whereas those that are only applicable for *elaborate* ones are marked with ♣. The properties that are common to both are not marked. We refer to Figure 9 for an illustration.

Definition 2 (Extended Instance for $h = 2$)**.** An *extended instance* of 2-Layer Crossing Minimization is given by $(G, V_1, V_2, \mathsf{ExtBnd}, \pi, k)$ where,

1. G, with $V(G) = V_1 \uplus V_2$, and $E(G) = \mathsf{RegularEdges} \uplus \mathsf{WeightedEdges} \uplus \mathsf{BoundaryEdges}$. Here, $E(G)$ could be a multiset.

2. $\mathsf{BoundaryEdges} = \{\mathsf{leftbound}, \mathsf{rightbound}\}$ is a set of two *boundary edges* of the instance. We refer to $\mathsf{leftbound}$ and $\mathsf{rightbound}$ as the left and right boundary edges of the instance, respectively.

3. ♦ G is connected. Further, $\mathsf{ExtBnd} = \emptyset$ and it is not relevant.
   ♣ $\mathsf{ExtBnd} = \mathsf{LeftExtBnd} \uplus \mathsf{RightExtBnd} \subseteq V_1$ is a set of at most $3\sqrt{k^\star}$ vertices, where $\mathsf{endpt}(\mathsf{leftbound}, 1) \in \mathsf{LeftExtBnd}$ and $\mathsf{endpt}(\mathsf{rightbound}, 1) \in \mathsf{RightExtBnd}$. Further, each vertex v is reachable from some vertex of $\mathsf{ExtBnd} \cup \mathsf{Endpoints}(\mathsf{BoundaryEdges})$.

4. ♦ π is not relevant.
   ♣ $\pi = (\pi_1, \pi_2)$, where for each $i \in [2]$, π_i is a partial order on V_i such that, π_i is the union of $\lambda \geq 0$ different relations $\pi_i^1, \pi_i^2, \ldots, \pi_i^\lambda$, where for each $1 \leq j \leq \lambda$, π_i^j is a total order on $V_i^j \subseteq V_i$, and the sets $\left\{V_i^j : 1 \leq j \leq \lambda\right\}$ are not necessarily disjoint. Here, λ always remains bounded by $4\log(k^\star)$.
   Among these partial orders, π_i^1 is a total order on ExtBnd where the vertices appear in the following order from left to right: $\mathsf{endpt}(\mathsf{leftbound}, 1)$, vertices of $\mathsf{LeftExtBnd} \setminus \mathsf{endpt}(\mathsf{leftbound}, 1)$ in some order, vertices of $\mathsf{RightExtBnd} \setminus \mathsf{endpt}(\mathsf{rightbound}, 1)$ in some order, $\mathsf{endpt}(\mathsf{rightbound}, 1)$.

5. Each $e \in \mathsf{WeightedEdges}$ is referred to as a *multi-edge*, and has an associated weight/multiplicity $\mu(e)$ (which is a positive integer).

[3] Note that if we start with the kernelized instance, then n and m are polynomial in the original parameter $k^\star$; however we will only use the kernelization to derive our final theorem.

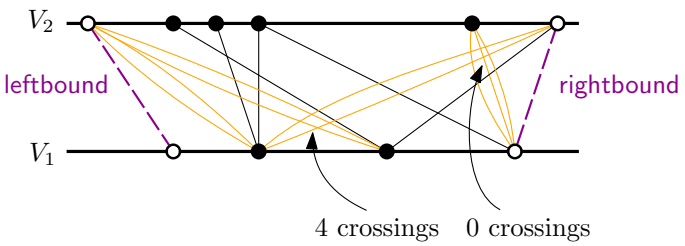

Figure 9: Example of an extended instance of 2-Layer Crossing Minimization with a corresponding drawing. BoundaryEdges = {leftbound, rightbound} are shown as dashed purple edges. All multi-edges in WeightedEdges are shown in orange, and parallel edges indicate multiplicity. RegularEdges are shown in black. Crossings are counted with multiplicity in accordance with item 4 in Definition 3.

6. Each multi-edge $e \in$ WeightedEdges is incident to exactly one vertex of Endpoints(BoundaryEdges). Further, for each vertex $v \in$ Endpoints(BoundaryEdges), the total sum of multiplicities of multi-edges incident to v is at most $2\sqrt{k^\star}$.

Next, we define the notion of drawing of (normal/elaborate) extended instances.

Definition 3 (Drawing of an extended instance.)**.** Let $\mathcal{I} = (G, V_1, V_2, \mathsf{ExtBnd}, \pi, k)$ be an extended instance of 2-Layer Crossing Minimization. We say that $\sigma = (\sigma_1, \sigma_2)$ is a drawing of $\mathcal{I}$ if the following properties are satisfied for each $i \in [2]$ (see Figure 9).

1. σ_i is a permutation of V_i.
2. For $i \in [2]$, and $u \in V_i$, $\sigma_i(\mathsf{endpt}(\mathsf{leftbound}, i)) \leq \sigma_i(u) \leq \sigma_i(\mathsf{endpt}(\mathsf{rightbound}, i))$. That is, the endpoints of leftbound, rightbound are placed leftmost and rightmost on the respective layers.
3. If for some $u, v \in V_i$, if $\pi_i(u) < \pi_i(v)$, then $\sigma_i(u) < \sigma_i(v)$, i.e., σ_i is compatible with π_i.
4. ♣ In the order σ_1, the vertices of LeftExtBnd must be placed consecutively (i.e., no vertex outside LeftExtBnd can have two vertices of LeftExtBnd placed on its either sides), and analogously for RightExtBnd.[4]

Further, we count the crossings in the drawing σ in the following way:

1. Constraints on σ imply that no edge of BoundaryEdges is crossed by any other edge.
2. For any pair of edges $e_1, e_2 \in$ RegularEdges, if they cross according to σ (in the regular sense), then we count it as a single crossing.
3. If an edge $e_r \in$ RegularEdges and an edge $e_m \in$ WeightedEdges with multiplicity $\mu(e_m)$ cross (in the regular sense), then we count it as $\mu(e_m)$ crossings.
4. Let $e_1, e_2 \in$ WeightedEdges be two multi-edges.
   - If e_1, e_2 are incident to different endpoints of the same edge $e \in$ BoundaryEdges, then they will necessarily cross in any drawing σ, that satisfies the constraints above. However we count zero crossings in σ.[5]
   - Otherwise, if e_1, e_2 cross (in the regular sense), then we count as $\mu(e_1) \cdot \mu(e_2)$ crossings.

[4]Note that, this condition, along with the compatibility π_i^1, implies that a prefix (resp. suffix) of σ_1 is given by π_i^1.

[5]These crossings will be already accounted for while creating the extended instance.

From original instance to extended instances. Consider an original instance $\mathcal{I}^o = (H, U_1, U_2, k^\star)$ of 2-Layer Crossing Minimization. Let us suppose that H is connected – otherwise we can handle each connected component separately. First, we try each value of $0 \leq k \leq k^\star$ (these are $k^\star + 1$ guesses), such that if $\mathcal{I}$ is a Yes-instance, then it admits a drawing σ with k crossings, but not $k-1$ crossings. Next, we guess the leftmost vertex u_1^l and the rightmost vertex u_1^r in U_1, according to some hypothetical unknown drawing $\sigma = (\sigma_1, \sigma_2)$. Note that there are at most n^2 choices for these vertices. Then, for a fixed choice of u_1^l, u_1^r we add two new vertices $u_2^l, u_2^r \in U_2$ to the graph, and add two new edges (u_1^l, u_2^l), and (u_1^r, u_2^r). Let H' denote the resulting graph. It is straightforward to check, that assuming that the guess for u_1^l, u_1^r is correct w.r.t. σ, one can obtain a new drawing $\sigma' = (\sigma_1', \sigma_2')$ of the new graph H', where $\sigma_1' = \sigma_1$, and σ_2' places the new vertices u_2^l, u_2^r at leftmost and rightmost positions, respectively. Further, this drawing does not create any additional crossings. We let $\mathsf{leftbound} = (u_1^l, u_2^l), \mathsf{rightbound} = (u_1^r, u_2^r)$, and $\mathsf{BoundaryEdges} = \{\mathsf{leftbound}, \mathsf{rightbound}\}$. We let $\pi = (\pi_1, \pi_2)$ to be empty partial orders, since they are not relevant in normal instances. All the edges of $E(H') \setminus \mathsf{BoundaryEdges}$ are $\mathsf{RegularEdges}$. Finally, we define $\mathsf{ExtBnd} = \emptyset$, since again it is irrelevant for normal instances. Note that the graph H' is connected, since H was connected. From the preceding discussion, the following observation is immediate.

Observation 2. $\mathcal{I}^o = (H, U_1, U_2, k^\star)$ is a Yes-instance of 2-Layer Crossing Minimization if and only if at least one of the at most $n^{\mathcal{O}(1)}$ extended instances $\mathcal{I}$ obtained after the guessing above is an extended Yes-instance.

Henceforth, we may drop the qualifier *extended* from *extended instance*, and simply say *instance*. Further, the graph G in an instance $\mathcal{I} = (G, V_1, V_2, \mathsf{ExtBnd}, \pi, k)$ is said to be the *underlying graph* of the instance $\mathcal{I}$, and k is said to be its parameter.

Definition 4. Let $\mathcal{I}$ be an instance with underlying graph G and let σ be a drawing of $\mathcal{I}$. Then, for any edge $e \in E$, we define $\mathsf{cross}(e, \sigma)$ as the set of edges that cross e in the drawing σ. If the drawing σ is clear from the context, then we may drop σ from the notation.

Definition 5 (Blue and red edges)**.** Let $\mathcal{I}$ be a Yes-instance with underlying graph G and parameter k, and let σ be a corresponding drawing with at most k crossings. We say that an edge $e \in E(G)$ is a *blue edge* in σ if $|\mathsf{cross}(e, \sigma)| \leq \sqrt{k}$, and we say that e is a *red edge*, otherwise. We denote the set of blue edges by $\mathsf{Blue}(\sigma)$ and the set of red edges by $\mathsf{Red}(\sigma)$. If the drawing is clear from the context, then we may simply say *blue* and *red* edge. See Figure 10.

We have the following observation.

Observation 3. Let $\mathcal{I}$ be a Yes-instance with underlying graph G, and consider any drawing σ with k crossings. Then, for any $t \geq 0$, any subset of $E(G)$ of size larger than $2\sqrt{k} + t$ (where multi-edges are counted with multiplicities) contains at least $t+1$ blue edges w.r.t. σ.

Proof. Let $S \subseteq E(G)$ be a subset greater than $2\sqrt{k} + t$, and suppose for contradiction, that S contains at most t blue edges. Therefore, the number of red edges in S is larger than $2\sqrt{k}$. Since each red edge is crossed by more than $\sqrt{k}$ edges in σ, and every crossing is counted at most twice in this manner, the total number of crossings is strictly larger than k, which contradicts σ has at most k crossings. Note that this argument is oblivious to the convention adopted in Definition 3 regarding the crossing of edges of $\mathsf{WeightedEdges}$. □

4.1 Algorithm

We use n and m to denote the number of vertices and edges (with multiplicities), respectively.[6] Next, we perform $n^{\mathcal{O}(1)}$ guesses in the original instance in order to transform it into an extended

[6]Note that if we start with the kernelized instance, then n and m are polynomial in the original parameter $k^\star$; however we will only use the kernelization to derive our final theorem.

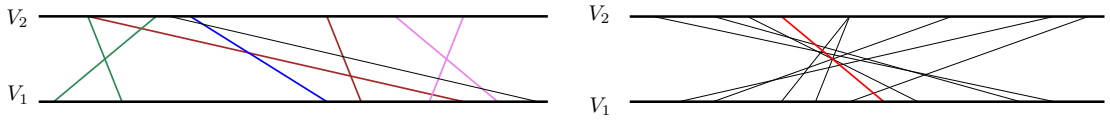

Figure 10: Examples of a blue edge (above) and red edge (below) w.r.t. a drawing from Definition 5. This figure also illustrates the notion of left and right from Definition 7. The crossing between the two green edges is to the left of the blue edge, and that between two pink edges is to its right. The crossing between the two brown edges is neither to its left nor to its right, because (at least) one of the two brown edges crosses the blue edge.

instance. This increases the number of vertices and edges by at most 2, and from Observation 2 it follows that the original instance is a YES-instance iff for at least one of the guesses, we have an YES-instance. For each extended instance corresponding to each of the guesses, we will run the following algorithm.

Let $\mathcal{I} = (G, V_1, V_2, \mathsf{ExtBnd}, \pi, k)$ be the extended instance given as an input. Let c be a sufficiently large constant. When $k \leq c\sqrt{k^\star}$, our algorithm directly solves the given instance by a careful enumeration. Otherwise, when $k > c\sqrt{k^\star}$, we are in the recursive case, explained later.

Base case. At a high level, we want to guess the subset of at most $2k$ crossing edges and the ordering of their endpoints, and check whether the rest of the graph (specifically, each connected component in the graph) can be drawn without crossings, such that it respects the given partial orders π and other constraints. However, due to the way we count the crossings between the multi-edges incident to the different endpoints of a boundary edge (Item 4 in Definition 3), it may happen that a drawing contains crossings; but we do not want to count such drawings. To handle this subtlety, we proceed as follows.

Suppose $\mathcal{I}$ is a YES-instance, then let σ be a drawing of $\mathcal{I}$ with k crossings. Let $X \subseteq E(G)$ be the set of at most $2k$ edges involved in crossings in σ – note that the crossings are counted according to Definition 3. First, we guess X, for which there are at most $\binom{m}{k} = n^{\mathcal{O}(\sqrt{k^\star})}$ choices, since $k = \mathcal{O}(\sqrt{k^\star})$. Let $Y^a := \mathsf{Endpoints}(X)$, and note that $|Y^a| = \mathcal{O}(\sqrt{k^\star})$. Next, let $Y^b := \mathsf{Endpoints}(\mathsf{WeightedEdges})$. Note that due to Item 6 in Definition 2, it follows that $|Y^b| = \mathcal{O}(\sqrt{k^\star})$. Let $Y := Y^a \cup Y^b \cup \mathsf{Endpoints}(\mathsf{BoundaryEdges})$. Let $Y_i = Y \cap V_i$ for $i \in [2]$. We guess a permutation ζ_i of Y_i that is compatible with the given ordering π, and let $\zeta = (\zeta_1, \zeta_2)$. Note that the number of choices for ζ are bounded by $(\mathcal{O}(\sqrt{k^\star}))! = 2^{\mathcal{O}(\sqrt{k^\star}\log(k^\star))}$. Henceforth, we focus on the guess where the guessed permutation ζ is compatible with σ.

Then, we use the algorithm of Proposition 2 ([3]) to find, for each connected component C in $G - (X \cup \mathsf{WeightedEdges})$ a drawing $\sigma'(C)$ without any crossings that respects the given partial order π restricted to C, if such a drawing exists. This can be done in polynomial time due to Proposition 2. If such a drawing does not exist for some component C, then we terminate this guess. Otherwise, we obtain a combined drawing σ' of G by gluing together the orderings of components C and X in an appropriate order, that respects the given partial ordering π.

On the other hand, if for all guesses X and the ordering of the endpoints of edges in X, if either the number of crossings is larger than k, or if at least one connected component of $G - X$ cannot be drawn without crossings, then we return that $\mathcal{I}$ is a no-instance. From the above discussion, the following lemma easily follows.

This completes the description of how we handle the first base case when $k < c\sqrt{k^\star}$.

Lemma 1. *Let $\mathcal{I}$ be an extended instance of* 2-Layer Crossing Minimization *with underlying graph G, and parameter $k \leq c\sqrt{k^\star}$, where c is a sufficiently large constant. Then, the algorithm returns a drawing σ' of $\mathcal{I}$ with k crossings in time $n^{\mathcal{O}(\sqrt{k^\star})}$ iff $\mathcal{I}$ is an extended* YES*-instance.*

Otherwise, we proceed to the recursive case, explained next.

Recursive case. Since the base case is not applicable, we know that $k > c\sqrt{k^\star}$. At a high level, our recursive algorithm consists of the following steps.

1. **Divide.** First, we prove some combinatorial properties of a "balanced separator" in this step, which is defined by a set of two edges, called lsep, rsep. This is a balanced separator, in that the number of crossings to the left of lsep and that to the right of rsep, is at most $k/2$; in addition it has certain additional nice properties. This lets us divide the instance into three parts – left, middle, and right. Of these, the left and the right parts become sub-instances (in fact, extended instances) with parameters bounded by $k/2$, whereas the middle part has certain special properties, which lets us solve it efficiently.

2. **Guess the interfaces.** In this step, we "guess" the two separator edges, as well as other auxiliary sets of vertices and edges, whose existence is showed in the previous step. For this, we have $n^{\mathcal{O}(\sqrt{k})}$ choices.

3. **Creating sub-instances.** For each of the guesses made in the previous step, we formally create the two sub-instances, which takes polynomial time.

4. **Conquer left and right.** For each such pair of sub-instances corresponding to $n^{\mathcal{O}(\sqrt{k}))}$ guesses, we recursively call the algorithm on the left and right sub-instances. Since the parameter is bounded by $k/2$ in each sub-instance, this inductively takes

$$n^{\mathcal{O}(\sqrt{k})} \cdot T(k/2) \leq n^{\mathcal{O}(\sqrt{k}))} \cdot n^{\mathcal{O}(\sqrt{k/2})}$$

   time.

5. **Obtaining a combined drawing.** Assuming that for some guess, both the recursive calls corresponding to the left and right instances output the corresponding drawings, then we combine the drawings—along with the guesses made for the middle subproblem—in order to obtain a combined drawing for the current instance. This takes polynomial time.

Let $T_2(n,k)$ denote an upper bound on the running time of our algorithm on an extended instance on a graph with at most n vertices and parameter bounded by k. From the above description, it follows that $T_2(n,k)$ satisfies the following recurrence.[7]

$$T_2(n,k) = \begin{cases} n^{\mathcal{O}(\sqrt{k^\star})} & \text{if } k \leq c\sqrt{k^\star} \\ n^{\mathcal{O}(\sqrt{k})} \cdot T_2(n, \frac{k}{2}) + n^{\mathcal{O}(1)} & \text{if } k > c\sqrt{k^\star} \end{cases} \tag{1}$$

It can be shown that this recurrence solves to $T_2(n,k^\star) = n^{\mathcal{O}(\sqrt{k^\star})}$.

4.2 Step 1: Divide

Definition 6 (Ordering between edges)**.** See Figure 10. Let e_a, e_b be two distinct edges in E. Further, let $\pi = (\pi_1, \pi_2)$, where each π_i is a (partial) order on V_i, for $i \in [2]$. We say that e_a is *to the left of* e_b *w.r.t.* π if for each $i \in [2]$, we have that $\mathsf{endpt}(e_a, i) \leq \mathsf{endpt}(e_b, i)$, with at least one inequality being strict (since $e_a \neq e_b$). In this case, we also say that e_b is *to the right of* e_a.

Note that "left of" and "right of" relations are partial orders on the edges.

Observation 4. Let $\tau = (\tau_1, \tau_2)$, where each τ_i is permutation of V_i for $i \in [2]$. Then, for any pair of distinct edges $e_a, e_b \in E(G)$, either e_a and e_b cross, or one of the two edges is to the left of the other w.r.t. τ. In particular, for any blue edge $b \in E(G)$, except a subset of at most $\sqrt{k}$ edges of E that cross b in π, each edge in E_{12} is either to the left of b or to the right of b w.r.t. π.

[7] We will give a formal proof of this after giving a formal description of the algorithm.

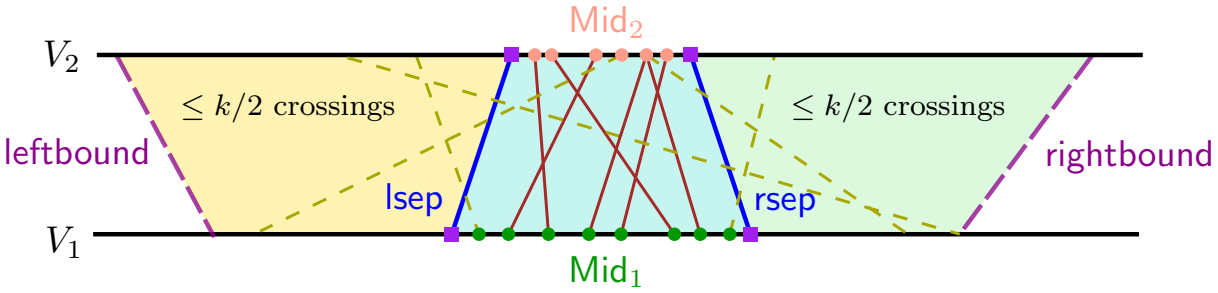

Figure 11: Illustration of a separator from Definition 8. Sep = {lsep, rsep} shown in blue. Edges of $M :=$ MiddleEdges(Sep) are shown in brown, and those of CrossingEdges(Sep) shown in olive (these are the edges that cross lsep or rsep). Vertices of Mid_1 are shown in dark green and those of Mid_2 in light pink. Vertices of Endpoints(Sep) are shown in purple.

Definition 7 (Ordering between an edge and a crossing)**.** Let $\sigma = (\sigma_1, \sigma_2)$ be a drawing of an instance $\mathcal{I}$ with an underlying graph G. Consider an edge $e \in E(G)$. We say that a crossing between edges $e_a, e_b \in E(G)$ (possibly $e = e_a$ or $e = e_b$ but not both) *is to the left (resp. right) of* e *w.r.t.* σ if (i) e_a and e_b cross in σ, and (ii) both e_a and e_b are to the left (resp. right) of e w.r.t. σ. See Figure 10.

Note that when σ is clear from the context, we will omit "according to σ" when using left/right relations. For the rest of this section, we will fix an instance $\mathcal{I}$ of 2-Layer Crossing Minimization that is an Yes-instance for k crossings, but is a No-instance for at most $k-1$ crossings. We refer to this type of instance as a *k-minimal* Yes*-instance*. We now introduce the following crucial definition of a *separator*.

Definition 8. Let $\mathcal{I}$ be a k-minimal Yes-instance, and let σ be any drawing of $\mathcal{I}$ with k crossings. We say that Sep = {lsep, rsep} is a *separator* w.r.t. σ, if the following properties hold (see Figure 11).

1. lsep, rsep $\in E(G)$ are distinct blue edges w.r.t. σ and they do not cross each other.
2. The number of crossings to the left of lsep, and that to the right of rsep are each at most $\frac{k}{2}$.
3. ♦ Define
$$\mathsf{Mid}_i := \{u \in V_i : \sigma_i(\mathsf{endpt}(\mathsf{lsep}, i)) \leq \sigma_i(u) \leq \sigma_i(\mathsf{endpt}(\mathsf{rsep}, i))\}\,.$$
That is, Mid_i is the set of vertices that are drawn between the respective endpoints of the two edges lsep and rsep.
For $i \in [2]$, it holds that $|\mathsf{Mid}_i| < 4\sqrt{k} + 2$.
4. $|\mathsf{MiddleEdges}(\mathsf{Sep}, \sigma)| \leq 2\sqrt{k}$, where
$$\mathsf{MiddleEdges}(\mathsf{Sep}, \sigma) := \{uv \in E : u \in \mathsf{Mid}_1, v \in \mathsf{Mid}_2\} \setminus \{\mathsf{lsep}, \mathsf{rsep}\}\,.$$
5. $|\mathsf{CrossingEdges}(\mathsf{Sep}, \sigma)| \leq 2\sqrt{k}$, where $\mathsf{CrossingEdges}(\mathsf{Sep}, \sigma) := \mathsf{cross}(\mathsf{lsep}, \sigma) \cup \mathsf{cross}(\mathsf{rsep}, \sigma)$

Lemma 2. *Let $\mathcal{I}$ be a k-minimal* Yes*-instance, and let σ be any drawing of $\mathcal{I}$ with k crossings. Then, there exists a separator* Sep *w.r.t. σ.*

Proof. Let us define a lexicographic order $\prec$ on the edges of E using the order of their endpoints in V_1 using σ_1, with ties broken using the endpoints in V_2 using σ_2. More formally, for two distinct edges $e_a, e_b \in E$, we have $e_a \prec e_b$ iff $\sigma_1(\mathsf{endpt}(e_a, 1)) < \sigma_1(\mathsf{endpt}(e_b, 1))$, or $\sigma_1(\mathsf{endpt}(e_a, 1)) = \sigma_1(\mathsf{endpt}(e_b, 1))$ and $\sigma_2(\mathsf{endpt}(e_a, 2)) < \sigma_2(\mathsf{endpt}(e_b, 2))$ (multi-edges are treated as a single edge for this order). Let rsep $\in E(G)$ be the first blue edge, according to $\prec$, that has more than $\frac{k}{2}$ crossings

to its left. First we claim that rsep exists, since rightbound—which is blue, and has all ($k > \frac{k}{2}$) crossings to its left— satisfies the stated property.

By the definition of rsep, the number of crossings to its left is larger than $\frac{k}{2}$, which implies the following: (1) the number of crossings to its right is at most $\frac{k}{2}$ (note that some crossings may not be either to the left or to the right of rsep), and (2) the number of edges to its left is larger than $\frac{k}{2}$. Further, (2) implies that leftbound is one of the edges that is to the left of rsep, since leftbound is to the left of every other edge. In particular, this implies that rsep $\neq$ leftbound. See Figure 11 for an illustration.

Now, we define lsep to be the last edge according to $\prec$ such that: (i) lsep is blue, (ii) lsep $\prec$ rsep, and (iii) lsep does not cross rsep according to σ. First, lsep exists because leftbound satisfies conditions (i) - (iii) – indeed leftbound is not equal to rsep as shown above. Further, since lsep $\prec$ rsep, lsep is to the left of rsep. Now, if the number of crossings to the left of lsep is larger than $\frac{k}{2}$, then this contradicts the choice of rsep as the lexicographically first blue edge with more than $\frac{k}{2}$ crossings to its left. Therefore, the number of crossings to the left of lsep is at most $\frac{k}{2}$. This shows that $\mathsf{Sep} \coloneqq \{\mathsf{lsep}, \mathsf{rsep}\}$ satisfies the first two properties from Definition 8. It remains to show that it also satisfies the third property from Definition 8. Before this, we will show that $|\mathsf{MiddleEdges}(\mathsf{Sep})| \leq 2\sqrt{k}$, as claimed in the statement of the lemma.

Let us use $M \coloneqq \mathsf{MiddleEdges}(\mathsf{Sep})$ for brevity, where the latter is defined in Definition 8. Suppose for the contradiction $|M| > 2\sqrt{k}$. Note that the definition of M, for each edge $e \in M$, lsep $\prec e \prec$ rsep, and e does not cross rsep. Then, by Observation 3, M contains at least one blue edge, say b. However, note that b satisfies conditions (i)–(iii) above, which contradicts the choice of lsep, since lsep is the lexicographically last such edge. This shows that $|\mathsf{MiddleEdges}(\mathsf{Sep})| \leq 2\sqrt{k}$.

♦ Now we will show the bound on the number of vertices in $\mathsf{Mid}_1, \mathsf{Mid}_2$. Note that this part is relevant only for normal instances. Note that $|\mathsf{Endpoints}(\mathsf{Sep}, 1)| \leq 2$ and $\mathsf{Endpoints}(\mathsf{Sep}, 1) \subseteq \mathsf{Mid}_1$, which accounts for the $+2$ factor in the bound. So now, consider any vertex $x \in \mathsf{Mid}_1 \setminus \mathsf{Endpoints}(\mathsf{Sep}, 1)$. Since G is a connected graph, there is at least one edge incident to x, say it is xy with $y \in V_2$. Consider one such edge. If $y \in \mathsf{Mid}_2$, then $xy \in M$, and $|M| \leq 2\sqrt{k}$. Therefore, the number of vertices in Mid_1 with at least one endpoint in Mid_2 is bounded by $2\sqrt{k}$. Otherwise, y satisfies either $\sigma_2(y) < \sigma_2(\mathsf{endpt}(\mathsf{lsep}, 2))$ or $\sigma_2(y) > \sigma_2(\mathsf{endpt}(\mathsf{rsep}, 2)))$. Therefore, it follows that the edge xy crosses either lsep or rsep. However, since lsep and rsep are blue edges, the number of edges crossing either of them is bounded by $\sqrt{k}$, which implies that the number of vertices in Mid_1 without a neighbor in Mid_2 is bounded by $2\sqrt{k}$. Therefore, $|\mathsf{Mid}_1| \leq 2 + 2\sqrt{k} + 2\sqrt{k} = 2 + 4\sqrt{k}$. An analogous argument shows the same bound on $|\mathsf{Mid}_2|$.

□

We introduce the following definition that will be later useful for proving the correctness of the algorithm.

Definition 9. Let $\mathcal{I}$ be a YES-instance with underlying graph G, and let σ be a drawing of $\mathcal{I}$ with k crossings. Consider a separator Sep w.r.t. σ along with the associated objects as in Definition 8 (see Figure 11). Then, we define the following division of vertices on each layer into left (L), middle (M), and right (R) parts, as follows.

$$
\begin{aligned}
V_1(\sigma, \mathsf{Sep}, \mathcal{L}) &\coloneqq \{u \in V_1 : \sigma_1(u) \leq \sigma_1(\mathsf{endpt}(\mathsf{lsep}, 1))\} \\
V_1(\sigma, \mathsf{Sep}, \mathcal{M}) &\coloneqq \{u \in V_1 : \sigma_1(\mathsf{endpt}(\mathsf{lsep}, 1) \leq \sigma_1(u) \leq \sigma_1(\mathsf{endpt}(\mathsf{rsep}, 1))\} \\
V_1(\sigma, \mathsf{Sep}, \mathcal{R}) &\coloneqq \{u \in V_1 : \sigma_1(\mathsf{endpt}(\mathsf{rsep}, 1)) \leq \sigma_1(u)\} \\
V_2(\sigma, \mathsf{Sep}, \mathcal{L}) &\coloneqq \{u \in V_2 : \sigma_2(u) \leq \sigma_2(\mathsf{endpt}(\mathsf{lsep}, 2))\} \\
V_2(\sigma, \mathsf{Sep}, \mathcal{M}) &\coloneqq \{u \in V_2 : \sigma_2(\mathsf{endpt}(\mathsf{lsep}, 2)) \leq \sigma_2(u) \leq \sigma_2(\mathsf{endpt}(\mathsf{rsep}, 2))\} \\
V_2(\sigma, \mathsf{Sep}, \mathcal{R}) &\coloneqq \{u \in V_2 : \sigma_2(\mathsf{endpt}(\mathsf{rsep}, 2)) \leq \sigma_2(u)\}
\end{aligned}
$$

Finally, for each $\mathcal{T} \in \{\mathcal{L}, \mathcal{M}, \mathcal{R}\}$, define $V(\sigma, \mathsf{Sep}, \mathcal{T}) \coloneqq V_1(\sigma, \mathsf{Sep}, \mathcal{T}) \cup V_2(\sigma, \mathsf{Sep}, \mathcal{T})$.

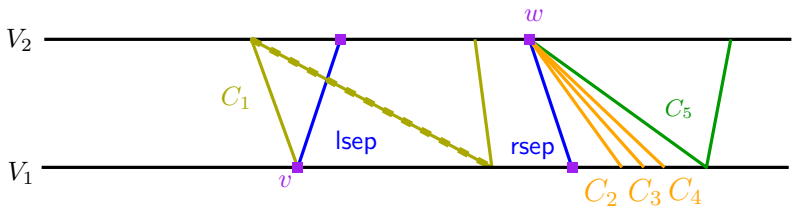

Figure 12: Illustration of classification of components from Definition 10. v, w are two vertices from Endpoints(Sep). $C_1 \in \mathsf{comps}(v)$ belongs to Special(v) since it contains an edge crossing lsep. Three components $C_2, C_3, C_4 \in \mathsf{comps}(w)$ consist of pendant vertices attached to w. Finally, $C_5 \in \mathsf{comps}(w)$ belongs to UpDown(w) since it contains a vertex in the same layer as w.

Specifically, in Figure 11, these sets correspond to the vertices belonging to the yellow, cyan, and light green shaded regions (along with the shared vertices of Endpoints(Sep)), respectively.

Henceforth, until the beginning of Section 4.2.1, we fix a *hypothetical* drawing $\sigma = (\sigma_1, \sigma_2)$ and the corresponding separator Sep corresponding to a k-minimal YES-instance $\mathcal{I}$, as guaranteed by Lemma 2, and the associated objects with Sep from Definition 8. These qualifiers/properties of σ and Sep will not be repeated in the definitions and lemma statements that follow, until the beginning of Section 4.2.1.

In the following definition, we classify the components attached to a vertex in Endpoints(Sep) into multiple classes. Components of each class satisfy certain properties, which will be subsequently used to derive more information on the number of such components, or the way in which they must be drawn.

Definition 10. Let $\mathcal{I} = (G, V_1, V_2, \mathsf{ExtBnd}, \pi, k)$ be an extended instance of 2-LAYER CROSSING MINIMIZATION and σ be a hypothetical drawing. Furthermore, let Sep be the separator corresponding to $\mathcal{I}$. Let $v \in$ Endpoints(Sep) be a vertex, and let $\mathsf{comps}_G(v)$ denote the set of connected components of $C_v - v$, where C_v is the connected component of G that contains v. We partition these components into multiple classes (subsets), called Special(v), π-Defined(v), Pendants(v), and UpDown(v) (we may omit the subscript G, if it is clear from the context). A component $C \in \mathsf{comps}_G(v)$ belongs to:

- Special(v), if it contains at least one edge of CrossingEdges(Sep) $\cup$ MiddleEdges(Sep), or a vertex of Endpoints(Sep).
- π-Defined(v), if $C \notin$ Special(v), and C contains a vertex $u \in \bigcup_{i\in[2]} \bigcup_{j\in[\lambda]} V_i^j$ [8]
- Pendants(v), if it does not belong to Special(v) or π-Defined(v), and C consists of only a single *pendant vertex* u attached to v. Note that the edge uv could belong to WeightedEdges.
- UpDown(v), if it does not belong to Special(v) or π-Defined(v), and contains vertices of both V_1 and V_2.

See Figure 12 for an illustration.

First, we have the following observation.

Observation 5. For any $v \in$ Endpoints(Sep), it holds that $|\mathsf{Special}(v)| \leq |\mathsf{CrossingEdges(Sep)}| + |\mathsf{MiddleEdges(Sep)}| + 2 \leq 4\sqrt{k} + 2$.

Next, we bound the number of components in UpDown(v) in the following lemma.

[8] Recall that V_i^j's are the $\lambda = \mathcal{O}(\log k^\star)$ subsets over which the total orders π_i^j are defined, and these λ total orders constitute π_i. Furthermore, $\bigcup_{i\in[2]} \bigcup_{j\in[\lambda]} V_i^j$ may not be equal to $V(G)$

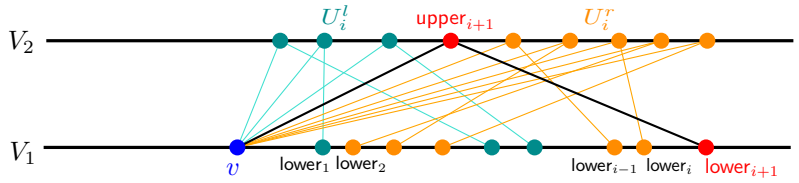

Figure 13: Illustration for the inductive proof in Lemma 3. Here, the vertices of U_i^l—which are to the left of upper^{i+1}—are shown in teal and that of U_i^r—which are to the right of upper^{i+1}—are shown in orange color.

Lemma 3. *For any $v \in \mathsf{Endpoints}(\mathsf{Sep})$, it holds that $|\mathsf{UpDown}(v)| \leq 4\sqrt{k}$.*

Proof. Fix any $v \in \mathsf{Endpoints}(\mathsf{Sep})$. Suppose w.l.o.g. that $v \in V_1$ since V_2 case is symmetric. Suppose for the contradiction that $|\mathsf{UpDown}(v)| > 4\sqrt{k}$.

Consider some component $C \in \mathsf{UpDown}(v)$. Note that C must contain a vertex, say $\mathsf{upper}(C) \in V_2$, that is adjacent to v, as well as a vertex $\mathsf{lower}(C)$, such that $\mathsf{lower}(C) \in V_1$ and is adjacent to $\mathsf{upper}(C)$ (since C does not belong to $\mathsf{Pendants}(v)$). Define $\mathsf{upper}(C)$ and $\mathsf{lower}(C)$ for each $C \in \mathsf{UpDown}(v)$ in this manner (in case of multiple choices for such vertices, select arbitrarily).

For any $C \in \mathsf{UpDown}(v)$, since $\mathsf{lower}(C) \in V_1$ and $v \in V_1$, either $\mathsf{lower}(C)$ is placed to the left of v or to the right of v according to σ_1. By pigeonhole principle, either the number of $\mathsf{lower}(C)$'s to the left or right of u is larger than $2\sqrt{k}$. W.l.o.g., suppose the number of components such that $\mathsf{lower}(C)$ is to the right of v is $r > 2\sqrt{k}$. Let us order the vertices $\mathsf{lower}(C)$ corresponding to such components, as $\mathsf{lower}_1, \mathsf{lower}_2, \ldots, \mathsf{lower}_r$ according to their left-to-right order in σ_1, and let $\mathsf{upper}_1, \mathsf{upper}_2, \ldots, \mathsf{upper}_r$ be the corresponding $\mathsf{upper}(C)$ vertices (note that this need not be their left-to-right order according to σ_2). Note that lower_j's and upper_j's are distinct, since they belong to different connected components in $G - v$. We will show that the number of crossings between the edges $E_r := \{(v, \mathsf{upper}_i), (\mathsf{lower}_i, \mathsf{upper}_i) : 1 \leq i \leq r\}$ is at least $\frac{r(r-1)}{2} \geq \frac{(r-1)^2}{2} > k$, which will yield the desired contradiction.

We prove the claim using induction, where we show that for any $i \geq 1$, the number of crossings among the edges of $E_i := \{(v, \mathsf{upper}_j), (\mathsf{lower}_j, \mathsf{upper}_j) : 1 \leq j \leq i\}$ is at least $\frac{i(i-1)}{2}$. The base case for $i = 1$ is trivial, since there are exactly 0 crossings among the edges of E_1 – since both the edges in E_1 are incident to upper_1. Now, suppose the claim is true for some $i \geq 1$. Let $U_i := \{\mathsf{upper}_j : 1 \leq j \leq i\}$, and let U_i^l, U_i^r denote the set of vertices from U that are to the left and right of upper_{i+1}, respectively. Notice that, for each $\mathsf{upper}_j \in U_i^l$, $\sigma_2(\mathsf{upper}_j) < \sigma_2(\mathsf{upper}_{i+1})$ and $\sigma_1(v) < \sigma_1(\mathsf{upper}_j)$, hence the edge $(v, \mathsf{upper}_{i+1})$ crosses the edge $(\mathsf{lower}_j, \mathsf{upper}_j)$ for each $\mathsf{upper}_j \in U_i^l$. On the other hand, for each $\mathsf{upper}_j \in U_i^r$, $\sigma_1(v) < \sigma_1(\mathsf{lower}_j) < \sigma_1(\mathsf{lower}_{i+1})$ and $\sigma_2(\mathsf{upper}_{i+1}) < \sigma_2(\mathsf{upper}_j)$, hence the edge $(\mathsf{lower}_{i+1}, \mathsf{upper}_{i+1})$ crosses both (v, upper_j) as well $(\mathsf{lower}_j, \mathsf{upper}_j)$, for each $\mathsf{upper}_j \in U_i^r$. Therefore, the number of crossings, where one edge is from E_i and the other edge is from $\{(v, \mathsf{upper}_{i+1}), (\mathsf{upper}_{i+1}, \mathsf{lower}_{i+1})\}$ is at least $l + 2r \geq l + r = i$ (see Figure 13). Then, by using the inductive hypothesis, the number of crossings among the edges of E_{i+1} is at least $\frac{i(i-1)}{2} + i = \frac{i(i+1)}{2}$. This finishes the proof. □

At this point, we are left with analyzing the behavior of components in $\mathsf{Pendants}(v)$. This depends on the structure of the separator $\mathsf{Sep}(\sigma)$ guaranteed by Lemma 2. In the following definition, we study three different cases.

Definition 11. Let $\mathsf{Sep}(\sigma) = \{\mathsf{lsep}, \mathsf{rsep}\}$ be the separator as guaranteed by Lemma 2. We have the following two different (non-exclusive) cases.

(i) Distinct Top (DT). When $\mathsf{endpt}(\mathsf{lsep}, 2) \neq \mathsf{endpt}(\mathsf{rsep}, 2)$.

(ii) Distinct Bottom (DB). When $\mathsf{endpt}(\mathsf{lsep}, 1) \neq \mathsf{endpt}(\mathsf{rsep}, 1)$.

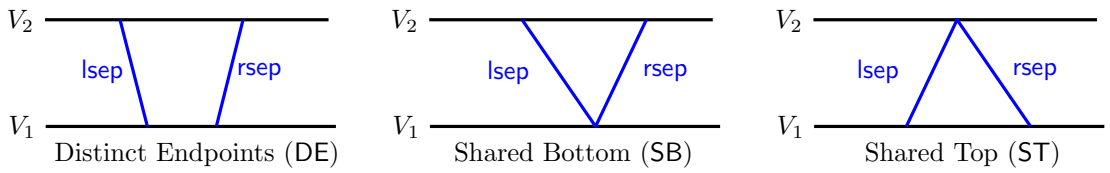

Figure 14: Illustration of three different cases from Definition 11

Based on this, we now have three different mutually exclusive cases (see Figure 14).

1. Distinct Endpoints (DE) case. When both DT and DB hold.

2. Shared Bottom (SB) case. When DT holds but DB does not. That is, $\mathsf{endpt}(\mathsf{lsep}, 2) \neq \mathsf{endpt}(\mathsf{rsep}, 2)$ and $\mathsf{endpt}(\mathsf{lsep}, 1) = \mathsf{endpt}(\mathsf{rsep}, 1)$.

3. Shared Top (ST) case. When DB holds but DT does not. That is, $\mathsf{endpt}(\mathsf{lsep}, 1) \neq \mathsf{endpt}(\mathsf{rsep}, 1)$ and $\mathsf{endpt}(\mathsf{lsep}, 2) = \mathsf{endpt}(\mathsf{rsep}, 2)$.

Lemma 4. *The following properties hold in the drawing σ.*

1. *In* DT case,
   - (a) *All the vertices of* $\mathsf{Pendants}(\mathsf{endpt}(\mathsf{lsep}, 2))$ *are placed to the left of* $\mathsf{endpt}(\mathsf{lsep}, 1)$.
   - (b) *All the vertices of* $\mathsf{Pendants}(\mathsf{endpt}(\mathsf{rsep}, 2))$ *are placed to the right of* $\mathsf{endpt}(\mathsf{rsep}, 1)$.
2. *In* DB case,
   - (c) *All the vertices of* $\mathsf{Pendants}(\mathsf{endpt}(\mathsf{lsep}, 1))$ *are placed to the left of* $\mathsf{endpt}(\mathsf{lsep}, 2)$.
   - (d) *All the vertices of* $\mathsf{Pendants}(\mathsf{endpt}(\mathsf{rsep}, 1))$ *are placed to the right of* $\mathsf{endpt}(\mathsf{rsep}, 2)$.

Thus, in DE case, *all (a), (b), (c), (d) hold. In* SB case, *(a), and (b) hold. In* ST case, *(c) and (d) hold.*

Proof. First let us consider DT case. Suppose that there exists a pendant vertex $w \in \mathsf{Pendants}(u_2)$ attached to $u_2 \coloneqq \mathsf{endpt}(\mathsf{lsep}, 2))$ that lies to the right of $u_1 \coloneqq \mathsf{endpt}(\mathsf{lsep}, 1)$. That is, $\sigma_1(u_1) < \sigma_1(w)$. Let $u_1' \coloneqq \mathsf{endpt}(\mathsf{rsep}, 1)$, and $u_2' \coloneqq \mathsf{endpt}(\mathsf{rsep}, 2)$. Now, either (i) $\sigma_1(w) < \sigma_1(u_1')$, or (ii) $\sigma_1(u_1') < \sigma_1(w)$. In case (i), $w \in \mathsf{Mid}_1$ and thus $u_2w \in \mathsf{MiddleEdges}(\mathsf{Sep})$, contradicting that $w \notin \mathsf{Special}(u_2)$. In case (ii), the edge u_2w crosses $\mathsf{rsep} = (u_1', u_2')$, since $\sigma_1(u_1) < \sigma_1(u_1')$ and $\sigma_1(u_1') < \sigma_1(w)$. This implies that $u_2w \in \mathsf{CrossingEdges}(\mathsf{Sep})$, again contradicting that $w \notin \mathsf{Special}(u_2)$. Similarly, one can prove (b) in DT case, and (c) and (d) in DB case.

Now in DE case, both DT and DB hold, which implies properties (a)–(d). In SB case, only DT holds, which implies properties (a) and (b). Finally, in ST case, only DB holds, which implies properties (c) and (d). □

Thus, we are left with analyzing the behavior of some of the pendant components in SB and ST cases. Specifically, in SB case, when $\mathsf{endpt}(\mathsf{lsep}, 1) = \mathsf{endpt}(\mathsf{rsep}, 1)$, we need to analyze which components in $\mathsf{Pendants}(\mathsf{endpt}(\mathsf{lsep}, 1))$ (which is the same as $\mathsf{Pendants}(\mathsf{endpt}(\mathsf{rsep}, 1))$) are placed to the left of $\mathsf{endpt}(\mathsf{lsep}, 2)$, and which of them are placed to the right of $\mathsf{endpt}(\mathsf{rsep}, 2)$. Note that there cannot be any that are placed in between the two vertices, since otherwise they would belong to $\mathsf{Special}(\mathsf{endpt}(\mathsf{lsep}, 1))$. Analogously, in ST case, when $\mathsf{endpt}(\mathsf{lsep}, 2) = \mathsf{endpt}(\mathsf{rsep}, 2)$, we need to analyze which components in $\mathsf{Pendants}(\mathsf{endpt}(\mathsf{lsep}, 2))$ (which is the same as $\mathsf{Pendants}(\mathsf{endpt}(\mathsf{rsep}, 2))$ are placed to the left of $\mathsf{endpt}(\mathsf{lsep}, 1)$, and which of them are placed to the right of $\mathsf{endpt}(\mathsf{rsep}, 1)$. In the next subsection, we will focus on SB case, and the ST case will be handled analogously. These cases are not equivalent due to weight, and, also, we need to take care of the placement of these pendant components to preserve the number of crossings on both sides. In short, we are going to show that all that matters is the number of these components on both sides and not the components itself.

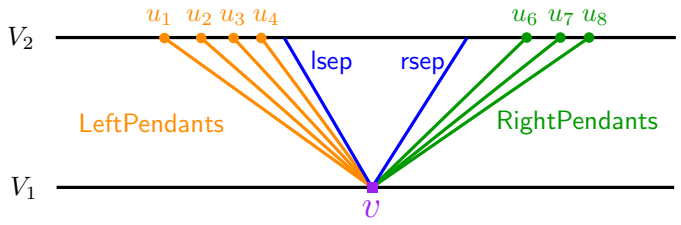

Figure 15: A separator in SB case, where $v = \mathsf{endpt}(\mathsf{lsep}, 1) = \mathsf{endpt}(\mathsf{rsep}, 1)$. A partition $(\mathsf{LeftPendants}, \mathsf{RightPendants})$ of $\mathsf{Pendants}(v)$, which can be thought of as a PendantGuess. It is compliant with $\mathsf{PendantBlueprint} = (4, 3)$, assuming that the multiplicity of each edge is 1. (Note that the other two endpoints of lsep, rsep are not in $\mathsf{Pendants}(v)$, since the corresponding components belong to $\mathsf{Special}(v)$ according to Definition 10.)

4.2.1 Handling SB case

In this subsection, we will make some simplifying assumptions. First, we are in SB case, when $\mathsf{endpt}(\mathsf{lsep}, 1) = \mathsf{endpt}(\mathsf{rsep}, 1)$, and for simplicity, we denote this shared endpoint by v. Similarly, let $u^l := \mathsf{endpt}(\mathsf{lsep}, 2)$ and $u^r := \mathsf{endpt}(\mathsf{rsep}, 2)$. If a component $C \in \mathsf{Pendants}(v)$, then it contains a pendant vertex w attached to v. In this case, we will use the vertex w and the component C interchangeably. Note here, that the edge vw in a pendant component may be a multi-edge with weight/multiplicity $\ell \geq 0$, in which case we also say that ℓ is the multiplicity of C or w. If vw is not a multi-edge, then we define the multiplicity to be 1.

In the following definition, we introduce the notion of nice families of drawings.

Definition 12. Let the shared endpoint be v and let $u^l := \mathsf{endpt}(\mathsf{lsep}, 2)$ and $u^r := \mathsf{endpt}(\mathsf{rsep}, 2)$.

- A *pendant-blueprint* is a tuple $\mathsf{PendantBlueprint} = (\mathsf{LeftPNum}, \mathsf{RightPNum})$, where $\mathsf{LeftPNum} + \mathsf{RightPNum} = \mathsf{TotalPNum}$, where TotalPNum denotes the total sum of multiplicities of all pendants in $\mathsf{Pendants}(v)$.
- A *pendant-guess* is a partition $\mathsf{PendantGuess} = (\mathsf{LeftPendants}, \mathsf{RightPendants})$ of $\mathsf{Pendants}(v)$. That is, the partition of components of $\mathsf{Pendants}(v)$ into left and right side, respectively. See Figure 15 for an illustration.
- Let $\mathsf{PendantGuess} = (\mathsf{LeftPendants}, \mathsf{RightPendants})$ be a pendant guess, and let $\mathsf{PendantBlueprint} = (\mathsf{LeftPendants}, \mathsf{RightPendants})$ be a pendant blueprint. We say that PendantGuess is PendantBlueprint-*compliant* if it satisfies that, the total sum of multiplicities of vertices in LeftPendants is equal to LeftPNum, and the total sum of multiplicities of vertices in RightPendants is equal to RightPNum.
- Let $\mathsf{PendantBlueprint} = (\mathsf{LeftPNum}, \mathsf{RightPNum})$ be a pendant-blueprint, and let $\mathsf{PendantGuess} = (\mathsf{LeftPendants}, \mathsf{RightPendants})$ be a PendantBlueprint-compliant pendant-guess. Then, we define a $\mathsf{NiceFamily}(\mathsf{PendantBlueprint}, \mathsf{PendantGuess}, v, \sigma)$ to be a family of drawings $\sigma' = (\sigma'_1, \sigma'_2)$ satisfying the following properties.

  1. $\sigma_1(w) = \sigma'_1(w)$ for all $w \in V_1$, i.e., the order of the vertices on the first layer in the drawing σ' is exactly the same as that of σ,
  2. σ'_2 and σ_2, when restricted to the vertices of $V_2 \setminus \mathsf{Pendants}(v)$ are the same,
  3. All the pendants in $\mathsf{LeftPendants}(v)$ are placed to the left of u^l in σ'_2.
  4. All the pendants in $\mathsf{RightPendants}(v)$ are placed to the right of u^r in σ'_2.

We also have the following definition.

Definition 13. Let $\mathcal{I}$ be a k-minimal YES-instance and σ, σ' be drawings with k crossings. Let Sep be a separator w.r.t. σ. We say that σ' is (Sep, σ)-compatible, if it satisfies the following properties.

- Sep is also a separator w.r.t. σ'.
- MiddleEdges(Sep, σ) = MiddleEdges(Sep, σ').
- CrossingEdges(Sep, σ) = CrossingEdges(Sep, σ').

Next, we state the following lemma.

Lemma 5. *Let $\mathcal{I}$ be a k-minimal* YES*-instance, let σ be a drawing with k crossings, and let* Sep *be a separator w.r.t. σ with* SB case. *Let v = endpt(lsep, 1) = endpt(rsep, 1).*

Then, there exists a PendantBlueprint *such that, for any* PendantGuess *that is* PendantBlueprint*-compliant, there exists a drawing $\sigma' \in$ NiceFamily(PendantBlueprint, PendantGuess, v, σ) such that σ' is* (Sep, σ)*-compatible.*

Intuitive explanation of the statement of the lemma. Before proving the lemma, let us illustrate what the lemma is trying to say with the help of an example of Figure 15. Suppose the drawing σ distributes Pendants(v) into left and right as illustrated in the example. As mentioned in the caption, PendantGuess = $(\{v_1, v_2, v_3, v_4\}, \{v_5, v_6, v_7\})$ is PendantBlueprint-compliant, where PendantBlueprint = $(4, 3)$. Now consider another PendantGuess$'$ = $(\{v_2, v_3, v_6, v_7\}, \{v_1, v_4, v_5\})$, which is also PendantBlueprint-compliant. Then, there exists a drawing σ' such that, (i) σ' places $\{v_2, v_3, v_6, v_7\}$ on the left of endpt(lsep, 2), $\{v_1, v_4, v_5\}$ to the right of endpt(rsep, 2), and does not change the ordering of the rest of the vertices (this is captured by the condition that $\sigma' \in$ NiceFamily(PendantBlueprint, PendantGuess$'$, v, σ)); and (ii) Sep and its associated objects remain unchanged in σ and σ' (this is captured by σ' being (Sep, σ)-compatible). Now let us prove the lemma formally.

Proof of Lemma 5. Let $w^\star \in$ Pendants(v), such that the edge $vw^\star$ is crossed q times in σ. We claim that each edge vw for any $w \in$ Pendants(v) is *also* crossed exactly q times in σ. Suppose for the contradiction that there exists some $w' \in$ Pendants(v) such that the edge vw' is crossed $q' < q$ times in σ (if $q' > q$, then swap the roles of w and w' in the following). Then, the drawing σ' is obtained by placing w just after w'. If $\mu(vw) > 0$ is the multiplicity of the edge vw, then the number of crossings in σ' is now $k - \mu(vw) \cdot q + \mu(vw) \cdot q' < k$. Thus, σ' is a drawing of $\mathcal{I}$ with fewer than k crossings, contradicting the k-minimality of $\mathcal{I}$.

Let PendantBlueprint $\coloneqq$ (LeftNum, RightNum). Here, LeftNum (RightNum) denotes the total sum of multiplicities of all pendants in Pendants(v) that are placed to the left of (resp. right of) u^l = endpt(lsep, 2) (resp. u^r = endpt(rsep, 2)) in σ_2.

Consider any pendant $w^l \in$ Pendants(v) that is placed to the left of u^l according σ_2, i.e., $\sigma_2(w^l) < \sigma_2(u^l)$. Similarly, let $w^r \in$ Pendants(v) that is placed to the right of u^r according to σ_2, i.e., $\sigma_2(w^r) > \sigma_2(u^r)$. We will use w^l and w^r in defining σ' in the next paragraph.

Now consider any PendantGuess = (LeftPendants, RightPendants) that is PendantBlueprint-compliant. We claim that, the drawing σ' obtained by modifying σ in the following way, satisfies the claimed properties: (1) relocate all the vertices in LeftPendants in the same place as w^l[9], (2) place all the vertices in RightPendants in the same place as w^r, and (3) the order of all other vertices remains unchanged. First, we observe that σ' thus obtained indeed belongs to NiceFamily(PendantBlueprint, PendantGuess, v, σ). Furthermore, since each edge vw for $w \in$ Pendants(v) is crossed exactly the same number of times in σ, it follows that the relocation of vertices used to obtain σ' does not change the number of times each edge is crossed in σ or σ'. Furthermore, note that σ already places LeftNum (resp. RightNum) pendants (counting multiplicities) from Pendants(v) to the left (resp. right) of u^l (resp. u^r), and the resulting drawing σ' does not alter this either. Therefore, Sep also satisfies all properties from Definition 8 w.r.t. σ' as well. This shows that σ' is (Sep, σ)-compatible. □

[9]More formally, if $a^l, b^l \in V_2 \setminus$ Pendants(v) are the vertices that are immediately to the left and right of w^l in σ_2, respectively, then we place all the vertices of LeftPendants between a^l and b^l in an arbitrary order.

4.3 Step 2. Guessing the Interfaces

In this step, we will "guess" various subsets of edges and vertices that can be used in the creation of two sub-instances in the next step. Each of the guesses is made by assuming that $\mathcal{I}$ is a k-minimal YES-instance with hypothetical solution drawing σ. These guesses will be made in multiple steps, with each step dependent on all possible choices of the previous steps. These guesses can be thought of as *branching* for each of the choices specified in all steps. Furthermore, if any of the guesses in a subsequent step contradicts the constraints imposed on a drawing (as in Definition 3), or a guess made in a prior step, then such a guess is considered invalid and terminated (i.e., that branch is not explored further). If all the guesses are terminated, or the corresponding recursive calls report that the sub-instances are NO-instances, then we will conclude that our assumption that $\mathcal{I}$ was a YES-instance was wrong, and thus we must have a NO-instance.

In each step, we specify at a high level the kind of object that we are trying to guess. In this algorithm, we label vertices using one or more labels from the following set $\{\mathcal{L}, \mathcal{M}, \mathcal{R}\}$, that are indicative of the sub-instance(s) that the vertex will belong to.[10] Whenever we are trying to guess a subset of edges, we follow the following convention: all the edges of a multi-edge are treated in the same manner, such as being chosen in a guess.

Step A: Separator. We start by guessing $\mathsf{Sep} = \{\mathsf{lsep}, \mathsf{rsep}\} \subseteq E$ that is a separator w.r.t. σ. Note that there are at most $m^2 = n^{\mathcal{O}(1)}$ choices. Furthermore, based on our guess, we can also conclude whether we are in DE, SB, or ST case, which will be important later. Next, we guess $\mathsf{Mid}_i \subseteq V_i \setminus \mathsf{Endpoints}(\mathsf{Sep}, i)$ for $i \in \{1, 2\}$, such that each is of size $4\sqrt{k} + 2$. There are at most $n^{\mathcal{O}(\sqrt{k})}$ choices for this. Overall, we have at most $n^{\mathcal{O}(\sqrt{k})}$ choices.

Step B: Labeling $\mathsf{Endpoints}(\mathsf{Sep})$**.** DE case. Vertices in $\mathsf{Endpoints}(\mathsf{lsep})$ receive label $\{\mathcal{L}, \mathcal{M}\}$, and that in $\mathsf{Endpoints}(\mathsf{rsep})$ receive $\{\mathcal{M}, \mathcal{R}\}$.

SB case. $\mathsf{endpt}(\mathsf{lsep}, 1) = \mathsf{endpt}(\mathsf{rsep}, 1)$ receives labels $\{\mathcal{L}, \mathcal{M}, \mathcal{R}\}$, $\mathsf{endpt}(\mathsf{lsep}, 2)$ receives labels $\{\mathcal{L}, \mathcal{M}\}$, and $\mathsf{endpt}(\mathsf{rsep}, 2)$ receives labels $\{\mathcal{M}, \mathcal{R}\}$.

ST case. $\mathsf{endpt}(\mathsf{lsep}, 2) = \mathsf{endpt}(\mathsf{rsep}, 2)$ receives labels $\{\mathcal{L}, \mathcal{M}, \mathcal{R}\}$, $\mathsf{endpt}(\mathsf{lsep}, 1)$ receives labels $\{\mathcal{L}, \mathcal{M}\}$, and $\mathsf{endpt}(\mathsf{rsep}, 1)$ receives labels $\{\mathcal{M}, \mathcal{R}\}$.

Step C: Labeling MiddleEdges **and** CrossingEdges**.** First, we label all endpoints of all the edges of MiddleEdges with the label $\mathcal{M}$.

Next, we guess two sets of edges $\mathsf{cross}(\mathsf{lsep}), \mathsf{cross}(\mathsf{rsep})$ of size at most $\sqrt{k}$ that cross $\mathsf{lsep}, \mathsf{rsep}$, respectively. Then, we define $\mathsf{CrossingEdges} := \mathsf{cross}(\mathsf{lsep}) \cup \mathsf{cross}(\mathsf{rsep})$. Note that

$$|\mathsf{CrossingEdges}| \leq 2\sqrt{k}$$

and the number of guesses is bounded by $n^{\mathcal{O}(\sqrt{k})}$.

We now describe feasible pairs for labels of the endpoints of CrossingEdges; and each guess corresponds to using labeling the endpoint of each edge in CrossingEdges by one of the feasible (ordered) pairs. However, if the guess for the label is *incompatible* with a previously assigned label, then we consider it as an invalid guess and proceed to the next one. Note that, if a vertex has previously been assigned multiple labels, then the incompatible label(s) is a label that has not been assigned to the said vertex, if any.

- For $e \in \mathsf{cross}(\mathsf{lsep})$, the feasible label pairs are $(\mathcal{L}, \mathcal{M}), (\mathcal{M}, \mathcal{L}), (\mathcal{L}, \mathcal{R}), (\mathcal{R}, \mathcal{L})$.
- For $e \in \mathsf{cross}(\mathsf{rsep})$, the feasible label pairs are $(\mathcal{R}, \mathcal{M}), (\mathcal{M}, \mathcal{R}), (\mathcal{L}, \mathcal{R}), (\mathcal{R}, \mathcal{L})$.

Overall, we have at most $\binom{m}{2\sqrt{k}}^4 \cdot 2^{\mathcal{O}(\sqrt{k})} = n^{\mathcal{O}(\sqrt{k})}$ choices. Let $\mathsf{CrossLeft}, \mathsf{CrossRight} \subseteq \mathsf{Endpoints}(\mathsf{CrossingEdges})$ denote the subsets of vertices that receive labels $\mathcal{L}, \mathcal{R}$, respectively.

[10] We do not explicitly define a middle sub-instance in the algorithm, but the label $\mathcal{M}$ is defined for convenience.

Step D: Labeling $\mathsf{comps}(v)$ for $v \in \mathsf{Endpoints}(\mathsf{Sep})$ Recall that for each $v \in \mathsf{Endpoints}(\mathsf{Sep})$, each component $C \in \mathsf{comps}(v)$ can be classified into $\mathsf{Special}(v)$, $\pi\text{-}\mathsf{Defined}(v)$, $\mathsf{UpDown}(v)$, and $\mathsf{Pendants}(v)$ according to Definition 10. Now we describe how to handle each of the components separately.

▶ $\mathsf{Special}(v)$**.** Some of the vertices in such components are already labeled. We will return to the remaining vertices in these components at a later step.

▶ $\pi\text{-}\mathsf{Defined}(v)$**.** Recall that each π_i for $i \in [2]$ is comprised of $\lambda = \mathcal{O}(\log k^\star)$ total orders $\pi_i^1, \pi_i^2, \ldots, \pi_i^\lambda$ for $V_i^1, V_i^2, \ldots, V_i^\lambda$, respectively. For each $i \in [2]$, and for each $j \in [\lambda]$, we do the following.

- We guess two vertices $u(j,l), w(j,r) \in V_i^j$ with $\pi_i^j(u(j,l)) < \pi_i^j(w(j,r))$.
- Any $C \in \pi\text{-}\mathsf{Defined}(v)$ that contains a vertex $u \in V_i^j$ with $\pi_i^j(u) \leq \pi_i^j(u(j,l))$, we label all the vertices of C using $\mathcal{L}$.
- Any $C \in \pi\text{-}\mathsf{Defined}(v)$ that contains a vertex $w \in V_i^j$ with $\pi_i^j(w) \geq \pi_i^j(w(j,r))$, we label all the vertices of C using $\mathcal{R}$.

If any of the guesses made in this step is incompatible with a previous step, or with a prior guess made in this step, then we terminate this guess. Note that the number of guesses made in this step is at most $n^{\mathcal{O}(\lambda)} = n^{\mathcal{O}(\log k^\star)} = n^{\mathcal{O}(\sqrt{k})}$, since $k = \Omega(\sqrt{k^\star})$.

▶ $\mathsf{UpDown}(v)$**.** By Lemma 3, we know that $|\mathsf{UpDown}(v)| \leq 8\sqrt{k}$ for each $v \in \mathsf{Endpoints}(\mathsf{Sep})$. Thus, if the number of such components is more than $8\sqrt{k}$, then we terminate the guess. Otherwise, we guess the label of each of them. The number of guesses for this is bounded by $\binom{n}{8\sqrt{k}} \cdot 2^{8\sqrt{k}} = n^{\mathcal{O}(\sqrt{k})}$.

▶ $\mathsf{Pendants}(v)$**.** We distinguish between DE, SB, ST cases and use Lemma 4 to label components as applicable. In DE case, we label all the components in $\bigcup_{v \in \mathsf{Endpoints}(\mathsf{Sep})} \mathsf{Pendants}(v)$. However, in SB case (resp. ST case), we are left with labeling $\mathsf{Pendants}(\mathsf{endpt}(\mathsf{lsep}, 1))$ (resp. $\mathsf{Pendants}(\mathsf{endpt}(\mathsf{lsep}, 2))$), which we do next.

We focus on the SB case and show how in accordance with Definition 12 and Lemma 5. The ST case is handled analogously. Let $v := \mathsf{endpt}(\mathsf{lsep}, 1) = \mathsf{endpt}(\mathsf{lsep}, 2)$, and let $\mathsf{TotalNum}$ denote the total sum of the multiplicities of $\mathsf{Pendants}(v)$. First, we guess a $\mathsf{PendantBlueprint} = (\mathsf{LeftPNum}, \mathsf{RightPNum})$, for which there are at most n possibilities. Then, for each choice of $\mathsf{PendantBlueprint}$, we use a knapsack-style dynamic programming to find a $\mathsf{PendantBlueprint}$-compatible $\mathsf{PendantGuess} = (\mathsf{LeftPendants}, \mathsf{RightPendants})$ in polynomial time, if one exists. If such a partition does not exist, then we move to the next guess. Otherwise, we label the vertices of $\mathsf{LeftPendants}$ using $\mathcal{L}$ and that of $\mathsf{RightPendants}$ using $\mathcal{R}$.

Step E: Label propagation. Let A denote the subset of vertices that have already received a label, and let $A^{\mathcal{L}}, A^{\mathcal{R}} \subseteq A$ denote the subsets of A that have received the respective label. Further, let $O \subseteq A$ denote the subset of vertices that have received a unique label, and again let $O^{\mathcal{L}}, O^{\mathcal{R}} \subseteq O$ denote the partition of O into two disjoint subsets. At this point, some of the vertices of the graph may still be unlabeled, which we label during this phase. To this end, we consider normal (♦) and elaborate (♣) instances separately.

♦ Define the following sets:

$$\mathsf{Origin}^{\mathcal{L}} = \mathsf{Endpoints}(\{\mathsf{leftbound}, \mathsf{lsep}\}) \cup \mathsf{CrossLeft}$$
$$\mathsf{Origin}^{\mathcal{R}} = \mathsf{Endpoints}(\{\mathsf{rightbound}, \mathsf{rsep}\}) \cup \mathsf{CrossRight}$$

♣ Define the following sets:

$$\mathsf{Origin}^{\mathcal{L}} = (\mathsf{LeftExtBnd} \cap A^{\mathcal{L}}) \cup \mathsf{Endpoints}(\{\mathsf{leftbound}, \mathsf{lsep}\}) \cup \mathsf{CrossLeft}$$
$$\mathsf{Origin}^{\mathcal{R}} = (\mathsf{RightExtBnd} \cap A^{\mathcal{R}}) \cup \mathsf{Endpoints}(\{\mathsf{rightbound}, \mathsf{rsep}\}) \cup \mathsf{CrossRight}$$

Now, regardless of ♦ or ♣, we proceed as follows.

1. For each $u \in \mathsf{Origin}^{\mathcal{L}}$, and each unlabeled $w \notin A$ that is reachable from u in the graph $G - (O^{\mathcal{M}} \cup O^{\mathcal{R}})$, we label it using $\mathcal{L}$.
2. For each $u \in \mathsf{Origin}^{\mathcal{R}}$, and each unlabeled $w \notin A$ that is reachable from u in the graph $G - (O^{\mathcal{L}} \cup O^{\mathcal{M}})$, we label it using $\mathcal{R}$.
3. For each $u \in \mathsf{Origin}^{\mathcal{M}}$, and each unlabeled $w \notin A$ that is reachable from u in the graph $G - (O^{\mathcal{L}} \cup O^{\mathcal{R}})$, we label it using $\mathcal{M}$.

During this process, if a previously unlabeled vertex, i.e., $w \notin A$ receives multiple labels, then we terminate the guess.

Otherwise, for $\mathcal{T} \in \{\mathcal{L}, \mathcal{M}, \mathcal{R}\}$, let $V^{\mathcal{T}}$ denote the subset of vertices that have received the respective label.

Step F: Partial Order. Recall that we are given a partial orders π_1, π_2. For each $i \in [2]$, we will guess permutation between certain subsets of vertices, which will be used to extend π_i to a new partial order π'_i.
Let $S \coloneqq \mathsf{Mid}_1 \cup \mathsf{Mid}_2 \cup \mathsf{Endpoints}(\mathsf{CrossingEdges})$. The bounds on the respective sets imply that $|S| = \mathcal{O}(\sqrt{k})$. For each $i \in [2]$, let $S_i \coloneqq V_i \cap S$. We guess permutations $\pi_i^{\lambda+1}$ of S_i for each $i \in [2]$. Since $|S_i| = \mathcal{O}(\sqrt{k})$, it follows that the total number of guesses is $|S_1|! \cdot |S_2|! = (\mathcal{O}(\sqrt{k})!)^2 = 2^{\mathcal{O}(\sqrt{k}\log k)}$. Let $\pi^{\lambda+1} = (\pi_1^{\lambda+1}, \pi_2^{\lambda+1})$. We say that $\pi^{\lambda+1}$ is *valid* if the following condition is met.

- ♣ For each $i \in [2]$, for any distinct $u, v \in S_i$, if $\pi_i^{\lambda+1}(u) < \pi_i^{\lambda+1}(v)$, then, either u and v are incomparable in π_i, or $\pi_i(u) < \pi_i(v)$.
- $\pi^{\lambda+1}$ satisfies the following properties (that must be satisfied by the separator):
  - $\mathsf{lsep}, \mathsf{rsep}$ do not cross
  - For each $e \in \mathsf{Sep}$, the set of edges crossing e is exactly equal to $\mathsf{cross}(e)$,
  - $\mathsf{MiddleEdges}$ is exactly the set of edges that lie between $\mathsf{lsep}, \mathsf{rsep}$

Note that the validity of $\pi^{\lambda+1}$ can be checked in polynomial time. If a guess for $\pi_i^{\lambda+1}$ is invalid, then we terminate it and move to the next guess.

♣ We define $\pi^{\lambda+2}$ as an ordering over $\mathsf{ExtBnd} \cup \mathsf{Endpoints}(\mathsf{Sep}, 1)$ by combining parts of $\pi_1^{\lambda+1}$, and one of the constituent total orders of π^2 that is a permutation of the set ExtBnd. Then, we define $\pi'_1 = \pi_1 \cup \pi_1^{\lambda+1} \cup \pi_1^{\lambda+2}$, and $\pi'_2 \coloneqq \pi_2 \cup \pi_2^{\lambda+1}$. Note that π'_i is a valid extension of π_i, since $\pi^{\lambda+1}$ is valid. Henceforth, we assume that we are working with such a $\pi' = (\pi'_1, \pi'_2)$ for some guess for $\pi^{\lambda+1}$.

Step G: Parameter division. Let k_{current} denote the number of crossings among the edges of $\mathsf{CrossingEdges} \cup \mathsf{MiddleEdges} \cup \mathsf{Sep}$ according to ζ. If $k_{\mathsf{current}} > k$, then we terminate the current guess. Otherwise, let $k' \coloneqq k - k_{\mathsf{current}}$. We guess two non-negative integers $0 \leq k^{\mathcal{L}}, k^{\mathcal{R}} \leq k/2$ such that $k' = k^{\mathcal{L}} + k^{\mathcal{R}}$. Note that the number of such guesses is bounded by k^2.

The following lemma follows from the description of Step 2.

Lemma 6. *The total number of guesses made in steps A through G, for (i) separator* Sep *and its associated sets of vertices and edges, (ii) labels of vertices of* $V(G)$*, and (iii) division of* k *into smaller parameters, is bounded by* $n^{\mathcal{O}(\sqrt{k})}$.

4.4 Step 3: Creating sub-instances.

For each set of guesses made in the previous step (as stated in Lemma 6), we create a pair of sub-instances, a left sub-instance $\mathcal{I}^{\mathcal{L}}$, and a right sub-instance $\mathcal{I}^{\mathcal{R}}$. In this step, we construct each such pair of sub-instances corresponding to a particular guess. Then, in the next step, we will recursively call our algorithm to solve each of the two sub-instances.

Creating left sub-instance. Let $G^{\mathcal{L}} = G[V^{\mathcal{L}}]$, where recall that $V^{\mathcal{L}}$ is a set of all the vertices with label $\mathcal{L}$ (including a vertex with multiple labels). Next, we add certain multi-edges in the graph.

Creating mutli-edges. For each edge $uv \in \mathsf{cross}(\mathsf{lsep})$, we identify a multi-edge $e_{add}(uv)$ to be added in each case. After the case analysis, we will discuss the multiplicity of this edge, and actually add it to the graph.

- $u \in V_1$ has label $\mathcal{L}$ and $v \in V_2$ has label has label $\mathcal{M}$ or $\mathcal{R}$.
  Operation. Define $e_{add}(uv) \coloneqq (u, \mathsf{endpt}(\mathsf{lsep}, 2))$.
- $v \in V_2$ has label $\mathcal{L}$, and $u \in V_1$ has label $\mathcal{M}$ or $\mathcal{R}$.
  Operation. Define $e_{add}(uv) \coloneqq (\mathsf{endpt}(\mathsf{lsep}, 1), v)$.

Now, we proceed as follows:

- If an edge is defined as e_{add} of multiple edges, say $e_1, e_2, \ldots$, then its multiplicity is defined as the sum of the respective multiplicities of $e_1, e_2, \ldots$ – here we adopt the convention that, if an edge e_i belongs to $\mathsf{RegularEdges}$, then its multiplicity is 1.
- Finally, we add e_{add} with the combined multiplicity to $\mathsf{WeightedEdges}^{\mathcal{L}}$. Here, if e_{add} is already in $\mathsf{RegularEdges}$, we do not remove it.

Let $G^{\mathcal{L}}$ denote the resulting graph. The left sub-instance is an extended instance $\mathcal{I}^{\mathcal{L}} = (G^{\mathcal{L}}, V_1^{\mathcal{L}}, V_2^{\mathcal{L}}, \mathsf{ExtBnd}^{\mathcal{L}}, \pi^{\mathcal{L}}, k^{\mathcal{L}})$, where:

- For each $i \in [2]$, $V_i^{\mathcal{L}} \coloneqq V_i \cap V^{\mathcal{L}}$, and $\pi_i^{\mathcal{L}} = \pi_i'$ projected to $V^{\mathcal{L}}$.
- $\mathsf{BoundaryEdges}^{\mathcal{L}} = \{\mathsf{leftbound}^{\mathcal{L}}, \mathsf{rightbound}^{\mathcal{L}}\}$, where $\mathsf{leftbound}^{\mathcal{L}} = \mathsf{leftbound}$, $\mathsf{rightbound}^{\mathcal{L}} = \mathsf{lsep}$.
- ♦ Let $\mathsf{ExtBnd}^{\mathcal{L}} = \mathsf{LeftExtBnd}^{\mathcal{L}} = \mathsf{RightExtBnd}^{\mathcal{L}} = \emptyset$, since these sets are irrelevant for normal extended instances.
  ♣ For elaborate extended instances, let $\mathsf{LeftExtBnd}^{\mathcal{L}} = \mathsf{LeftExtBnd} \cap V^{\mathcal{L}}$ and $\mathsf{RightExtBnd}^{\mathcal{L}} = \mathsf{endpt}(\mathsf{lsep}, 1)$, and $\mathsf{ExtBnd}^{\mathcal{L}} = \mathsf{LeftExtBnd}^{\mathcal{L}} \cup \mathsf{RightExtBnd}^{\mathcal{L}}$.

At this point, we perform the following check based on whether $\mathcal{I}$ was a normal (♦), or an elaborate (♣) extended instance.
♦ If $G^{\mathcal{L}}$ is not connected, then we terminate the guess.
♣ If each $u \in V^{\mathcal{L}}$ is reachable from some vertex of $v \in \mathsf{ExtBnd}^{\mathcal{L}}$ in $G^{\mathcal{L}}$, then we terminate the guess.

The right sub-instance $\mathcal{I}^{\mathcal{R}} = (G^{\mathcal{R}}, V_1^{\mathcal{R}}, V_2^{\mathcal{R}}, \mathsf{ExtBnd}^{\mathcal{R}}, \pi^{\mathcal{R}}, k^{\mathcal{R}})$ is created analogously, and we perform the connectivity check as described above; we omit the description.

Further, we note the following auxiliary observations, that satisfy the constraints on the extended instances, as mentioned in Definition 2.

- ♣ The parameter in each recursive call reduces by at least a factor of $1/2$ (and the base case is triggered when $k \leq c\sqrt{k^\star}$), and each time we add two total orders to the relation π, it follows that the total number of partial orders remains bounded by $4 \log k^\star$ at each step as required in Item 4.
- ♣ The size of $\mathsf{ExtBnd}^{\mathcal{L}}, \mathsf{ExtBnd}^{\mathcal{R}}$ remains bounded by ExtBnd, which was originally $3\sqrt{k^\star}$, satisfying Item 3.
- Note that, in the original instance $\mathsf{WeightedEdges} = \emptyset$, and in each call of the algorithm, the total sum of multiplicities of $\mathsf{WeightedEdges}$ incident to a vertex $u = \mathsf{endpt}(e, i)$ for some $e \in \mathsf{Sep}$, and some $i \in [2]$ added in that call, is at most the number of crossings the corresponding separator edge e is involved in–which is bounded by $\sqrt{k}$. Again, since the

parameter decreases by a constant factor in each recursive call, it holds that the total sum of multiplicities of all edges incident to a vertex, remains bounded by $\sum_{\ell \geq 0} \sqrt{(1/2)^\ell \cdot k^\star} = 2\sqrt{k^\star}$. Item 6

From the previous discussion, the following claim follows.

Claim 4.1. *If $\mathcal{I}$ was a valid normal (resp. elaborate) extended instance, then $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}$ are valid normal (resp. elaborate) extended instances.*

4.5 Step 4: Conquer left and right.

For each guess in Step 2, we obtain a pair of sub-instances $\mathcal{I}^{\mathcal{L}}$ and $\mathcal{I}^{\mathcal{R}}$, with each having a parameter at most $k/2$. We focus on one pair of instances that we solve recursively in time $2 \cdot n^{\mathcal{O}(\sqrt{k})}$. There are two possibilities. The first one is that at least one of the recursive calls returns that the corresponding sub-instance is a No-instance, in which case we proceed to the next guess. Hence, assume that both recursive calls return that $\mathcal{I}^{\mathcal{L}}$ and $\mathcal{I}^{\mathcal{R}}$ are Yes-instances, and in addition they also output an ordering of the corresponding vertices.

4.6 Step 5: Obtaining the Final Drawing.

If, for any of the non-terminated guesses, all three recursive calls reporting that the respective sub-instances are Yes-instances, along with the corresponding orderings, then we can combine these orderings to obtain orderings of V_1, V_2 that has at most k crossings, and respect the given partial order π.

Specifically, suppose $\sigma^{\mathcal{L}}, \sigma^{\mathcal{R}}$ be the drawings of $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}$ returned by the corresponding recursive calls. Then, we obtain the final drawing of $\mathcal{I}$ by defining $\varsigma_i' \coloneqq \sigma_i^{\mathcal{L}} \circ \pi_i'(\mathsf{Mid}_i) \circ \sigma_i^{\mathcal{R}}$ for $i \in [2]$. We return $\sigma' = (\sigma_1', \sigma_2')$ as our final drawing. Otherwise, if for each guess, either (i) it is terminated due to incompatibility, or (ii) one of the three sub-instances corresponding to the guess is found to be a No-instance, then we output that $\mathcal{I}$ is a No-instance.

Lemma 7. *Let $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}$ be a pair of sub-instances (with parameters $k^{\mathcal{L}}, k^{\mathcal{R}}$) corresponding to some guess made in Step 2. Further, let $\sigma^{\mathcal{L}}, \sigma^{\mathcal{R}}$ be valid drawings of the $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}$. Then, if we obtain a combined drawing σ' as in Step 5, then σ' is a valid drawing of $\mathcal{I}$.*

Proof. Recall that, in step F we guess the permutations $\pi_i^{\lambda+1}$ of S_i, and we use it in step G to determine the number of crossings k_{current} among the edges of $\mathsf{CrossingEdges} \cup \mathsf{MiddleEdges} \cup \mathsf{Sep}$. Then, we define $k = k' + k_{\mathsf{current}}$, and divide it into $k^{\mathcal{L}} + k^{\mathcal{R}}$, which are used as parameters for the respective sub-instances.

Next, we claim that all the k_{current} crossings among the edges are not counted in any of the sub-instances. To this end, first observe that the edges of $\mathsf{MiddleEdges}$ are not involved in any of the subproblems. Next, consider any pair of edges $e_1, e_2 \in \mathsf{CrossingEdges} \cup \mathsf{Sep}$, such that the crossing between e_1 and e_2 is counted in k_{current}. Note that both e_1, e_2 cannot be in Sep, since the guessed permutation must satisfy that these edges do not cross. Then, suppose $e_1 \in \mathsf{Sep}$ and $e_2 \in \mathsf{CrossingEdges}$. Then in each of the (at most) two sub-instances, the edge e_2 is replaced by a multi-edge e_2' incident to one of the endpoints of e. Thus, in any drawing of the respective sub-instance, the edge e_1 and e_2' cannot cross. [11] Finally, consider the case when $e_1, e_2 \in \mathsf{CrossingEdges}$. Since e_1, e_2 cross, we have two cases: (a) $e_1 \in \mathsf{cross}(e_1')$ and $e_2 \in \mathsf{cross}(e_2')$ for distinct edges $e_1', e_2' \in \mathsf{Sep}$, or (b) $e_1, e_2 \in \mathsf{cross}(e)$ for some $e \in \mathsf{Sep}$. In case (a), the multi-edges replacing e_1 and e_2 belong to completely different sub-instances (note that we do not have a middle sub-instance). Thus, we are left with case (b), when they cross the same edge in the separator. Then,

[11] Note that in different sub-instances, the multi-edge e_2' that replaces e_2 is different; however, in each of the sub-instances, it holds that e_1 cannot cross with any of the replacing e_2'.

in each of the corresponding sub-instance, there is a multi-edge e'_1 that replaces e_1, and a multi-edge e'_2 that replaces e_2. If e'_1 and e'_2 are incident to the same endpoint of e, then they cannot cross. Alternatively, if e'_1 and e'_2 are incident to different endpoints of e, then due to Item 4 in Definition 3, we do not count such a crossing. In any case, we have that the crossing between $e, e' \in \mathsf{CrossingEdges} \cup \mathsf{MiddleEdges} \cup \mathsf{Sep}$ is not counted in any of the sub-instances.

Next, we note that in $\mathcal{I}^{\mathcal{L}}$, the left boundary edge $\mathsf{leftbound}$ (and left extended boundary $\mathsf{LeftExtBnd}$, in the case of ♣ elaborate instance) is derived from the current instance $\mathcal{I}$, and thus are placed at the appropriate locations. An analogous property holds about $\mathcal{I}^{\mathcal{R}}$. Next, we note that the orderings $\pi^{\mathcal{L}}, \pi^{\mathcal{R}}$ is obtained by the projecting π' (which is an extension of π) to the respective vertex subsets. Since the respective drawings are compatible with these orders, it follows that the combined drawing is compatible with π', and hence with π. Furthermore, since π' also contains the orderings $\pi^{\lambda+1}$, the respective drawings also satisfy the ordering on the vertices of $\mathsf{LeftExtBnd} \cup \mathsf{Endpoints}(\mathsf{Sep}, 1)$.

Finally, since $\sigma^{\mathcal{L}}, \sigma^{\mathcal{R}}$ are valid drawings for $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}$ respectively, when we obtain the combined drawing σ' by gluing the three drawings, it is straightforward to verify that Definition 3 are satisfied. This shows that σ' is a valid drawing for $\mathcal{I}$ with k crossings. □

4.7 Proof of Correctness

Lemma 8. *$\mathcal{I}$ is an* Yes*-instance for parameter k iff the algorithm returns a drawing σ' of $\mathcal{I}$ with at most k crossings.*

Proof. In the base case when $k \leq c\sqrt{k^\star}$, the correctness of the algorithm follows via Lemma 1. Thus, we consider the recursive case. Suppose inductively that the statement is true for all values of parameter strictly smaller than $k > c\sqrt{k^\star}$, and we want to show the statement holds for k.

Forward direction. Suppose $\mathcal{I}$ is a k-minimal Yes-instance, and let σ be a drawing of $\mathcal{I}$ with k crossings. Consider the guess where the following objects are correctly guessed in the algorithm:

1. Separator $\mathsf{Sep}(\sigma)$ as guaranteed by Lemma 2,
2. Sets $\mathsf{CrossingEdges}(\mathsf{Sep})$ and $\mathsf{MiddleEdges}(\mathsf{Sep})$ and the relative ordering between the corresponding vertices is guessed according to σ.
3. For each $v \in \mathsf{Endpoints}(\mathsf{Sep})$, by Lemma 3, $|\mathsf{UpDown}(v)| \leq 4\sqrt{k}$. Then, for each component $C \in \mathsf{UpDown}(v)$, if all the vertices of $C \cap V_i$ belong to the left of $\mathsf{endpt}(\mathsf{lsep}, i)$ (resp. to the right of $\mathsf{endpt}(\mathsf{rsep}, i)$) for each $i \in [2]$, then consider the guess where it is labeled $\mathcal{L}$ (resp. $\mathcal{R}$).

Now we distinguish between three cases, namely DT, SB, ST.

▶ DE case. Here, all components of $\mathsf{Pendants}(v), v \in \mathsf{Endpoints}(\mathsf{Sep})$ behave according to Lemma 4, which is how the algorithm labels the respective components.

▶ SB case. In this case, all components of $\mathsf{Pendants}(\mathsf{endpt}(\mathsf{lsep}, 2)) \cup \mathsf{Pendants}(\mathsf{endpt}(\mathsf{rsep}, 2))$ behave according to Lemma 4, and the algorithm also labels such components accordingly.
Now, we are left with labeling the components of $\mathsf{Pendants}(\mathsf{endpt}(\mathsf{lsep}, 1))$. In this case, let $(\mathsf{LeftNum}, \mathsf{RightNum})$ be the pair of integers guaranteed by Lemma 5, and consider the guess corresponding to $(\mathsf{LeftNum}, \mathsf{RightNum})$. In this case, Lemma 5 implies that a $(\mathsf{LeftNum}, \mathsf{RightNum})$-partition of $\mathsf{Pendants}(v)$ exists, and one such partition, say $(\mathsf{LeftPendants}, \mathsf{RightPendants})$ will be found by the algorithm. Lemma 5 implies that there exists a drawing σ' in the family $\mathsf{NiceFamily}(\sigma, \mathsf{endpt}(\mathsf{lsep}, 1), \mathsf{LeftNum}, \mathsf{RightNum}, \mathsf{LeftPendants}, \mathsf{RightPendants})$, w.r.t. which $\mathsf{Sep}(\sigma)$ is still a separator, and furthermore, σ' is compatible with all the guesses made so far. Henceforth, we consider this guess, and use σ' instead of σ in the subsequent analysis. For ease of notation, let us rename $\sigma \leftarrow \sigma'$.

▶ ST case. This case is handled in a symmetric manner as SB case, and we omit the description.

Let σ be the drawing obtained at the end after appropriate redefintion (if any), as described above. In the following claim, we show that the labeling process correctly labels all the vertices w.r.t. the (new) drawing σ.

Claim 4.2. *Let σ be the (new) drawing as described above, and consider the guess as defined above. Then, after the vertex labeling process of step H corresponding to this particular guess, we have that $V^{\mathcal{T}} = V(\sigma, \mathsf{Sep}, \mathcal{T})$, for each $\mathcal{T} \in \{\mathcal{L}, \mathcal{M}, \mathcal{R}\}$. Note that the definition of the sets $V(\sigma, \mathsf{Sep}, \mathcal{T})$ can be found in Definition 9.*

Proof of Claim 4.2. We consider ♣ elaborate instances first, since the proof for the ♦ normal instances is only simpler.

Let us consider a vertex $u \in V(\sigma, \mathsf{Sep}, \mathcal{L})$. By of Definition 14, u is reachable from some vertex $v \in \mathsf{ExtBnd} \cup \mathsf{Endpoints}(\mathsf{BoundaryEdges})$, say via a path $P(uv)$. We consider different cases based on v and a path $P(u \rightsquigarrow v)$. Let $P(uv) = \langle u = w_0, w_1, w_2, \ldots, w_\ell = v \rangle$. Let w_{i+1} be the first vertex (if any) along $P(u \rightsquigarrow v)$, such $w_0, w_1, \ldots, w_i \in V(\sigma, \mathcal{L})$, and $w_{i+1} \notin V(\sigma, \mathcal{L})$. Otherwise, let $w_i = w_\ell = v$. We consider these two cases separately; see Figure 16.

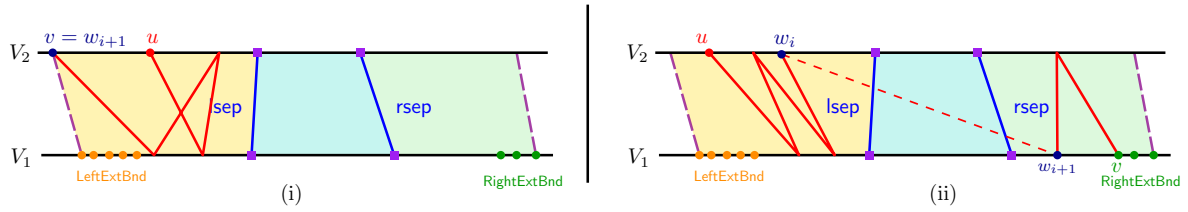

Figure 16: Illustration of the two cases in Claim 4.2. In case (i) on the left, the entire path contains vertices of $V(\sigma, \mathcal{L})$, and the path ends at vertex $v \in \mathsf{Origin}^{\mathcal{L}}$. In case (ii) on the right, the path goes outside $V(\sigma, \mathsf{Sep}, \mathcal{L})$; however to do so, it must contain a crossing edge (shown as dashed red), in which case its left endpoint $w_i \in \mathsf{CrossLeft} \subseteq \mathsf{Origin}^{\mathcal{L}}$.

(i) If $w_i = v$, then since $w_i \in V(\sigma, \mathcal{L})$, it cannot be the case that $w_i \in (\mathsf{RightExtBnd} \cup \mathsf{Endpoints}(\mathsf{rightbound})) \setminus \mathsf{Endpoints}(\mathsf{Sep})$. Thus, it implies that $w_i \in \mathsf{LeftExtBnd} \cup \mathsf{Endpoints}(\mathsf{leftbound}) \cup \mathsf{CrossLeft} = \mathsf{Origin}^{\mathcal{L}}$.

(ii) If $w_i \neq v$, then we know that w_{i+1}, the vertex immediately after w_i in $P(uv)$ satisfies that $w_{i+1} \notin V(\sigma, \mathcal{L})$. Then, it follows that the edge (w_i, w_{i+1}) crosses some edge of Sep–implying that $w_i \in \mathsf{CrossLeft}$.

This implies that, for any vertex $u \in V(\sigma, \mathsf{Sep}, \mathcal{L}) \setminus \mathsf{Origin}^{\mathcal{L}}$, and any path $P(u \rightsquigarrow v)$ to a vertex $v \in \mathsf{ExtBnd} \cup \mathsf{Endpoints}(\mathsf{BoundaryEdges})$, the last vertex w with label $\mathcal{L}$ must belong to the set $\mathsf{Origin}^{\mathcal{L}}$. Further, the subpath $P(u \rightsquigarrow w)$ does not contain any vertex with a label $\mathcal{M}$ or $\mathcal{R}$. This also implies that, all the vertices of $V(\sigma, \mathcal{L})$ are reachable from some vertex $w' \in \mathsf{Origin}^{\mathcal{L}}$ in the graph $G - (O^{\mathcal{M}} \cup O^{\mathcal{R}})$. Thus, all the unlabeled vertices in $V(\sigma, \mathcal{L})$ get labeled $\mathcal{L}$ in the label propagation phase. This shows that $V(\sigma, \mathcal{L}) \subseteq V^{\mathcal{L}}$. Similar proofs also show that $V(\sigma, \mathsf{Sep}, \mathcal{M}) \subseteq V^{\mathcal{M}}$, and $V(\sigma, \mathsf{Sep}, \mathcal{R}) \subseteq V^{\mathcal{R}}$. Now, consider a vertex of $u' \in (V(\sigma, \mathsf{Sep}, \mathcal{M}) \cup V(\sigma, \mathsf{Sep}, \mathcal{R})) \setminus \mathsf{Origin}^{\mathcal{L}}$. However, such a vertex u' belongs to $O^{\mathcal{M}} \cup O^{\mathcal{R}}$, and hence is not part of the graph $G - (O^{\mathcal{M}} \cup O^{\mathcal{R}})$. Thus, it cannot receive label $\mathcal{L}$, which shows that $V^{\mathcal{L}} \subseteq V(\sigma, \mathsf{Sep}, \mathcal{L})$. By combining both containments, we obtain that $V^{\mathcal{L}} = V(\sigma, \mathsf{Sep}, \mathcal{L})$. Similar proofs also show that $V^{\mathcal{M}} = V(\sigma, \mathsf{Sep}, \mathcal{M})$, and $V^{\mathcal{R}} = V(\sigma, \mathsf{Sep}, \mathcal{R})$, and are therefore omitted.

The proof for ♦ normal instances is simpler, since the graph G is connected. Therefore, each vertex u is reachable from any other vertex, and we can select an arbitrary vertex to play the part of v in the proof above. □

After accounting for the k_{current} crossings among the endpoints of $\mathsf{CrossingEdges} \cup \mathsf{MiddleEdges} \cup \mathsf{Sep}$ in σ, the rest of the k' crossings are either to the left of lsep or to the right of rsep (this argument is similar to Lemma 7). Let the number of these crossings be $k^{\mathcal{L}}, k^{\mathcal{M}}$, respectively, and consider the

guess where these numbers are correctly guessed. Further, since we label all the vertices correctly, it follows that all the vertices of $V^{\mathcal{L}}$ (resp. $V^{\mathcal{R}}$) are reachable from $\mathsf{Origin}^{\mathcal{L}}$ (resp. $\mathsf{Origin}^{\mathcal{R}}$).

♦ For normal extended instances, we note that any $u \rightsquigarrow v$ path, where $u, v \in V^{\mathcal{L}}$, must either be completely contained inside $G'^{\mathcal{L}} = G[V^{\mathcal{L}}]$, use an edge of $\mathsf{cross}(\mathsf{lsep}) \cup \mathsf{lsep}$. In any case, due to the way we add extra edges to $\mathsf{WeightedEdges}^{\mathcal{L}}$, it follows that u and v are also reachable in $G^{\mathcal{L}}$.

♣ Now, consider elaborate extended instances. Then, we observe that vertices of $\mathsf{CrossLeft}$ (resp. $\mathsf{CrossRight}$) are directly connected to one of the vertices of $\mathsf{Endpoints}(\{\mathsf{lsep}\})$ (resp. $\mathsf{Endpoints}(\{\mathsf{rsep}\})$) in the graph $G^{\mathcal{L}}$ (resp. $G^{\mathcal{R}}$).

In either case, we have shown that these guesses are not terminated by the connectivity check done in Step 3 after creating the instances $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}$, respectively. Then, Claim 4.1 implies that $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}$ are valid extended instances. Further, σ restricted to the respective vertex sets $V^{\mathcal{L}}, V^{\mathcal{R}}$ gives a valid drawing of the respective instances. This implies that $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}$ are YES-instances for the parameters $k^{\mathcal{L}}, k^{\mathcal{R}}$, respectively.

Next, we claim that, since $\mathcal{I}$ is a k-minimal instance, both of these instances must also be minimal instances for the respective parameters. suppose for contradiction that $\mathcal{I}^{\mathcal{L}}$ is a YES-instance, but is not $k^{\mathcal{L}}$-minimal YES-instance, then it admits a drawing $(\sigma')^{\mathcal{L}}$ with $(k')^{\mathcal{L}} < k^{\mathcal{L}}$ crossings. Then, we can apply an argument from Lemma 7, in order to obtain a drawing for $\mathcal{I}$ with fewer than k crossings, contradicting the k-minimality of $\mathcal{I}$. A similar argument also holds for $\mathcal{I}^{\mathcal{R}}$.

Then, by induction (since $k^{\mathcal{L}}, k^{\mathcal{R}} \leq k/2$) the respective recursive calls return valid drawings $\sigma^{\mathcal{L}}, \sigma^{\mathcal{R}}$ of the respective instances. Then, Lemma 7 implies that that in Step 5, we return a combined drawing σ' corresponding to this guess.

Reverse direction. The reverse direction is trivial, since the valid drawing σ' of $\mathcal{I}$ witnesses that $\mathcal{I}$ is a YES-instance.

□

In the following lemma, we argue that the algorithm runs in subexponential time.

Lemma 9. *For any input extended instance $\mathcal{I}$ with parameter k, the algorithm runs in time $T_2(k) \leq n^{\mathcal{O}(\sqrt{k})}$.*

Proof. The base case corresponds to when $k \leq c\sqrt{k^\star}$, for some constant c. In this case, Lemma 1 implies that the algorithm runs in time $n^{\mathcal{O}(\sqrt{k})}$.

Otherwise, consider the recursive case, and suppose the claim is true for all $k' \leq k-1$, where $k > c\sqrt{k^\star}$ is the parameter of the given instance $\mathcal{I}$. Lemma 6 implies that the total number of guesses is bounded by $n^{\mathcal{O}(\sqrt{k})}$, and corresponding to each guess that is not terminated, we make two recursive calls to the algorithm with parameters at most $k/2$ each. Thus, $T_2(n,k)$ satisfies the recurrence given in Equation (1). Then, the claim follows via induction. □

By combining Observation 2, Lemma 8, Lemma 9, and the fact that due to linear-time kernelization algorithm of Proposition 3, we have $n = (k^\star)^{\mathcal{O}(1)}$, we obtain the following theorem (where we rename $k^\star$ as k).

Theorem 6. *On an n vertex graph,* 2-LAYER CROSSING MINIMIZATION *can be solved in time $2^{\mathcal{O}(\sqrt{k}\log k)} + n \cdot k^{\mathcal{O}(1)}$, where k denotes the number of crossings.*

5 Subexponential Algorithm for 3-Layer Crossing Minimization

In this section, we will design our subexponential FPT algorithm for 3-LAYER CROSSING MINIMIZATION running in time $2^{\mathcal{O}(k^{2/3}\log k)} + nk^{\mathcal{O}(1)}$. The overall structure of this algorithm is similar to that for 2-LAYER CROSSING MINIMIZATION– the algorithm is recursive, and is based on dividing the instance into multiple sub-instances with the help of a separator (consisting of at most 4 edges, and two special subsets of vertices) with at most αk crossings its either side. However, the details

are different in the following two crucial aspects: (i) the structure of the separator is more complex, which necessitates handling multiple cases (like DE, SB, ST CASES for 2-LAYER CROSSING MINIMIZATION), and (ii) the analysis of connected components in $G - v$, where v belongs to one of the endpoints of the separator edges, is more involved. Recall that, in the case of two layers, we needed to handle the set of pendant vertices attached to v, where we showed the existence of another, more structured, separator-compatible drawing in Lemma 5. For three layers, this requires a much more sophisticated analysis. Another notable aspect we would like to highlight is that, at various places in the algorithm for three layers, we will use the two layer algorithm as a subroutine.

5.1 Setting Up

First, we adopt the same notation set up in Definition 1 for $\mathsf{endpt}(\cdot), \mathsf{Endpoints}(\cdot)$ and extend it to three layers. Further, in any (extended) instance, we partition the set of edges into $E_{12} \cup E_{23}$, where E_{12} (resp. E_{23}) is the set of edges with one endpoint in V_1 and another in V_2 (resp. one endpoint in V_2 and another in V_3). Next, we describe the notion of an extended instance.

We denote the *original* instance of 3-LAYER CROSSING MINIMIZATION as $(H, U_1, U_2, U_3, k^\star)$. In the course of the recursive algorithm, our algorithm will produce *extended instances* of 3-LAYER CROSSING MINIMIZATION, defined as follows.

Definition 14 (Extended Instance for $h = 3$)**.** An *extended instance* of 3-LAYER CROSSING MINIMIZATION is given by $(G, V_1, V_2, V_3, \mathsf{ExtBnd}, \pi, k_{12}, k_{23}, k)$ where,

1. $V(G) = V_1 \uplus V_2 \uplus V_3$,
2. $E(G) = \mathsf{RegularEdges} \uplus \mathsf{WeightedEdges} \uplus \mathsf{BoundaryEdges}$.
   The set of edges between V_1 and V_2 respectively are denoted by E_{12}, and those between V_2 and V_3 are denoted E_{23}.
3. $\mathsf{ExtBnd} = (\mathsf{LeftExtBnd}, \mathsf{RightExtBnd})$, where
   - $\mathsf{BoundaryEdges} = \{\mathsf{leftbound}_{12}, \mathsf{rightbound}_{12}, \mathsf{leftbound}_{23}, \mathsf{rightbound}_{23}\}$ is a set of at most four *boundary edges* of the instance, where $\mathsf{leftbound}_{12}, \mathsf{rightbound}_{12} \in E_{12}$ (both may be equal), and $\mathsf{leftbound}_{23}, \mathsf{rightbound}_{23} \in E_{23}$ (both may be equal).
   - $\mathsf{LeftExtBnd}, \mathsf{RightExtBnd} \subseteq V_2$ is a set of at most $3\sqrt{k^\star}$ vertices, where
     (i) $\mathsf{endpt}(\mathsf{leftbound}_{12}, 2), \mathsf{endpt}(\mathsf{leftbound}_{23}, 2) \in \mathsf{LeftExtBnd}$, and
     (ii) $\mathsf{endpt}(\mathsf{rightbound}_{12}, 2), \mathsf{endpt}(\mathsf{rightbound}_{23}, 2) \in \mathsf{RightExtBnd}$.
   - Further, each $v \in V(G)$ is reachable from some vertex in $\mathsf{Endpoints}(\mathsf{BoundaryEdges}) \cup \mathsf{LeftExtBnd} \cup \mathsf{RightExtBnd}$.
4. $\pi = (\pi_1, \pi_2, \pi_3)$, where for each $i \in [3]$, π_i is a partial order on V_i such that, π_i is the union of $\lambda \geq 0$ different relations $\pi_i^1, \pi_i^2, \ldots, \pi_i^\lambda$, where for each $1 \leq j \leq \lambda$, π_i^j is a total order on $V_i^j \subseteq V_i$, and the sets $\left\{V_i^j : 1 \leq j \leq \lambda\right\}$ are not necessarily disjoint.[12] Here, λ always remains bounded by $4\log(k^\star)$.
   One of the orderings π_2^j has ground set $V_2^j = \mathsf{ExtBnd}$, and it orders these vertices in the following way from left to right:
   $u_1^l <$ vertices of $\mathsf{LeftExtBnd} \setminus \{u_1^l, u_2^l\}$ in some order $< u_2^l < u_1^r <$ vertices of $\mathsf{RightExtBnd} \setminus \{u_1^r, u_2^r\}$ in some order $< u_2^r$. Here, $\{u_1^l, u_2^l\} = \{\mathsf{endpt}(\mathsf{leftbound}_{12}, 2), \mathsf{endpt}(\mathsf{leftbound}_{23}, 2)\}$, and $\{u_1^r, u_2^r\} = \{\mathsf{endpt}(\mathsf{rightbound}_{12}, 2), \mathsf{endpt}(\mathsf{rightbound}_{23}, 2)\}$.
5. Each $e \in \mathsf{WeightedEdges}$ is referred to as a *multi-edge*, and has an associated weight/multiplicity $\mu(e)$ (which is a non-negative integer).

[12] The fact that each π_i is a union of λ total orders is crucial only at a few specific steps in the algorithm, which is where we will make this notation explicit.

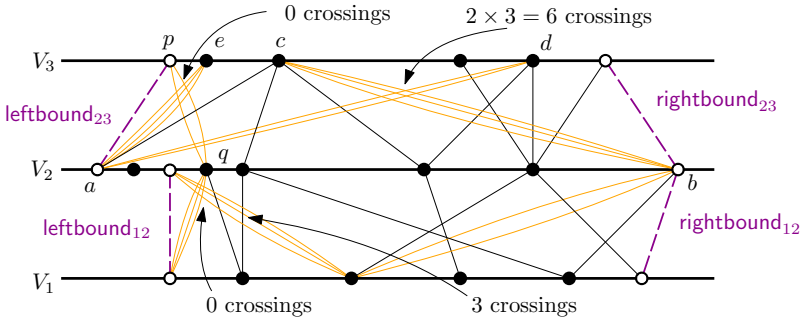

Figure 17: Left: Drawing of an extended instance for $h = 3$. BoundaryEdges are shown as dashed magenta edges. Note that the top right and top right boundary edges share an endpoint. WeightedEdges(shown in orange) are incident only to Endpoints(BoundaryEdges) (shown as unfilled nodes). As explained in Definition 15, crossings between multi-edges ad and bc are counted with multiplicities ($2 \times 3 = 6$), however, those between ae and pq are counted as 0, since these multi-edges are incident to different endpoints of the same boundary edge, namely $\mathsf{leftbound}_{23}$.

6. Each multi-edge $e \in \mathsf{WeightedEdges}$ is incident to exactly one vertex of Endpoints(BoundaryEdges). Further, for each vertex $v \in \mathsf{Endpoints(BoundaryEdges)}$, the total sum of multiplicities of multi-edges incident to v is at most $2\sqrt{k^\star}$.

7. $k = k_{12} + k_{23}$, where k, k_{12}, k_{23} are non-negative integers.

Next, we define the notion of a drawing of an extended instance, along the same lines as Definition 3.

Definition 15 (Drawing of an extended instance.)**.** Let $\mathcal{I} = (G, V_1, V_2, V_3, \pi, k_{12}, k_{23}, k)$ be an extended instance of 3-Layer Crossing Minimization. We say that $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ is a drawing of $\mathcal{I}$ if the following properties are satisfied.

1. σ_i is a permutation of V_i for each $i \in [3]$.

2. Ordering of vertices on respective lines:

   (a) $\sigma_1(\mathsf{endpt}(\mathsf{leftbound}_{12}, 1)) \leq \sigma_1(u_1) \leq \sigma_1(\mathsf{endpt}(\mathsf{rightbound}_{12}, 1))$ for all $u_1 \in V_1$.

   (b) Let $\ell_{12} = \mathsf{endpt}(\mathsf{leftbound}_{12}, 2), \ell_{23} = \mathsf{endpt}(\mathsf{leftbound}_{23}, 2)$, $r_{12} = \mathsf{endpt}(\mathsf{rightbound}_{12}, 2)$, and $r_{23} = \mathsf{endpt}(\mathsf{rightbound}_{23}, 2)$. Then,
   - $\min\{\sigma_2(\ell_{12}), \sigma_2(\ell_{23})\} \leq \sigma_2(u^l) \leq \max\{\sigma_2(\ell_{12}), \sigma_2(\ell_{23})\}$ for all $u^l \in \mathsf{LeftExtBnd}$,
   - $\min\{\sigma_2(r_{12}), \sigma_2(r_{23})\} \leq \sigma_2(u^r) \leq \max\{\sigma_2(r_{12}), \sigma_2(r_{23})\}$ for all $u^r \in \mathsf{RightExtBnd}$,
   - $\min\{\sigma_2(\ell_{12}), \sigma_2(\ell_{23})\} \leq \sigma_2(u) \leq \max\{\sigma_2(r_{12}), \sigma_2(r_{23})\}$ for all $u \in V_2$.

   (c) $\sigma_3(\mathsf{endpt}(\mathsf{leftbound}_{23}, 3)) \leq \sigma_3(u_3) \leq \sigma_3(\mathsf{endpt}(\mathsf{rightbound}_{23}, 3))$ for all $u_3 \in V_3$.

   (d) For each $e \in E_{12}$, for each $i \in [2]$, $\sigma_i(\mathsf{endpt}(\mathsf{leftbound}_{12}, i)) \leq \sigma_i(\mathsf{endpt}(e, i)) \leq \sigma_i(\mathsf{endpt}(\mathsf{rightbound}_{12}, i)$

   (e) For each $e \in E_{23}$, for each $j \in \{2, 3\}$, $\sigma_j(\mathsf{endpt}(\mathsf{leftbound}_{23}, j)) \leq \sigma_j(\mathsf{endpt}(e, j)) \leq \sigma_j(\mathsf{endpt}(\mathsf{rightbound}_{23}, j))$.

3. For each $u, v \in V_i$, if $\pi_i(u) < \pi_i(v)$, then $\sigma_i(u) < \sigma_i(v)$, i.e., σ_i is compatible with π_i.

Further, we count the crossings in the drawing σ in the following way:

- Constraints on σ imply that no edge of BoundaryEdges is crossed by any other edge.
- For any pair of edges $e_1, e_2 \in \mathsf{RegularEdges}$, if they cross according to σ (in the usual sense), then we count it as a single crossing.

- If an edge $e_r \in \mathsf{RegularEdges}$ and an edge $e_m \in \mathsf{WeightedEdges}$ with multiplicity $\mu(e_m)$ cross in σ (in the usual sense), then we count it as $\mu(e_m)$ crossings.
- Let $e_1, e_2 \in \mathsf{RegularEdges}$ be two multi-edges.
  - If e_1, e_2 are incident to different endpoints of the same edge $e \in \mathsf{BoundaryEdges}$, then they will necessarily cross in any drawing σ, that satisfies the constraints above. However we count zero crossings in σ. [13]
  - Otherwise, if e_1, e_2 cross in σ (in the usual sense), then we count as $\mu(e_1) \cdot \mu(e_2)$ crossings.

From the original instance to extended instance. Given the original instance $\mathcal{I}^o = (H, U_1, U_2, U_3, k^\star)$ of 3-Layer Crossing Minimization. First, we try each value of $0 \leq k \leq k^\star$ (these are $k+1$ guesses). Next, we also guess the partition $k = k_{12} + k_{23}$, such that if $\mathcal{I}$ is a Yes-instance, then it admits a drawing σ with k crossings, then there are k_{12} crossings among the edges of E_{12} and k_{23} crossings among the edges of E_{23}. There are $k \leq k^\star$ guesses for this. Next, we guess the leftmost and rightmost vertices in U_1 and U_3 in σ, for which there are at most 4 choices, and hence the number of guesses is at most n^4. Fix one such guess, and let us call the leftmost (resp. rightmost) vertices in U_1 and U_3 as u_1^l, u_3^l (resp. u_1^r, u_3^r), respectively. Now, we add two new vertices u_2^l, u_2^r, and add edges $(u_1^l, u_2^l), (u_2^l, u_3^l), (u_1^r, u_2^r), (u_2^r, u_3^r)$, respectively, and denote them as $\mathsf{leftbound}_{12}, \mathsf{leftbound}_{23}, \mathsf{rightbound}_{12}, \mathsf{rightbound}_{23}$, respectively. By slightly abusing the notation, we continue to use U_i for the new sets of vertices. Note that we have added 2 additional vertices and 4 additional edges, which only increases the size of the instance by $\mathcal{O}(1)$. Now, it can be easily observed that, given a drawing for the original instance $\mathcal{I}^o$ that satisfies $\pi_i(u_i^l) \leq \pi_i(u) \leq \pi_i(u_i^r)$ for $i \in \{1, 3\}$ (if the guess is correct, then σ is such a drawing), one can obtain a new drawing of the new instance by placing u_2^l at the extreme left on V_2 and u_2^r on the extreme right on V_2. Further, this does not create any additional crossings. Thus, the following observation is immediate.

Observation 6. $\mathcal{I}^o$ is a Yes-instance iff at least one of the at most $n^{\mathcal{O}(1)}$ extended instances $\mathcal{I}$ obtained after this guessing is an extended Yes-instance.

Henceforth, we may drop the qualifier *extended* from *extended instance*, and simply say *instance*/Yes-*instance*, respectively. Further, the graph G in an instance $\mathcal{I} = (G, V_1, V_2, V_3, \pi, k, k_{12}, k_{23})$ is said to be the *underlying graph* of the instance $\mathcal{I}$, and k is said to be its parameter.

Next, we continue to adopt the notation of $\mathsf{cross}(\cdot)$ and that of red and blue edges from Definition 4 and Definition 5. Further, note that Observation 3 continues to hold.

5.2 Algorithm

We use n and m to denote the number of vertices and edges, respectively. [14] Next, we transform it into an extended instance, as described above. This increases the number of vertices and edges by at most 4 each, and from Observation 6 it follows that the kernelized instance is a Yes-instance iff for at least one of the guesses, we have an Yes-instance. For each extended instance corresponding to each of the guesses, we will run the following algorithm.

Let $\mathcal{I} = (G, V_1, V_2, V_3, \pi, k, k_{12}, k_{23})$ be the extended instance given as an input. Let c be a sufficiently large constant. In two special cases, our algorithm directly solves the given instance by a careful enumeration. These two cases correspond to, when (i) $k \leq c\sqrt{k^\star}$, or (ii) $\min\{|E_{12}|, |E_{23}|\} \leq c\sqrt{k^\star}$, respectively. We first explain these two base cases. Otherwise, when $\min\{|E_{12}|, |E_{23}|\} > c\sqrt{k^\star}$, and $k > c\sqrt{k^\star}$, we are in the recursive case, explained later.

[13] These crossings will be already accounted for while creating the extended instance.

[14] Note that if we start with the kernelized instance, then n and m are polynomial in the original parameter $k^\star$; however we will only use the kernelization to derive our final theorem.

Base case. We have the following sub-cases.

Case 1. $k < c\sqrt{(k^\star)}$**.** This base case is handled exactly as in 2-Layer Crossing Minimization via Proposition 2 ([3]), which also holds for three layers.

Case 2. $\min\{|E_{12}|, |E_{23}|\} < c\sqrt{k^\star}$**.** [15] Note here that we count the edges without multiplicity similar to the analogous base case for 2-Layer Crossing Minimization. Now, suppose w.l.o.g. that $|E_{23}| < c\sqrt{k^\star}$ (the other case is exactly analogous). Let $V_2' \subseteq V_2$ be the set of all the vertices defined as follows:

$$V_2' \coloneqq \{u \in V_2 : u \text{ has a neighbor in } V_3\} \cup \mathsf{ExtBnd}.$$

We know that $|V_3| \leq c\sqrt{k^\star}$ (note that each vertex of V_3 must have at least one edge incident to it), and $|V_2'| \leq c\sqrt{k^\star} + \alpha\sqrt{k^\star}$. We guess permutations ζ_2, ζ_3 of V_2', V_3 for which there are $(k^\star)^{\mathcal{O}(\sqrt{k^\star})}$ choices. We terminate the guess for any choice that violates the conditions of a drawing of $\mathcal{I}$, given in Definition 15, specifically, this includes the case when the number of crossings among the edges of E_{23} according to ζ is not equal to k_{23}. Otherwise, for each non-terminated guess, we create an instance $\mathcal{I}' = (G', W_1, W_2, \mathsf{ExtBnd}', \pi', k')$ of 2-Layer Crossing Minimization as follows.

- $W_1 = V_1$ and $W_2 \coloneqq \{u \in V_2 : u \text{ has a neighbor in } V_1\} \cup \mathsf{ExtBnd}$.
- $\mathsf{ExtBnd}' = \mathsf{ExtBnd} = \mathsf{LeftExtBnd}' \cup \mathsf{RightExtBnd}'$, where $\mathsf{LeftExtBnd}' = \mathsf{LeftExtBnd}, \mathsf{RightExtBnd}' = \mathsf{RightExtBnd}$.
- $\mathsf{BoundaryEdges}' = \{\mathsf{leftbound}' \cup \mathsf{rightbound}'\}$, where $\mathsf{leftbound}' = \mathsf{leftbound}_{23}, \mathsf{rightbound}' = \mathsf{rightbound}_{23}$.
- $k' = k_{12}$
- $\pi' = (\pi_1', \pi_2')$, where $\pi_1' = \pi_1$, and $\pi_2' = \pi_2 \cup \zeta_2$.

It is straightforward to check that the instance $\mathcal{I}'$ is a valid elaborate extended instance of 2-Layer Crossing Minimization as per Definition 2. Then, we solve this instance using Theorem 6 (without kernelization) and find a drawing if it is a Yes-instance. Each such call takes time $n^{\mathcal{O}(\sqrt{k})} = n^{\mathcal{O}(\sqrt{k^\star})}$, and there are at most $(k^\star)^{\mathcal{O}(\sqrt{k^\star})}$ recursive calls. Thus, the overall time taken in this base case is at most $n^{\mathcal{O}(\sqrt{k^\star})}$.

If for some instance $\mathcal{I}'$, the algorithm returns a drawing $\varsigma' = (\varsigma_1', \varsigma_2')$ of $\mathcal{I}'$, then we can obtain a combined drawing $\sigma' = (\sigma_1', \sigma_2', \sigma_3')$ of $\mathcal{I}$ as follows: $\sigma_1' = \varsigma_1'$, $\sigma_2' = \zeta_2 \cup \varsigma_2'$, $\sigma_3' = \zeta_3$. It is straightforward to check that σ' is a valid drawing of $\mathcal{I}$ with k crossings. Otherwise, if all guesses are terminated, or the two-layer algorithm returns No, then we conclude that $\mathcal{I}$ is a No-instance. This finishes the description of the second base case.

Lemma 10. *Let $\mathcal{I}$ be an extended instance of* 3-Layer Crossing Minimization *with underlying graph G and parameter k. If either (i) $k \leq c\sqrt{(k^\star)}$, or (ii) $\min\{|E_{12}|, |E_{23}|\} \leq c\sqrt{k^\star}$. Then, the base case of the algorithm as described above returns a drawing σ of $\mathcal{I}$ with k crossings in time $n^{\mathcal{O}(\sqrt{k^\star})}$ iff $\mathcal{I}$ is an extended* Yes*-instance.*

Otherwise, we proceed to the recursive case, explained next.

[15] This base case very closely resembles the way we solve the "middle" subproblem in the recursive case. In fact, we could create an extended instance corresponding to the middle subproblem and make a recursive call, which will immediately trigger this base case. However, we explicitly give the reduction to the two-layer case in the middle subproblem for the ease of presentation.

Recursive case. Since the base case is not applicable, we know that $k > c\sqrt{k^\star}$, and $\min\{|E_{12}|, |E_{23}|\} > c\sqrt{k^\star}$ for some sufficiently large constant c. In this case, at a high level, our recursive algorithm consists of the following steps.

1. **Divide.** First, we prove some combinatorial properties of a "nice separator" in this step, which is defined by a set of at most 4 edges, and $\mathcal{O}(\sqrt{k})$ additional relevant vertices and edges. This separator divides the instance into at most three instances having certain properties. We refer to these sub-instances as *left, middle*, and *right* sub-instances. Note that, of these, the left and right sub-instances will also be extended instances.

2. **Guess the interfaces.** In this step, we actually guess various components of the separator (whose existence is guaranteed in the previous step, assuming that we are dealing with a YES-instance), other relevant information about the separator, and the rest of the interface between the sub-instances, i.e., the vertices and edges that span across multiple sub-instances.
   Due to the properties of the separator, this will constitute $n^{\mathcal{O}(k^{2/3})}$ guesses.

3. **Creating sub-instances.** Formally defining the sub-instances based on the choices guessed so far.
   The construction of each sub-instance takes $k^{\mathcal{O}(1)}$ time.

4. **Conquer left and right.** Recursively solving the left and right instances.
   Inductively, this takes time $n^{\mathcal{O}((3k/4)^{2/3})}$.

5. **Conquer middle.** If, for a particular guess, both sub-instances are found to be YES-instances, then we will use the solutions for left and right sub-problems, plus some $n^{\mathcal{O}(\sqrt{k})}$ guesses to reduce the problem to 2-LAYER CROSSING MINIMIZATION, which can be solved in time $n^{\mathcal{O}(\sqrt{k})}$ using the algorithm from the previous section (Theorem 6).

6. **Obtaining a combined drawing.** Assuming that for some guess, both the recursive calls corresponding to left and right instances output the corresponding drawings, and we find a "compatible" drawing for the middle subproblem via Theorem 6, then we combine the drawings along with the guesses made for the middle subproblem, in order to obtain a combined drawing for the current sub-instance. This step takes polynomial time for each guess.

Let $T_3(n, k)$ denote an upper bound on the running time of our algorithm on an extended instance with a graph with at most n vertices and with parameter bounded by k. From the above description, it follows that $T_3(n, k)$ satisfies the following recurrence.

$$T_3(n,k) \leq \begin{cases} n^{\mathcal{O}(k^{2/3})} \cdot (T_3(n, \frac{3k}{4}) + T_2(n,k)) + n^{\mathcal{O}(1)} & \text{if } k > c\sqrt{k^\star} \text{ and } |E| > c\sqrt{k^\star} \\ n^{\mathcal{O}(\sqrt{k^\star})} & \text{if } k \leq c\sqrt{k^\star} \text{ or } |E| \leq c\sqrt{k^\star} \end{cases} \tag{2}$$

Here, $T_2(n,k) = n^{\mathcal{O}(\sqrt{k})}$ is the running time of the algorithm for 2-LAYER CROSSING MINIMIZATION when we directly use the algorithm of Theorem 6 without the kernelization step. Then, it can be shown that this recurrence solves to $T_3(n, k^\star) = n^{\mathcal{O}((k^\star)^{2/3})}$.

5.3 Step 1: Divide.

This step is subdivided into two steps. In Section 5.3.1, we will show the existence of a "balanced separator" and explore its various properties. Then, in Section 5.3.2, we will show a technical claim that lets us obtain a more structured drawing (for a YES-instance) that is still compatible with the separator from the previous step. Now, we explore each of these steps in more detail.

5.3.1 Existence and Properties of a Separator

Definition 16 (Bottom- and Top-heavy instances)**.** If, in the given instance we have that $k_{12} \geq k_{23}$, i.e., $k_{12} \geq \frac{k}{2}$, then say that such an instance is *bottom-heavy*; and otherwise if $k_{23} < k_{12}$, then we say that an instance is *top-heavy*.

For the rest of this section while describing the algorithm, we will assume that we are dealing with a bottom-heavy instance, without repeating this assumption at each step. Specifically, this assumption is crucial while defining the structure of the separator defined in this subsection, which follows. Otherwise, the top-heavy instances can be dealt with interchanging the roles of the first and third lines in the definition of separator (Definition 17) and the rest of the algorithm.

We adopt and naturally extend the notions of edge orderings from Definition 6 as well as that of an ordering between an edge and a crossing from Definition 7 to 3-Layer Crossing Minimization. We note that an analogous version of Observation 4 holds in 3-Layer Crossing Minimization as well.

Definition 17. Let $\mathcal{I}$ be a k-minimal bottom-heavy Yes-instance, and let σ be any drawing of $\mathcal{I}$ with k crossings. We say that $\mathsf{Sep} = \{\mathsf{lsep}_{12}, \mathsf{rsep}_{12}, \mathsf{lsep}_{23}, \mathsf{rsep}_{23}\}$ is a *separator* w.r.t. σ, if the following properties hold (see Figure 18 for an example of Sep and associated objects.).

1. $\mathsf{lsep}_{12}, \mathsf{rsep}_{12} \in E_{12}$ are two distinct blue edges w.r.t. σ and they do not cross each other
   $\mathsf{lsep}_{23}, \mathsf{rsep}_{23} \in E_{23}$ are two distinct blue edges w.r.t. σ and they do not cross each other.
2. $\sigma_1(\mathsf{endpt}(\mathsf{lsep}_{12}), 1) \leq \sigma_1(\mathsf{endpt}(\mathsf{rsep}_{12}), 1)$,
   $\sigma_2(\mathsf{endpt}(\mathsf{lsep}_{23}), 2) \leq \sigma_2(\mathsf{endpt}(\mathsf{lsep}_{12}, 2)) \leq \sigma_2(\mathsf{endpt}(\mathsf{rsep}_{12}, 2)) \leq \sigma_2(\mathsf{endpt}(\mathsf{rsep}_{12}, 2))$,
   $\sigma_3(\mathsf{endpt}(\mathsf{lsep}_{23}), 3) \leq \sigma_3(\mathsf{endpt}(\mathsf{rsep}_{23}), 3)$.
3. The number of crossings to the left of lsep_{12}, and that to the right of rsep_{12} is at most $\frac{k_{12}}{2}$.
4. Let $\mathsf{MiddleEdges}(\mathsf{Sep}, \sigma)$ denote the set of all edges $u_1u_2 \in E_{12}$ other than $\mathsf{lsep}_{12}, \mathsf{rsep}_{12}$, such that
   $$\sigma_i(\mathsf{endpt}(\mathsf{lsep}_{12}), i) \leq \sigma_i(u_i) \leq \sigma_i(\mathsf{endpt}(\mathsf{rsep}_{12}), i) \text{ for } i \in [2].$$
   Then $|\mathsf{MiddleEdges}(\mathsf{Sep}, \sigma)| \leq 2\sqrt{k}$.
5. Let $V_2^{conn} \coloneqq \{u \in V_2 : N(u) \cap V_3 \neq \emptyset\}$, and define
   $$\mathsf{GapLeft}(\mathsf{Sep}, \sigma) \coloneqq \{v \in V_2^{conn} : \sigma_2(\mathsf{endpt}(\mathsf{lsep}_{23}), 2) < \sigma_2(v) < \sigma_2(\mathsf{endpt}(\mathsf{lsep}_{12}), 2)\}$$
   $$\mathsf{GapRight}(\mathsf{Sep}, \sigma) \coloneqq \{v \in V_2^{conn} : \sigma_2(\mathsf{endpt}(\mathsf{rsep}_{12}), 2) < \sigma_2(v) < \sigma_2(\mathsf{endpt}(\mathsf{rsep}_{23}), 2)\}$$
   Then, $|\mathsf{GapLeft}(\mathsf{Sep}, \sigma)|, |\mathsf{GapRight}(\mathsf{Sep}, \sigma)| \leq 3\sqrt{k}$.
6. $|\mathsf{CrossingEdges}(\mathsf{Sep}, \sigma)| \leq 4\sqrt{k}$, where $\mathsf{CrossingEdges}(\mathsf{Sep}, \sigma) \coloneqq \bigcup_{e \in \mathsf{Sep}} \mathsf{cross}(e, \sigma)$.

When the separator Sep and/or the drawing σ are obvious from the context, we may simply use $\mathsf{GapLeft}, \mathsf{GapRight}$, $\mathsf{MiddleEdges}$, and $\mathsf{CrossingEdges}$.

We state an immediate consequence of the preceding definition.

Observation 7. Let $\mathcal{I}$ be a k-minimal bottom heavy extended Yes-instance, let σ be a drawing of $\mathcal{I}$ with k crossings, and let Sep be a separator w.r.t. σ. Let k_{12}^l, k_{12}^r denote the number of crossings that are to the left of lsep_{12} and to the right of rsep_{12}, respectively, and k_{23}^l, k_{23}^r denote the number of crossings to the left of lsep_{23} and to the right of rsep_{23}, respectively. Then, $k^l, k^r \leq 3k/4$, where $k^l \coloneqq k_{12}^l + k_{23}^l$ and $k^r \coloneqq k_{12}^r + k_{23}^r$.

Proof. Let $k_{12} = \alpha k$ and $k_{23} = (1-\alpha)k$ for some $\alpha \geq 1/2$, since $k_{12} \geq k/2 \geq k_{23}$. Then, since Sep is a separator, $k_{12}^l \leq k_{12}/2 = \alpha k/2$, and $k_{12}^r \leq k_{23} = (1-\alpha)k$. Therefore, $k_{23}^l + k_{23}^r \leq k(\alpha/2 + 1 - \alpha) = (1 - \alpha/2)k \leq 3k/4$, where the last inequality follows from $\alpha \geq 1/2$. Analogous argument works for k^r. □

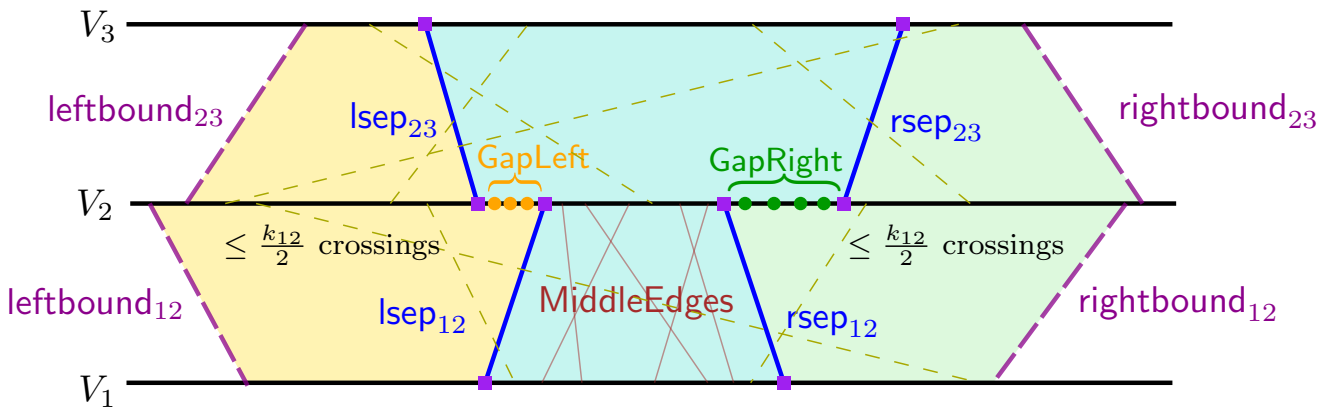

Figure 18: Illustration of a separator from Definition 17. $\mathsf{Sep} = \{\mathsf{lsep}_{12}, \mathsf{rsep}_{12}, \mathsf{lsep}_{23}, \mathsf{rsep}_{23}\}$ are shown in blue, and $\mathsf{Endpoints}(\mathsf{Sep})$ are shown in purple. Edges of $M := \mathsf{MiddleEdges}(\mathsf{Sep})$ shown in brown and that of $\mathsf{CrossingEdges}(\mathsf{Sep})$ are shown in dashed olive – these are the edges that cross at least one edge of Sep. Vertices of GapLeft are shown in orange and that of GapRight in green.

Next, we state the following important lemma that shows the existence of a separator, which forms the basis of our divide-and-conquer algorithm.

Lemma 11 (Separator Lemma for 3-Layer Crossing Minimization)**.** *Let $\mathcal{I}$ be a bottom-heavy k-minimal* Yes*-instance and let σ be any drawing of $\mathcal{I}$ with k crossings. Then, there exists a separator* Sep *w.r.t. σ.*

Proof. Note that due to k-minimality and bottom-heaviness of $\mathcal{I}$, we know that the number of crossings among the edges of E_{12} is larger than $k_{12}/2$; note here that $k_{12}/2 < k_{12}$, since $k_{12} \geq k/2 > c/2$ for some absolute constant c. We divide the proof into four parts: (I) constructing $\mathsf{lsep}_{12}, \mathsf{rsep}_{12}$, (II) bounding the size of MiddleEdges, (III) constructing rsep_{23} and analyzing GapRight, and (IV) constructing lsep_{23} and analyzing GapLeft. The first two parts are exactly identical to the proof of Lemma 2; nevertheless we include them here for the sake of completeness.

Part (I): Constructing $\mathsf{lsep}_{12}, \mathsf{rsep}_{12}$. We define a lexicographic order $\prec$ on the edges of E using the order of their endpoints in V_1 using σ_1, with ties broken using the endpoints in V_2 using σ_2. More formally, for two distinct edges $e_a, e_b \in E$, we have $e_a \prec e_b$ iff $\sigma_1(\mathsf{endpt}(e_a, 1)) < \sigma_1(\mathsf{endpt}(e_b, 1))$, or $\sigma_1(\mathsf{endpt}(e_a, 1)) = \sigma_1(\mathsf{endpt}(e_b, 1))$ and $\sigma_2(\mathsf{endpt}(e_a, 2)) < \sigma_2(\mathsf{endpt}(e_b, 2))$ (multi-edges are treated as a single edge for this order). Let $\mathsf{rsep}_{12} \in E(G)$ be the first blue edge, according to $\prec$, that has more than $\frac{k_{12}}{2}$ crossings to its left. First we claim that rsep_{12} exists, since $\mathsf{rightbound}_{12}$—which is blue, and has all ($k_{12} > \frac{k_{12}}{2}$) crossings to its left— satisfies the stated property.

By the definition of rsep_{12}, the number of crossings to its left is larger than $\frac{k_{12}}{2}$, which implies the following: (a) the number of crossings to its right is at most $\frac{k_{12}}{2} \leq \frac{k}{2}$ (note that some crossings may not be either to the left or to the right of rsep_{12}). (b) the number of edges to its left is at least $\frac{k_{12}}{2} > 0$. See Figure 18 for an illustration. Further, (b) implies that $\mathsf{leftbound}_{12}$ is one of the $\frac{k_{12}}{2}$ edges that is to the left of rsep_{12}, which specifically implies that $\mathsf{leftbound}_{12} \neq \mathsf{rsep}_{12}$.

Now, we define lsep_{12} to be the last edge according to $\prec$ such that: (i) lsep_{12} is blue, (ii) $\mathsf{lsep}_{12} \prec \mathsf{rsep}_{12}$, and (iii) lsep_{12} does not cross rsep_{12} according to σ. First, lsep_{12} exists because $\mathsf{leftbound}_{12}$ satisfies conditions (i) - (iii) – indeed $\mathsf{leftbound}_{12} \neq \mathsf{rsep}_{12}$ as shown above. Further, since $\mathsf{lsep}_{12} \prec \mathsf{rsep}_{12}$, lsep_{12} is to the left of rsep_{12} and in particular the two edges do not cross each other. Now, if the number of crossings to the left of lsep_{12} is larger than $\frac{k_{12}}{2}$, then this contradicts the choice of rsep_{12} as the lexicographically first blue edge with more than $\frac{k_{12}}{2}$ crossings to its left. Therefore, the number of crossings to the left of lsep_{12} is at most $\frac{k_{12}}{2}$. This finishes the first part.

Part (II): Bounding the size of $\mathsf{MiddleEdges}$. Suppose for the contradiction $|\mathsf{MiddleEdges}| > 2\sqrt{k}$. Note that the definition of $\mathsf{MiddleEdges}$, for each edge $e \in \mathsf{MiddleEdges}$, $\mathsf{lsep}_{12} \prec e \prec \mathsf{rsep}_{12}$, and e does not cross rsep_{12}. Then, by Observation 3, $\mathsf{MiddleEdges}$ contains at least one blue edge, say b. However, note that b satisfies conditions (i)–(iii) above, which contradicts the choice of lsep_{12}, since lsep_{12} is the lexicographically last such edge. This shows that $|\mathsf{MiddleEdges(Sep)}| \leq 2\sqrt{k}$.

Part (III): Constructing rsep_{23} and analyzing $\mathsf{GapRight}$. Now we define a similar lexicographic order $\prec'$ on the edges of E_{23}, except first we consider the order of the V_2 endpoints, and then the ties are broken using V_3 endpoints (multi-edges are treated as a single edge for this order).

Let rsep_{23} be a lexicographically first blue edge according to $\prec'$ such that (i) $\mathsf{rsep}_{23} \neq \mathsf{leftbound}_{23}$, and (ii) $\sigma_2(\mathsf{endpt}(\mathsf{rsep}_{12}), 2) \leq \sigma_2(\mathsf{endpt}(\mathsf{rsep}_{23}), 2)$, if it exists. [16] Then, consider the set of vertices $\mathsf{GapRight}$ as defined above. First, if $\mathsf{endpt}(\mathsf{rsep}_{23}, 2) = \mathsf{endpt}(\mathsf{rsep}_{12}, 2)$, then $\mathsf{GapRight} = \emptyset$. Now, consider $\mathsf{endpt}(\mathsf{rsep}_{23}, 2) \neq \mathsf{endpt}(\mathsf{rsep}_{12}, 2)$, and suppose for the sake of contradiction that $|\mathsf{GapRight}| > 2\sqrt{k}$. Let $E^r = \{wz \in E_{23} : w \in \mathsf{GapRight}, z \in V_3\}$. Note that $|E^r| \geq |\mathsf{GapRight}| > 2\sqrt{k}$, since each $w \in \mathsf{GapRight}$ contributes at least one edge to E^r, and these edges are distinct. Then, by Observation 3, E^r contains at least one blue edge b. Note that $b \prec' \mathsf{rsep}_{23}$, since $\sigma_2(\mathsf{endpt}(b, 2)) < \sigma_2(\mathsf{endpt}(\mathsf{rsep}_{23}, 2))$ by construction, which contradicts the choice of rsep_{23} as the lexicographically first such edge. Thus, $|\mathsf{GapRight}| \leq 2\sqrt{k} \leq 3\sqrt{k}$.

Part (IV). Constructing lsep_{23} and analyzing $\mathsf{GapLeft}$. We define lsep_{23} to be the lexicographically last blue edge (if it exists) such that (i) $\sigma_2(\mathsf{endpt}(\mathsf{lsep}_{23}), 2) \geq \sigma_2(\mathsf{endpt}(\mathsf{lsep}_{12}), 2)$, and (ii) lsep_{23} does not cross rsep_{23}. [17] Now consider $\mathsf{GapLeft}$ as defined above. First, if $\mathsf{endpt}(\mathsf{lsep}_{23}, 2) = \mathsf{endpt}(\mathsf{lsep}_{12}, 2)$, then $\mathsf{GapLeft} = \emptyset$. Now, suppose $\mathsf{endpt}(\mathsf{rsep}_{23}, 2) \neq \mathsf{endpt}(\mathsf{rsep}_{12}, 2)$, and suppose for the sake of contradiction that, $|\mathsf{GapLeft}| > 2\sqrt{k}$. Then, let $E^l = \{wz \in E_{23} : w \in \mathsf{GapLeft}, z \in V_3\}$. Note that by a similar argument, $|E^l| \geq |V^l| > 3\sqrt{k}$, since each $w \in V^l$ contributes at least one edge to E^r. Let $E'^r \subseteq E^r$ be the set of edges that do not cross rsep_{23}. Note that $|E'^r| \geq |E^r| - \sqrt{k} > 2\sqrt{k}$, since rsep_{23} is a blue edge. Then, by Observation 3, E'^r contains at least one blue edge, say b'. Further, $\mathsf{lsep}_{23} \prec' b$ since $\sigma_2(\mathsf{endpt}(\mathsf{lsep}_{23}, 2)) < \sigma_2(\mathsf{endpt}(b', 2))$. This contradicts the choice of lsep_{23} as the lexicographically last edge. This shows that $|\mathsf{GapLeft}| \leq 3\sqrt{k}$.

□

Note that a symmetric version of Definition 17 and Lemma 11 holds for top-heavy instances. We omit the definition and the proof.

In the following definition, we divide (not necessarily in disjoint sets) the vertex sets into three parts based on a drawing of an extended instance. These definitions will be used later on during the analysis of the algorithm.

Definition 18. Let $\mathcal{I}$ be a YES-instance with underlying graph G, and let σ be a drawing of $\mathcal{I}$ with k crossings. Consider a separator Sep w.r.t. σ along with the associated objects as in Definition 17. Then, we define the following division of vertices on each layer into left (L), middle (M), and right

[16] Note that, if $\sigma_2(\mathsf{endpt}(\mathsf{rsep}_{12}, 2)) \leq \sigma_2(\mathsf{rightbound}_{23}, 2))$, then $\mathsf{rightbound}_{23}$ satisfies both the conditions (indeed $\mathsf{rightbound}_{23} \neq \mathsf{leftbound}_{23}$ since $|E_{23}| > c\sqrt{k^\star}$). In case rsep_{23} does not exist, then we define rsep_{23} to be a "phantom (placeholder) edge", and $\mathsf{GapLeft} = \emptyset$. When such an edge will be chosen for dividing the problem into sub-problems, the right sub-problem will be a degenerate instance with zero edges in E_{23}, which can be then solved by the second base case. We omit the details.

[17] A degenerate corner case arises when $\mathsf{leftbound}_{23}$ does not satisfy the conditions, and hence lsep_{23} does not exist. This can be handled analogous to the previous case. We omit the details.

(R) parts, as follows.

$$
\begin{aligned}
V_1(\sigma, \mathsf{Sep}, \mathcal{L}) &\coloneqq \{u \in V_1 : \sigma_1(u) \leq \sigma_1(\mathsf{endpt}(\mathsf{lsep}_{12}, 1))\} \\
V_1(\sigma, \mathsf{Sep}, \mathcal{M}) &\coloneqq \{u \in V_1 : \sigma_1(\mathsf{endpt}(\mathsf{lsep}_{12}, 1) \leq \sigma_1(u) \leq \sigma_1(\mathsf{endpt}(\mathsf{rsep}_{12}, 1))\} \\
V_1(\sigma, \mathsf{Sep}, \mathcal{R}) &\coloneqq \{u \in V_1 : \sigma_1(\mathsf{endpt}(\mathsf{rsep}_{12}, 1)) \leq \sigma_1(u)\} \\
V_2(\sigma, \mathsf{Sep}, \mathcal{L}) &\coloneqq \{u \in V_2 : \sigma_2(u) \leq \sigma_2(\mathsf{lsep}_{23}, 2)\} \cup \mathsf{GapLeft} \cup \mathsf{endpt}(\mathsf{lsep}_{12}, 2) \\
V_2(\sigma, \mathsf{Sep}, \mathcal{M}) &\coloneqq \{u \in V_2^{conn} : \sigma_2(\mathsf{endpt}(\mathsf{lsep}_{23}, 2)) \leq \sigma_2(u) \leq \sigma_2(\mathsf{endpt}(\mathsf{lsep}_{12}, 2))\} \\
&\quad \cup \{u \in V_2 : \sigma_2(\mathsf{endpt}(\mathsf{lsep}_{12}, 2)) \leq \sigma_2(u) \leq \sigma_2(\mathsf{endpt}(\mathsf{rsep}_{12}, 2))\} \\
&\quad \cup \{u \in V_2^{conn} : \sigma_2(\mathsf{endpt}(\mathsf{rsep}_{12}, 2)) \leq \sigma_2(u) \leq \sigma_2(\mathsf{endpt}(\mathsf{rsep}_{23}, 2))\} \\
V_2(\sigma, \mathsf{Sep}, \mathcal{R}) &\coloneqq \{u \in V_2 : \sigma_2(\mathsf{endpt}(\mathsf{rsep}_{23}, 2)) \leq \sigma_2(u)\} \cup \mathsf{GapRight} \cup \mathsf{endpt}(\mathsf{rsep}_{12}, 2) \\
V_3(\sigma, \mathsf{Sep}, \mathcal{L}) &\coloneqq \{u \in V_3 : \sigma_3(u) \leq \sigma_3(\mathsf{endpt}(\mathsf{lsep}_{23}, 3))\} \\
V_3(\sigma, \mathsf{Sep}, \mathcal{M}) &\coloneqq \{u \in V_3 : \sigma_3(\mathsf{endpt}(\mathsf{lsep}_{23}, 1) \leq \sigma_3(u) \leq \sigma_3(\mathsf{endpt}(\mathsf{rsep}_{23}, 1))\} \\
V_3(\sigma, \mathsf{Sep}, \mathcal{R}) &\coloneqq \{u \in V_3 : \sigma_3(\mathsf{endpt}(\mathsf{rsep}_{23}, 3)) \leq \sigma_3(u)\}
\end{aligned}
$$

Finally, for each $\mathcal{T} \in \{\mathcal{L}, \mathcal{M}, \mathcal{R}\}$, define $V(\sigma, \mathcal{T}) \coloneqq V_1(\sigma, \mathsf{Sep}, \mathcal{T}) \cup V_2(\sigma, \mathsf{Sep}, \mathcal{T}) \cup V_3(\sigma, \mathsf{Sep}, \mathcal{T})$.

Now, fix a k-minimal YES-instance, a corresponding drawing σ of $\mathcal{I}$ with k crossings, and a separator Sep (and the associated objects from Definition 17) as guaranteed by Lemma 11. In the next part, we classify the components in $G - v$ for $v \in \mathsf{Endpoints}(\mathsf{Sep})$ into multiple types, and then analyze their behavior in σ. This is largely similar to the two layer case; except that the analogue of pendant components for the three layer case is technically much more challenging to analyze, which is the focus of the next subsection.

Definition 19. Let $v \in \mathsf{Endpoints}(\mathsf{Sep})$ be a vertex, and let $\mathsf{comps}(v)$ denote the set of connected components of $C_v - v$, where C_v is the component of G that contains v. We partition these components into different classes (subsets), called $\mathsf{Special}(v)$, $\pi\text{-}\mathsf{Defined}(v)$, $\mathsf{UpDown}(v)$, and $\mathsf{Peculiar}(v)$ (we may omit the subscript G). A component $C \in \mathsf{comps}_G(v)$ belongs to:

- $\mathsf{Special}(v)$, if $E(G[C \cup \{v\}])$ contains an edge of $\mathsf{CrossingEdges}(\mathsf{Sep}) \cup \mathsf{MiddleEdges}(\mathsf{Sep})$, or if C contains a vertex of $\mathsf{GapLeft} \cup \mathsf{GapRight} \cup \mathsf{Endpoints}(\mathsf{Sep}) \setminus \{v\}$.
- $\pi\text{-}\mathsf{Defined}(v)$, if $C \notin \mathsf{Special}(v)$, and C contains a vertex $u \in \bigcup_{i \in [3]} \bigcup_{j \in [\lambda]} V_i^j$ [18]
- $\mathsf{UpDown}(v)$, if it does not belong to $\mathsf{Special}(v) \cup \pi\text{-}\mathsf{Defined}(v)$, and C contains at least one vertex on the same line as v.
- $\mathsf{Peculiar}(v)$, if it does not belong to $\mathsf{Special}(v) \cup \pi\text{-}\mathsf{Defined}(v)$, and C does not contain a vertex on the same layer as v.
  We also define a subset $\mathsf{Pendants}(v) \subseteq \mathsf{Peculiar}(v)$ consisting of components that consist of a single pendant vertex adjacent to v (possibly with a multi-edge).
  Note that if $v \in V_2$, then $\mathsf{Pendants}(v) = \mathsf{Peculiar}(v)$. However, if $v \in V_1 \cup V_3$, then we define $\mathsf{PeculiarNP}(v) \coloneqq \mathsf{Peculiar}(v) \setminus \mathsf{Pendants}(v)$ (peculiar non-pendants), and these are the components that contain vertices of the remaining two layers other than that of v.

First, the following observation follows from the bounds on $\mathsf{CrossingEdges}(\mathsf{Sep})$, $\mathsf{MiddleEdges}(\mathsf{Sep})$, $\mathsf{GapLeft}$, and $\mathsf{GapRight}$.

Observation 8. For any $v \in \mathsf{Endpoints}(\mathsf{Sep})$, it holds that $|\mathsf{Special}(v)| \leq 12\sqrt{k} + 8 = \mathcal{O}(\sqrt{k})$.

Next, we bound the size of UpDown in the following lemma, which is done via arguments almost identical to that in Lemma 3.

[18] Recall that V_i^j's are the λ subsets over which the total orders π_i^j are defined, and these λ total orders constitute π_i.

Lemma 12. *For any $v \in \mathsf{Endpoints}(\mathsf{Sep})$, it holds that $|\mathsf{UpDown}(v)| \leq 8\sqrt{k}$.*

Proof. Consider any $v \in \mathsf{Endpoints}(\mathsf{Sep}) \cap (V_1 \cap V_3)$. Then, one can use arguments exactly as in Lemma 3 to show that, in fact, $|\mathsf{UpDown}(v)| \leq 4\sqrt{k} \leq 8\sqrt{k}$. Now, consider $v \in \mathsf{Endpoints}(\mathsf{Sep}) \cap V_2 = \mathsf{Endpoints}(\mathsf{Sep}, 2)$, and suppose that $|\mathsf{UpDown}(v)| > 8\sqrt{k}$. Then, we classify the components of $\mathsf{UpDown}(v)$ into two types, say (12) and (32), where type 12 components contain a neighbor of v in V_1, and its neighbor in V_2 (it may possibly contain vertices of V_3); whereas type 32 components contain a neighbor of v in V_3 and its neighbor in V_2. If a component satisfies both conditions, then arbitrarily place it in one of them. At least the number of components of either of the two types of components must be larger than $4\sqrt{k}$. Then, one can use an argument similar to that in Lemma 3 to arrive at a contradiction. This proves that for any $v \in \mathsf{Endpoints}(\mathsf{Sep}, 2)$, it holds that $|\mathsf{UpDown}(v)| \leq 8\sqrt{k}$, which finishes the proof. □

To analyze the peculiar components, we again need to consider different cases based on the structure of the separator.

Definition 20. Let $\mathsf{Sep} = \{\mathsf{lsep}_{12}, \mathsf{rsep}_{12}, \mathsf{lsep}_{23}, \mathsf{rsep}_{23}\}$ be the separator as guaranteed by Lemma 11. We have the following different (non-exclusive) cases.

(i) Distinct Bottom (DB)**:** if $\mathsf{endpt}(\mathsf{lsep}_{12}, 1) \neq \mathsf{endpt}(\mathsf{rsep}_{12}, 1)$.
Shared Bottom (SB): if $\mathsf{endpt}(\mathsf{lsep}_{12}, 1) = \mathsf{endpt}(\mathsf{rsep}_{12}, 1)$.

(ii) Distinct Top (DT)**:** if $\mathsf{endpt}(\mathsf{lsep}_{12}, 1) \neq \mathsf{endpt}(\mathsf{rsep}_{12}, 1)$.
Shared Top (ST): if $\mathsf{endpt}(\mathsf{lsep}_{12}, 1) = \mathsf{endpt}(\mathsf{rsep}_{12}, 1)$.

Based on this, we have the following disjoint cases: (DB, DT), (DB, ST), (SB, DT), (SB, ST).

Next, we introduce an additional definition that further classifies the components of .

Definition 21. We say that a vertex subset $C \in V_1 \cup V_2$ (resp.$C \in V_2 \cup V_3$) is to the left of $e \in E_{12}$ (resp. E_{23}), if for each $i \in \{1, 2\}$ (resp. $i \in \{2, 3\}$) it holds that for all $u \in C \cap V_i$, $\sigma_i(u) \leq \sigma_i(\mathsf{endpt}(e, i))$. Analogously, we define "right of" relation if the inequality is flipped.

- For each $v \in \{\mathsf{endpt}(\mathsf{lsep}_{12}, 1), \mathsf{endpt}(\mathsf{rsep}_{12}, 1)\}$, let $\mathsf{Peculiar}^l(v) \subseteq \mathsf{Peculiar}(v)$ (resp. $\mathsf{PeculiarNP}^l(v) \subseteq \mathsf{PeculiarNP}(v)$) be the set of components C that are to the left of lsep_{23}.
- For each $v \in \{\mathsf{endpt}(\mathsf{lsep}_{12}, 1), \mathsf{endpt}(\mathsf{rsep}_{12}, 1)\}$, let $\mathsf{Peculiar}^r(v) \subseteq \mathsf{Peculiar}(v)$ (resp. $\mathsf{PeculiarNP}^r(v) \subseteq \mathsf{PeculiarNP}(v)$) be the set of components C that are to the right of rsep_{23}.
- For each $v \in \{\mathsf{endpt}(\mathsf{lsep}_{23}, 1), \mathsf{endpt}(\mathsf{rsep}_{23}, 1)\}$, let $\mathsf{Peculiar}^l(v) \subseteq \mathsf{Peculiar}(v)$ (resp. $\mathsf{PeculiarNP}^l(v) \subseteq \mathsf{PeculiarNP}(v)$) be the set of components C that are to the left of lsep_{23}.
- For each $v \in \{\mathsf{endpt}(\mathsf{lsep}_{23}, 1), \mathsf{endpt}(\mathsf{rsep}_{23}, 1)\}$, let $\mathsf{Peculiar}^r(v) \subseteq \mathsf{Peculiar}(v)$ (resp. $\mathsf{PeculiarNP}^r(v) \subseteq \mathsf{PeculiarNP}(v)$) be the set of components C that are to the left of rsep_{23}.

We next prove the following lemma.

Lemma 13. *The following bounds hold:*

1. $\min\left\{|\mathsf{PeculiarNP}^l(\mathsf{endpt}(\mathsf{lsep}_{12}, 1)), \mathsf{PeculiarNP}^l(\mathsf{endpt}(\mathsf{lsep}_{23}, 3))|\right\} \leq \sqrt{k}$.
2. $\min\{|\mathsf{PeculiarNP}^r(\mathsf{endpt}(\mathsf{rsep}_{12}, 1)), \mathsf{PeculiarNP}^r(\mathsf{endpt}(\mathsf{rsep}_{23}, 3))|\} \leq \sqrt{k}$.

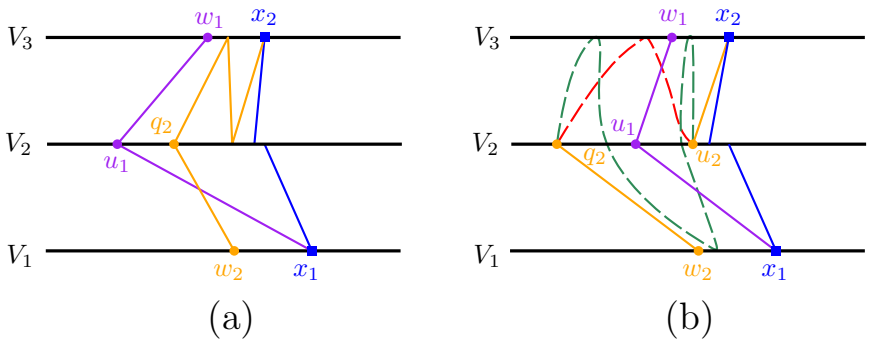

Figure 19: Illustration for the proof of Lemma 13. In case (b), we show two possible paths from q_2 to u_2 – the green path crosses x_1u_1; whereas the red path crosses u_1w_1.

Proof. We will show the first item, and the proof of the second one is analogous. Let us simplify the notation and say $x_1 \coloneqq \mathsf{endpt}(\mathsf{lsep}_{12}, 1), x_2 \coloneqq \mathsf{endpt}(\mathsf{lsep}_{23}, 3)$. Suppose for the contradiction that $|\mathsf{PeculiarNP}^l(x_1)| > \sqrt{k}$ and $|\mathsf{PeculiarNP}^l(x_1)| > \sqrt{k}$. Note that $\mathsf{PeculiarNP}^l(x_1) \cap \mathsf{PeculiarNP}^l(x_2) = \emptyset$, since a component $C \in \mathsf{PeculiarNP}^l(x_1)$ contains a neighbor of x_1 and hence in $G - \mathsf{endpt}(x_2)$, $V(C) \cup \{x_2\}$ is part of one connected component whose vertex set is a strict superset of that of C. In the next paragraph, we will show that each pair of components $C_1 \in \mathsf{PeculiarNP}^l(x_1)$ and $C_2 \in \mathsf{PeculiarNP}^l(x_2)$ must account for a distinct crossing. This implies that the number of crossings between the components of $\mathsf{PeculiarNP}^l(x_1)$ and $\mathsf{PeculiarNP}^l(x_2)$ is larger than k, a contradiction.

Let $u_1 \in V(C_1) \cap V_2$ be the rightmost neighbor of x_1 in C, and let $w_1 \in V(C_1) \cap V_3$ be the rightmost vertex in $V(C_1) \cap V_3$. Similarly, define $u_2 \in V(C_2) \cap V_2$ as the rightmost neighbor of x_2 in C_2 and let $w_2 \in V_1 \cap V(C_2)$ be the rightmost vertex of $V(C_2) \cap V_1$. If w_2 has a neighbor q_2 that is to the right of u_1, then w_2q_2 crosses x_1u_1 (see Figure 19, case (a)). Otherwise, if all neighbors of w_2 are to the left of u_1. Let q_2 be one such neighbor of w_2. There exists a path between u_2 and q_2 that is entirely within C_1. Since u_1 is between q_2 and u_2, such a path must cross either the edge x_1u_1 or the edge u_1w_1 (see Figure 19, case (b)).

□

In the next observation, we consolidate some properties regarding the components $\mathsf{comps}(v)$ for $v \in \mathsf{Endpoints}(\mathsf{Sep})$ in different cases.

Lemma 14.

1. *In* DB *case,*

(a) $\mathsf{PeculiarNP}^l(\mathsf{endpt}(\mathsf{lsep}_{12}, 1) = \mathsf{PeculiarNP}(\mathsf{endpt}(\mathsf{lsep}_{12}, 1))$. *Further, each* $C \in \mathsf{Pendants}(\mathsf{lsep}_{12}, 1)$ *lies to the left of* $\mathsf{endpt}(\mathsf{lsep}_{12}, 1)$.

(b) $\mathsf{PeculiarNP}^r(\mathsf{endpt}(\mathsf{rsep}_{12}, 1) = \mathsf{PeculiarNP}(\mathsf{endpt}(\mathsf{rsep}_{12}, 1))$. *Further, each* $C \in \mathsf{Pendants}(\mathsf{rsep}_{12}, 1)$ *lies to the left of* $\mathsf{endpt}(\mathsf{rsep}_{12}, 1)$.

2. *In* DT *case,*

(a) $\mathsf{PeculiarNP}^l(\mathsf{endpt}(\mathsf{lsep}_{23}, 3) = \mathsf{PeculiarNP}(\mathsf{endpt}(\mathsf{lsep}_{23}, 3))$.

(b) $\mathsf{PeculiarNP}^r(\mathsf{endpt}(\mathsf{rsep}_{23}, 1) = \mathsf{PeculiarNP}(\mathsf{endpt}(\mathsf{rsep}_{23}, 3))$.

3. *In* SB *case, when* $\mathsf{endpt}(\mathsf{lsep}_{12}, 1) = \mathsf{endpt}(\mathsf{rsep}_{12}, 1) = v_b$, *say, then* $(\mathsf{Peculiar}^l(v_b), \mathsf{Peculiar}^r(v_b))$ *is a partition of* $\mathsf{Peculiar}(v_b)$.

4. *In* ST *case, when* $\mathsf{endpt}(\mathsf{lsep}_{23}, 3) = \mathsf{endpt}(\mathsf{rsep}_{23}) = v_t$, *say, then* $(\mathsf{PeculiarNP}^l(v_t), \mathsf{PeculiarNP}^r(v_t))$ *is a partition of* $\mathsf{PeculiarNP}(v_t)$.

Proof.

1. DB case. We give the proof of item 1 and the proof of 2 is analogous. Let $v = \mathsf{endpt}(\mathsf{lsep}_{12}, 1)$, and we will prove that $\mathsf{PeculiarNP}^l(v) = \mathsf{PeculiarNP}(v)$. Consider a component $C \in \mathsf{PeculiarNP}(v)$, and suppose for the contradiction that there exists some edge $e = uw \in E(C)$ that is not to the left of lsep_{23}. If $e = \mathsf{lsep}_{23}$, or if e crosses lsep_{23}, then C would belong to $\mathsf{Special}(v)$, which is a contradiction. Now, suppose that $e = uw$ is to the right of lsep_{23}, where $u \in V_2$ and $w \in V_3$. Since C is a connected component in $G - v$, there is a path $P = (v = v_0, v_1, \ldots, v_\ell = u)$ such that $vv_1 \in E(G)$, and $(v_i, v_{i+1}) \in E(C)$ for $1 \leq i \leq \ell - 1$. Let us also observe that, except for $v \in V_1$, all other vertices in P alternate between vertices of V_2 and V_3, with odd indexed vertices belonging to V_2 and even indexed vertices belonging to V_3, – if P contained a vertex of V_1, then $C \in \mathsf{UpDown}(v)$, a contradiction.

   If $\sigma_3(\mathsf{endpt}(\mathsf{rsep}_{23}, 3)) < \sigma_3(w)$, then we say that this is **case 1**, otherwise if $\sigma_3(\mathsf{endpt}(\mathsf{lsep}_{23}, 3)) < \sigma_(w) < \sigma_3(\mathsf{endpt}(\mathsf{rsep}_{23}, 3))$, then we say that this is **case 2**.

   In **case 1**, note that there exists an edge of the path P must either cross lsep_{23}, rsep_{23}, or pass through a vertex of $\mathsf{GapRight}$. In each of the cases, this implies that $C \in \mathsf{Special}(v)$, which is a contradiction.

   Now, let us consider **case 2**. Let $v_j \in C \cap V_2$, $j > 0$ be the first vertex along the path that lies to the right of $\mathsf{endpt}(\mathsf{lsep}_{23}, 2)$. Note that v_j exists, since the last vertex, i.e., u satisfies the condition. We will show that in this case, C must belong to $\mathsf{Special}(v)$, a contradiction.

   First, if $j = 1$, then let us consider two cases. (i) If $\sigma_2(\mathsf{endpt}(\mathsf{lsep}_{23}, 2)) < \sigma_2(v_1) < \sigma_2(\mathsf{endpt}(\mathsf{lsep}_{12}, 2))$, then $v_1 \in \mathsf{GapLeft}$. (ii) If $\sigma_2(v_1) < \sigma_2(\mathsf{endpt}(\mathsf{lsep}_{12}, 2)) < \sigma_2(v_1) < \sigma_2(\mathsf{endpt}(\mathsf{rsep}_{12}, 2))$, then $vv_1 \in \mathsf{MiddleEdges}(\mathsf{Sep})$. (iii) Finally, if $\sigma_2(v_1) > \sigma_2(\mathsf{endpt}(\mathsf{rsep}_{12}, 2))$, then vv_1 crosses rsep_{12}. Thus, in each of the three sub-cases, we obtain that $C \in \mathsf{Special}(v)$, which is a contradiction. Note that this same argument also shows the second claim about components from $\mathsf{Pendants}(v)$.

   Next, suppose $j > 1$. First, note that $j \geq 3$, and $v_{j-1} \in V_3$. (i) If $\sigma_3(v_{j-1}) < \sigma_3(\mathsf{endpt}(\mathsf{lsep}_{23}, 3))$, then the edge $v_{j-1}v_j$ crosses lsep_{12}. (ii) Otherwise, $\sigma_3(\mathsf{endpt}(\mathsf{lsep}_{23}, 3)) < \sigma_3(v_{j-1})$. Note that $v_{j-2} \in V_2$ satisfies that $\sigma_2(v_{j-2}) < \sigma_2(\mathsf{endpt}(\mathsf{lsep}_{23}, 2))$. However, this implies that the edge $v_{j-2}v_{j-1}$ crosses lsep_{23}. Thus, again in each of the two sub-cases, we obtain that $C \in \mathsf{Special}(v)$, a contradiction.

2. DT case. Again, we prove the first item, and the proof of the second item is analogous. Again, for convenience, let $v = \mathsf{endpt}(\mathsf{lsep}_{23}, 3)$, and consider any $C \in \mathsf{PeculiarNP}(v)$. Suppose for contradiction that $C \notin \mathsf{PeculiarNP}^l(v)$, which implies that C contains an edge uw with $u \in V_2, w \in V_1$ that is not to the left of lsep_{12}. Let us again consider a path $P = (v = v_0, v_1, v_2, \ldots, v_\ell = u)$ where $vv_1 \in E(G)$ and $(v_i, v_{i+1}) \in E(C)$. Again, the vertices must alternate between V_2 and V_1 respectively. Let $v_j \in V_2$ be the first vertex along the path that is to the right of $\mathsf{endpt}(\mathsf{lsep}_{12}, 2)$.

   If $j = 1$, then consider the following cases. (i) $\sigma_1(v_2) < \sigma_1(\mathsf{endpt}(\mathsf{lsep}_{12}, 1))$, then the edge v_1v_2 crosses lsep_{12}. (ii) If $\sigma_1(\mathsf{endpt}(\mathsf{lsep}_{12}, 1)) < \sigma_1(v_2) < \sigma_1(\mathsf{endpt}(\mathsf{rsep}_{12}, 1))$, then $v_1v_2 \in \mathsf{MiddleEdges}(\mathsf{Sep})$. (iii) If $\sigma_1(\mathsf{endpt}(\mathsf{rsep}_{12}, 1)) < \sigma_2(v_2)$, then v_1v_2 crosses rsep_{12}. Thus, in each case $C \in \mathsf{Special}(v)$, a contradiction.

   If $j > 1$, then this proof is analogous to a similar case from the previous part, and hence we omit it.

3. SB case. In this case, suppose for contradiction that there exists some $C \in \mathsf{Peculiar}(v_b)$ such that $C \notin \mathsf{Peculiar}^l(v_b) \cup \mathsf{Peculiar}^r(v_b)$. We show that $C \in \mathsf{Special}(v_b)$, a contradiction. First, if C is a pendant vertex u, then using an argument similar to DB case above, we can show that either $u \in \mathsf{GapLeft} \cup \mathsf{GapRight}$, or the edge $v_bu \in \mathsf{MiddleEdges}(\mathsf{Sep})$, a contradiction. Note that for some $i \in \{2, 3\}$, there exists a vertex $w \in V(C) \cap V_i$ such that $\sigma_i(\mathsf{endpt}(\mathsf{lsep}_{23}, i)) < \sigma_i(w) < \sigma_i(\mathsf{endpt}(\mathsf{rsep}_{23}, i))$. Suppose w.l.o.g. that $w \in V_2$. Then, let $w' \in V_3$ be a neighbor of w in C. Note that w' must also satisfy $\sigma_3(\mathsf{endpt}(\mathsf{lsep}_{23}, 3)) < \sigma_3(w') < \sigma_3(\mathsf{endpt}(\mathsf{rsep}_{23}, 3))$ – otherwise ww' edge crosses either lsep_{23} or rsep_{23}, a contradiction. However, if w' lies between the top endpoints

of lsep_{23} and rsep_{23}, then we can show that $C \in \mathsf{Special}(v_b)$ using an argument similar to the DB case above, and arrive at a contradiction.

4. ST case. Here, the argument is exactly similar to the SB case, and is therefore omitted. Note that in this case, the argument regarding the pendant vertices does not hold, since the edges of $\mathsf{MiddleEdges}(\mathsf{Sep})$ are only defined in E_{23}.

□

Handling Pendant Components. In this step, we will handle the pendant components attached to $\mathsf{Endpoints}(\mathsf{Sep})$. At a high level, the idea is similar to that used for the 2 layer case, except that we may have at most 8 vertices in $\mathsf{Endpoints}(\mathsf{Sep})$. [19]

Definition 22.

- A *pendant-blueprint* is a collection of tuples

$$\mathsf{PendantBlueprint} = \{(\mathsf{LeftPNum}(v), \mathsf{MidPNum}(v), \mathsf{RightPNum}(v)) : v \in \mathsf{Endpoints}(\mathsf{Sep})\},$$

such that, (i) for each $v \in \mathsf{Endpoints}(\mathsf{Sep})$, $\mathsf{LeftPNum}(v) + \mathsf{MidPNum}(v) + \mathsf{RightPNum}(v) = \mathsf{TotalPNum}(v)$, which denotes the total sum of multiplicities of $\mathsf{Pendants}(v)$, and (ii) for each $v' \in \mathsf{Endpoints}(\mathsf{Sep}, 1)$, $\mathsf{MidPNum}(v) = 0$.

- A *pendant-guess* is a collection of tuples

$$\mathsf{PendantGuess} = \{(\mathsf{LeftPendants}(v), \mathsf{MidPendants}(v), \mathsf{RightPendants}(v)) : v \in \mathsf{Endpoints}(\mathsf{Sep})\}$$

where (i) for each $v \in \mathsf{Endpoints}(\mathsf{Sep})$, $(\mathsf{LeftPendants}(v), \mathsf{MidPendants}(v), \mathsf{RightPendants}(v))$ is a partition of $\mathsf{Pendants}(v)$, and (ii) for each $v' \in \mathsf{Endpoints}(\mathsf{Sep}, 1)$, $\mathsf{MidPendants}(v') = \emptyset$.

- We say that a $\mathsf{PendantGuess} = \{(\mathsf{LeftPendants}(v), \mathsf{MidPNum}(v), \mathsf{RightPendants}(v)) : v \in \mathsf{Endpoints}(\mathsf{Sep})\}$ is $\mathsf{PendantBlueprint}$-*compliant* if, for each $v \in \mathsf{Endpoints}(v)$ the total sum of multiplicities of vertices in $\mathsf{LeftPendants}(v)$ (resp. $\mathsf{MidPNum}(v), \mathsf{RightPNum}(v)$) is equal to $\mathsf{LeftPendants}(v)$ (resp. $\mathsf{MidPendants}(v), \mathsf{RightPe}$

- Let $\mathsf{PendantBlueprint}$ be a pendant-blueprint, and let $\mathsf{PendantGuess}$ be a $\mathsf{PendantBlueprint}$-compliant pendant-guess. Then, we define a $\mathsf{NicePFamily}(\mathsf{PendantBlueprint}, \mathsf{PendantGuess}, \mathsf{Sep}, \sigma)$ to be a family of drawings σ' satisfying the following properties.
  - For each $i \in [3]$, σ_i and σ'_i, when restricted to $V_i \setminus \bigcup_{v \in \mathsf{Endpoints}(\mathsf{Sep})} \mathsf{Pendants}(v)$, are the same.
  - For each $v \in \mathsf{Endpoints}(\mathsf{Sep}, 1) \cup \mathsf{Endpoints}(\mathsf{Sep}, 3)$, σ'_2 places (i) all the vertices in $\mathsf{LeftPendants}(v)$ to the left of $\mathsf{endpt}(\mathsf{lsep}_{23}, 2)$, (ii) all the vertices of $\mathsf{MidPendants}(v)$ between $\mathsf{endpt}(\mathsf{lsep}_{23}, 2)$ and $\mathsf{endpt}(\mathsf{rsep}_{23}, 2)$, and (iii) all the vertices in $\mathsf{RightPendants}(v)$ to the right of $\mathsf{endpt}(\mathsf{rsep}_{23}, 2)$.
  - For each $v_2 \in \mathsf{Endpoints}(\mathsf{Sep}, 2)$,
    - σ'_1 places (i) all the vertices in $\mathsf{LeftPendants}(v)$ to the left of $\mathsf{endpt}(\mathsf{lsep}_{12}, 1)$, (ii) all the vertices in $\mathsf{MidPendants}(v)$ between $\mathsf{endpt}(\mathsf{lsep}_{23}, 2)$ and $\mathsf{endpt}(\mathsf{rsep}_{23}, 2)$, and (iii) all the vertices in $\mathsf{RightPendants}(v)$ to the right of $\mathsf{endpt}(\mathsf{rsep}_{12}, 1)$,
    - σ'_3 places (i) all the vertices in $\mathsf{LeftPendants}(v)$ to the left of $\mathsf{endpt}(\mathsf{lsep}_{23}, 3)$, (ii) all the vertices in $\mathsf{MidPendants}(v)$ between $\mathsf{endpt}(\mathsf{lsep}_{23}, 3)$ and $\mathsf{endpt}(\mathsf{rsep}_{23}, 3)$, and (iii) all the vertices in $\mathsf{RightPendants}(v)$ to the right of $\mathsf{endpt}(\mathsf{rsep}_{23}, 3)$.

[19] By a more careful case analysis, it may be shown that we do not have to handle each of the 8 vertices, which further restricts the number of guesses; however since this does not substantially affect the running time, we handle all the vertices in $\mathsf{Endpoints}(\mathsf{Sep})$ in a unified fashion.

Next, we introduce the notion of separator-compatible drawings.

Definition 23. Let $\mathcal{I}$ be a k-minimal YES-instance and σ, σ' be drawings with k crossings. Further, let Sep be a separator w.r.t. σ. We say that σ' is a (Sep, σ)-*compatible drawing* if it satisfies the following properties.

- Sep is also a separator w.r.t. σ', such that the number of crossings on each side of each edge in Sep is same in σ and σ',
- $\mathsf{GapLeft}(\mathsf{Sep}, \sigma) = \mathsf{GapLeft}(\mathsf{Sep}, \sigma')$, and $\mathsf{GapRight}(\mathsf{Sep}, \sigma) = \mathsf{GapRight}(\mathsf{Sep}, \sigma')$.
- $\mathsf{MiddleEdges}(\mathsf{Sep}, \sigma) = \mathsf{MiddleEdges}(\mathsf{Sep}, \sigma')$, and $\mathsf{CrossingEdges}(\mathsf{Sep}, \sigma) = \mathsf{CrossingEdges}(\mathsf{Sep}, \sigma')$.

Next, we state a transitive property of separator-compatibility.

Observation 9. Let $\mathcal{I}$ be a k-minimal extended instance, and let $\sigma_1, \sigma_2, \sigma_3$ be drawings of $\mathcal{I}$ with k crossings. Let Sep be a separator w.r.t. σ_1. If σ_2 is (Sep, σ_1)-compatible, and σ_3 is (Sep, σ_2)-compatible, then σ_3 is (Sep, σ_1)-compatible.

The proof of the following lemma is also via the same kinds of arguments used in the proof of Lemma 5, and thus is omitted.

Lemma 15. *Let $\mathcal{I}$ be a k-minimal* YES*-instance, let σ be a drawing with k crossings, and let* Sep *be a separator guaranteed by Lemma 2.*

Then, there exists a $\mathsf{PendantBlueprint}$ *such that, for any* $\mathsf{PendantGuess}$ *that is* $\mathsf{PendantBlueprint}$*-compliant, there exists a drawing $\sigma' \in \mathsf{NicePFamily}(\mathsf{PendantBlueprint}, \mathsf{PendantGuess}, \mathsf{Sep}, \sigma)$ such that σ' is (Sep, σ)-compatible.*

To summarize, in (DB, DT) case, we can conclude the fate of all the components (i.e., which side of the separator edges they belong to) for $\bigcup_{v \in \mathsf{Endpoints}(\mathsf{Sep})} \mathsf{PeculiarNP}(v)$. On the other hand, in (SB, ST) case, we know that, either (i) all except a $\mathcal{O}(\sqrt{k})$ components in $\mathsf{PeculiarNP}(\mathsf{endpt}(\mathsf{lsep}_{12}, 1))$ (resp. $\mathsf{PeculiarNP}(\mathsf{endpt}(\mathsf{lsep}_{23}, 3))$) lie to the left of lsep_{23} (resp. to the right of rsep_{12}), or (ii) vice versa. Thus, we can guess which of the two cases we are in, and in each case, two subsets of size $\mathcal{O}(\sqrt{k})$, respectively. This leaves us with (SB, DT) and (ST, DB) case, in which case, we know that the $\mathsf{PeculiarNP}(v)$ for the shared endpoint v are partitioned into $(\mathsf{PeculiarNP}^l(v), \mathsf{PeculiarNP}^r(v))$; however, we do not know how to find such a partition. The goal of the next subsection is to show that such a partition can be found in a structured way.

5.3.2 Reordering Peculiar Components in (SB, DT)/(ST, DB) Case.

Throughout the subsubsection, we will focus on (SB, DT) case, wherein $\mathsf{endpt}(\mathsf{lsep}_{12}, 1) = \mathsf{endpt}(\mathsf{rsep}_{12}, 1)$. The (ST, DB) case is handled symmetrically by focusing on the corresponidng shared endpoint $\mathsf{endpt}(\mathsf{lsep}_{23}, 3) = \mathsf{endpt}(\mathsf{rsep}_{23}, 3)$, and by flipping the roles of V_1 and V_3. We omit the repetition.

We note down the assumptions that will be made throughout this subsubsection. We have a bottom-heavy k-minimal instance $\mathcal{I}$ with a drawing σ with k crossings, and a Sep w.r.t. σ, which is of the case (SB, DT) case, with $\mathsf{endpt}(\mathsf{lsep}_{12}, 1) = \mathsf{endpt}(\mathsf{rsep}_{12}, 1)$, which we denote by v throughout the subsection. Further, the vertices $u \in V_2$ with $\sigma_2(\mathsf{endpt}(\mathsf{lsep}_{23}, 2)) < \sigma_2(u) < \sigma_2(\mathsf{endpt}(\mathsf{rsep}_{23}, 2))$ and $w \in V_3$ with $\sigma_3(\mathsf{endpt}(\mathsf{lsep}_{23}, 3)) \leq \sigma_3(w) \leq \sigma_3(\mathsf{endpt}(\mathsf{rsep}_{23}, 3))$ will not be relevant for any of the arguments/discussion of this subsubsection, and we will tacitly ignore cases involving such vertices.

Further, let us use $\mathcal{C}, \mathcal{C}^l, \mathcal{C}^r$ instead of $\mathsf{PeculiarNP}, \mathsf{PeculiarNP}^l, \mathsf{PeculiarNP}^r$, respectively. Note that these sets are defined w.r.t. a drawing σ and its corresponding separator Sep. First, we will analyze the properties of the components of $\mathcal{C}$, then subsequently we will modify σ to obtain another separator-compatible drawing σ'.

Setup.

Definition 24 (light and heavy components)**.** We say that a component $C \in \mathcal{C}$ is σ-*light* if the total number of crossings involving at least one edge of $E(C) \cup \{vu : u \in C\}$, is at most $\sqrt{k}$; otherwise we say that C is σ-*heavy*. We denote the subsets of light and heavy components as $\mathsf{Heavy}(v, \mathcal{C}, \sigma)$ and $\mathsf{Light}(v, \mathcal{C}, \sigma)$, respectively.

The following remark is due regarding the simplification of the notation.

Remark 1. In the notions introduced in Definition 24 and later, we initially make the dependence on $v, \mathcal{C}$, and σ explicit in the notation/terminology, and subsequently simplify it by omitting $v, \mathcal{C}, \sigma$ from the notation, if the respective objects are obvious from the context (indeed, they are fixed for the rest of the subsection, except for σ which will be modified at the end). For example, henceforth $\mathsf{Heavy}(v, \mathcal{C}, \sigma)$ is abbreviated to simply Heavy.

We state the following simple observation.

Observation 10. The number of heavy components is bounded by $2\sqrt{k}$, i.e., $|\mathsf{Heavy}| \leq 2\sqrt{k}$.

Proof. Suppose for contradiction that $|\mathsf{Heavy}| > 2\sqrt{k}$, then total number of crossings in σ is more than $2\sqrt{k} \cdot \sqrt{k}/2$, since each $C \in \mathsf{Heavy}$ accounts for more than $\sqrt{k}$ crossings, and since each crossing is counted at most twice in this manner (since, for example, the pair of crossing edges may belong to a pair of two different heavy components). This contradicts the fact that σ is a drawing with k crossings. □

Leftmost and rightmost light components.

Definition 25 (leftmost and rightmost light components)**.** Let $\prec_l$ denote a total ordering on $\mathcal{C}$, where for $C_1, C_2 \in \mathcal{C}$, $C_1 \prec_l C_2$ if the leftmost neighbor of v in C_1 is to the left of the leftmost neighbor of v in C_2. Similarly, let $\prec_r$ denote a total ordering on $\mathcal{C}$ using the rightmost neighbors of v in components (note that $C_1 \prec_l C_2$ does not necessarily imply that $C_2 \prec_r C_1$, and vice versa). Let $\mathsf{LeftmostLC} \in \mathsf{Light}$ be the leftmost light component (according to $\prec_l$), and $\mathsf{RightmostLC} \in \mathsf{Light}$ be the rightmost light component (according to $\prec_r$). Note that any component C satisfying $C \prec_l \mathsf{LeftmostLC}$ or $C \prec_r \mathsf{RightmostLC}$ must be a heavy component.
Let $\mathsf{Extreme} \subseteq \mathcal{C}$ denote all the components C such that, (i) $C \prec_l \mathsf{LeftmostLC}$, or (ii) $C \prec_r \mathsf{RightmostLC}$, or (iii) at least one edge of C crosses an edge of $\mathsf{LeftmostLC}$ or $\mathsf{RightmostLC}$.

The following observation follows immediately from the definition.

Observation 11. $|\mathsf{Extreme}| \leq 4\sqrt{k}$.

Finally, we define the notion of star vertices, which are crucial to the result shown in this subsubsection.

Definition 26 (Star vertices)**.** we say that a vertex $p \in R_2$ is a $(v, \mathcal{C}, \sigma)$-*left-star vertex* (resp. $(v, \mathcal{C}, \sigma)$-*right-star vertex*) if there is an edge incident to p that crosses an edge of the form $vu^l \in E_{12}$, where $u^l \in \mathsf{LeftmostLC} \cap V_2$ is the leftmost neighbor of v in $\mathsf{LeftmostLC}$ (resp. $u^r \in \mathsf{RightmostLC} \cap V_2$ is the rightmost neighbor of v in $\mathsf{RightmostLC}$). Let $\mathsf{LeftStar}, \mathsf{RightStar}$ denote the sets of left-star and right-star vertices respectively. When the left vs. right distinction is not important, we refer to the vertices of $\mathsf{LeftStar} \cup \mathsf{RightStar}$ as simply *star vertices*. Let Star denote the resultant set of star vertices, and they are are enumerated as $\mathsf{st}_1, \mathsf{st}_2, \ldots, \mathsf{st}_q$ from left to right according to σ_2. See Figure 20 for an illustration.

Next, we state and prove following straightforward observations that follow immediately from the definitions.

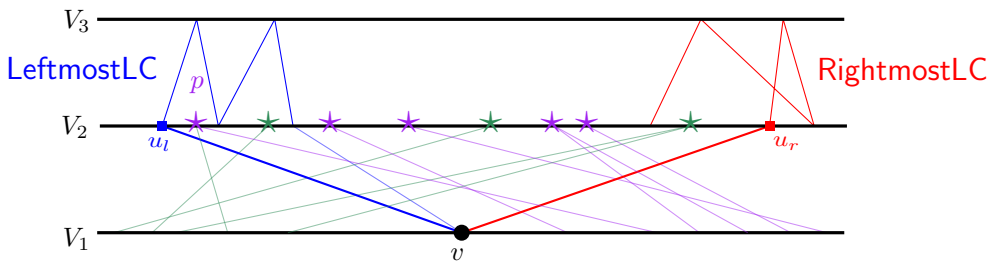

Figure 20: Illustration of components LeftmostLC, RightmostLC and star vertices. Left-star vertices (shown in green) have at least one incident edge crossing vu_l, and right-star (purple) have at least one incident edge crossing vu_r. Note that a vertex can be both left- and right-star vertex, e.g., p.

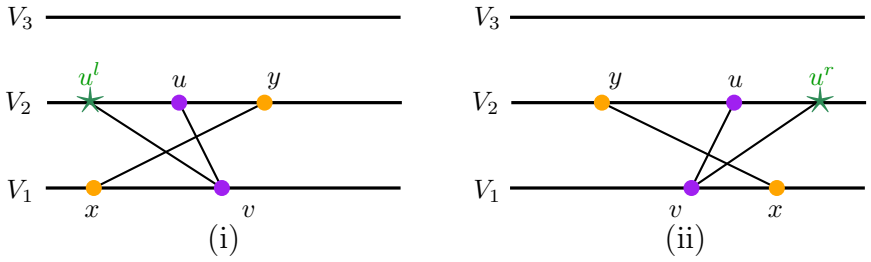

Figure 21: Two cases from the proof of Claim 5.1.

Observation 12. No star vertex belongs to a component in $\mathcal{C}$.

Proof. A star vertex u has an incident edge that crosses an edge incident to v, and hence u is adjacent to a vertex $w \in V_1$, where $w \neq v$. Hence, the connected component in $G - v$ that contains u, also contains $w \in V_1$, implying that $C \in \mathsf{UpDown}(v)$. □

Observation 13. $|\mathsf{LeftStar}|, |\mathsf{RightStar}| \leq \sqrt{k}$.

Proof. A star vertex has an incident edge that crosses an edge vu_l for some $u_l \in \mathsf{LeftmostLC}$. However, since LeftmostLC is a light component, it is involved in at most $\sqrt{k}$ crossings. Therefore, the number of vertices in LeftStar is bounded by $\sqrt{k}$. A similar argument shows that $|S_r| \leq \sqrt{k}$. □

Claim 5.1. *Let $u, y \in V_2$ be two distinct vertices such that (1) u is a neighbor of v, (2) there exists some neighbor $x \in V_1$ of y, and (3) the edges vu and xy cross each other. Then, y must be a star vertex.*

Proof. Since vu and xy cross, either (i) $\sigma_1(v) < \sigma_1(x)$ and $\sigma_2(y) < \sigma_2(u)$, or (ii) $\sigma_1(v) > \sigma_1(x)$ and $\sigma_2(y) > \sigma_2(u)$ (See Figure 21). In case (i), note that $\sigma_2(y) < \sigma_2(u) \leq \sigma_2(u_r)$. We also have that $\sigma_1(v) < \sigma_1(x)$. Therefore, xy crosses vu^r as well, which implies that y is a star vertex. Case (ii) is symmetric – we have that $\sigma_2(u^l) \leq \sigma_2(u) < \sigma_2(y)$, and $\sigma_2(x) < \sigma_2(v)$. Therefore, xy crosses vu^l implying that y is a star vertex. □

Claim 5.2. *Let u, u' be two distinct neighbors of v such that $\sigma_2(\mathsf{st}_i) < \sigma_2(u) < \sigma_2(u') < \sigma_2(\mathsf{st}_{i+1})$, where $\mathsf{st}_i, \mathsf{st}_{i+1} \in \mathsf{Star}$ are some pair of consecutive star vertices according to σ_2 (note that $1 \leq i < |\mathsf{Star}|$). Then, the edges vu and vu' cross the same set of edges in σ.*

Proof. Suppose for the sake of contradiction that there exists an edge xy with $x \in V_1, y \in V_2$, such that xy crosses vu but not vu' (the other case is symmetric). Since xy crosses vu, it must be the case that (i) $\sigma_1(v) < \sigma_1(x)$ and $\sigma_2(u) > \sigma_2(y)$, or (ii) $\sigma_1(v) > \sigma_1(x)$ and $\sigma_2(u) < \sigma_2(y)$. See Figure 22.

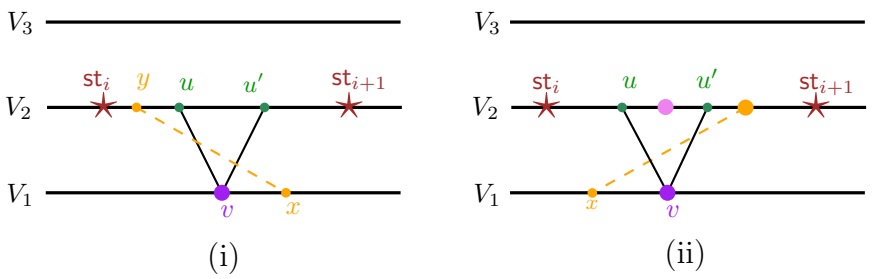

Figure 22: Illustration for the proof of Claim 5.2. In case (i), an edge xy that crosses vu must also cross vu'. In case (ii), there are two possible locations for the vertex y, shown in orange and purple, respectively. In the orange case, the edge xy again must cross both vu and vu'; whereas in the purple case, the vertex y must be a star vertex, contradicting that st^i and st^{i+1} are consecutive star vertices.

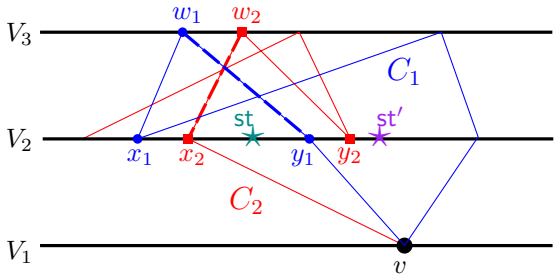

Figure 23: Illustration for the proof of Claim 5.3. Components C_1 (blue) and C_2 (red) surround a star vertex st (C_1 happens to surround st' as well). Due to the positional assumptions in the first case in the proof of Claim 5.3, edges y_1w_1 and x_2w_2 must cross (shown as thick dashed edges).

First, consider case (i). Since $\sigma_2(y) < \sigma_2(u) < \sigma_2(u')$ and $\sigma_1(v) < \sigma_1(x)$, it follows that vu' and xy cross as well.

Now consider case (ii). We have that $\sigma_2(u) < \sigma_2(u')$, $\sigma_2(u) < \sigma_2(y)$, and $\sigma_1(x) < \sigma_1(v)$. If $\sigma_2(u') < \sigma_2(y)$, then xy crosses vu'. Otherwise, $\sigma_2(x_l) \leq \sigma_2(\mathsf{st}_i) < \sigma_2(u) < \sigma_2(u') < \sigma_2(y) \leq \sigma_2(\mathsf{st}^{i+1}) \leq \sigma_2(x_r)$. However, by Claim 5.1, we obtain that y must be a star vertex, which contradicts that st_{i+1} is the next consecutive star vertex after st_i. □

Next, we introduce the notion of components that "surround" a star vertex, and then bound the number of such components.

Definition 27. We say that a component C *surrounds* a star vertex st, if it contains two vertices that are placed to the either side of st in σ_2 (see Figure 23). Let $\mathsf{Surround} \subseteq \mathcal{C} \setminus (\mathsf{Heavy} \cup \mathsf{Extreme})$ denote the subset of components C such that C surrounds some $u \in \mathsf{Star}$. Note that no pendant component can belong to $\mathsf{Surround}$.

We show the following claim regarding the necessity of crossings between components surrounding the same star vertex.

Claim 5.3. *Let $C_1, C_2 \in \mathsf{Surround}$ be distinct components that surround a star vertex $\mathsf{st} \in \mathsf{Star}$. Then, there exist edges $e_1 \in E(C_1)$ and $e_2 \in E(C_2)$ that cross each other.*

Proof. Since C_1 surrounds st, it contains two vertices that are placed to the either side of st. Further, since C_1 is a connected component in $G - v$, it must contain a vertex $w_1 \in V_3 \cap V(C_1)$ such that w_1 has two neighbors in C_1, say x_1 and y_1 such that x_1 is to the left of st, and y_1 is to the right of st. Similarly, define $w_2, x_2, y_2 \in V(C_2)$, respectively.

Suppose first that $\sigma_3(w_1) < \sigma_3(w_2)$, i.e., w_1 is to the left of w_2. Then, since $\sigma_2(x_2) < \sigma_2(\mathsf{st}) < \sigma_2(y_1)$, the edges x_2w_2 and y_1w_1 cross each other (see Figure 23). In the other case, when $\sigma_3(w_2) < \sigma_3(w_1)$, we can observe symmetrically that the edges x_1w_1 and y_2w_2 cross each other. This proves the claim. □

In the following lemma, we bound the number of components that surround some star vertex.

Lemma 16. $|\mathsf{Surround}| \leq 5 \cdot (k)^{2/3}$.

Proof. Let us number the left-star vertices from right to left (in σ_2) as $\mathsf{st}^l_1, \mathsf{st}^l_2, \ldots$, and analogously, let us number the right-star vertices from left to right as $\mathsf{st}^r_1, \mathsf{st}^r_2, \ldots$. Let $t = (k)^{1/3}$, and let $\mathsf{Star}' = \{\mathsf{st}^l_i \in \mathsf{LeftStar} : i \leq t\} \cup \{\mathsf{st}^r_i \in \mathsf{RightStar} : i \leq t\}$. Note that $|\mathsf{Star}'| \leq 2t = 2(k)^{2/3}$. In order to bound the size of $\mathsf{Surround}$, we classify each $C \in \mathsf{Surround}$ into two types: C is class 1 if it surrounds some star vertex of Star'; otherwise, if it only surrounds some vertex of $\mathsf{Star} \setminus \mathsf{Star}'$, then it is class 2. In the following two claims, we bound the number of components of each class separately.

Claim 5.4. *The number of class 1 components in* $\mathsf{Surround}$ *is at most* $4 \cdot (k)^{2/3}$.

Proof of Claim 5.4. We arbitrarily number the vertices of Star' as $\mathsf{st}'_1, \mathsf{st}'_2, \ldots, \mathsf{st}'_z$, where $z = |\mathsf{Star}'| \leq 2(k^{1/3})$. Let n_i denote the number of components that surround st'_i, and w.l.o.g. assume $n_i > 0$ for each i – otherwise such star vertex plays no role in the following analysis, and can be ignored. Let $m_i = n_i - 1 \geq 0$ for each $1 \leq i \leq z$. Hence, the total number of class 1 components is at most $\sum_{i=1}^{z} n_i = \sum_{i=1}^{z} m_i + z$. Since $z \leq 2(k)^{2/3}$, it suffices to show that $\sum_{i=1}^{z} m_i \leq 2 \cdot (k)^{2/3}$.

By Claim 5.3 any pair of components that surround the same $\mathsf{st}_i \in \mathsf{Star}$ must have a pair of crossing edges. Therefore, the number of crossings among the components surrounding st_i is at least $\frac{n_i(n_i-1)}{2} \geq \frac{m_i^2}{2}$. However, since there are at most k crossings, it follows that $\sum_{i=1}^{z} \frac{m_i^2}{2} \leq k$, which implies that, $\sum_{i=1}^{z} m_i^2 \leq 2k$. Now, by using Cauchy–Schwarz inequality, we obtain that

$$\left(\sum_{i=1}^{z} m_i\right)^2 \leq \left(\sum_{i=1}^{z} m_i^2\right) \cdot z \leq 2kz.$$

This implies that, $\sum_{i=1}^{z} m_i \leq \sqrt{2kz} \leq \sqrt{2k \cdot 2(k)^{1/3}} = 2 \cdot \sqrt{(k)^{4/3}} = 2 \cdot (k)^{2/3}$. □

Claim 5.5. *The number of class 2 components in* $\mathsf{Surround}$ *is at most* $(k)^{2/3}$.

Proof of Claim 5.5. We will show that each component of class 2 is involved in at least $(k)^{1/3}$ distinct relevant crossings, which immediately implies the lemma.

Fix one such class 2 component C, and note that it surrounds a star vertex $\mathsf{st} \notin \mathsf{Star}'$. Without loss of generality, suppose st is a left-star vertex (the right-star case is symmetric), say $\mathsf{st}^l_i \in \mathsf{LeftStar} \setminus \mathsf{Star}'$, which implies that $i > t$. Further, since C is not a class 1 component, it does not surround any $\mathsf{st}^l_{i'}$ with $i' \leq t$. However, C contains a neighbor w_l of v, and by assumption, it also lies to the left of all of $\mathsf{st}^l_{i'}$ with $i' \leq t$.

Since C is a light component, it holds that $\mathsf{LeftmostLC} \prec_l C$, where recall that $\mathsf{LeftmostLC}$ is the leftmost component from Definition 25, with $u^l \in \mathsf{LeftStar}$ being the leftmost neighbor of v. Similarly, let w^l denote teh leftmost neighbor of v in C. This implies that $\sigma(u^l) < \sigma(w^l) < \sigma(\mathsf{st}^l_{i'})$ for all $i' \leq t$. Further, since an edge e incident to $\mathsf{st}^l_{i'}$ crosses vu_l, the other endpoint of e, say $u_b \in V_1$ lies to the left of v. However, this implies that e also crosses the edge vw^l (See Figure 24). Since this holds for each $i \leq t$, the edge vw^l crosses at least t distinct edges, thus accounting for at least t distinct crossings.

□

With Claim 5.4 and Claim 5.5, we conclude the proof of Lemma 16.

□

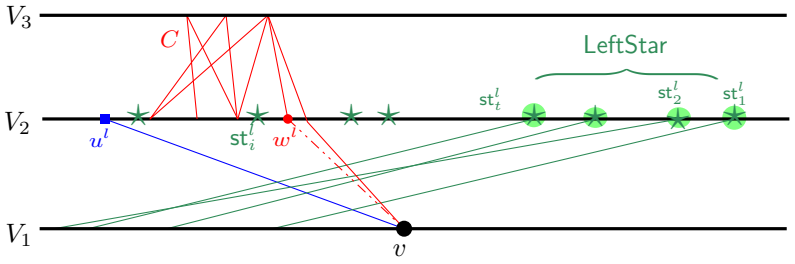

Figure 24: Illustration for Claim 5.5. Component C (shown in red) surrounds star vertex $\mathsf{st}_i^l \notin \mathsf{Star}'$. For each $i' \leq k^{1/3}$, the edge vw_l (dashed red) crosses all edges incident to $\mathsf{st}_{i'}^l$ that also cross the edge vu^l (blue). Thus, C accounts for at least $(k)^{1/3}$ crossings.

Component Types.

Next, we define the notion of a type of a component.

Definition 28 (Original and reduced type)**.** Let $C \in \mathsf{PeculiarNP}(C) = \mathcal{C} \setminus \mathsf{Pendants}(v)$ be a component in $G - v$. Let $\mathsf{m}_{12}^{\mathsf{orig}}(C) \coloneqq |M_1^{\mathsf{orig}}2|\,|\{uv : u \in C \cap V_2\}|$, i.e., the number of edges between v and the vertices of C (including multiplicities), and $\mathsf{m}_{23}^{\mathsf{orig}}(C) \coloneqq |E(C)|$, i.e., the number of edges in C.
Then, the *original type* of C is defined as the pair $(\mathsf{m}_{12}^{\mathsf{orig}}(C), \mathsf{m}_{23}^{\mathsf{orig}}(C))$, and the *reduced type* of C is defined as $(\mathsf{m}_{12}(C), \mathsf{m}_{23}(C)) \in \{1, 2, \ldots, (k)^{1/3}, *, L\}^2$, where

$$(\mathsf{m}_{12}(C), \mathsf{m}_{23}(C)) = \begin{cases} (\mathsf{m}_{12}^{\mathsf{orig}}(C), \mathsf{m}_{23}^{\mathsf{orig}}(C)) & \text{if } 0 < \mathsf{m}_{12}^{\mathsf{orig}}(C) < (k)^{1/3} \text{ and } \mathsf{m}_{23}^{\mathsf{orig}}(C) < (k)^{1/3} \\ (*, L) & \text{if } \mathsf{m}_{23}^{\mathsf{orig}}(C) \geq (k)^{1/3} \\ (L, *) & \text{if } \mathsf{m}_{23}^{\mathsf{orig}}(C) < (k)^{1/3} \text{ and } \mathsf{m}_{12}^{\mathsf{orig}} \geq (k)^{1/3}. \end{cases}$$

Here, the meaning of $(*, L)$ type is that $|E_{23}|$ is *larger* than $(k)^{1/3}$ (hence, L), whereas the number of neighbors of v in C is arbitrary (hence, $*$). We interpret $(L, *)$ similarly.

For any pendant $C = \{u\} \in \mathsf{Pendants}(v)$. Then, with $\mathsf{m}_{12}^{\mathsf{orig}}(C)$ denote the multiplicity of the edge vu. Then, the original type of C is defined to be $(\mathsf{m}_{12}^{\mathsf{orig}}(C), 0)$, and the reduced type of C is defined as $(1, 0)$.

Finally, we define $\mathsf{Types} \coloneqq \{1, \ldots, (k)^{1/3}\} \times \{1, 2, \ldots, (k)^{1/3}\} \cup \{(*, L), (L, *), (1, 0)\}$.

The following observations are immediate from the definition of original/reduced types.

Observation 14.

1. $|\mathsf{Types}| \leq k^{2/3} + 3 = \mathcal{O}(k^{2/3})$.
2. $C \in \mathsf{Pendants}(v)$, iff the original type of C is $(i, 0)$ for some $i \geq 1$. Otherwise, $C \in \mathsf{PeculiarNP}(v)$ iff the original type of C is (i, j) with $i, j > 0$.
3. The original/reduced type of a component entirely depends on its structural properties, and is *independent* of a drawing.

Definition 29 (Exceptional and nice components)**.** Let $\mathsf{Exceptional}(v, \mathcal{C}, \sigma) \coloneqq \mathsf{Heavy} \cup \mathsf{Extreme} \cup \mathsf{Surround}$, and let $\mathsf{nice}(v, \mathcal{C}, \sigma) \coloneqq \mathcal{C} \setminus \mathsf{Exceptional}(v, \mathcal{C}, \sigma)$. In the drawing σ, for each $(i, j) \in \mathsf{Types}$, the set of all *nice components* of reduced type (i, j) is denoted by $\mathcal{T}_\sigma(i, j)$, and for each $(i', j') \in \mathbb{N}^2$, the set of all nice components of original type (i', j') is denoted by $\mathcal{T}'_\sigma(i', j')$. [20]

From Observation 10, Observation 11, and Lemma 16, we immediately obtain the following corollary.

Corollary 1. $|\mathsf{Exceptional}| \leq 11 \cdot (k)^{2/3} = \mathcal{O}(k^{2/3})$.

[20] Note that the sets $\mathcal{T}_\sigma(\cdot, \cdot), \mathcal{T}'_\sigma(\cdot, \cdot)$ are subsets of $\mathsf{nice}(v, \mathcal{C}, \sigma)$ and hence depend on the drawing σ.

Rearrangement of Nice Components

Now we are left with analyzing the behavior of nice components. Note that by definition, each $C \in \mathsf{nice}(\mathcal{C})$ is a light component, and it does not surround any star vertex. We need the following definition to analyze the behavior of such nice components.

Definition 30 (Slot)**.** Let us now order the star vertices of Star from left to right as $\mathsf{st}_1, \mathsf{st}_2, \ldots, \mathsf{st}_q$, and we define set of vertices $\mathsf{slot}_i(\sigma) \coloneqq \{u \in V_2 : \sigma_2(\mathsf{st}_i) < \sigma_2(u) < \sigma_2(\mathsf{st}_{i+1})\}$ as an Star-*slot*, or simply a *slot*, defined by two consecutive star vertices $\mathsf{st}_i, \mathsf{st}_{i+1}$. We say that a component C *belongs to* a slot $\mathsf{slot}_i(\sigma)$ in σ, if $V(C) \cap V_2 \subseteq \mathsf{slot}_i(\sigma)$.

The following observation follows from the definition of $\mathsf{nice}(\mathcal{C})$.

Observation 15. Each $C \in \mathsf{nice}$ belongs to some slot $\mathsf{slot}_i(\sigma)$.

We further define the following notion – note that this is defined w.r.t. *any* drawing $\varsigma = (\varsigma_1, \varsigma_2, \varsigma_3)$, not *just* for a drawing σ fixed at the beginning of the section.

Definition 31 (Consecutiveness)**.**

- We say that a set of vertices Q is *consecutive* in ς if for each $i \in [3]$, $Q_i \coloneqq Q \cap V_i$ satisfies the following property: (a) either $|Q_i| \leq 1$, or (b) there exist two distinct vertices $u_l, u_r \in Q_i$, such that $Q_i = \{u \in V_i : \varsigma_i(u_l) \leq \varsigma_i(u) \leq \varsigma_i(u_r)\}$.
- We say that a collection of vertex-subsets $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_\ell\}$ is *consecutive* in ς, if (a) $\bigcup_{Q_j \in \mathcal{Q}} Q_j$ is consecutive in ς, and moreover (b) each $Q_j \in \mathcal{Q}$ is consecutive in ς. [21]

In the following definitions, we introduce the notion of a guess.

Definition 32. For each $C \in \mathsf{PeculiarNP}(v)$, let $\mathsf{opt}(C)$ denote the optimal number of crossings required to draw C on 2 lines. Further, for a subset $\mathcal{S} \subseteq \mathsf{PeculiarNP}(v)$, let $\mathsf{opt}(\mathcal{S}) \coloneqq \sum_{C \in \mathcal{S}} \mathsf{opt}(C)$.

Definition 33. A *component-blueprint* is a tuple $\mathsf{ComponentBlueprint} = (\mathsf{LeftNum}, \mathsf{RightNum}, \mathsf{optL}, \mathsf{optR}, \mathcal{E})$, where

- $\mathcal{E} \subseteq \mathcal{T}(L, *) \cup \mathcal{T}(*, L) \subseteq \mathsf{nice}(\mathcal{C})$ is a subset of components of size at most $k^{2/3}$.
- $\mathsf{LeftNum}, \mathsf{RightNum} : \mathsf{Types} \to \mathbb{N}$, where, for each $(i, j) \in \mathsf{Types}$, it holds that, $\mathsf{LeftNum}(i, j) + \mathsf{RightNum}(i, j) = |\mathcal{T}(i, j) \setminus \mathcal{E}|$.
- $\mathsf{optL}, \mathsf{optR} : \mathsf{Types} \to \mathbb{N}$, where, for each $(i, j) \in \mathsf{Types}$, it holds that, $\mathsf{optL}(i, j) + \mathsf{optR}(i, j) = \mathsf{opt}(\mathcal{T}(i, j) \setminus \mathcal{E})$.

Finally, we define the notion of blueprint-compliant guess.

Definition 34. A *component-guess* is a tuple $\mathsf{ComponentGuess} = (\mathsf{LeftComps}, \mathsf{RightComps})$ where, $\mathsf{LeftComps}, \mathsf{RightComps} : \mathsf{Types} \to \mathcal{C}$ where for each $(i, j) \in \mathsf{Types}$, $\mathsf{LeftComps}(i, j), \mathsf{RightComps}(i, j)$ are disjoint subsets of $\mathcal{T}(i, j)$.

Let $\mathsf{ComponentGuess} = (\mathsf{LeftComps}, \mathsf{RightComps})$ be a component-guess, and $\mathsf{ComponentBlueprint} = (\mathsf{LeftNum}, \mathsf{RightNum}, \mathsf{optL}, \mathsf{optR}, \mathcal{E})$ be a component-blueprint. We say that the guess $\mathsf{ComponentGuess}$ is $\mathsf{ComponentBlueprint}$-*compliant* if, for each $(i, j) \in \mathcal{T}$),

- $|\mathsf{LeftComps}(i, j)| = \mathsf{LeftNum}(i, j)$, and $|\mathsf{RightComps}(i, j)| = \mathsf{RightNum}(i, j)$,
- $\mathsf{opt}(\mathsf{LeftComps}(i, j)) = \mathsf{optL}(i, j)$, and $\mathsf{opt}(\mathsf{RightComps}(i, j)) = \mathsf{optR}(i, j)$.

Definition 35. Let $\mathcal{G} = (\mathsf{LeftComps}, \mathsf{RightComps})$ be a guess. We define a family $\mathsf{NiceCFamily}(\mathsf{ComponentBlueprint}, \mathsf{ComponentGuess}, \mathsf{Sep}, \sigma)$ to be a family of drawings σ' satisfying the following properties:

[21] Note that these two conditions together imply that the sets $Q_j \in \mathcal{Q}$ are placed "one after the other" in some arbitrary order, and no vertex not in any of Q_j's is placed anywhere "in between" the sets.

- σ and σ', when respected to $V(G) \setminus \bigcup_{(i,j)\in \mathsf{Types}}(\mathsf{LeftComps}(i,j) \cup \mathsf{RightComps}(i,j))$ are equal, and
- Let $V_L \coloneqq \bigcup_{(i,j)\in \mathsf{Types}} \mathsf{LeftComps}(i,j)$ and $V_R \coloneqq \bigcup_{(i,j)\in \mathsf{Types}} \mathsf{RightComps}(i,j)$. Then, for each $\ell \in \{2,3\}$, $\sigma_\ell(u) < \sigma_\ell(\mathsf{endpt}(\mathsf{lsep}_{23}, \ell))$ for each $u \in V_L \cap V_\ell$, and $\sigma_\ell(w) > \sigma_\ell(\mathsf{endpt}(\mathsf{rsep}_{23}, \ell))$ for each $w \in V_R \cap V_\ell$.

Finally, we state the main lemma of our section.

Lemma 17. *Let $\mathcal{I}$ be a k-minimal bottom-heavy* YES*-instance, let σ be a drawing with k crossings. Let $\mathsf{Sep}(\sigma)$ be a separator w.r.t. σ in (SB, DT) case, where $v \coloneqq \mathsf{endpt}(\mathsf{lsep}_{12}, 1) = \mathsf{endpt}(\mathsf{rsep}_{12}, 1)$. Then, there exists a blueprint $\mathsf{ComponentBlueprint}$, such that, for any $\mathsf{ComponentBlueprint}$-compliant guess $\mathsf{ComponentGuess}$, there exists a drawing σ' such that (i) $\sigma \in \mathsf{NiceCFamily}(\mathsf{ComponentBlueprint}, \mathsf{ComponentGu}$ and (ii) σ' is (Sep, σ)-compatible.*

Proof. First, let us introduce some notation that will be used throughout the lemma. Let $\mathsf{nice} \coloneqq \mathsf{nice}(\mathcal{C}) \subseteq \mathcal{C} = \mathsf{PeculiarNP}(v)$. Let $\mathsf{CompEdges} = \bigcup_{C\in \mathsf{nice}} E(C)$, and $\mathsf{ConnEdges} = \bigcup_{C\in \mathsf{nice}} \mathsf{ConnEdges}(C)$, where $\mathsf{ConnEdges}(C) \coloneqq \{vu : u \in C \cap V_2\}$.

For a drawing ς, we say that a component $C \in \mathsf{nice}$ is ς-left (resp. ς-right), if in the drawing ς, for $\ell \in \{2,3\}$, all the vertices of $C \cap V_\ell$ are to the left of $\mathsf{endpt}(\mathsf{lsep}_{23}, \ell)$ (resp. to the right of $\mathsf{endpt}(\mathsf{rsep}_{23}, \ell)$).

Transforming σ. In the beginning, σ' is initialized to σ. Throughout this proof, we will gradually transform σ by reordering the vertices involved in $R \coloneqq \bigcup_{C\in \mathsf{nice}} C$, while keeping the relative ordering of the rest of the vertices unchanged. This will be done in multiple steps. Before this, however, we set up things. Further, during this process, we will also ensure that, σ' remains (Sep, σ)-compatible (note that in the beginning, $\sigma' = \sigma$ and thus this property holds initially). Specifically, during each step of this process, we ensure the following:

1. Only σ'_2, σ'_3 are modified by only changing the relative ordering between the vertices of $R \cap V_2$ and $R \cap V_3$, respectively, whereas σ'_1 remains unchanged.
2. The sets of left- and right-star vertices w.r.t. $v, \mathcal{C}, \sigma'$ remain unchanged throughout the procedure.
3. The number of crossings remains the same, and moreover, the number of crossings on the left of $\mathsf{lsep}_{12}, \mathsf{lsep}_{23}$, and that to the right of $\mathsf{rsep}_{12}, \mathsf{rsep}_{23}$, remains the same.
4. for any nice component $C \in \mathsf{nice}(\mathcal{C})$, the number of crossings involving at least one edge of $E(C)$ does not increase.

At the end, when we will have achieved property 2, it follows that the final drawing σ' at that point will be a (Sep, σ)-well-behaved drawing.

This gradual transformation can be broadly divided into three steps, which we explain in detail below.

Step 1. Making each $C \in \mathsf{nice}$ consecutive. By Observation 15, each $C \in \mathsf{nice}$ belongs to a slot S_i. Now, let $e = uw \in E(C) \subseteq E_{23}$ (with $u \in V_2, w \in V_3$) be an edge crosses the minimum number of edges outside $E(C)$. Let $E_{out} \subseteq E_{23} \setminus E(C)$ denote the set of edges that cross e. Let $p^l(C) \in V_2 \setminus C$ (resp. $p^r(C) \in V_2 \setminus C$) be the rightmost (resp. leftmost) vertex in $V_2 \setminus C$ according to σ'_2 that is to the left (resp. right) of u. Similarly, define $q^l(C), q^r(C) \in V_3 \setminus C$ w.r.t. w and σ'_3. Then, we modify σ'_2 by moving all the vertices of $C \cap V_2$ between $p^l(C)$ and $p^r(C)$, while maintaining the same relative order among themselves. Similarly, we move all the vertices of $C \cap V_3$ between $q^l(C)$ and $q^r(C)$ while maintaining the same relative order among themselves. This ensures that C is consecutive in the resulting drawing ς.

We now argue that the number of crossings does not increase after the transformation. First, note that by Observation 15, $V(C) \cap V_2 \subseteq \mathsf{slot}_i$ for some slot slot_i, and we only move the vertices of $C \cap V_2$ within the same slot. Therefore, via an argument similar to Claim 5.2, for each edge $e_{12} \in \mathsf{ConnEdges}(C)$, the set of edges crossing e_{12} does not change due to the modification. Further, since we maintain the same relative order among the vertices of $V(C) \cap V_2$ and $V(C) \cap V_3$, the set of edges of $E(C)$ that cross any edge e_{23} also does not change. On the other hand, for each $e_{23} \in E(C)$, the set of edges from $E_{23} \setminus E(C)$ is now exactly equal to E_{out}. Due to the choice of e, it follows that the number of edges crossing any e_{23} does not increase after the transformation. However, due to k-minimality of $\mathcal{I}$, it follows that the number of crossings before and after the transformation must remain the same. After iteratively transforming σ' in this manner for each $C \in \mathsf{nice}$, in the final drawing σ', we ensure the following properties for each $C \in \mathsf{nice}$:

1. C is consecutive in σ'. Further, if C was σ-left (resp. σ-right), then C is σ'-left (resp. σ-right).
2. All the vertices in $V(C) \cap V_2$ belong to the interval $\left\{p \in V_2 : \sigma'_2(p^l(C)) < \sigma'_2(p) < \sigma' 2(p^r(C))\right\}$ defined by the vertices $p^l(C), p^r(C)$,
3. All the vertices in $V(C) \cap V_3$ belong to the interval $\left\{q \in V_2 : \sigma'_2(q^l(C)) < \sigma'_2(q) < \sigma'_2(q^r(C))\right\}$ defined by the vertices $q^l(C), q^r(C)$, and
4. For any distinct $C, C' \in \mathsf{nice}$, the edges of $E(C)$ and $E(C')$ do not cross each other.

These properties imply that, the final drawing σ' obtained at the end of step 1 is (Sep, σ)-compatible, where σ is the original drawing from the beginning of the proof. We refer to the pair of intervals defined in items 2 and 3 above as the *sub-slot* defined by $p^l(C), p^r(C), q^l(C), q^r(C)$, and denote it by $\mathsf{subslot}(C)$. We note that, for any $C \in \mathsf{nice}$, since $E(C) \cup \{uv : u \in C\}$ does not contain any edge of $\mathsf{CrossingEdges} \cup \mathsf{MiddleEdges}$, it holds that either (i) for each $u \in \mathsf{lsep}_{23}(C) \cap V_i$, $\sigma'_i(u) < \sigma'_i(\mathsf{endpt}(\mathsf{lsep}_{23}, i))$, for $i \in \{2, 3\}$ or (ii) for each $u \in \mathsf{lsep}_{23}(C) \cap V_i$, $\sigma'_i(u) > \sigma'_i(\mathsf{endpt}(\mathsf{rsep}_{23}, i))$, for $i \in \{2, 3\}$. Informally, either the entire subslot—and hence—is to the left of lsep_{12}, or to the right of rsep_{23}.

Further, let $E_{out}(C) \subseteq E_{23} \setminus E(C)$ to be the set of all edges that cross all of the edges of E_{23} in σ'.

Step 2. Constructing an Extended Component Blueprint. Recall that each reduced type $(i, j) \in \mathsf{Types} \setminus \{(L, *), (*, L)\}$ is the same as the corresponding original type; whereas the two special types $(L, *), (*, L)$ consist of multiple original types. We first extend the notion of a component-blueprint to original types, instead of reduced types. Note that an important difference here is that, this definition does not the "exceptional set" $\mathcal{E}$.

Definition 36. An *extended component-blueprint* is a tuple

$$\mathsf{ExtComponentBlueprint} = (\mathsf{LeftNum}, \mathsf{RightNum}, \mathsf{optL}, \mathsf{optR}),$$

such that,

- $\mathsf{LeftNum}, \mathsf{RightNum} : \mathbb{N}^2 \to \mathbb{N}$, where for each original type $(i', j') \in \mathbb{N}^2$, it holds that $\mathsf{LeftNum}(i', j') + \mathsf{RightNum}(i', j') = |\mathcal{T}'(i', j')|$.
- $\mathsf{optL}, \mathsf{optR} : \mathbb{N}^2 \to \mathbb{N}$, where for each original type $(i', j') \in \mathbb{N}^2$, it holds that $\mathsf{optL}(i', j') + \mathsf{optR}(i', j') = \mathsf{opt}(\mathcal{T}'(i', j'))$.

We similarly define the notion of an extended component-guess, and the compliance thereof with an extended component-blueprint.

Next, we will look at the current σ' to define an $\mathsf{ExtComponentBlueprint}^o = (\mathsf{LeftNum}^o, \mathsf{RightNum}^o, \mathsf{optL}^o, \mathsf{optR}^o)$ that will be used as a reference point. To this end, define for each original type $(i', j') \in \mathbb{N}^2$,

- $(\mathsf{LeftComps}^o(i', j'), \mathsf{RightComps}^o(i', j'))$ denote the partition of $\mathcal{T}'(i', j')$, where $\mathsf{LeftComps}^o(i', j')$ is the set of components that are σ'-left, and $\mathsf{RightComps}^o(i', j')$ is the set of components that are σ'-right.

- $\mathsf{LeftNum}^o(i',j') \coloneqq |\mathsf{LeftComps}^o(i',j')|$, and $\mathsf{RightNum}^o(i',j') \coloneqq |\mathsf{RightComps}^o(i',j')|$.
- $\mathsf{optL}^o(i',j') \coloneqq \mathsf{opt}(\mathsf{LeftComps}^o(i',j'))$, and $\mathsf{optR}^o(i',j') \coloneqq \mathsf{opt}(\mathsf{RightComps}^o(i',j'))$.

Now, consider any $\mathsf{ExtComponentGuess} = (\mathsf{LeftComps}, \mathsf{RightComps})$ that is $\mathsf{ExtComponentBlueprint}^o$-compliant. In this step, we will iteratively modify σ' such that, for each $(i',j') \in \mathbb{N}^2$, all the components in $\mathsf{LeftComps}(i',j')$ are σ'-left, and are placed consecutively, and similarly, all the components in $\mathsf{RightComps}(i',j')$ are σ'-right, and are placed consecutively.

We iterate over all original types $(i',j') \in \mathbb{N}^2$ in an arbitrary order. Fix one such $(i',j') \in \mathbb{N}^2$, and let $C^l(i',j') \coloneqq \arg\min_{C \in \mathsf{LeftComps}(i',j')} |\mathsf{SlotEdges}(C)| \cdot i' + |E_{out}(C)| \cdot j'$, where $\mathsf{SlotEdges}(C) \subseteq E_{12}$ denotes the set of edges that cross any edge of the form vu, where u belongs to the same as C. Let $\mathsf{subslot}^l(i',j') \coloneqq \mathsf{subslot}(C(i',j'))$. We transform σ' as follows: for each $C' \in \mathsf{LeftComps}(i',j')$, we move all the vertices of C' to the $\mathsf{subslot}^l(i',j')$ while maintaining the same relative order of vertices of C'. We place all such components one after the other in an arbitrary order in this subslot. We perform a similar transformation for $\mathsf{RightComps}(i',j')$. Note that after completely processing each original type (i',j'), the resultant drawing σ'' at the end of the iteration, is (Sep, σ')-compatible, where σ' is the drawing before the start of the iteration.

Note that for $(i',j') \in \left\{1, 2, \ldots, (k)^{1/3}\right\}^2$, the original and reduced types are the same. Hence, this step makes all such $\mathsf{LeftComps}(i',j'), \mathsf{RightComps}(i',j')$ consecutive. For each (i',j'), let $c^l_{12}(i',j') \coloneqq |\mathsf{SlotEdges}(C^l(i,j))|, c^l_{23}(i',j') \coloneqq |E_{out}(C^l(i,j)|$, where $C^l(i,j)$ is the component that is used to relocate all other components. Analogously, define $c^r_{12}(i',j')$ and $c^r_{23}(i',j')$.

Step 3. Handling reduced type $(*, L)$, $(L, *)$. Let $T_{orig}(L,*)$ be the set of all original types whose corresponding reduced type is $(L,*)$. Note that these consist of original types (i',j'), where $i < k^{1/3}$ and $j \geq (k)^{1/3}$. Also define $T_{orig}(*,L)$ as the set of original types (i',j') with $i \geq (k)^{1/3}$.

We first consider the left side. We classify the original types $T_{orig}(L,*)$ into two sub-categories $(i',j') \in T_{orig}(i',j')$ is *left-category 0* if $c^l_{12}(i',j') = 0$; and *left-category 1* if $c^l_{12}(i',j') \geq 1$. Similarly, classify the original types $T_{orig}(*,L)$ into two sub-categories as follows: $(i',j') \in T_{orig}(*,L)$ is *left-category 0* if $c^l_{23}(i',j') = 0$; and *left-category 1* if $c^l_{23}(i',j') \geq 1$. Similarly, define the notion of right-category 0 and right-category 1, respectively.

First, we show how to handle left-category 0 types from $(L,*)$. Let these original types be $(i_1,j_1),(i_2,j_2),\ldots,(i_q,j_q)$. By definition, for each $1 \leq p \leq q$, $c^l_{12}(i_p,j_p) = 0$. Let $\ell = \arg\min_{1 \leq p \leq q} c^l_{23}(i_p,j_p)$. We move all components in $\mathsf{LeftComps}(i_p,j_p)$ to the subslot containing all the components of $\mathsf{LeftComps}(i_\ell,j_\ell)$, while preserving the relative order within the components of $\mathsf{LeftComps}(i_p,j_p)$ for each $1 \leq p \leq q$, as well as relative order of vertices within each component. Note that, for each component $C \in \mathsf{LeftComps}(i_p,j_p)$, the number of crossings from outside $E(C)$, before the movement is $c_{23}(i_p,j_p)$, and that after the movement is $c_{23}(i_\ell,j_\ell) \leq c_{23}(i_p,j_p)$. Therefore, we do not increase the number of crossings in this transformation.

We handle left-category 0 types from $(*,L)$, along with reduced type $(1,0)$ in a very similar manner, which we describe for completeness. Let $(i_1,j_1),(i_2,j_2),\ldots,(i_q,j_q)$, indicate the original types corresponding to $(*,L)$ such that $1 \leq p \leq q$, $c^l_{23}(i_p,j_p) = 0$. Let $\ell = \arg\min_{1 \leq p \leq q} c^l_{12}(i_p,j_p)$. We move all components in $\mathsf{LeftComps}(i_p,j_p)$ to the subslot containing all the components of $\mathsf{LeftComps}(i_\ell,j_\ell)$, while preserving the relative order within the components of $\mathsf{LeftComps}(i_p,j_p)$ for each $1 \leq p \leq q$, as well as relative order of vertices within each component. Note that, for each component $C \in \mathsf{LeftComps}(i_p,j_p)$, the number of crossings of $\mathsf{ConnEdges}$ from outside, before the movement is $c^l_{12}(i_p,j_p)$, and that after the movement is $c^l_{12}(i_\ell,j_\ell) \leq c^l_{12}(i_p,j_p)$. Therefore, we do not increase the number of crossings in this transformation.

We do a similar transformation for right-category 0 types from $T_{orig}(L,*)$ and $T_{orig}(*,L)$ to move the components $\mathsf{RightComps}(\cdot,\cdot)$ of the respective types. We omit the description.

Finally, we handle left/right-category 1 types from $(*, L)$ and $(L, *)$ by showing that the total number of components belonging to such types is bounded by $(k)^{2/3}$. Formally, we show the following claim.

Claim 5.6.

$$\sum_{\substack{(i,j):\max(i,j)\geq(k)^{1/3} \\ (i,j) \text{ is left/right-category 1}}} |\mathcal{T}'(i,j)| \leq (k)^{2/3}.$$

Proof of Claim 5.6. We prove that each component C of original type (i, j) with (i, j) being left/right-category 1, accounts for a distinct set of $(k)^{1/3}$ crossings. Since the total number of crossings is k, this will show the claim.

First consider $C \in \mathsf{LeftComps}(i', j')$ with $(i', j') \in T_{orig}(L, *)$, such that (i', j') is left-category 1 (the proof for right-category 1 is analogous). Since (i', j') is left-category 1, $c^l_{12}(i, j) \geq 1$. Note that $|\mathsf{ConnEdges}| \geq k^{1/3}$. By the previous step, all the components in $\mathsf{LeftComps}'(i', j')$ belong to a single slot, and by Claim 5.2, all the edges in $\mathsf{ConnEdges}$ cross the same non-empty set of edges – where the non-emptiness follows from $c^l_{12}(i, j) \geq 1$. Thus, there are $i \geq (k)^{1/3}$ such edges, and each edge is involved in at least $(k)^{1/3}$ crossings.

Now let us consider $C \in \mathsf{LeftComps}(i', j')$, where $(i', j') \in T_{orig}(*, L)$ and (i', j') is left-category 1 (the proof for right-category 1 is analogous). Since (i', j') is category 1, $c^l_{23}(i', j') \geq 1$. Further, all the components in $\mathcal{T}'(i', j')$ belong to a single slot. Further, by step 1, the edges of different components $C, C' \in \mathsf{LeftComps}(i', j')$ do not cross each other; and all the edges of $\bigcup_{C\in\mathsf{LeftComps}(i',j')} E(C)$ cross the same non-empty set of edges not in $\bigcup_{C\in\mathsf{LeftComps}(i',j')} E(C)$ – here, the non-emptiness follows from $c^l_{23}(i', j') \geq 1$. Thus, for each $C \in \mathsf{LeftComps}(i', j')$, each edge in $E(C)$ is crossed at least once, and there are at least $j \geq (k)^{1/3}$ edges in $E(C)$, which amounts to at least $(k)^{1/3}$ crossings for each $C \in \mathsf{LeftComps}(i', j')$. □

Let $\mathcal{E}$ denote the set of all components $C \in \mathcal{T}'(i', j')$, where (i', j') is left/right-category 1. Claim 5.6 shows that $|\mathcal{E}| \leq k^{2/3}$. Now, we use this $\mathcal{E}$ to obtain a component-blueprint and component-guess.

Recall the extended component-blueprint $\mathsf{ExtComponentBlueprint}^o = (\mathsf{LeftNum}^o, \mathsf{RightNum}^o, \mathsf{optL}^o, \mathsf{optR}^o)$ defined earlier. Now, define $\mathsf{ComponentBlueprint}^p = (\mathsf{LeftNum}^p, \mathsf{RightNum}^p, \mathsf{optL}^p, \mathsf{optR}^p, \mathcal{E})$, where for each $(i, j) \in \mathsf{Types} \setminus \{(L, *), (*, L)\}$, define $\mathsf{LeftNum}^p(i, j) = \mathsf{LeftNum}^o(i, j)$, $\mathsf{RightNum}^p(i, j) = \mathsf{RightNum}^o(i, j)$, $\mathsf{optL}^p(i, j) = \mathsf{optL}^o(i, j)$, and $\mathsf{optR}^p(i, j) = \mathsf{optR}^o(i, j)$. On the other hand, for $(i, j) \in \{(L, *), (*, L)\}$, define

- $\mathsf{LeftNum}^p(i, j) = \sum_{(i',j')T_{orig}(i,j) \text{ and left-category } 0} \mathsf{LeftNum}^o(i', j')$,
- $\mathsf{RightNum}^p(i, j) = \sum_{(i',j')\in T_{orig}(i,j) \text{ and right-category } 0} \mathsf{RightNum}^o(i', j')$,
- $\mathsf{optL}^p(i, j) = \sum_{(i',j')T_{orig}(i,j) \text{ and left-category } 0} \mathsf{optL}^o(i', j')$,
- $\mathsf{optR}^p(i, j) = \sum_{(i',j')T_{orig}(i,j) \text{ and left-category } 0} \mathsf{optR}^o(i', j')$.

Now, notice that, any $\mathsf{ComponentBlueprint}^p$-compliant guess $\mathsf{ComponentGuess}^p$ can be extended to $\mathsf{ExtComponentBlueprint}^o$-compliant $\mathsf{ExtComponentGuess}^o$, by considering any partition of the category 1 types with appropriate constraints, which is essentially a partition of components in $\mathcal{E}$. Then, applying the 3-step proof as described above, we can show that, for the $\mathsf{ComponentBlueprint}^p$ as defined above, for any $\mathsf{ComponentBlueprint}^p$-compliant $\mathsf{ComponentGuess}$, the drawing σ' obtained at the end belongs to $\mathsf{NiceCFamily}(\mathsf{ComponentBlueprint}^p, \mathsf{ComponentGuess}, \mathsf{Sep}, \sigma)$, and also is (Sep, σ)-compatible. This completes the proof.

□

5.4 Step 2: Guessing the Interfaces

In this step, we will "guess" various subsets of edges and vertices that can be used in the creation of two sub-instances in the next step. Each of the guesses are made by supposing that $\mathcal{I}$ is a k-minimal YES-instance with an unknown drawing σ. Further, we describe the algorithm corresponding to bottom-heavy case, i.e., when $k_{12} \geq k_{23}$, which implies that $k_{12} \geq k/2$ – top-heavy case is symmetric by exchanging the roles of lines 1 and 3. These guesses will be made in multiple steps, with each step being dependent on all possible choices of the previous steps. Further, if any of the guesses in a subsequent step contradicts a guess done at a previous step, then such a guess is deemed invalid and is terminated. These guesses can be thought of *branching* on each of the specified choices in all steps. If all the guesses are terminated, or the corresponding recursive calls report that the sub-instances are NO-instances, then we will conclude that our assumption that $\mathcal{I}$ was a YES-instance was wrong, and thus we must have a NO-instance.

In each step, we specify at a high level the kind of object that we are trying to guess. Whenever we are trying to guess a subset of edges, we follow the following convention: all the edges of a multi-edge are treated in the same manner, such as being chosen in a guess.

Step A: Separator. We start by guessing the separator $\mathsf{Sep} = \{\mathsf{lsep}_{12}, \mathsf{rsep}_{12}, \mathsf{lsep}_{23}, \mathsf{rsep}_{23}\}$ w.r.t. a hypothetical (unknown) drawing σ. Note that there are at most $m^4 = n^{\mathcal{O}(1)}$ choices. Furthermore, based on our guess, we can also conclude whether we are in (DB, DT), (DB, ST), (SB, DT), or (SB, ST) case. Next, we guess $\mathsf{MiddleEdges}(\mathsf{Sep}) \subseteq E_{12}$ of size at most $2\sqrt{k}$, and two sets $\mathsf{GapLeft}, \mathsf{GapRight} \subseteq V_2^{conn}$ of size at most $3\sqrt{k}$, as in the definition of Definition 17. There are at most $n^{\mathcal{O}(\sqrt{k})}$ choices for this. Overall, we have at most $n^{\mathcal{O}(\sqrt{k})}$ choices.

Step B: Labeling $\mathsf{Endpoints}(\mathsf{BoundaryEdges}) \cup \mathsf{Endpoints}(\mathsf{Sep}) \cup \mathsf{GapLeft} \cup \mathsf{GapRight}$. The vertices of $\mathsf{Endpoints}(\mathsf{leftbound}_{12})$ and $\mathsf{Endpoints}(\mathsf{leftbound}_{23})$ receive labels $\mathcal{L}$. The vertices of $\mathsf{Endpoints}(\mathsf{lsep}_{12}) \cup \mathsf{Endpoints}(\mathsf{lsep}_{23}) \cup \mathsf{GapLeft}$ are each given two labels $\mathcal{L}, \mathcal{M}$. Similarly, the vertices of $\mathsf{Endpoints}(\mathsf{rsep}_{12}) \cup \mathsf{Endpoints}(\mathsf{rsep}_{23}) \cup \mathsf{GapRight}$ are given two labels $\mathcal{M}, \mathcal{R}$. Note that the assignment of labels in this step does not overwrite previous labels. For example, in SB/ST case the shared bottom (resp. top) endpoint of $\mathsf{lsep}_{12}, \mathsf{lsep}_{23}$ (resp. $\mathsf{lsep}_{23}, \mathsf{rsep}_{23}$) gets all three labels. Similarly, if $\mathsf{lsep}_{12}, \mathsf{lsep}_{23}$ share top endpoint, then the shared endpoint will also receive all three labels. Note that there is no guessing involved in this step.

Step C: CrossingEdges and labeling their endpoints. For each edge $e \in \mathsf{Sep}$, we guess $\mathsf{cross}(e)$, which is a set of at most $\sqrt{k}$ edges that cross e. Let $\mathsf{CrossingEdges} := \bigcup_{e \in \mathsf{Sep}} \mathsf{cross}(e)$.

We now describe feasible pairs for labels of the endpoints of CrossingEdges; and each guess corresponds to using labeling the endpoint of each edge in CrossingEdges by one of the feasible (ordered) pairs. However, if the guess for the label is *incompatible* with a previously assigned label, then we consider it as an invalid guess and proceed to the next one. Note that, if a vertex has previously been assigned multiple labels, then the incompatible label(s) is a label that has not been assigned to the said vertex, if any.

- For $e \in \mathsf{cross}(\mathsf{lsep}_{12}) \cup \mathsf{cross}(\mathsf{lsep}_{23})$, the feasible label pairs are $(\mathcal{L}, \mathcal{M}), (\mathcal{M}, \mathcal{L}), (\mathcal{L}, \mathcal{R}), (\mathcal{R}, \mathcal{L})$.
- For $e \in \mathsf{cross}(\mathsf{rsep}_{12}) \cup \mathsf{cross}(\mathsf{rsep}_{23})$, the feasible label pairs are $(\mathcal{R}, \mathcal{M}), (\mathcal{M}, \mathcal{R}), (\mathcal{L}, \mathcal{R}), (\mathcal{R}, \mathcal{L})$.

Overall, we have at most $\binom{m}{4\sqrt{k}}^4 \cdot 2^{\mathcal{O}(\sqrt{k})} = n^{\mathcal{O}(\sqrt{k})}$ choices. Further, let $\mathsf{CrossLeft}, \mathsf{CrossMid}, \mathsf{CrossRight} \subseteq \mathsf{Endpoints}(\mathsf{CrossingEdges})$ denote the set of vertices that receive labels $\mathcal{L}, \mathcal{M}, \mathcal{R}$, respectively.

Step D: Edges incident to $\mathsf{GapLeft} \cup \mathsf{GapRight}$.

- Consider an edge $uv \in E_{12} \setminus \mathsf{CrossingEdges}$, where $v \in \mathsf{GapLeft}$ (resp. $v \in \mathsf{GapRight}$). Then, we label u as $\mathcal{L}$ (resp. $\mathcal{R}$).

- Consider an edge $uv \in E_{23} \setminus \mathsf{CrossingEdges}$, where $v \in \mathsf{GapLeft} \cup \mathsf{GapRight}$. Then, we label u as $\mathcal{M}$.

Step E: Handling $\mathsf{comps}(v)$ for $v \in \mathsf{Endpoints}(\mathsf{Sep})$. Recall that for each $v \in \mathsf{Endpoints}(\mathsf{Sep})$, each component $C \in \mathsf{comps}(v)$ can be classified into $\mathsf{Special}(v), \pi\text{-}\mathsf{Defined}(v), \mathsf{UpDown}(v), \mathsf{PeculiarNP}(v)$, and $\mathsf{PeculiarNP}(v)$ according to Definition 19. Now we describe how to handle each of the components separately.

▶ $\mathsf{Special}(v)$**.** Some of the vertices in such components are already labeled. We will return to the remaining vertices in these components at a later step.

▶ $\pi\text{-}\mathsf{Defined}(v)$**.** Recall that each π_i for $i \in [3]$ is comprised of $\lambda = \mathcal{O}(\log k^\star)$ total orders $\pi_i^1, \pi_i^2, \ldots, \pi_i^\lambda$ for $V_i^1, V_i^2, \ldots, V_i^\lambda$, respectively. For each $i \in [3]$, and for each $j \in [\lambda]$, we do the following.

- We guess two vertices $u(j,l), w(j,r) \in V_i^j$ with $\pi_i^j(u(j,l)) < \pi_i^j(w(j,r))$.
- Any $C \in \pi\text{-}\mathsf{Defined}(v)$ that contains a vertex $u \in V_i^j$ with $\pi_i^j(u) \leq \pi_i^j(u(j,l))$, we label all the vertices of C using $\mathcal{L}$.
- Any $C \in \pi\text{-}\mathsf{Defined}(v)$ that contains a vertex $w \in V_i^j$ with $\pi_i^j(w) \geq \pi_i^j(w(j,r))$, we label all the vertices of C using $\mathcal{R}$.

If any of the guesses made in this step is incompatible with a previous step, or with a prior guess made in this step, then we terminate this guess. Note that the number of guesses made in this step is at most $n^{\mathcal{O}(\log k^\star)} = n^{\mathcal{O}(k^{2/3})}$ since $k = \Omega((k^\star)^{2/3})$.

▶ $\mathsf{UpDown}(v)$**.** By Lemma 12, the number of such components in an YES-instance is bounded by $8\sqrt{k}$; otherwise we terminate this guess. Otherwise, for each such component, we guess a label from $\{\mathcal{L}, \mathcal{R}\}$ and use that label for all the vertices of the component. The number of guesses is $n^{\mathcal{O}(\sqrt{k})}$.

▶ $\mathsf{Pendants}(v)$ **for all** $v \in \mathsf{Endpoints}(\mathsf{Sep})$**.**

- We guess a pendant-blueprint

$$\mathsf{PendantBlueprint} = \{(\mathsf{LeftPNum}(v), \mathsf{MidPNum}(v), \mathsf{RightPNum}(v)) : v \in \mathsf{Endpoints}(\mathsf{Sep})\}$$

  in accordance with Definition 22. Note that the number of guesses for pendant-blueprints is bounded by $n^{\mathcal{O}(1)}$.
- Now, fix a $\mathsf{PendantBlueprint}$. For each $v \in \mathsf{Endpoints}(\mathsf{Sep})$, we use a variant of 3D knapsack to find a partition (if one exists) $(\mathsf{LeftPendants}(v), \mathsf{MidPendants}(v), \mathsf{RightPendants}(v))$ of $\mathsf{Pendants}(v)$ for each $v \in \mathsf{Endpoints}(\mathsf{Sep})$ in polynomial time, such that the resulting $\mathsf{PendantGuess}$ is compatible with $\mathsf{PendantBlueprint}$ (as defined in Definition 34). If such a partition does not exist, then we terminate the guess.
- Then, for each $v \in \mathsf{Endpoints}(v)$, we label each pendant in $\mathsf{LeftPendants}(v), \mathsf{MidPendants}(v)$, and $\mathsf{RightPendants}(v)$ with label $\mathcal{L}, \mathcal{M}$ and $\mathcal{R}$, respectively.

Note that the number of guesses in this step is bounded by $n^{\mathcal{O}(1)}$, and otherwise the step takes polynomial time.

At this point, for all $v \in \mathsf{Endpoints}(\mathsf{Sep}, 2)$, we have labeled all the components in $\mathsf{comps}(v)$ (other $\mathsf{Special}(v)$ as mentioned above). Thus, we are left with handling $\mathsf{PeculiarNP}(v)$ for $v \in \mathsf{Endpoints}(\mathsf{Sep}, 1) \cup \mathsf{Endpoints}(\mathsf{Sep}, 3)$.

▶ $\mathsf{PeculiarNP}(v)$ **for all** $v \in \mathsf{Endpoints}(\mathsf{Sep})$**.** We consider different cases based.

(i) (DB, DT) case: Here, $v \in \mathsf{Endpoints}(\mathsf{Sep}, 1) \cup \mathsf{Endpoints}(\mathsf{Sep}, 3)$ can be labeled with $\mathcal{L}/\mathcal{R}$ as appropriate in accordance with Lemma 14.

(ii) (SB, ST) case, Lemma 13, we can guess two subsets of $P_1 \subseteq \mathsf{PeculiarNP}(\mathsf{endpt}(\mathsf{lsep}_{12}, 1))$ and $P_2 \subseteq \mathsf{PeculiarNP}(\mathsf{endpt}(\mathsf{lsep}_{23}, 3))$, each of size at most $\sqrt{k}$. Then, we consider the following two possibilities (guesses).

1. Label all components of P_1 using $\mathcal{L}$, and all components of $\mathsf{PeculiarNP}(\mathsf{endpt}(\mathsf{lsep}_{12}, 1)) \setminus P_1$ using $\mathcal{R}$.
   Label all components of P_2 using $\mathcal{R}$, and label all components of $\mathsf{PeculiarNP}(\mathsf{endpt}(\mathsf{lsep}_{23}, 3)) \setminus P_2$ using $\mathcal{L}$.
2. Label all components of P_1 using $\mathcal{R}$, and all components of $\mathsf{PeculiarNP}(\mathsf{endpt}(\mathsf{lsep}_{12}, 1)) \setminus P_1$ using $\mathcal{L}$.
   Label all components of P_2 using $\mathcal{L}$, and label all components of $\mathsf{PeculiarNP}(\mathsf{endpt}(\mathsf{lsep}_{23}, 3)) \setminus P_2$ using $\mathcal{R}$.

(iii) (SB, DT) case. For the rest of the description of step E, let $v := \mathsf{endpt}(\mathsf{lsep}_{12}, 1) = \mathsf{endpt}(\mathsf{rsep}_{12}, 1)$, and $\mathcal{C} := \mathsf{PeculiarNP}(v)$.

- We guess a subset $\mathsf{Exceptional} \subseteq \mathcal{C}$ of size at most $11 \cdot k^{2/3}$ as in Lemma 16. Note that there are at most $n^{\mathcal{O}(k^{2/3})}$ guesses. Let $\mathcal{C}' := \mathcal{C} \setminus \mathsf{Exceptional}$.
- For each $C \in \mathcal{C}'$, we use the two layer algorithm of Theorem 6 to find an optimal number of crossings, $\mathsf{opt}(C) \leq k$ (if $\mathsf{opt}(C) > k$, then clearly we have a No-instance). This takes time $|V(C)|^{\mathcal{O}(\sqrt{k})} = n^{\mathcal{O}(\sqrt{k})}$ (note that we skip the kernelization step), and obtain a drawing of C.
- According to Definition 19, we classify $\mathcal{C}'$ into reduced types, and use $\mathcal{T}(i,j) \subseteq \mathcal{C}'$ to denote the set of components of type (i,j). Note that the number of reduced types is at most $k^{2/3}+2$.
- Now, we guess a component-blueprint $\mathsf{ComponentBlueprint}$ according to Definition 33. Note that the number of guesses for a blueprint is bounded by $n^{\mathcal{O}(k^{2/3})}$. Fix one such $\mathsf{ComponentBlueprint} = (\mathsf{LeftNum}, \mathsf{RightNum}, \mathsf{optL}, \mathsf{optR}, \mathcal{E})$.
- We try to construct a $\mathsf{ComponentBlueprint}$-compliant $\mathsf{ComponentGuess} = (\mathsf{LeftComps}, \mathsf{RightComps})$ as follows: for each reduced type $(i,j) \in \mathsf{Types}$, we try to find a partition of $\mathcal{T}(i,j) \setminus \mathcal{E}$ into $(\mathsf{LeftComps}(i,j), \mathsf{RightComps}(i,j))$ using a version of 4D knapsack (this runs in polynomial time) that satisfies the following two conditions:
  - $|\mathsf{LeftComps}(i,j)| = \mathsf{LeftNum}(i,j)$, and $|\mathsf{RightComps}(i,j)| = \mathsf{RightNum}(i,j)$
  - $\mathsf{opt}(\mathsf{LeftComps}(i,j)) = \mathsf{optL}(i,j)$, and $\mathsf{opt}(\mathsf{RightComps}(i,j)) = \mathsf{optR}(i,j)$.

  If such $\mathsf{ComponentGuess}$ does not exist, then we terminate the current guess.
- Otherwise, for each component $C \in \bigcup_{(i,j)\in\mathsf{Types}} \mathsf{LeftComps}(i,j)$, we label it using $\mathcal{L}$, and for each component $C \in \bigcup_{(i,j)\in\mathsf{Types}} \mathsf{RightComps}(i,j)$, we label it using $\mathcal{R}$.

(iv) (DB, ST) case: this case is handled symmetrically as (SB, DT) case as explained above in (iii). We omit the description.

Step F: Label propagation. At this point, some of the vertices of the graph may still be unlabeled.

Let A denote the set of vertices that have already received a label, and let $A^{\mathcal{L}}, A^{\mathcal{M}}, A^{\mathcal{R}} \subseteq A$ denote the subsets of vertices that have received label $\mathcal{L}, \mathcal{M}, \mathcal{R}$, respectively (note that, if a vertex has multiple labels, then it belongs to multiple subsets). Further, let $O \subseteq A$ denote the subset of vertices that have only received one label, and again $O^{\mathcal{L}}, O^{\mathcal{M}}, O^{\mathcal{R}}$ denote the corresponding partition of O.

Next, we define three sets of vertices, as follows.

$$\begin{aligned}
\mathsf{Origin}^{\mathcal{L}} &:= \mathsf{LeftExtBnd} \cup \mathsf{Endpoints}(\{\mathsf{leftbound}_{12}, \mathsf{leftbound}_{23}, \mathsf{lsep}_{12}, \mathsf{lsep}_{23}\}) \cup \mathsf{GapLeft} \cup \mathsf{CrossLeft} \\
\mathsf{Origin}^{\mathcal{R}} &:= \mathsf{RightExtBnd} \cup \mathsf{Endpoints}(\{\mathsf{rsep}_{12}, \mathsf{rsep}_{23}, \mathsf{rightbound}_{12}, \mathsf{rightbound}_{23}\}) \cup \mathsf{GapRight} \cup \mathsf{CrossRight} \\
\mathsf{Origin}^{\mathcal{M}} &:= \mathsf{Endpoints}(\mathsf{Sep}) \cup \mathsf{Endpoints}(\mathsf{MiddleEdges}) \cup \mathsf{GapLeft} \cup \mathsf{GapRight} \cup \mathsf{CrossMid} \qquad (3)
\end{aligned}$$

Now, we proceed as follows.

1. For each $u \in \mathsf{Origin}^{\mathcal{L}}$, and each unlabeled $w \notin A$ that is reachable from u in the graph $G - (O^{\mathcal{M}} \cup O^{\mathcal{R}})$, we label it using $\mathcal{L}$.

2. For each $u \in \mathsf{Origin}^{\mathcal{R}}$, and each unlabeled $w \notin A$ that is reachable from u in the graph $G - (O^{\mathcal{L}} \cup O^{\mathcal{M}})$, we label it using $\mathcal{R}$.
3. For each $u \in \mathsf{Origin}^{\mathcal{M}}$, and each unlabeled $w \notin A$ that is reachable from u in the graph $G - (O^{\mathcal{L}} \cup O^{\mathcal{R}})$, we label it using $\mathcal{M}$.

During this process, we terminate the guess, if any of the following happens.

- A vertex $w \notin A$ receives two different labels.
- A vertex $w \in V_1 \setminus A$ that has at least one neighbor in V_2 receives label $\mathcal{M}$.
- A vertex $w \in V_2 \setminus A$ that has at least one neighbor in V_1 receives label $\mathcal{M}$.

Otherwise, let $V^{\mathcal{T}}$ denote the set of vertices that have received the label $\mathcal{T}$, for $\mathcal{T} \in \{\mathcal{L}, \mathcal{M}, \mathcal{R}\}$, respectively. Note that these sets are not disjoint.

Step G: Partial Order. For each $i \in [3]$, we will guess permutation $\pi_i^{\lambda+1}$ (total order) of a certain subset of vertices of V_i in the drawing σ, which will be used to extend π_i to a new partial order π_i'. Let $S \coloneqq \mathsf{Endpoints}(\mathsf{BoundaryEdges})\mathsf{Endpoints}(\mathsf{Sep}) \cup \mathsf{Endpoints}(\mathsf{MiddleEdges}) \cup \mathsf{Endpoints}(\mathsf{CrossingEdges}) \cup \mathsf{GapLeft} \cup \mathsf{GapRight}$. From the bounds on the respective sets of vertices and edges, it is straightforward to check that $|S| = \mathcal{O}(\sqrt{k})$. For each $i \in [3]$, let $S_i \coloneqq V_i \cap S$. We guess permutations $\pi_i^{\lambda+1}$ of S_i for each $i \in [3]$. Since $|S_i| = \mathcal{O}(\sqrt{k})$, it follows that the total number of guesses is $|S_1|! \cdot |S_2|! \cdot |S_3|! = (\mathcal{O}(\sqrt{k})!)^3 = 2^{\mathcal{O}(\sqrt{k}\log k)}$. Next, we define $\pi_2^{\lambda+2}$ as an ordering over $\mathsf{ExtBnd} \cup \mathsf{Endpoints}(\mathsf{Sep}, 2) \cup \mathsf{GapLeft} \cup \mathsf{GapRight}$ by combining parts of $\pi_2^{\lambda+1}$, and one of the constituent total orders of π^2 of the set ExtBnd. Let $\pi^{\lambda+1} = (\pi_1^{\lambda+1}, \pi_2^{\lambda+1}, \pi_3^{\lambda+1})$. We say that $\pi^{\lambda+1}$ is *valid* if the following conditions are satisfied:

- For each $i \in [3]$, for any distinct $u, v \in S_i$ with $\pi_i^{\lambda+1}(u)$ π_i defines an order between u and v, then both π_i and $\pi_i^{\lambda+1}$ are order u and v consistently. [22]
- $\pi^{\lambda+1}$ satisfies the following properties (that must satisfied by the separator):
  - $\mathsf{lsep}_{12}, \mathsf{rsep}_{12}$ do not cross, $\mathsf{rsep}_{12}, \mathsf{rsep}_{23}$ do not cross
  - For each $e \in \mathsf{Sep}$, the set of edges crossing e is exactly equal to $\mathsf{cross}(e)$,
  - $\mathsf{MiddleEdges}$ is exactly the set of edges that lie between $\mathsf{lsep}_{12}, \mathsf{rsep}_{12}$

Note that the validity of $\pi^{\lambda+1}$ can be checked in polynomial time. If a guess for $\pi_i^{\lambda+1}$ is invalid, then we terminate it and move to the next guess. Otherwise, for each $i \in \{1, 3\}$, let $\pi_i' = \pi_i \cup \pi_i^{\lambda+1}$, and $\pi_2' = \pi_2 \cup \pi_2^{\lambda+1} \cup \pi_2^{\lambda+2}$, and note that π_i' is a valid extension of π_i, since $\pi^{\lambda+1}$ is valid. Henceforth, we assume that we are working with such a $\pi' = (\pi_1', \pi_2', \pi_3')$ for some guess for $\pi^{\lambda+1}$.

Step H: Parameter division. First, let k_{special} denote the number of crossings (with multiplicity) of the following form according to the prior guess:

- $e_1 \in \mathsf{cross}(\mathsf{lsep}_{23}), e_2 \in \mathsf{cross}(\mathsf{rsep}_{23})$ such that e_1 and e_2 cross according to their ordering in $\pi^{\lambda+1}$.

Let k_{current} denote the number of crossings among the edges of $\mathsf{CrossingEdges} \cup \mathsf{MiddleEdges} \cup \mathsf{Sep}$ according to $\pi_i^{\lambda+1}$, except those counted in k_{special}. If $k_{\mathsf{current}} > k$, then we terminate the current guess. Otherwise, let $k' \coloneqq k - k_{\mathsf{current}}$. Let k' denote the final value. We guess three non-negative integers $0 \leq k^{\mathcal{L}}, k^{\mathcal{M}}, k^{\mathcal{R}}$ such that $k^{\mathcal{L}}, k^{\mathcal{R}} \leq 3k/4$, and $k^{\mathcal{L}} + k^{\mathcal{M}} + k^{\mathcal{R}} = k'$. Further, we also guess $k_{12}^{\mathcal{L}}, k_{23}^{\mathcal{L}}, k_{12}^{\mathcal{R}}, k_{23}^{\mathcal{R}} \geq 0$ such that $k^{\mathcal{L}} = k_{12}^{\mathcal{L}} + k_{23}^{\mathcal{L}}$, and $k^{\mathcal{R}} = k_{12}^{\mathcal{R}} + k_{23}^{\mathcal{R}}$. Note that the number of such guesses is bounded by $k^{\mathcal{O}(1)}$.

From the discussion above, the following lemma follows.

[22] Note that, it is possible that u and v are incomparable in π_i, but since they both belong to S_i, they are comparable in $\pi_i^{\lambda+1}$.

Lemma 18. *The total number of guesses made in steps A through H, for (i) separator* Sep *and its associated sets of vertices and edges, (ii) labels of vertices of* $V(G)$*, (iii) partial orders* $\pi' i$*, and (iv) division of* k *into smaller parameters, is bounded by* $n^{\mathcal{O}(k^{2/3})}$*.*

In the subsequent steps, we assume that we are working with one such fixed guess for the separator Sep and its associated objects, labels of vertices, partial orders $\pi' = (\pi'_1, \pi'_2, \pi'_3)$, and the smaller parameters.

5.5 Step 3: Creating sub-instances.

For each set of guesses made in the previous step (as stated in Lemma 18), we create three sub-instances (in fact, extended instances), (i) a left sub-instance $\mathcal{I}^{\mathcal{L}}$, and (ii) a right sub-instance $\mathcal{I}^{\mathcal{R}}$, and (iii) a middle sub-instance $\mathcal{I}^{\mathcal{M}}$. Among these, $\mathcal{I}^{\mathcal{L}}$ and $\mathcal{I}^{\mathcal{R}}$ are instances of 3-Layer Crossing Minimization, whereas $\mathcal{I}^{\mathcal{M}}$ is an instance of 2-Layer Crossing Minimization. In this step (Step 3), we describe the construction of $\mathcal{I}^{\mathcal{L}}$ and $\mathcal{I}^{\mathcal{R}}$, and in the next step (Step 4) we will recursively call our algorithm to find a drawing for each of the two sub-instances; or to conclude that either or both of them are No-instances, in which case we terminate the guess. Then, in Step 5, we will use the drawings of $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}$ found by the recursive algorithm to construct the middle sub-instance $\mathcal{I}^{\mathcal{M}}$ of 2-Layer Crossing Minimization, which will be solved using the algorithm of Theorem 6.

Let $V^{\mathcal{L}}, V^{\mathcal{M}}, V^{\mathcal{R}}$ denote the set of vertices that have received labels $\mathcal{L}, \mathcal{M}, \mathcal{R}$, respectively – note that, if a vertex has received multiple labels, then it belongs to all the respective subsets. Now, we proceed to the description of $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}$.

Creating Left Sub-instance

▶ **Base graph.** Let $G'^{\mathcal{L}} = G[V^{\mathcal{L}}]$, with its edge-set partitioned into regular and multi-edges, denoted by $\mathsf{RegularEdges}^{\mathcal{L}}$ and $\mathsf{WeightedEdges}^{\mathcal{L}}$, respectively.

▶ **Boundary edges.** Let $\mathsf{BoundaryEdges}^{\mathcal{L}} := \left\{\mathsf{leftbound}^{\mathcal{L}}_{12}, \mathsf{rightbound}^{\mathcal{L}}_{12}, \mathsf{rightbound}^{\mathcal{L}}_{12}, \mathsf{rightbound}^{\mathcal{L}}_{23}\right\}$, where $\mathsf{leftbound}^{\mathcal{L}}_{12} := \mathsf{leftbound}_{12}, \mathsf{leftbound}_{23} := \mathsf{leftbound}_{23}, \mathsf{rightbound}^{\mathcal{L}}_{12} := \mathsf{lsep}_{12}, \mathsf{rightbound}^{\mathcal{L}}_{23} := \mathsf{lsep}_{23}$.

▶ **Extended Boundary.** Let $\mathsf{LeftExtBnd}^{\mathcal{L}} := \mathsf{LeftExtBnd} \cap V^{\mathcal{L}}$, $\mathsf{RightExtBnd}^{\mathcal{L}} := \mathsf{GapLeft} \setminus \mathsf{LeftExtBnd}^{\mathcal{L}}$. Then, $\mathsf{ExtBnd}^{\mathcal{L}} = \mathsf{LeftExtBnd}^{\mathcal{L}} \cup \mathsf{RightExtBnd}^{\mathcal{L}}$.

▶ **Multi-edges** Next, we add a few more multi-edges to the set $\mathsf{WeightedEdges}^{\mathcal{L}}$, as follows:
For each (multi-)edge $uv \in \mathsf{CrossingEdges}$ with at least one endpoint in $V^{\mathcal{L}}$, we identify a multi-edge $e_{add}(uv)$ to be added in each case. After the case analysis, we will discuss the multiplicity of this edge, and actually add it to the graph.

- $uv \in E_{12}$, where $u \in V_1$ has a label $\mathcal{L}$ and $v \in V_2$ has label $\mathcal{M}$ or $\mathcal{R}$.
  Note that, since $uv \in \mathsf{CrossingEdges}$, the vertex v cannot belong to GapLeft and hence cannot have a label $\mathcal{L}$ – this is ensured while guessing the labels.
  Operation. $e_{add}(uv) := (u, \mathsf{endpt}(\mathsf{lsep}_{12}, 2))$.
- $vu \in E_{12}$, where $u \in V_2$ has label $\mathcal{L}$ and $v \in V_1$ has label $\mathcal{M}$ or $\mathcal{R}$.
  Operation. $e_{add}(uv) := (u, \mathsf{endpt}(\mathsf{lsep}_{12}, 1))$ to $\mathsf{WeightedEdges}^{\mathcal{L}}$.
- $vu \in E_{23}$, where $u \in V_3$ has label $\mathcal{L}$ and $v \in V_2$ has label $\mathcal{M}$ or $\mathcal{R}$.
  Note that the vertex v may belong to $\mathsf{GapLeft} \subseteq V^{\mathcal{L}}$, and u cannot be equal to $\mathsf{endpt}(\mathsf{lsep}_{23}, 3)$.
  Operation. If $v \in \mathsf{GapLeft} \cup \{\mathsf{endpt}(\mathsf{lsep}_{12}, 2)\}$, then we also delete the edge vu from $E(G^{\mathcal{L}})$. $e_{add} := (\mathsf{endpt}(\mathsf{lsep}_{23}, 2), u)$.

- $uv \in E_{23}$, where $u \in V_2$ has label $\mathcal{L}$ and $v \in V_3$ has label $\mathcal{M}$ or $\mathcal{R}$.
  Note that, since $uv \in \mathsf{CrossingEdges}$, the vertex u cannot belong to $\mathsf{GapLeft}$, and hence cannot have a label $\mathcal{L}$ – this is ensured while guessing the labels.
  Operation. $e_{add}(uv) \coloneqq (u, \mathsf{endpt}(\mathsf{lsep}_{23}, 3))$.

Now, we proceed as follows:

- If an edge is defined as e_{add} of multiple edges, say $e_1, e_2, \ldots$, then its multiplicity is defined as the sum of the respective multiplicities of $e_1, e_2, \ldots$ – here we adopt the convention that, if an edge e_i belongs to $\mathsf{RegularEdges}$, then its multiplicity is 1.
- Finally, we add e_{add} with the combined multiplicity to $\mathsf{WeightedEdges}^{\mathcal{L}}$. Here, if e_{add} is already in $\mathsf{RegularEdges}$, we do not remove it.

Let $G^{\mathcal{L}}$ denote the resulting graph after adding the edges in this manner.

▶ **Vertex set.** For each $i \in [3]$, let $V_i^{\mathcal{L}} \coloneqq V^{\mathcal{L}} \cap V_i$, and let $\pi_i^{\mathcal{L}}$ denote π_i' projected to the vertices of $V^{\mathcal{L}}$.

▶ **Partial order** Let $\pi^{\mathcal{L}} = (\pi_1^{\mathcal{L}}, \pi_2^{\mathcal{L}}, \pi_3^{\mathcal{L}})$. Observe that $(\pi^{\mathcal{L}})_2^{\lambda+2}$ contains an ordering of $\mathsf{ExtBnd}^{\mathcal{L}}$ that satisfies the required properties.

▶ **Final instance.** Let $\mathcal{I}^{\mathcal{L}} = (G^{\mathcal{L}}, V_1^{\mathcal{L}}, V_2^{\mathcal{L}}, V_3^{\mathcal{L}}, \mathsf{ExtBnd}^{\mathcal{L}}, \pi^{\mathcal{L}}, k_1^{\mathcal{L}}, k_2^{\mathcal{L}}, k^{\mathcal{L}})$.

At this point, we check whether in the graph $G^{\mathcal{L}}$, each vertex of $V(G^{\mathcal{L}})$ is reachable from some vertex of $\mathsf{ExtBnd}^{\mathcal{L}} \cup \mathsf{Endpoints}(\mathsf{BoundaryEdges})$ in $G^{\mathcal{L}}$. If not, we terminate the guess. Otherwise, we continue to the next step.

At this point, we note the following observations akin to the two-layer algorithm.

- As required in Item 4, the total number of partial orders in each π_i remains bounded by $\mathcal{O}(\log k^\star)$, since parameter reduces by at least a constant factor in each step.
- Note that $|\mathsf{LeftExtBnd}^{\mathcal{L}}| \leq |\mathsf{LeftExtBnd}| \leq \alpha\sqrt{k^\star}$, and $|\mathsf{LeftExtBnd}^{\mathcal{R}}| \leq |\mathsf{GapLeft}| \leq 3\sqrt{k}$, and analogous bounds hold for $|\mathsf{RightExtBnd}^{\mathcal{R}}|, |\mathsf{LeftExtBnd}^{\mathcal{R}}|$, respectively. Thus, $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}$ satisfy Item 3.
- Using a similar argument as in two layer case, it follows that the total sum of multiplicities of $\mathsf{WeightedEdges}$incident to each vertex in $\mathsf{Endpoints}(\mathsf{Sep})$ remains bounded by $2\sqrt{k^\star}$, satisfying Item 6

Creating Right Sub-instance

The right sub-instance $\mathcal{I}^{\mathcal{R}} = (G^{\mathcal{R}}, V_1^{\mathcal{R}}, V_2^{\mathcal{R}}, V_3^{\mathcal{R}}, \mathsf{ExtBnd}^{\mathcal{R}}, \pi^{\mathcal{R}}, k_1^{\mathcal{R}}, k_2^{\mathcal{R}}, k^{\mathcal{R}})$ is created analogously, and we omit the description. Again, we check whether there exists some vertex of $V(G^{\mathcal{R}})$ that is not reachable from some vertex of $\mathsf{Endpoints}(\mathsf{BoundaryEdges}^{\mathcal{R}}) \cup \mathsf{ExtBnd}^{\mathcal{R}}$ in $G^{\mathcal{R}}$, and terminate the guess. Otherwise, we proceed to the next step.

From the preceding constructions of $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}$—assuming that the guesses are not terminated—it is straightforward to verify that they satisfy the properties of extended instances from Definition 14. Thus, we have the following claim, whose proof is omitted.

Claim 5.7. *$\mathcal{I}^{\mathcal{L}}$ and $\mathcal{I}^{\mathcal{R}}$ are valid extended instances of* 3-Layer Crossing Minimization.

5.6 Step 4: Conquer left and right.

For each guess made in Step 2, we have defined a pair of sub-instances $\mathcal{I}^{\mathcal{L}}$ and $\mathcal{I}^{\mathcal{R}}$ in Step 3, with each having parameter at most $3k/4$. For each such pair of instances, which we solve recursively in time $T_3(k^{\mathcal{L}}) + T_3(k^{\mathcal{R}}) \leq 2T_3(3k/4) \leq 2 \cdot n^{\mathcal{O}((3k/4)^{2/3})}$. There are two possibilities. (i) At least one of the recursive calls returns that the corresponding sub-instance is a No-instance, in which case we proceed to the next guess. (ii) Otherwise, if both of the recursive calls return that $\mathcal{I}^{\mathcal{L}}$ and $\mathcal{I}^{\mathcal{R}}$ are Yes-instances along with the respective drawings $\sigma^{\mathcal{L}}, \sigma^{\mathcal{R}}$, then we proceed to the next step.

5.7 Step 5: Conquer Middle.

For each $i \in [3]$, let $M_i := V_i \cap V^{\mathcal{M}} \cap S$, where $S = \mathsf{Endpoints(Sep)} \cup \mathsf{Endpoints(MiddleEdges)} \cup \mathsf{Endpoints(CrossingEdges)} \cup \mathsf{GapLeft} \cup \mathsf{GapRight}$ as defined in step G. Note that M_i is a subset of the set S_i defined in Step H, for which we guess the permutation $\pi_i^{\lambda+1}$.

Note that, the relative order among the vertices of M_i for each $i \in [3]$ is already determined by $\pi_i^{\lambda+1}$. In particular, note that that $M_1 = V^{\mathcal{M}} \cap V_1$, and $\pi_1^{\lambda+1}$ completely determines the ordering of M_1. Now we are left with determining the order of vertices in $V^{\mathcal{M}} \cap V_2$ and $V^{\mathcal{M}} \cap V_3$ that respects the partial order π'. To this end, we define an instance of 2-Layer Crossing Minimization, as follows.

Defining a two-layer instance. Note that the base graph for the instance of 2-Layer Crossing Minimization will be defined on the vertex subsets $V^{\mathcal{M}} \cap V_2$ and $V^{\mathcal{M}} \cap V_3$, respectively. Later, we will rename them to first and second layer, respectively.

▶ **Base graph.** Let $G'^{\mathcal{M}} = G[V^{\mathcal{M}} \cap (V_2 \cap V_3)]$, with its edge-set partitioned into regular and multi-edges, denoted by $\mathsf{RegularEdges}^{\mathcal{M}}$ and $\mathsf{WeightedEdges}^{\mathcal{M}}$, respectively.

▶ **Extended boundary.** Let $\mathsf{LeftExtBnd}^{\mathcal{M}} := \mathsf{GapLeft} \cup \mathsf{endpt}(\mathsf{lsep}_{12}, 2) \cup \mathsf{endpt}(\mathsf{lsep}_{23}, 2)$, and $\mathsf{RightExtBnd}^{\mathcal{M}} := \mathsf{GapRight} \cup \mathsf{endpt}(\mathsf{rsep}_{12}, 2) \cup \mathsf{endpt}(\mathsf{rsep}_{23}, 2)$. Then, $\mathsf{ExtBnd}^{\mathcal{M}} := \mathsf{LeftExtBnd}^{\mathcal{M}} \cup \mathsf{RightExtBnd}^{\mathcal{M}}$.

▶ **Boundary edges.** Let $\mathsf{BoundaryEdges}^{\mathcal{M}} := \left\{\mathsf{leftbound}^{\mathcal{M}}, \mathsf{rightbound}^{\mathcal{M}}\right\}$, where $\mathsf{leftbound}^{\mathcal{M}} := \mathsf{lsep}_{23}$, $\mathsf{rightbound}^{\mathcal{L}} := \mathsf{rsep}_{23}$.

▶ **Multi-edges.** For each (multi-)edge $uv \in \mathsf{CrossingEdges} \cap E_{23}$ with at least one endpoint in $V^{\mathcal{M}}$, we add a multi-edge to the graph $G^{\mathcal{M}}$, based on the following cases.

- $vu \in E_{23}$, where $u \in V_3$ has label $\mathcal{L}$ and $v \in V_2$ has label $\mathcal{M}$ or $\mathcal{R}$.
  Note that the vertex v may belong to $\mathsf{GapLeft} \subseteq V^{\mathcal{L}}$, and u cannot be equal to $\mathsf{endpt}(\mathsf{lsep}_{23}, 3)$.
  Operation. First, if $v \in \mathsf{GapLeft} \cup \{\mathsf{endpt}(\mathsf{lsep}_{12}, 2)\}$, then we also delete the edge vu from $E(G^{\mathcal{L}})$. We add a multi-edge $(\mathsf{endpt}(\mathsf{lsep}_{23}, 2), u)$ to $\mathsf{WeightedEdges}^{\mathcal{M}}$.
- $uv \in E_{23}$, where $u \in V_2$ has label $\mathcal{L}$ and $v \in V_3$ has label $\mathcal{M}$ or $\mathcal{R}$.
  Note that, since $uv \in \mathsf{CrossingEdges}$, the vertex u cannot belong to $\mathsf{GapLeft}$, and hence cannot have a label $\mathcal{L}$ – this is ensured while guessing the labels.
  Operation. We add a new multi-edge $(u, \mathsf{endpt}(\mathsf{lsep}_{23}, 3))$ to $\mathsf{WeightedEdges}^{\mathcal{M}}$.

For each edge considered in each of the preceding cases, if the original edge $uv \in \mathsf{WeightedEdges}$, then the newly added edge gets multiplicity equal to $\mu(uv)$; otherwise, it gets multiplicity equal to 1.
Let $G^{\mathcal{M}}$ denote the resulting graph after adding the edges in this manner.

▶ **Vertex set and partial Order.** Let $W_1 := V^{\mathcal{M}} \cap V_2$ and $W_2 := V^{\mathcal{M}} \cap V_3$, and let $\pi_1^{\mathcal{M}} = \pi_2'$ projected to the vertices of W_1, and $\pi_2^{\mathcal{M}} = \pi_3'$ projected to the vertices of W_2. Let $\pi^{\mathcal{M}} = (\pi_1^{\mathcal{M}}, \pi_2^{\mathcal{M}})$. Note that, $\mathsf{GapLeft}$ (resp. $\mathsf{GapRight}$) must contain all the vertices of V_2^{conn} that are

between $\mathsf{endpt}(\mathsf{lsep}_{23}, 2)$ and $\mathsf{endpt}(\mathsf{lsep}_{12}, 2)$ (resp. $\mathsf{endpt}(\mathsf{rsep}_{12}, 2)$ and $\mathsf{endpt}(\mathsf{rsep}_{23}, 2)$). Thus, if $(\pi^{\mathcal{M}})_1^{\lambda+2}$, which is $\pi_2^{\lambda+2}$ projected to V_2 if violates this condition, then we terminate the guess.

▶ **Final instance.** Let $\mathcal{I}^{\mathcal{M}} = (G^{\mathcal{M}}, W_1^{\mathcal{M}}, V_2^{\mathcal{M}}, \pi^{\mathcal{M}}, k^{\mathcal{M}})$.

Finally, we check whether each vertex of $V(G^{\mathcal{M}})$ is reachable from some vertex of $\mathsf{ExtBnd}^{\mathcal{M}} \cup \mathsf{Endpoints}(\mathsf{BoundaryEdges}^{\mathcal{M}})$ in the constructed graph $G^{\mathcal{M}}$. If not, we terminate the guess. Otherwise, it can be verified that the constructed instance satisfies all the properties of an elaborate extended instance of 2-Layer Crossing Minimization from Definition 14, which we state in the following claim.

Claim 5.8. *$\mathcal{I}^{\mathcal{M}}$ is a valid elaborate extended instance of* 2-Layer Crossing Minimization.

Given this claim, we call the recursive algorithm of Theorem 6 (without kernelizing) to solve it in time $T_2(n, k) \leq n^{\mathcal{O}(\sqrt{k})}$. Either, the algorithm returns that $\mathcal{I}^{\mathcal{M}}$ is a No-instance, or it returns a drawing $\sigma^{\mathcal{M}}$. In the former case, we terminate the guess. Otherwise, we proceed to the following final step.

5.8 Step 6: Obtaining the Final Drawing.

If, for any of the non-terminated guesses, all three recursive calls reporting that the respective sub-instances are Yes-instances, along with the corresponding orderings, then we can combine these orderings to obtain orderings of V_1, V_2 that has at most k crossings, and respect the given partial order π.

Specifically, suppose $\sigma^{\mathcal{L}}, \sigma^{\mathcal{M}}, \sigma^{\mathcal{R}}$ be the drawings of $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}$ returned by the corresponding recursive calls. Then, we obtain the final drawing of $\mathcal{I}$ by defining $\varsigma_i := \sigma_i^{\mathcal{L}} \circ \sigma_i^{\mathcal{M}} \circ \sigma_i^{\mathcal{R}}$ for $i \in [3]$. We return $\varsigma = (\varsigma_1, \varsigma_2, \varsigma_3)$ as our final drawing. In the following lemma, we show that the combined drawing is a valid drawing of $\mathcal{I}$ with k crossings.

Lemma 19. *Let $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}, \mathcal{I}^{\mathcal{M}}$ be the three sub-instances defined in Steps 3 and 5 (with respective parameters $k^{\mathcal{L}}, k^{\mathcal{R}}, k^{\mathcal{M}}$) corresponding to some guess made in Step 2. Further, let $\sigma^{\mathcal{L}}, \sigma^{\mathcal{R}}, \sigma^{\mathcal{M}}$ be valid drawings of the $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}, \sigma^{\mathcal{M}}$. Then, if we obtain a combined drawing σ' as in Step 6, then σ' is a valid drawing of $\mathcal{I}$.*

Proof. Recall that, in step H we guess the permutations $\pi_i^{\lambda+1}$ of S_i, and we use it in step I to determine the number of crossings k_{current} among the edges of $\mathsf{CrossingEdges} \cup \mathsf{MiddleEdges} \cup \mathsf{Sep}$. Then, we define $k' = k - k_{\mathsf{current}}$, and divide it into $k^{\mathcal{L}} + k^{\mathcal{M}} + k^{\mathcal{R}}$, which are used as parameters for the respective sub-instances.

Next, we claim that all the k_{current} crossings among the edges are not counted in any of the sub-instances. To this end, first observe that the edges of $\mathsf{MiddleEdges}$ are not involved in any of the subproblems, since the instance $\mathcal{I}^{\mathcal{M}}$ is only defined over layers 2 and 3. The proof is analogous to that of Lemma 7, except that we need to be careful about the following case. Consider $e_1, e_2 \in \mathsf{CrossingEdges}$ where $e_1 \in \mathsf{cross}(\mathsf{lsep}_{23})$, $e_2 \in \mathsf{cross}(\mathsf{rsep}_{23})$ and suppose e_1, e_2 cross according to $\pi_i^{\lambda+1}$. Then, note that the crossing between e_1, e_2 is a special crossing as defined in Step H, which is not counted in k_{current}. Therefore, it implies that any crossing between $e_1, e_2 \in \mathsf{CrossingEdges}$, that contributes towards k_{current} must satisfy that e_1, e_2 cross the same edge of Sep (corresponding to case (b) in Lemma 7). Then the rest of the proof follows.

Next, we note that in $\mathcal{I}^{\mathcal{L}}$, the left boundary edges ($\mathsf{leftbound}_{12}, \mathsf{leftbound}_{23}$) and left extended boundary ($\mathsf{LeftExtBnd}$) are derived from the current instance $\mathcal{I}$, and thus are placed at the appropriate locations. An analogous property holds about $\mathcal{I}^{\mathcal{R}}$. Next, we note that the orderings $\pi^{\mathcal{L}}, \pi^{\mathcal{M}}, \pi^{\mathcal{R}}$ is obtained by the projecting π' (which is an extension of π) to the respective vertex subsets. Since the respective drawings are compatible with these orders, it follows that the

combined drawing is compatible with π', and hence with π. Furthermore, since π' also contains the orderings $\pi^{\lambda+1}$, the respective drawings also satisfy the ordering on the vertices of $\mathsf{GapLeft} \cup \mathsf{GapRight} \cup \mathsf{ExtBnd} \cup \mathsf{Endpoints}(\mathsf{Sep}, 2)$.

Finally, since $\sigma^{\mathcal{L}}, \sigma^{\mathcal{M}}, \sigma^{\mathcal{R}}$ are valid drawings for $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{M}}, \mathcal{I}^{\mathcal{R}}$, when we obtain the combined drawing σ by gluing the three drawings, it is straightforward to verify that $\mathsf{GapLeft}$ and $\mathsf{GapRight}$ are placed consecutively at appropriate locations between the vertices of $\mathsf{Endpoints}(\mathsf{Sep}, 2)$, and all of the conditions of Definition 15 are satisfied. This shows that σ' is a valid drawing for $\mathcal{I}$.

□

Otherwise, for each guess, if either it is terminated, or at least one sub-instance is a No-instance, then we return that the current extended instance is a No-instance.

5.9 Proof of Correctness

Lemma 20. *$\mathcal{I}$ is an extended* Yes*-instance of* 3-Layer Crossing Minimization *for parameter k iff the algorithm returns a drawing σ'.*

Proof. In the base case when $k \leq c\sqrt{k^\star}$, or $\min\{|E_{12}|, |E_{23}|\} \leq c\sqrt{k^\star}$, the correctness of the algorithm follows via Lemma 10. Thus, we consider the recursive case. Suppose inductively that the statement is true for all values of parameter strictly smaller than $k > c\sqrt{k^\star}$ (with no assumption on $\min\{|E_{12}|, |E_{23}|\}$), and we want to show that the statement holds for parameter k.

Forward direction. Suppose $\mathcal{I}$ is a k-minimal Yes-instance, and let σ be a drawing of $\mathcal{I}$ with k crossings. Consider the guess where the following objects are correctly guessed in the algorithm:

1. Separator Sep as guaranteed by Lemma 11, the corresponding edge-sets $\mathsf{CrossingEdges}(\mathsf{Sep})$, $\mathsf{MiddleEdges}(\mathsf{Sep})$, vertex-sets $\mathsf{GapLeft}, \mathsf{GapRight}$, and the relative ordering of the vertices involved in these sets,

2. For each $v \in \mathsf{Endpoints}(\mathsf{Sep})$, by Lemma 12, $|\mathsf{UpDown}(v)| \leq 8\sqrt{k}$. Then, for each component $C \in \mathsf{UpDown}(v)$, if all the vertices of $C \cap V_i$ belong to the left of $\mathsf{endpt}(\mathsf{lsep}, i)$ (resp. to the right of $\mathsf{endpt}(\mathsf{rsep}, i)$) for each $i \in [3]$, then consider the guess where it is labeled $\mathcal{L}$ (resp. $\mathcal{R}$).

3. For each $i \in [3]$, and $j \in [\lambda]$, the vertices $v(j,l), v(j,r) \in V_i^j$ that are immediately to the left of $\mathsf{endpt}(\mathsf{lsep}_{23}, 2)$ and to the right of $\mathsf{endpt}(\mathsf{rsep}_{23}, 2)$, respectively. Note that for each connected component $C \in \pi\text{-}\mathsf{Defined}(v)$ containing a vertex $u \in V_i^j$ with $\pi_i^j(u) \leq \pi_i^j(v(j,l))$ (resp. $\pi_i^j(u) \leq \pi_i^j(v(j,r))$), all the vertices of C must belong to $V(\sigma, \mathsf{Sep}, \mathcal{L})$ (resp. $V(\sigma, \mathsf{Sep}, \mathcal{R})$). Hence, all the vertices belonging to $\pi\text{-}\mathsf{Defined}(v)$ also get correctly labeled.

4. Consider the blueprint $\mathsf{PendantBlueprint}$ guaranteed by Lemma 15. Since our algorithm tries all possible pendant-blueprints, consider the guess where it considers such $\mathsf{PendantBlueprint}$. Then, it finds a $\mathsf{PendantGuess}$ that is compatible with $\mathsf{PendantBlueprint}$. Then, Lemma 15 implies that for any guess $\mathsf{PendantGuess}$ that is $\mathsf{PendantBlueprint}$-compliant, and in particular the $\mathsf{PendantGuess}$ that is constructed in the algorithm, there exists a drawing $\sigma' \in \mathsf{NicePFamily}(\mathsf{PendantBlueprint}, \mathsf{Pe}$ that is (Sep, σ)-separator-compatible. Henceforth, we rename $\sigma' \leftarrow \sigma$ and proceed to the next step in the analysis.

5. Next, if we are in (DB, DT) or (ST, SB) case, consider where we correctly guess at most $\mathcal{O}(\sqrt{k})$ components of $\mathsf{PeculiarNP}(v)$ for $v \in \mathsf{Endpoints}(\mathsf{Sep}, 1) \cup \mathsf{Endpoints}(\mathsf{Sep}, 3)$ in accordance with Lemma 14, along with their labels.
   Otherwise, in (SB, DT) case, consider the $\mathsf{ComponentGuess}$ guaranteed by Lemma 17, and again, since our algorithm tries all possible pendant-blueprints, consider the guess where it considers such $\mathsf{ComponentBlueprint}$. Then, it finds a $\mathsf{ComponentGuess}$ that is compatible with $\mathsf{ComponentBlueprint}$. Then, Lemma 17 implies that for any guess $\mathsf{ComponentGuess}$ that is

ComponentBlueprint-compliant, and in particular the ComponentGuess that is constructed in the algorithm, there exists a drawing

$$\sigma' \in \mathsf{NiceCFamily}(\mathsf{ComponentBlueprint}, \mathsf{ComponentGuess}, \mathsf{Sep}, \sigma)$$

that is (Sep, σ)-separator-compatible. Again, we rename $\sigma \leftarrow \sigma'$. The other case, (DB, ST) is handled analogously.

Let σ be the drawing obtained at the end after appropriate redefintions as described above. In the following claim, we show that the rest of the labeling process correctly labels all the vertices.

Claim 5.9. *Let σ be the drawing after appropriate redfinitions as described above, and consider the guess as described above. Then, after the vertex labeling process of step H corresponding to this particular guess, we have that $V^{\mathcal{T}} = V(\sigma, \mathsf{Sep}, \mathcal{T})$, for each $\mathcal{T} \in \{\mathcal{L}, \mathcal{M}, \mathcal{R}\}$. Note that the definition of the sets $V(\sigma, \mathsf{Sep}, \mathcal{T})$ can be found in Definition 18.*

Proof Claim 5.9. Let us consider a vertex $u \in V(\sigma, \mathsf{Sep}, \mathcal{L})$. By Item 3 of Definition 14, u is reachable from some vertex $v \in \mathsf{ExtBnd} \cup \mathsf{Endpoints}(\mathsf{BoundaryEdges})$, say via a path $P(uv)$. We consider different cases based on v and a path $P(uv)$. Let $P(uv) = \langle u = w_0, w_1, w_2, \ldots, w_\ell = v \rangle$. Let w_{i+1} be the first vertex (if any) along $P(uv)$, such $w_0, w_1, \ldots, w_i \in V(\sigma, \mathsf{Sep}, \mathcal{L})$, and $w_{i+1} \notin V(\sigma, \mathsf{Sep}, \mathcal{L})$. Otherwise, let $w_i = w_\ell = v$. We consider these two cases separately.

(i) If $w_i = v$, then since $w_i \in V(\sigma, \mathsf{Sep}, \mathcal{L})$, it cannot be the case that $w_i \in (\mathsf{RightExtBnd} \cup \mathsf{Endpoints}(\{\mathsf{rightbound}_{12}, \mathsf{rightbound}_{23}\})) \setminus \mathsf{Endpoints}(\mathsf{Sep})$. Thus, it implies that $w_i \in \mathsf{LeftExtBnd} \cup \mathsf{Endpoints}(\{\mathsf{leftbound}_{12}, \mathsf{leftbound}_{23}\}) \cup \mathsf{CrossLeft}$.

(ii) If $w_i \neq v$, then we know that w_{i+1}, the vertex immediately after w_i in $P(uv)$ satisfies that $w_{i+1} \notin V(\sigma, \mathsf{Sep}, \mathcal{L})$. Then, it follows that either the edge (w_i, w_{i+1}) crosses some edge of Sep–implying that $w_i \in \mathsf{CrossLeft}$, or that $w_{i+1} \in V_3$, and $w_i \in \mathsf{GapLeft}$.

In any case, for any vertex $u \in V(\sigma, \mathsf{Sep}, \mathcal{L}) \setminus \mathsf{Origin}^{\mathcal{L}}$, and any path $P(uv)$ to a vertex $v \in \mathsf{ExtBnd} \cup \mathsf{Endpoints}(\mathsf{BoundaryEdges})$, there exists a vertex $w_i \in P(uv)$ with label $\mathcal{L}$, such that $w_i \in \mathsf{Origin}^{\mathcal{L}}$, and further the entire subpath $P(uw_i)$ does not contain any vertex in $V(\sigma, \mathsf{Sep}, \mathcal{L}) \cup V(\sigma, \mathsf{Sep}, \mathcal{R})$. This also implies that, all the vertices of $V(\sigma, \mathsf{Sep}, \mathcal{L})$ are reachable from some vertex $w_i \in \mathsf{Origin}^{\mathcal{L}}$ in the graph $G - (O^{\mathcal{M}} \cup O^{\mathcal{R}})$. Thus, all the unlabeled vertices in $V(\sigma, \mathsf{Sep}, \mathcal{L})$ get labeled $\mathcal{L}$ in the label propagation phase. This shows that $V(\sigma, \mathsf{Sep}, \mathcal{L}) \subseteq V^{\mathcal{L}}$. Similar proofs also show that $V(\sigma, \mathsf{Sep}, \mathcal{L}) \subseteq V^{\mathcal{M}}$, and $V(\sigma, \mathsf{Sep}, \mathcal{R}) \subseteq V^{\mathcal{R}}$. Now, consider a vertex of $u' \in (V(\sigma, \mathsf{Sep}, \mathcal{L}) \cup V(\sigma, \mathsf{Sep}, \mathcal{R})) \setminus \mathsf{Origin}^{\mathcal{L}}$. However, such a vertex u' belongs to $O^{\mathcal{M}} \cup O^{\mathcal{R}}$, and hence is not part of the graph $G - (O^{\mathcal{M}} \cup O^{\mathcal{R}})$. Thus, it cannot receive label $\mathcal{L}$, which shows that $V^{\mathcal{L}} \subseteq V(\sigma, \mathsf{Sep}, \mathcal{L})$. By combining both containments, we obtain that $V^{\mathcal{L}} = V(\sigma, \mathsf{Sep}, \mathcal{L})$. Similar proofs also show that $V^{\mathcal{M}} = V(\sigma, \mathsf{Sep}, \mathcal{L})$, and $V^{\mathcal{R}} = V(\sigma, \mathsf{Sep}, \mathcal{R})$, and are therefore omitted. □

After accounting for the k_{current} crossings among the endpoints of $\mathsf{CrossingEdges} \cup \mathsf{MiddleEdges} \cup \mathsf{Sep}$ in σ, the rest of the k' crossings are either to the left of lsep or to the right of rsep, or within the edges of E_{23} whose endpoints are labeled $\mathcal{M}$. Let the number of these crossings be $k^{\mathcal{L}}, k^{\mathcal{R}}, k^{\mathcal{M}}$, respectively, and consider the guess where these numbers are correctly guessed, and for $k^{\mathcal{L}}, k^{\mathcal{R}}$, also suppose that their partition between $k^{\mathcal{L}}_{12}, k^{\mathcal{L}}_{23}, k^{\mathcal{R}}_{12}, k^{\mathcal{R}}_{23}$ is correctly guessed.

Further, since we label all the vertices correctly, it follows that all the vertices of $V^{\mathcal{L}}, V^{\mathcal{R}}$ are reachable from $\mathsf{Origin}^{\mathcal{L}}, \mathsf{Origin}^{\mathcal{R}}$ respectively. Then, we observe that vertices of $\mathsf{CrossLeft}$ (resp. $\mathsf{CrossRight}$) are directly connected to one of the vertices of $\mathsf{Endpoints}(\{\mathsf{lsep}_{12}, \mathsf{lsep}_{23}\})$ (resp. $\mathsf{Endpoints}(\{\mathsf{rsep}_1$ in the graph $G^{\mathcal{L}}$ (resp. $G^{\mathcal{R}}$), which shows that these guesses are not terminated by the connectivity check done in Step 3 after creating the instances $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}$, respectively. This shows that these guesses are not terminated by the connectivity check done in Step 3 after creating the instances $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}$, respectively. Then, Claim 5.7 imply that $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}, \mathcal{I}^{\mathcal{M}}$ are valid extended instances. Next, σ restricted to the respective vertex sets $V^{\mathcal{L}}, V^{\mathcal{R}}$ gives a valid drawing of the respective instances. This implies that $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}$ are YES-instances for the parameters $k^{\mathcal{L}}, k^{\mathcal{R}}$, respectively. Further, by

using an argument similar to the one from Lemma 8, we can also infer that $\mathcal{I}^{\mathcal{L}}, \mathcal{I}^{\mathcal{R}}$ are *minimal* YES-instances for the respective parameters. Then, by induction (since $k^{\mathcal{L}}, k^{\mathcal{R}} \leq 3k/4$) the respective recursive calls return valid drawings $\sigma^{\mathcal{L}}, \sigma^{\mathcal{R}}$ of the respective instances, implying that we proceed to Step 4 and create $\mathcal{I}^{\mathcal{M}}$.

Again, note that all the vertices of $V(G^{\mathcal{M}})$ are labeled correctly, which implies that they are reachable from $\mathsf{Origin}^{\mathcal{M}}$. Again, we observe that vertices of $\mathsf{CrossMid}$ are directly connected to one of the vertices of $\mathsf{Endpoints}(\{\mathsf{lsep}_{23}, \mathsf{rsep}_{23}\})$ in the graph $G^{\mathcal{M}}$. Recall that in the elaborate extended instance $\mathcal{I}^{\mathcal{M}}$ constructed, $\mathsf{lsep}_{23}, \mathsf{rsep}_{23}$ become the boundary edges $\mathsf{leftbound}^{\mathcal{M}}, \mathsf{rightbound}^{\mathcal{M}}$, and $\mathsf{GapLeft}, \mathsf{GapRight}$ become $\mathsf{LeftExtBnd}, \mathsf{RightExtBnd}$, respectively; and the preceding argument shows that each vertex in $V^{\mathcal{M}}$ is reachable from some vertex of $\mathsf{Endpoints}(\mathsf{BoundaryEdges}^{\mathcal{M}}) \cup \mathsf{ExtBnd}^{\mathcal{M}}$. This implies that the guess is not terminated, and then Claim 5.8 implies that $\mathcal{I}^{\mathcal{M}}$ is a valid elaborate extended instance of 2-LAYER CROSSING MINIMIZATION, and since the order on $\mathsf{Endpoints}(\mathsf{Sep}, 2) \cup \mathsf{GapLeft} \cup \mathsf{GapRight}$ is compatible with σ, it follows that σ restricted to the vertices of $V^{\mathcal{M}}$ gives a valid drawing of $\mathcal{I}^{\mathcal{M}}$. Therefore, $\mathcal{I}^{\mathcal{M}}$ is a YES-instance. Again, we can argue that $\mathcal{I}^{\mathcal{M}}$ is a *minimal* YES-instance, and via Lemma 8, we obtain a corresponding drawing $\sigma^{\mathcal{M}}$. Thus, in Step 6, we combine the three drawing and return a drawing σ'.

Reverse direction. The reverse direction is trivial, since the valid drawing σ' returned by the algorithm is a witness that $\mathcal{I}$ is a YES-instance. □

In the following lemma, we argue that the algorithm runs in subexponential time.

Lemma 21. *For any input extended instance $\mathcal{I}$ of* 3-LAYER CROSSING MINIMIZATION *with parameter k, the algorithm runs in time $T_3(k) \leq n^{\mathcal{O}(k^{2/3})}$.*

Proof. The base case corresponds to when $k \leq c\sqrt{k^\star}$, or when $\min\{|E_{12}|, |E_{23}|\} \leq c\sqrt{k^\star}$, for some constant c. Then, the running time of the algorithm is $n^{\mathcal{O}(\sqrt{k^\star})}$ via Lemma 10.

Otherwise, consider the recursive case, and suppose the claim is true for all $k' \leq k-1$, where $k > c\sqrt{k^\star}$ is the parameter of the given instance $\mathcal{I}$. Lemma 6 implies that the total number of guesses is bounded by $n^{\mathcal{O}(k^{2/3})}$, and corresponding to each guess that is not terminated, we make two recursive calls to the algorithm for 3-LAYER CROSSING MINIMIZATION with parameters at most $3k/4$ each. If for one such guess, both recursive calls return a drawing, then we also solve an instance of 2-LAYER CROSSING MINIMIZATION using Theorem 6 (without the kernelization step), where the parameter is at most k. Thus, it takes time $T_2(n,k) = n^{\mathcal{O}(\sqrt{k})}$ time. Thus, $T_3(n,k)$ satisfies the recurrence given in Equation (2), and then the claim follows by induction. □

By combining Observation 6, Lemma 20, Lemma 21, and the fact that due to linear-time kernelization algorithm of Theorem 4, we have $n = (k^\star)^{\mathcal{O}(1)}$, we obtain the following theorem (where we rename $k^\star$ as k).

Theorem 7. *On an n vertex graph,* 3-LAYER CROSSING MINIMIZATION *can be solved in time $2^{\mathcal{O}(k^{2/3}\log k)} + n \cdot k^{\mathcal{O}(1)}$, where k denotes the number of crossings.*

6 Kernelization for 3-Layer Crossing Minimization

The whole section is the proof of Theorem 4, which we restate here.

Theorem 4. 3-LAYER CROSSING MINIMIZATION *admits a kernel with $\mathcal{O}(k^8)$ vertices and edges. Furthermore, the kernelization algorithm can be implemented to run in $k^{\mathcal{O}(1)} \cdot n$ time.*

In Section 6.1, we give the kernelization algorithm and prove its correctness. In Section 6.2, we show that using some additional steps, we can implement the algorithm to run in time which is linear in n.

6.1 Construction of the kernel

Let (G, V_1, V_2, V_3, k) where (V_1, V_2, V_3) is a 3-partition of $V(G)$ be an instance of 3-Layer Crossing Minimization. We give a polynomial time algorithm that either correctly concludes that (G, V_1, V_2, V_3, k) is a no-instance, or constructs an equivalent instance $(G', V_1', V_2', V_3', k)$, with $\mathcal{O}(k^8)$ vertices and edges.

By Proposition 1, one can check in time $\mathcal{O}(n)$ whether G admits a 3-layer drawing respecting (V_1, V_2, V_3) without crossings. If this is the case, we immediately return a trivial yes-instance and stop. We also stop and output a trivial no-instance if $k = 0$ and G does not admit such a drawing. From now on, we assume that G has no drawing respecting (V_1, V_2, V_3) without crossings and that $k \geq 1$.

We apply a sequence of reduction rules. Almost each of the rules operates as follows: Suppose G contains a sufficiently large subgraph H that can be (a) separated from G by deleting a small (at most 6) number of vertices and (b) drawn on two or three layers without crossing. Then, we derive an equivalent but smaller instance of the problem by removing some "irrelevant" vertices of H from G. Occasionally, we may need to introduce new edges, but the number of vertices in the new equivalent instance always decreases. None of the rules changes the parameter k. The rules and the proof of their correctness (or, using the established expression in the area, their safety) depend on the number of vertices required to separate H from G. The rules are exhaustively applied whenever possible. If, by a rule, we delete a vertex or a set of vertices, we assume that the sets V_1, V_2, V_3 are modified accordingly by the deletion of vertices.

First, we address connected components that can be drawn without crossings. Trivially, such components are irrelevant because they can be drawn independently without creating any crossings.

Reduction Rule 6.1. If G has a connected component H that admits a 3-layer drawing respecting $(V_1 \cap V(H), V_2 \cap V(H), V_3 \cap V(H))$ without crossings, then set $G := G - V(H)$.

For a vertex $v \in V(G)$ and $i \in \{1, 2, 3\}$, let $P_i(v)$ be the set of pendent vertices in V_i adjacent to v. The following reduction rule reduces the number of pendent neighbors of the vertices from V_2.

Reduction Rule 6.2. If there is $u \in V_2$ such that $|P_i(u)| > k + 1$ for some $i \in \{1, 3\}$ then select (arbitrarily) vertex $v \in P_i(u)$ and set $G := G - v$.

Lemma 22. *Reduction Rule 6.2 is safe.*

Proof. Assume without loss of generality that the rule is applied for $u \in V_2$ and $v \in P_1(u)$, and let $G' = G - v$ and $V_1' = V_1 \setminus \{v\}$. Trivially, if G admits a 3-layer drawing respecting (V_1, V_2, V_3) with at most k crossing then G' has a 3-layer drawing respecting (V_1', V_2, V_3) with at most k crossing. Suppose that G' admits a 3-layer drawing $\sigma' = (\sigma_1', \sigma_2, \sigma_3)$ respecting (V_1', V_2, V_3) with at most k crossing. Because $|P_1(u)| > k + 1$, u is adjacent to at least $k + 1$ pendent vertices in V_1'. Since the number of crossings is upper bounded by k, there is $w \in P_1(u) \setminus \{v\}$ such that uw does not cross any other edge. We construct the order σ_1 of V_1 from σ_1' by inserting v immediately after w. Then uv does not cross any edge of G. Thus, $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ is 3-layer drawing respecting (V_1, V_2, V_3) with at most k crossing. This concludes the proof. □

We apply a similar rule to reduce the number of vertices in V_2. Let $u \in V_1$ and $v \in V_3$. We denote by $Q(u, v)$ the set of vertices of V_2 of degree two, that is, u and v are the only neighbors of any $w \in Q(u, v)$.

Reduction Rule 6.3. If there are $u \in V_1$ and $v \in V_3$ such that $|Q(u, v)| > k + 1$ then select (arbitrarily) vertex $w \in Q(u, v)$ and set $G := G - w$.

Lemma 23. *Reduction Rule 6.2 is safe.*

Proof. Suppose that the rule is applied for $u \in V_1$, $v \in V_3$, and $w \in Q(u,v)$, and let $G' = G - w$ and $V_2' = V_2 \setminus \{w\}$. As in the poof of Lemma 22, it is sufficient to show that if G' has a 3-layer drawing respecting (V_1, V_2', V_3) with at most k crossing then G has a 3-layer drawing respecting (V_1, V_2, V_3) with at most k crossing. Assume that $\sigma' = (\sigma_1, \sigma_2', \sigma_3)$ is such a drawing of G'. Because $|Q(u,v)| > k+1$, there is $x \in Q(u,v) \setminus \{w\}$ such that edges ux and vx do not cross any edge in the drawing. We construct the order σ_2 of V_2 from σ_2' by inserting w immediately after w. Then, neither uw nor vw cross an edge of G. We obtain that $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ is 3-layer drawing respecting (V_1, V_2, V_3) with at most k crossing. This concludes the proof of the claim. □

Let $u \in V_1 \cup V_3$ be a cut vertex of G. We say that a connected component H of $G - u$ is *nice* if (i) $V(H) \subseteq V_2 \cup V_3$ if $u \in V_1$ and $V(H) \subseteq V_1 \cup V_2$ if $u \in V_3$, and (ii) H admits a 2-layer drawing without crossings respecting either $(V_2 \cap V(H), V_3 \cap V(H))$ if $u \in V_1$ or $(V_1 \cap V(H), V_2 \cap V(H))$ if $u \in V_3$. The next rule is used to reduce the number of nice components.

Reduction Rule 6.4. If there is $u \in V_1 \cup V_3$ such that the number of nice connected components of $G - u$ is at least $2k+2$, then select a nice component H and set $G := G - V(H)$; we select a nice component of size one if such a component H exists and choose arbitrary H, otherwise.

Lemma 24. *Reduction Rule 6.4 is safe.*

Proof. By symmetry, let us assume that the rule is applied for $u \in V_1$ and $V(H) \subseteq V_2 \cup V_3$ for the deleted nice component H. Let $G' = G - V(H)$, $V_2' = V_2 \setminus V(H)$, and $V_3' = V_3 \setminus V(H)$. Again, it is sufficient to show that if G' has a 3-layer drawing respecting (V_1, V_2', V_3') with at most k crossing then G has a 3-layer drawing respecting (V_1, V_2, V_3) with at most k crossing. Assume that $\sigma' = (\sigma_1, \sigma_2', \sigma_3')$ is a 3-layer drawing of G' with at most k crossings. Because the number of nice connected components of $G - u$ is at least $2k+2$, there is a nice component C distinct from H such that neither edges of C nor edges uv for $v \in V(C)$ cross other edges. Since we cannot apply Reduction Rule 6.1, C cannot be a connected component of G. Thus, there is a vertex of C adjacent to u. Denote by x the last, with respect to σ_2', vertex of $V_2 \cap V(C)$ adjacent to u.

Suppose first that H is trivial, that is, it has a single vertex. Then we construct a 3-layer drawing $\sigma = (\sigma_1, \sigma_2, \sigma_3')$ of G respecting (V_1, V_2, V_3) by modifying σ_2' the same way as in the proof of Lemma 22. Namely, we insert the unique vertex of H immediately after x in σ_2'. It does not create a new crossing because ux does not cross any edge in σ'.

From now on, we assume that H has at least two vertices. By the choice of H, there is no nice component of size one and, therefore, C also has at least two vertices and has vertices from both V_2 and V_3. Let y be the last vertex of $V_2 \cap V(C)$ and z be the last vertex of $V_3 \cap V(C)$ with respect to σ_2' and σ_3', respectively. It may happen that $x = y$. Suppose that there is $v \in V_2' \setminus V(C)$ such that $\sigma_2'(x) < \sigma_2'(v) < \sigma_2'(y)$. Because C is connected, we have that v has no neighbors in V_3 and is adjacent only to some vertices $w \in V_1$ with $\sigma_1(u) \leq \sigma_2(w)$. This observation allows us to modify σ_2' as follows. Let $v_1, \ldots, v_\ell$ be all the vertices of $V_2 \setminus V(C)$ such that $\sigma_2'(x) < \sigma_2'(v_i) < \sigma_2'(y)$ for $i \in \{1, \ldots, \ell\}$ (the set of these vertices may be empty) and assume that $\sigma_2'(v_1) < \cdots < \sigma_2'(v_\ell)$. We construct σ_2'' from σ_2' by placing $v_1, \ldots, v_\ell$ following their order in σ_2' after y in the order of V_2'. Notice that this modification does not create any new crossing. Furthermore, for $\sigma'' = (\sigma_1, \sigma_2'', \sigma_3')$, we have that either u, y, and z are the last vertices in σ_1, σ_2'', and σ_3', respectively, or u separates the vertices that are after u, y, and z with respect to σ'' from the vertices that are before them. More formally, it holds that (a) there is no edge ab with $a \in V_2'$ and $b \in V_3'$ such that either $\sigma_2''(a) \leq \sigma_2''(y)$ and $\sigma_3'(b) > \sigma_3'(z)$ or $\sigma_2''(a) > \sigma_2''(y)$ and $\sigma_3'(b) \leq \sigma_3'(z)$, and (b) there is no edge ab with $a \in V_1$ and $b \in V_2'$ such that either $\sigma_1(a) < \sigma_a(u)$ and $\sigma_2''(b) > \sigma_2''(y)$ or $\sigma_1(a) > \sigma_1(u)$ and $\sigma_2''(b) \leq \sigma_2''(y)$. This allows us to construct a 3-layer drawing $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ of G respecting (V_1, V_2, V_3) without increasing the number of crossings.

Because H admits a 2-layer drawing without crossings respecting $(V_2 \cap V(H), V_3 \cap V(H))$, there are the corresponding orders $\hat{\sigma}_2$ and $\hat{\sigma}_3$ of $V_2 \cap V(H)$ and $V_3 \cap V(H)$, respectively. We construct $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ by modifying σ_2'' and σ_3'. To construct σ_2, we insert the vertices of $V_2 \cap V(H)$ in

σ_2'' immediately after y following $\hat{\sigma}_2$. Similarly, σ_3 is constructed from σ_3' by inserting the vertices of $V_3 \cap V(H)$ immediately after z following $\hat{\sigma}_3$. This concludes the proof. □

By the following five rules, we detect pieces of the graph with only particular drawings and reduce their size. We start with pieces that can be separated by four vertices from the remaining graph.

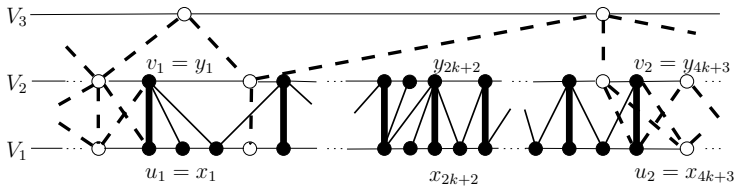

Figure 25: Applying Reduction Rule 6.5; the vertices of A are black bullets and the vertices of $B \setminus A$ are white; the edges of H are solid lines and the edges outside H are drawn by dashed lines; the edges of M are highlighted by thick lines.

Informally, Reduction Rule 6.5 does the following. Assume that H is a subgraph of G that has a 2-layer drawing in layers V_1 and V_2 without crossings. Moreover, assume that in such a drawing, the endpoints of the leftmost edge u_1v_1 and the rightmost edge u_2v_2 of H separate it from the remaining part of G. Then, if H contains a matching M of size $4k+3$, at least one of the endpoints of the central edge in M could be safely removed. (We also must add one or two edges to maintain connectivity.)

Reduction Rule 6.5 (4-Separation A)**.** Suppose that there is a separation (A, B) of G with $A \cap B = \{u_1, u_2, v_1, v_2\}$ such that (see Figure 26)

(i) $H = G[A]$ is a connected graph with $A \subseteq V_1 \cup V_2$, $u_1, u_2 \in V_1$, and $v_1, v_2 \in V_2$,

(ii) $u_1v_1, u_2v_2 \in E(G)$,

(iii) H has a 2-layer drawing $\sigma_H = (\sigma_1^H, \sigma_2^H)$ without crossings respecting $(V_1 \cap V(H), V_2 \cap V(H))$, where σ_i^H is an order of $V_i \cap V(H)$ for $i \in \{1, 2\}$, such that (a) for every $w \in V_1$, $\sigma_1^H(u_1) \leq \sigma_1^H(w) \leq \sigma_1^H(u_2)$ and (b) for every $w \in V_2$, $\sigma_2^H(v_1) \leq \sigma_2^H(w) \leq \sigma_2^H(v_2)$,

(iv) H has a matching M of size $4k + 3$ with $u_1v_1, u_2v_2 \in M$.

Then let $M = \{x_1y_1, \ldots, x_{4k+3}y_{4k+3}\}$, where $x_1 = u_1$, $y_1 = v_1$, $x_{4k+3} = u_2$, $y_{4k+3} = v_2$, $\sigma_1^H(x_1) < \cdots < \sigma_1^H(x_{4k+3})$, and $\sigma_2^H(y_1) < \cdots < \sigma_2^H(y_{4k+3})$, and do the following:

(a) find the unique $z_1 \in \{x_{2k+1}, y_{2k+1}\}$ that has a neighbor either after x_{2k+1} in σ_1^H or after y_{2k+1} in σ_2^H, respectively, and find the unique $z_2 \in \{x_{2k+3}, y_{2k+3}\}$ that has a neighbor either before x_{2k+1} in σ_1^H or before y_{2k+1} in σ_2^H, respectively,

(b) delete every vertex $w \in V_1 \cap V(H)$ with $\sigma_1^H(x_{2k+1}) < \sigma_1^H(w) < \sigma_1^H(x_{2k+3})$,

(c) delete every vertex $w \in V_2 \cap V(H)$ with $\sigma_2^H(y_{2k+1}) < \sigma_2^H(w) < \sigma_2^H(y_{2k+3})$,

(d) if either $z_1 \in V_1$ and $z_2 \in V_2$ or $z_1 \in V_2$ and $z_2 \in V_1$ then make z_1 and z_2 adjacent to maintain connectivity,

(e) if $z_1 \in V_1$ and $z_2 \in V_1$ then restore y_{2i+2} and make it adjacent to z_1 and z_2,

(f) if $z_1 \in V_2$ and $z_2 \in V_2$ then restore x_{2i+2} and make it adjacent to z_1 and z_2.

We note that we use a complicated procedure of adding edges to ensure connectivity for simplifying the running time analysis in the next subsection. In particular, we guarantee that the rule does not create new pendent vertices and ensure that the graph H' obtained from H does not contain a matching of the same size as M if M is a maximum size matching satisfying the conditions of the rule.

Lemma 25. *Reduction Rule 6.5 is safe.*

Proof. First, we observe that because H has a 2-layer drawing $\sigma_H = (\sigma_1^H, \sigma_2^H)$ without crossings respecting $(V_1 \cap V(H), V_2 \cap V(H))$ such that (iii)(a) and (iii)(b) are fulfilled, we can order the endpoints of the edges of the matching M to satisfy the conditions that $\sigma_1^H(x_1) < \cdots < \sigma_1^H(x_{4k+3})$, and $\sigma_2^H(y_1) < \cdots < \sigma_2^H(y_{4k+3})$. Thus, the rule can be executed if there is a separation (A, B) satisfying (i)–(iv). Note also that because at least one of the vertices x_{2k+2} and y_{2k+2} is deleted by the rule, it reduces the size of the graph.

Suppose that the rule is applied for a separation (A, B) and denote by G' the graph obtained by applying the rule. Let also $D \subseteq V_1 \cup V_2$ be the set of deleted vertices, $V_1' = V_1 \setminus D$, $V_2' = V_2 \setminus D$.

Assume that $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ is a 3-layer drawing of G respecting (V_1, V_2, V_3) with at most k crossings. Notice that because the number of crossings is at most k, there are $i \in \{1, \ldots, 2k+1\}$ and $j \in \{2k+3, \ldots, 4k+3\}$ such that the edges x_iy_i and x_jy_j do not cross any edge. Without loss of generality, we assume that $\sigma_1(x_i) < \sigma_1(x_j)$ and $\sigma_2(y_i) < \sigma_2(y_j)$. Otherwise, we just reverse the orders in σ. We set

$$U_1 = \{w \in V_1 \cap V(H) \colon \sigma_1(x_i) < \sigma_1(w) < \sigma_1(x_j)\}$$

and

$$U_2 = \{w \in V_2 \cap V(H) \colon \sigma_2(y_i) < \sigma_2(w) < \sigma_2(y_j)\},$$

and define

$$W_1 = \{w \in V_1 \setminus V(H) \colon \sigma_1(x_i) < \sigma_1(w) < \sigma_1(x_j)\}$$

and

$$W_2 = \{w \in V_2 \setminus V(H) \colon \sigma_2(y_i) < \sigma_2(w) < \sigma_2(y_j)\}.$$

Observe that because H is connected and x_iy_i and x_jy_j do not cross any edge, we have that

$$U_1 = \{w \in V_1 \cap V(H) \colon \sigma_1^H(x_i) < \sigma_1^H(w) < \sigma_1^H(x_j)\}$$

and

$$U_2 = \{w \in V_2 \cap V(H) \colon \sigma_2^H(y_i) < \sigma_2^H(w) < \sigma_2^H(y_j)\}.$$

Furthermore, if $xy \in E(G) \setminus E(H)$ with $x \in V_1$ and $y \in V_2$ such that $x \in W_1$ or $y \in W_2$, then $x \in W_1$, $y \in W_2$, and xy crosses at least one edge of H in the drawing of G.

We construct a 3-layer drawing $\sigma' = (\sigma_1', \sigma_2', \sigma_3)$ of G' respecting (V_1', V_2', V_3) with no more crossing than σ as follows. We obtain σ_1' from σ_1 by reordering the vertices of $U_1 \cap V(G')$ and W_1. First, we place the vertices of $U_1 \cap V(G')$ between x_i and x_j following their order in σ_1^H and then place the vertices of W_1 between x_{2k+1} and x_{2k+3} following their order in σ_1. Similarly, σ_2' is obtained from σ_2 by reordering the vertices of $U_2 \cap V(G')$ and W_2: we place the vertices of $U_2 \cap V(G')$ between y_i and y_j following their order in σ_2^H and then place the vertices of W_2 between y_{2k+1} and y_{2k+3} following their order in σ_2. Observe that an edge $xy \in E(H) \cap E(G')$ with $\sigma_1'(x_i) \leq \sigma_1'(x) \leq \sigma_1'(x_j)$ and $\sigma_2'(y_i) \leq \sigma_2'(y) \leq \sigma_1'(y_j)$ does not cross any edge of G'. Further, the edges constructed by the rules, that is, either $x_{2k+1}y_{2k+3}$, or $y_{2k+1}x_{2k+3}$, or $x_{2k+1}y_{2k+2}$ with $y_{2k+2}x_{2k+3}$, or $y_{2k+1}x_{2k+2}$ with $x_{2k+2}y_{2k+3}$, cross only the edges $xy \in E(G')$ with $x \in W_1$ and $y \in W_2$ and xy crosses exactly one edge constructed by the rule. This implies that in σ' the number of crossings does not exceed the number of crossings in σ.

For the opposite direction, assume that there is a 3-layer drawing $\sigma' = (\sigma_1', \sigma_2', \sigma_3)$ of G' respecting (V_1', V_2', V_3) with at most k crossings. We are using almost the same arguments as for

the forward implication to show that there is a 3-layer drawing $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ of G respecting (V_1, V_2, V_3) such that the number of crossings in σ is at most the number of crossings in σ'.

Because the number of crossings is at most k, there are $i \in \{1, \ldots, 2k+1\}$ and $j \in \{2k+3, \ldots, 4k+3\}$ such that the edges x_iy_i and x_jy_j do not cross any edge with respect to σ'. We can assume that $\sigma'_1(x_i) < \sigma'_1(x_j)$ and $\sigma'_2(y_i) < \sigma'_2(y_j)$. We define vertex sets

$$U_1 = \{w \in V_1 \cap V(H) \colon \sigma_1^H(x_i) < \sigma_1^H(w) < \sigma_1^H(x_j)\},$$

$$U_2 = \{w \in V_2 \cap V(H) \colon \sigma_2^H(y_i) < \sigma_2^H(w) < \sigma_2^H(y_j)\}.$$

We also introduce

$$U'_1 = \{w \in V'_1 \cap V(H) \colon \sigma'_1(x_i) < \sigma_1(w) < \sigma'_1(x_j)\}$$

and

$$U'_2 = \{w \in V'_2 \cap V(H) \colon \sigma'_2(y_i) < \sigma'_2(w) < \sigma'_2(y_j)\}.$$

We further define

$$W'_1 = \{w \in V_1 \setminus V(H) \colon \sigma'_1(x_i) < \sigma'_1(w) < \sigma'_1(x_j)\},$$

and

$$W'_2 = \{w \in V_2 \setminus V(H) \colon \sigma'_2(y_i) < \sigma'_2(w) < \sigma'_2(y_j)\}.$$

We remark that graph H' obtained from H by deleting the vertices of D and adding one or two edges is connected. Then, since x_iy_i and x_jy_j do not cross any edge, we have that $U'_1 \subseteq U_1$ and $U'_2 \subseteq U_2$. Moreover, if $xy \in E(G') \setminus E(H')$ with $x \in V_1$ and $y \in V_2$ such that $x \in W_1$ or $y \in W_2$, then $x \in W_1$ and $y \in W_2$ and xy crosses at least one edge of H' in the drawing of G'.

We construct σ_1 from σ'_1. First, we replace the vertices of U'_1 with the vertices of U_1 and put them between x_i and x_j following their order in σ_1^H. Then we place the vertices of W_1 immediately after x_{2k+2} following their order in σ'_1. The order σ_2 is constructed from σ'_2 in a similar way. We replace the vertices of U'_2 with the vertices of U_2 and put them between y_i and y_j following their order in σ_2^H. Then we place the vertices of W_2 immediately before y_{2k+2} following their order in σ'_2. We have that any edge $xy \in E(H)$ with $\sigma_1(x_i) \leq \sigma_1(x) \leq \sigma_1(x_j)$ and $\sigma_2(y_i) \leq \sigma_2(y) \leq \sigma_1(y_j)$ that is distinct from $x_{2k+2}y_{2k+2}$ does not cross any edge of G and $x_{2k+1}y_{2k+3}$ crosses only the edges $xy \in E(G)$ with $x \in W_1$ and $y \in W_2$. Therefore, the number of crossings in the constructed 3-layer drawing σ does not exceed the number of crossings in σ'. This concludes the proof of the claim. □

Reduction Rule 6.5 is applied for the drawing of H in the layers V_1 and V_2. Of course, by symmetry, the same arguments are applicable when the vertices of H are in layers V_2 and V_3. Since the statement of Reduction Rule 6.5 is already quite long and challenging to read. Thus, instead of putting it in the general form, capturing two symmetric cases (like we did in Reduction Rule 6.2), it is slightly preferable to have it as two reduction rules. The next Reduction Rule 6.6, up to the symmetry, is exactly the same as Reduction Rule 6.5. We state the rule (without proof of its safety) for completeness.

Reduction Rule 6.6 (4-Separation B)**.** Suppose that there is a separation (A, B) of G with a separator $A \cap B = \{u_1, u_2, v_1, v_2\}$ such that

(i) $H = G[A]$ is a connected graph with $A \subseteq V_2 \cup V_3$, $u_1, u_2 \in V_2$, and $v_1, v_2 \in V_3$,

(ii) $u_1v_1, u_2v_2 \in E(G)$,

(iii) H has a 2-layer drawing $\sigma_H = (\sigma_2^H, \sigma_3^H)$ without crossings respecting $(V_2 \cap V(H), V_3 \cap V(H))$, where σ_i^H is an order of $V_i \cap V(H)$ for $i \in \{2, 3\}$, such that (a) for every $w \in V_2$, $\sigma_2^H(u_1) \leq \sigma_2^H(w) \leq \sigma_2^H(u_2)$ and (b) for every $w \in V_3$, $\sigma_3^H(v_1) \leq \sigma_3^H(w) \leq \sigma_3^H(v_2)$,

(iv) H has a matching M of size $4k+3$ with $u_1v_1, u_2v_2 \in M$.

Then let $M = \{x_1y_1, \ldots, x_{4k+3}y_{4k+3}\}$, where $x_1 = u_1$, $y_1 = v_1$, $x_{4k+3} = u_2$, $y_{4k+3} = v_2$, $\sigma_2^H(x_1) < \cdots < \sigma_2^H(x_{4k+3})$, and $\sigma_3^H(y_1) < \cdots < \sigma_3^H(y_{4k+3})$, and do the following:

(a) find the unique $z_1 \in \{x_{2k+1}, y_{2k+1}\}$ that has a neighbor either after x_{2k+1} in σ_2^H or after y_{2k+1} in σ_3^H, respectively, and find the unique $z_2 \in \{x_{2k+3}, y_{2k+3}\}$ that has a neighbor either before x_{2k+1} in σ_2^H or before y_{2k+1} in σ_3^H, respectively,

(b) delete every vertex $w \in V_1 \cap V(H)$ with $\sigma_2^H(x_{2k+1}) < \sigma_2^H(w) < \sigma_2^H(x_{2k+3})$,

(c) delete every vertex $w \in V_2 \cap V(H)$ with $\sigma_3^H(y_{2k+1}) < \sigma_3^H(w) < \sigma_3^H(y_{2k+3})$,

(d) if either $z_1 \in V_2$ and $z_2 \in V_3$ or $z_1 \in V_3$ and $z_2 \in V_2$ then make z_1 and z_2 adjacent to maintain connectivity,

(e) if $z_1 \in V_2$ and $z_2 \in V_2$ then restore y_{2i+2} and make it adjacent to z_1 and z_2,

(f) if $z_1 \in V_3$ and $z_2 \in V_3$ then restore x_{2i+2} and make it adjacent to z_1 and z_2.

In 4-Separation Reduction Rules, we considered subgraphs with the vertices in two sets of the 3-partition. Now, we turn to the subgraphs with vertices in three sets. First is the case of a subgraph with one vertex in V_1 or V_3.

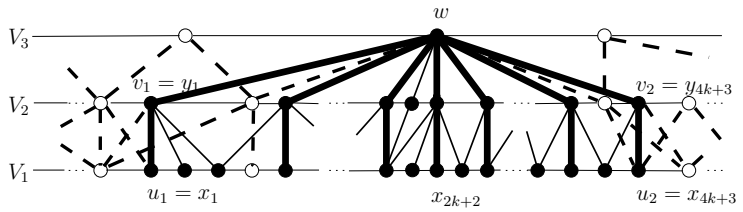

Figure 26: Applying Reduction Rule 6.7; the vertices of A are shown by black bullets and the vertices of $B \setminus A$ are white; the edges of H are shown by solid lines and the edges outside H are drawn by dashed lines; the edges of M and the edges wy_i for $i \in \{1, \ldots, 4k+3\}$ are highlighted by thick lines.

Reduction Rule 6.7 (5-Separation A)**.** Suppose that there is a separation (A, B) of G with a separator $A \cap B = \{u_1, u_2, v_1, v_2, w\}$ such that (see Figure 26)

(i) $G[A]$ is a graph with $A \cap V_3 = \{w\}$, $u_1, u_2 \in V_1$, and $v_1, v_2 \in V_2$,

(ii) $H = G[A] - w$ is connected,

(iii) $u_1v_1, u_2v_2, v_1w, v_2w \in E(G)$,

(iv) H has a 2-layer drawing $\sigma_H = (\sigma_1^H, \sigma_2^H)$ without crossings respecting $(V_1 \cap V(H), V_2 \cap V(H))$, where σ_i^H is an order of $V_i \cap V(H)$ for $i \in \{1, 2\}$, such that (a) for every $x \in V_1 \cap V(H)$, $\sigma_1^H(u_1) \le \sigma_1^H(x) \le \sigma_1^H(u_2)$ and (b) for every $x \in V_2 \cap V(H)$, $\sigma_2^H(v_1) \le \sigma_2^H(x) \le \sigma_2^H(v_2)$,

(v) H has a matching $M = \{x_1y_1, \ldots, x_{4k+3}y_{4k+3}\}$ of size $4k+3$ with $u_1v_1, u_2v_2 \in M$ and $y_1, \ldots, y_{4k+3} \in N_G(w)$.

Then assume that $x_1 = u_1$, $y_1 = v_1$, $x_{4k+3} = u_2$, $y_{4k+3} = v_2$, $\sigma_1^H(x_1) < \cdots < \sigma_1^H(x_{4k+3})$, and $\sigma_2^H(y_1) < \cdots < \sigma_2^H(y_{4k+3})$, and do the following:

(a) delete every vertex $z \in V_1 \cap V(H)$ with $\sigma_1^H(x_{2k+1}) < \sigma_1^H(z) < \sigma_1^H(x_{2k+3})$,

(b) delete every vertex $z \in V_2 \cap V(H)$ with $\sigma_2^H(y_{2k+1}) < \sigma_2^H(z) < \sigma_2^H(y_{2k+3})$ except y_{2k+2} and delete the edges incident to y_{2k+2},

(c) make x_{2k+1} and x_{2k+3} adjacent to y_{2k+2} to maintain connectivity.

We note that the deleted vertices and edges ensure that we do not create new paths of length two with their end-vertices in V_1 and V_3.

Lemma 26. *Reduction Rule 6.7 is safe.*

Proof. The proof follows the same lines as the proof of Lemma 25 and is, in fact, simpler. Because H has a 2-layer drawing $\sigma_H = (\sigma_1^H, \sigma_2^H)$ without crossings respecting $(V_1 \cap V(H), V_2 \cap V(H))$ such that (iv)(a) and (iv)(b) are fulfilled, the endpoints of the edges of the matching M can be ordered to satisfy the conditions that $\sigma_1^H(x_1) < \cdots < \sigma_1^H(x_{4k+3})$, and $\sigma_2^H(y_1) < \cdots < \sigma_2^H(y_{4k+3})$. Therefore, the rule is feasible if there is a separation (A, B) satisfying (i)–(v). Also, because x_{2k+2} is deleted by the rule, it reduces the size of the graph.

Suppose that the rule is applied for a separation (A, B) and denote by G' the graph obtained by the application of the rule. Let also $D \subseteq V_1 \cup V_2$ be the set of deleted vertices, $V_1' = V_1 \setminus D$, $V_2' = V_2 \setminus D$.

Let $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ be a 3-layer drawing of G respecting (V_1, V_2, V_3) with at most k crossings. Because the number of crossings is at most k, there are $i \in \{1, \ldots, 2k+1\}$ and $j \in \{2k+3, \ldots, 4k+3\}$ such that the edges x_iy_i, y_iw, x_jy_j, and y_jw do not cross any edge. Without loss of generality, we assume that $\sigma_1(x_i) < \sigma_1(x_j)$ and $\sigma_2(y_i) < \sigma_2(y_j)$. We define

$$U_1 = \{z \in V_1 \cap V(H) \colon \sigma_1(x_i) < \sigma_1(z) < \sigma_1(x_j)\}$$

and

$$U_2 = \{z \in V_2 \cap V(H) \colon \sigma_2(y_i) < \sigma_2(z) < \sigma_2(y_j)\}.$$

Because H is connected and x_iy_i, y_iw, x_jy_j, and y_jw do not cross any edge, we have that

$$U_1 = \{z \in V_1 \cap V(H) \colon \sigma_1^H(x_i) < \sigma_1^H(z) < \sigma_1^H(x_j)\}$$

and

$$U_2 = \{z \in V_2 \cap V(H) \colon \sigma_2^H(y_i) < \sigma_2^H(z) < \sigma_2^H(y_j)\}.$$

Moreover, there is no vertex $z \in (V_1 \cup V_2) \setminus (U_1 \cup U_2)$ such that $\sigma_1(x_i) < \sigma_1(z) < \sigma_1(x_j)$ if $z \in V_1$ and $\sigma_2(y_i) < \sigma_2(z) < \sigma_2(y_j)$ if $z \in V_2$.

We construct the 3-layer drawing $\sigma' = (\sigma_1', \sigma_2', \sigma_3)$ of G' respecting (V_1', V_2', V_3) that has no more crossing than σ by modifying σ_1 and σ_2. We place the vertices of $U_1 \cap V(G')$ between x_i and x_j following their order in σ_1^H and we place the vertices of $U_2 \cap V(G')$ between y_i and y_j following their order in σ_2^H. Since there are no crossing edges with their endpoints in U_1 or U_2, the number of crossings in σ' does not exceed the number of crossings in σ.

For the opposite direction, the proof is almost the same. Assume that there is a 3-layer drawing $\sigma' = (\sigma_1', \sigma_2', \sigma_3)$ of G' respecting (V_1', V_2', V_3) with at most k crossings. We construct a 3-layer drawing $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ of G respecting (V_1, V_2, V_3) such that the number of crossings in σ is at most the number of crossings in σ'. For this, we note that because the number of crossings is at most k, there are $i \in \{1, \ldots, 2k+1\}$ and $j \in \{2k+3, \ldots, 4k+3\}$ such that the edges x_iy_i, y_iw, x_jy_j, and y_jw do not cross any edge with respect to σ'. We can assume that $\sigma_1'(x_i) < \sigma_1'(x_j)$ and $\sigma_2'(y_i) < \sigma_2'(y_j)$. We set

$$U_1 = \{z \in V_1 \cap V(H) \colon \sigma_1^H(x_i) < \sigma_1^H(z) < \sigma_1^H(x_j)\}$$

and

$$U_2 = \{z \in V_2 \cap V(H) \colon \sigma_2^H(y_i) < \sigma_2^H(z) < \sigma_2^H(y_j)\},$$

and set

$$U_1' = \{z \in V_1' \cap V(H) \colon \sigma_1'(x_i) < \sigma_1(z) < \sigma_1'(x_j)\}$$

and

$$U_2' = \{z \in V_2' \cap V(H) \colon \sigma_2'(y_i) < \sigma_2'(z) < \sigma_2'(y_j)\}.$$

Because the graph H' obtained from H by deleting the vertices of D and making x_{2k+1} and x_{2k+3} adjacent to y_{2k+2} is connected and x_iy_i, y_iw, x_jy_j, and y_jw do not cross any edge, $U_1' \subseteq U_1$ and $U_2' \subseteq U_2$. Also, there is no vertex $z \in (V_1 \cup V_2) \setminus (U_1' \cup U_2')$ such that $\sigma_1'(x_i) \leq \sigma_1'(z) \leq \sigma_1'(x_j)$ if $z \in V_1$ and $\sigma_2'(y_i) \leq \sigma_2'(z) \leq \sigma_2'(y_j)$ if $z \in V_2$.

We construct σ_1 from σ_1' by replacing the vertices of U_1' with the vertices of U_1 and putting them between x_i and x_j following their order in σ_1^H. Similarly, σ_2 is constructed from σ_2' by replacing the vertices of U_2' with the vertices of U_2 and putting them between y_i and y_j following their order in σ_2^H. This concludes the proof. □

Similarly to 4-Separation Rules, there is a symmetric version of Reduction Rule 6.7, which we state for completeness.

Reduction Rule 6.8 (5-Separation B)**.** Suppose that there is a separation (A, B) of G with a separator $A \cap B = \{u_1, u_2, v_1, v_2, w\}$ such that

(i) $G[A]$ is a graph with $A \cap V_1 = \{w\}$, $u_1, u_2 \in V_2$, and $v_1, v_2 \in V_3$,

(ii) $H = G[A] - w$ is connected,

(iii) $u_1v_1, u_2v_2, v_1w, v_2w \in E(G)$,

(iv) H has a 2-layer drawing $\sigma_H = (\sigma_2^H, \sigma_3^H)$ without crossings respecting $(V_2 \cap V(H), V_3 \cap V(H))$, where σ_i^H is an order of $V_i \cap V(H)$ for $i \in \{2, 3\}$, such that (a) for every $x \in V_2 \cap V(H)$, $\sigma_2^H(u_1) \leq \sigma_2^H(x) \leq \sigma_2^H(u_2)$ and (b) for every $x \in V_3 \cap V(H)$, $\sigma_3^H(v_1) \leq \sigma_3^H(x) \leq \sigma_3^H(v_2)$,

(v) H has a matching $M = \{x_1y_1, \ldots, x_{4k+3}y_{4k+3}\}$ of size $4k + 3$ with $u_1v_1, u_2v_2 \in M$ and $x_1, \ldots, x_{4k+3} \in N_G(w)$.

Then assume that $x_1 = u_1$, $y_1 = v_1$, $x_{4k+1} = u_2$, $y_{4k+1} = v_2$, $\sigma_2^H(x_1) < \cdots < \sigma_2^H(x_{4k+3})$, and $\sigma_3^H(y_1) < \cdots < \sigma_3^H(y_{4k+3})$, and do the following:

(a) delete every vertex $z \in V_2 \cap V(H)$ with $\sigma_2^H(x_{2k+1}) < \sigma_2^H(z) < \sigma_2^H(x_{2k+3})$ except x_{2k+2} and delete the edges incident to x_{2k+2},

(b) delete every vertex $z \in V_3 \cap V(H)$ with $\sigma_3^H(y_{2k+1}) < \sigma_3^H(z) < \sigma_3^H(y_{2k+3})$,

(c) make x_{2k+2} adjacent to y_{2k+1} and y_{2k+3} to maintain connectivity.

In the following rule, we consider subgraphs separated by six vertices from two paths of length two. The general idea for this rule is very similar to the 4-Separation Rule. The main difference is that instead of a large matching in graph H, we have many disjoint paths passing from level V_1 to V_3.

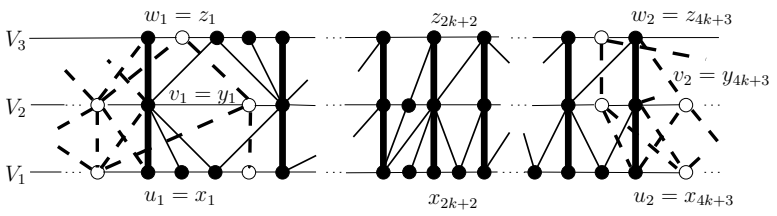

Figure 27: Applying Reduction Rule 6.9; the vertices of A are black bullets and the vertices of $B \setminus A$ are white; the edges of H are solid lines and the edges outside H are drawn by dashed lines; the edges of the paths in P are highlighted by thick lines.

Reduction Rule 6.9 (6-Separation)**.** Suppose that there is a separation (A, B) of G with a separator $A \cap B = \{u_1, v_1, w_1, u_2, v_2, w_2\}$ such that (see Figure 27)

(i) $H = G[A]$ is a connected graph, $u_1, u_2 \in V_1$, $v_1, v_2 \in V_2$, and $w_1, w_3 \in V_3$

(ii) $u_1v_1, v_1w_1, u_2v_2, v_2w_2 \in E(G)$, that is, $u_1v_1w_1$ and $u_2v_2w_2$ are paths of length two,

(iii) H has a 3-layer drawing $\sigma_H = (\sigma_1^H, \sigma_2^H, \sigma_2^H)$ without crossings respecting $(V_1 \cap V(H), V_2 \cap V(H), V_3 \cap V(H))$, where σ_i^H is an order of $V_i \cap V(H)$ for $i \in \{1, 2, 3\}$, such that (a) for every $x \in V_1 \cap V(H)$, $\sigma_1^H(u_1) \leq \sigma_1^H(x) \leq \sigma_1^H(u_2)$, (b) for every $x \in V_2 \cap V(H)$, $\sigma_2^H(v_1) \leq \sigma_2^H(x) \leq \sigma_2^H(v_2)$, and (c) for every $x \in V_3 \cap V(H)$, $\sigma_3^H(w_1) \leq \sigma_3^H(x) \leq \sigma_3^H(w_2)$,

(iv) H has a family of vertex disjoint path $P = \{x_1y_1z_1, \ldots, x_{4k+3}y_{4k+3}z_{4k+3}\}$ of size $4k+3$ with $u_1v_1w_1, u_2v_2w_2 \in P$.

Then assume that $x_1 = u_1$, $y_1 = v_1$, $z_1 = w_1$, $x_{4k+3} = u_2$, $y_{4k+3} = v_2$, $z_{4k+3} = w_2$, $\sigma_1^H(x_1) < \cdots < \sigma_1^H(x_{4k+3})$, $\sigma_2^H(y_1) < \cdots < \sigma_2^H(y_{4k+3})$, and $\sigma_3^H(z_1) < \cdots < \sigma_3^H(z_{4k+3})$ and do the following:

(a) delete every vertex $s \in V_1 \cap V(H)$ with $\sigma_1^H(x_{2k+1}) < \sigma_1^H(s) < \sigma_1^H(x_{2k+3})$,

(b) delete every vertex $s \in V_2 \cap V(H)$ with $\sigma_2^H(y_{2k+1}) < \sigma_2^H(s) < \sigma_2^H(y_{2k+3})$ except y_{2k+2} and delete the edges incident to y_{2k+2},

(c) delete every vertex $s \in V_3 \cap V(H)$ with $\sigma_3^H(z_{2k+1}) < \sigma_3^H(s) < \sigma_3^H(z_{2k+3})$,

(d) make x_{2k+1} and x_{2k+3} adjacent to y_{2k+2} to maintain connectivity.

We again note that deleting of vertices and edges is done in such a way that we cannot create new path of length two with the end-vertices in V_1 and V_3.

Lemma 27. *Reduction Rule 6.9 is safe.*

Proof. Again, we exploit the same ideas as in the proof of Lemma 25; now, we use the family of paths P instead of the matching M. The proof is very similar to the proof of Lemma 26. Because H has a 3-layer drawing $\sigma_H = (\sigma_1^H, \sigma_2^H, \sigma_3^H)$ without crossings respecting $(V_1 \cap V(H), V_2 \cap V(H), V_3 \cap V(H))$ such that (iii)(a)–(c) are fulfilled, the vertices of the paths of P can be ordered to satisfy the conditions that $\sigma_1^H(x_1) < \cdots < \sigma_1^H(x_{4k+3})$, $\sigma_2^H(y_1) < \cdots < \sigma_2^H(y_{4k+3})$, and $\sigma_3^H(y_1) < \cdots < \sigma_3^H(y_{4k+3})$. Thus, the rule is feasible. Also, the rule deletes at least two vertices. Suppose that the rule is applied for a separation (A, B) and denote by G' the graph obtained by applying the rule. Let also $D \subseteq V(H)$ be the set of deleted vertices, $V_1' = V_1 \setminus D$, $V_2' = V_2 \setminus D$, and $V_3' = V_3 \setminus D$.

Assume that $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ is a 3-layer drawing of G respecting (V_1, V_2, V_3) with at most k crossings. Because the number of crossings is at most k, there are $i \in \{1, \ldots, 2k+1\}$ and $j \in \{2k+3, \ldots, 4k+3\}$ such that the edges x_iy_i, y_iz_i, x_jy_j, and y_jz_j do not cross any edge. Without loss of generality, we assume that $\sigma_1(x_i) < \sigma_1(x_j)$, $\sigma_2(y_i) < \sigma_2(y_j)$, and $\sigma_3(z_i) < \sigma_3(z_j)$. We define

$$U_1 = \{s \in V_1 \cap V(H) \colon \sigma_1(x_i) < \sigma_1(s) < \sigma_1(x_j)\}$$

$$U_2 = \{s \in V_2 \cap V(H) \colon \sigma_2(y_i) < \sigma_2(s) < \sigma_2(y_j)\},$$

and

$$U_3 = \{s \in V_3 \cap V(H) \colon \sigma_3(z_i) < \sigma_2(s) < \sigma_3(z_j)\}.$$

Because H is connected and x_iy_i, y_iz_i, x_jy_j, and y_jz_j do not cross any edge, we have that

$$U_1 = \{s \in V_1 \cap V(H) \colon \sigma_1^H(x_i) < \sigma_1^H(s) < \sigma_1^H(x_j)\},$$

$$U_2 = \{s \in V_2 \cap V(H) \colon \sigma_2^H(y_i) < \sigma_2^H(s) < \sigma_2^H(y_j)\},$$

and

$$U_3 = \{s \in V_3 \cap V(H) \colon \sigma_3^H(z_i) < \sigma_3^H(s) < \sigma_3^H(z_j)\}.$$

Also, there is no vertex $s \in V(G) \setminus (U_1 \cup U_2 \cup U_3)$ such that $\sigma_1(x_i) < \sigma_1(s) < \sigma_1(x_j)$ if $s \in V_1$, $\sigma_2(y_i) < \sigma_2(s) < \sigma_2(y_j)$ if $s \in V_2$, and $\sigma_3(z_i) < \sigma_3(s) < \sigma_3(z_j)$ if $s \in V_3$.

We construct a 3-layer drawing $\sigma' = (\sigma'_1, \sigma'_2, \sigma'_3)$ of G' respecting (V'_1, V'_2, V'_3) by modifying σ as follows. We place the vertices of $U_1 \cap V(G')$ between x_i and x_j following their order in σ_1^H, we place the vertices of $U_2 \cap V(G')$ between y_i and y_j following their order in σ_2^H, and we place the vertices of $U_3 \cap V(G')$ between z_i and z_j following their order in σ_2^H. Because there are no crossing edges with their endpoints in U_1, U_2, or U_3, the number of crossings in σ' is at most the number of crossings in σ.

For the opposite direction, assume that there is a 3-layer drawing $\sigma' = (\sigma'_1, \sigma'_2, \sigma'_3)$ of G' respecting (V'_1, V'_2, V'_3) with at most k crossings. We construct a 3-layer drawing $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ of G respecting (V_1, V_2, V_3) such that the number of crossings in σ is at most the number of crossings in σ'. Because the number of crossings is at most k, there are $i \in \{1, \ldots, 2k+1\}$ and $j \in \{2k+3, \ldots, 4k+3\}$ such that the edges x_iy_i, y_iz_i, x_jy_j, and y_jz_i do not cross any edge with respect to σ'. We can assume that $\sigma'_1(x_i) < \sigma'_1(x_j)$, $\sigma'_2(y_i) < \sigma'_2(y_j)$, and $\sigma'_3(z_i) < \sigma'_3(z_j)$. Let

$$U_1 = \{s \in V_1 \cap V(H) \colon \sigma_1^H(x_i) < \sigma_1^H(s) < \sigma_1^H(x_j)\},$$

$$U_2 = \{s \in V_2 \cap V(H) \colon \sigma_2^H(y_i) < \sigma_2^H(s) < \sigma_2^H(y_j)\},$$

and

$$U_3 = \{s \in V_3 \cap V(H) \colon \sigma_3^H(z_i) < \sigma_3^H(s) < \sigma_3^H(z_j)\}.$$

We set

$$U'_1 = \{s \in V'_1 \cap V(H) \colon \sigma'_1(x_i) < \sigma_1(s) < \sigma'_1(x_j)\},$$

$$U'_2 = \{s \in V'_2 \cap V(H) \colon \sigma'_2(y_i) < \sigma'_2(s) < \sigma'_2(y_j)\},$$

and

$$U'_3 = \{s \in V'_3 \cap V(H) \colon \sigma'_3(z_i) < \sigma'_3(s) < \sigma'_3(z_j)\}.$$

Because the graph H' obtained from H by deleting the vertices of D and making x_{2k+1} and x_{2k+3} adjacent to y_{2k+2} is connected and x_iy_i, y_iz_i, x_jy_j, and y_jz_j do not cross any edges, $U'_1 \subseteq U_1$, $U'_2 \subseteq U_2$, and $U'_3 \subseteq U_3$. Furthermore, there is no vertex $s \in V(G) \setminus (U'_1 \cup U'_2 \cup U'_3)$ such that $\sigma'_1(x_i) < \sigma'_1(s) < \sigma'_1(x_j)$ if $s \in V_1$, $\sigma'_2(y_i) < \sigma'_2(s) < \sigma'_2(y_j)$ if $s \in V_2$, and $\sigma'_3(z_i) < \sigma'_3(s) < \sigma'_3(y_j)$ if $s \in V_3$.

We construct σ_1 from σ'_1 by replacing the vertices of U'_1 with the vertices of U_1 and putting them between x_i and x_j following their order in σ_1^H. Similarly, σ_2 is constructed from σ'_2 by replacing the vertices of U'_2 with the vertices of U_2 and putting them between y_i and y_j following their order in σ_2^H, and σ_3 is constructed from σ'_3 by replacing the vertices of U'_3 by the vertices of U_3 and putting them between z_i and z_j following their order in σ_3^H. This concludes the proof of the claim. □

Recall that the reduction rules are applied exhaustively. We show that if no rule could be applied then for yes-instances, the size of the graph is bounded.

Suppose that (G, V_1, V_2, V_3, k) is a yes-instance of 3-Layer Crossing Minimization. Then there is a 3-layer drawing $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ of G respecting (V_1, V_2, V_3) with at most k crossings. We denote by D the endpoints of crossing edges. Then $|D| \leq 4k$, $|D \cap V_2| \leq 2k$, and $|D \cap (V_1 \cup V_3)| \leq 2k$. First, we make the following observation.

Observation 16. Suppose that Reduction Rule 6.2 cannot be applied for (G, V_1, V_2, V_3, k). Then for every $u \in V_2$ and $i \in \{1, 3\}$, $|N_G(u) \cap (V_i \setminus D)| \leq k + 3$.

Proof. Let $u \in V_2$. We observe that the vertices of $N_G(u) \cap (V_i \setminus D)$ for $i \in \{1,3\}$ should be consecutive in σ_i because the edges uv with $v \in N_G(u) \cap (V_i \setminus D)$ do not cross any edge. Furthermore, at most two of them, namely the first and the last vertex of $N_G(u) \cap (V_i \setminus D)$ with respect to σ_i, may have neighbors distinct from u. This implies that $|N_G(u) \cap (V_i \setminus D)| \leq k+3$ because, otherwise, Reduction Rule 6.2 would be applied. This concludes the proof. □

For two (not necessarily distinct) vertices $u, v \in V_2$ with $\sigma_2(u) \leq \sigma_2(v)$, we say that $[u,v] = \{w \in V_2 \colon \sigma_2(u) \leq \sigma_2(w) \leq \sigma_2(v)\}$ is an *interval* of V_2. For interval $[u,v]$, we use $|[u,v]|$ to denote its size, that is, the number of vertices in $[u,v]$. We say that an interval $[u,v]$ is *safe* if $[u,v]$ contains no vertex incident with a crossing edge; $[u,v]$ is a *maximal* safe interval if $[u,v]$ is an inclusion maximal safe interval. Note that since the number of crossings is at most k, at most $2k$ vertices of V_2 are incident with crossing edges. Therefore, the number of maximal safe intervals is at most $2k+1$.

Lemma 28. *Suppose that Reduction Rule 6.1–Reduction Rule 6.6 cannot be applied for (G, V_1, V_2, V_3, k) and let $[x,y] \subseteq V_2$ be a safe interval such that any $u \in [x,y]$ has neighbors either only in V_1 or only in V_3. Then $|[x,y]| \leq 4(2k+3)(4k+2)$.*

Proof. For the sake of contradiction, assume that there is $[x,y]$ with $|[x,y]| > 4(2k+3)(4k+2)$. Then there is $X \subseteq [x,y]$ with $|X| > 2(2k+3)(4k+2)$ such that either every vertices of X has neighbors only in V_1 or every vertex of X has neighbors only in V_3. By symmetry, we assume without loss of generality that $N_G(X) \subseteq V_1$.

We observe that the vertices of X are included in at most two connected components of G. Otherwise, if there are three vertices $a, b, c \in X$ in distinct connected components of G such that $\sigma_2(a) < \sigma_2(b) < \sigma_2(c)$, the connected component H containing b has no crossing edges. This contradicts the assumption that Reduction Rule 6.1 cannot be applied. Thus, there is $Y \subseteq X$ such that $|Y| > (2k+3)(4k+2)$ and the vertices of Y are in the same connected component.

Consider a vertex $w \in V_1$ that has neighbors in Y. Notice that at most two neighbors of w in Y, namely, the first and the last neighbor with respect to σ_2, may be adjacent to vertices distinct from w because the edges wy with $y \in Y$ do not cross any edge and every $y \in Y$ has neighbors only in V_1. Then because Reduction Rule 6.4 cannot be applied, $|N_G(w) \cap Y| \leq 2k+3$. Furthermore, if $|N_G(w) \cap Y| = 2k+3$ then at least two neighbors of w have other (and distinct) neighbors in V_1. As $|Y| > (2k+3)(4k+2)$, we have that at least $4k+3$ vertices of V_2 have distinct neighbors in Y, that is, there a matching M of size $4k+3$ with endpoints of the edges in Y and V_1. The edges of M do not cross any edge. Let u_1 and u_2 be the first and the last endpoint, respectively, in V_1 with respect to σ_1, and let v_1 and v_2 be the first and the last endpoint, respectively, with respect to σ_2. Let $A_1 = \{w \in V_1 \colon \sigma_1(u_1) \leq \sigma_1(w) \leq \sigma_1(u_2)\}$ and $A_2 = \{w \in [x,y] \colon \sigma_2(v_1) \leq \sigma_2(w) \leq \sigma_2(v_2)$ and $N_G(w) \cap V_3 = \emptyset\}$. We obtain that Reduction Rule 6.5 can be applied for $A = A_1 \cup A_2$ and u_1, u_2, v_1, v_2. However, this contradicts the assumption that the reduction rules are not applicable. We conclude that $|[x,y]| \leq 4(2k+3)(4k+2)$. □

Lemma 29. *Suppose that Reduction Rule 6.1–Reduction Rule 6.8 cannot be applied for (G, V_1, V_2, V_3, k) and let $[u,v] \subseteq V_2$ be a safe interval. Let also $Z \subseteq [u,v]$ be the set of vertices having neighbors in both V_1 and V_3. Then for any $w \in V_1 \cup V_3$, w has at most $(k+3)(4k+2)(2k+3)$ neighbors in Z.*

Proof. The proof is by contradiction. Suppose that there is $w \in V_1 \cup V_3$ such that for $U = N_G(w) \cap Z$, it holds that $|U| > (k+3)(4k+2)(2k+3)$. By symmetry, we assume without loss of generality that $w \in V_1$.

For each vertex $x \in V_3$, at most two neighbors of x in U, namely, the first and the last neighbor with respect to σ_2, are adjacent to vertices distinct from x and w. This is because the edges xy and wy with $y \in U$ do not cross any edges. Since Reduction Rule 6.3 is not applicable, we have that $|N_G(x) \cap X| \leq k+3$.

Therefore, there is $W \subseteq U$ of size at least $(4k+2)(2k+3)+1$ such the vertices of W have distinct neighbors in V_3. Suppose that there is a connected component of $G-w$ that contains at least $4k+3$ vertices $x_1, \ldots, x_{4k+3} \in W$. Let us assume that $\sigma_2(x_1) < \cdots < \sigma_2(x_{4k+3})$. Denote by $y_1, \ldots, y_{4k+3}$ their respective distinct neighbors in V_3. Note that $\sigma_3(y_1) < \cdots < \sigma_2(y_{4k+3})$. Let $u_1 = x_1$, $u_2 = x_{4k+3}$, $v_1 = y_1$, and $v_2 = y_{4k+3}$. Also, set

$$A_1 = \{x \in V_2 \colon \sigma_2(x_1) \leq \sigma_2(x) \leq \sigma_2(x_{4k+3})\}$$

and

$$A_2 = \{y \in V_3 \colon \sigma_3(y_1) \leq \sigma_3(y) \leq \sigma_3(y_{4k+3})\}.$$

Then Reduction Rule 6.8 can be applied for $A = A_1 \cup A_2$ and w, u_1, u_2, v_1, v_2; which is a contradiction. Therefore, each connected component of $G-w$ containing vertices of W has at most $4k+2$ such vertices. Thus, there are at least $2k+4$ connected components of $G-w$ with vertices of W. At most two such components, namely the components containing the first and the last vertices of W with respect to σ_2, are not nice. Therefore, we have at least $2k+2$ connected components of $G-w$ that are nice. In this case, we can apply Reduction Rule 6.4, thus contradicting the assumption that the reduction rules are not applicable. It yields that the assumption $|W| > (4k+2)(2k+3)$ was wrong. This concludes the proof. □

Lemma 30. *Suppose that Reduction Rule 6.1–Reduction Rule 6.8 cannot be applied for (G, V_1, V_2, V_3, k) and let $[u, v] \subseteq V_2$ be a safe interval. Let also $R = \{y_1, \ldots, y_\ell\} \subseteq [u, v]$ be an inclusion maximal set of vertices such that each vertex of R has neighbors in both V_1 and V_3 and all these neighbors are distinct, and assume that $\sigma_2(y_1) < \cdots < \sigma_2(y_\ell)$. Then*

$$|[u, v]| \leq 4(\ell+1)(k+3)(4k+2)(2k+3)[4(2k+3)(4k+2)+1],$$

and if $\ell \geq 2$ then

$$|[y_1, y_\ell]| \leq 4(\ell-1)(k+3)(4k+2)(2k+3)[4(2k+3)(4k+2)+1].$$

Proof. For $i \in \{1, \ldots, \ell\}$, denote by x_i and z_i neighbors of y_i in V_1 and V_3, respectively, chosen in such a way that $x_1, \ldots, x_\ell$ and all $z_1, \ldots, z_\ell$ are distinct. We set $y_0 = u$ and $y_{\ell+1} = v$ (it may happen that $y_0 =$ or $y_{\ell+1} = y_\ell$). Let also x_0 and $x_{\ell+1}$ be the first and the last vertex, respectively, of V_1 with respect to σ_1 that is adjacent to a vertex of $[u, v]$. Similarly, let z_0 and $z_{\ell+1}$ be the first and the last vertex of V_3 with respect to σ_3 that is adjacent to a vertex of $[u, v]$. It may happen that $x_0 = x_1$ or $x_{\ell+1} = x_\ell$ or $z_0 = z_1$ or $z_{\ell+1} = z_\ell$. For $i \in \{0, \ldots, \ell\}$, denote $I_i = [y_i, y_{i+1}]$. We prove the following claim.

Claim 6.1. *For every $i \in \{0, \ldots, \ell\}$, $|I_i| \leq 4(k+3)(4k+2)(2k+3)[4(2k+3)(4k+2)+1]$.*

Proof of Claim 6.1. Consider arbitrary $i \in \{0, \ldots, \ell\}$. By Lemma 28, for a subinterval $[y', y''] \subseteq I_i$ such that any $y \in [y', y'']$ has neighbors either only in V_1 or only in V_3, it holds that $|[y', y'']| \leq 4(2k+3)(4k+2)$. Then I_i contains at least $\frac{|I_i|}{4(2k+3)(4k+2)+1}$ vertices having neighbors in both V_1 and V_3. Observe that if $y \in I_i \setminus \{y_i, y_{i+1}\}$ is adjacent to $x \in V_1$ and $z \in V_3$ then either $x \in \{x_i, x_{i+1}\}$ or $z \in \{z_i, z_{i+1}\}$ by the maximality of R. By Lemma 29, each of the vertices $x_i, x_{i+1}, y_i, y_{i+1}$ is adjacent to at most $(k+3)(4k+2)(2k+3)$ vertices of $[u, v]$ that have neighbors in both V_1 and V_2. Thus, the total number of such neighbors of $x_i, x_{i+1}, y_i, y_{i+1}$ is at most $4(k+3)(4k+2)(2k+3) \geq \frac{|I_i|}{4(2k+3)(4k+2)+1}$. Therefore, $|I_i| \leq 4(k+3)(4k+2)(2k+3)[4(2k+3)(4k+2)+1]$. This concludes the proof. □

By Claim 6.1, we have that

$$|[u, v]| \leq \sum_{i=0}^{\ell} |I_i| \leq 4(\ell+1)(k+3)(4k+2)(2k+3)[4(2k+3)(4k+2)+1],$$

and if $\ell \geq 2$ then

$$|[y_1, y_\ell]| \leq \sum_{i=1}^{\ell-1} |I_i| \leq 4(\ell-1)(k+3)(4k+2)(2k+3)[4(2k+3)(4k+2)+1],$$

This concludes the proof. □

Now we are ready to prove the crucial lemma.

Lemma 31. *Suppose that Reduction Rule 6.1–Reduction Rule 6.9 are not applicable for a yes-instance* (G, V_1, V_2, V_3, k) *of* 3-Layer Crossing Minimization. *Then* $|V(G)| \leq 2^{16}(k+4)^{11}$.

Proof. Suppose that (G, V_1, V_2, V_3, k) is a yes-instance of 3-Layer Crossing Minimization such that Reduction Rule 6.1–Reduction Rule 6.9 are not applicable. Recall that for a 3-layer drawing $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ of G respecting (V_1, V_2, V_3) with at most k crossings, we use D to denote the endpoints of crossing edges. Recall also that $|D| \leq 4k$, $|D \cap V_2| \leq 2k$, and $|D \cap (V_1 \cup V_3)| \leq 2k$. Notice that because Reduction Rule 6.1 is not applicable, G has no isolated vertices. That is, every vertex has at least one neighbor.

We claim that

$$|V_2| \leq 4(8k+6)(2k+1)(k+3)(4k+2)(2k+3)[4(2k+3)(4k+2)+1] + 2k. \tag{4}$$

To prove it, we upper bound the size of maximal safe intervals. Let $[u, v]$ be a maximal safe interval. Let also $R = \{y_1, \ldots, y_\ell\} \subseteq [u, v]$ be a set of maximum size such that each vertex of R has neighbors in both V_1 and V_3 and all these neighbors are distinct, and assume that $\sigma_2(y_1) < \cdots < \sigma_2(y_\ell)$. We claim that $\ell \leq 8k+5$.

For the sake of contradiction, assume that $\ell \geq 8k+6$. Notice that the vertices of $[u, v]$ are included in at most two connected components of G. Otherwise, if there are three vertices $a, b, c \in [u, v]$ in distinct connected components of G such that $\sigma_2(a) < \sigma_2(b) < \sigma_2(c)$, the connected component C containing b has no crossing edges contradicting the assumption that Reduction Rule 6.1 cannot be applied. This implies that the first $4k+3$ or the last $4k+3$ vertices of R are in the same connected component. By symmetry, we can assume without loss of generality that $y_1, \ldots, y_{4k+3}$ are in the same connected component. By the definition of R, for each $i \in \{1, \ldots, 4k+3\}$ there are neighbors x_i and z_i of y_i in V_1 and V_3 suchthat $x_1, \ldots, x_{4k+3}$ and all $z_1, \ldots, z_{4k+3}$ are distinct. Note that $\sigma_1(x_1) < \cdots < \sigma_1(x_{4k+3})$ and $\sigma_3(z_1) < \cdots < \sigma_3(z_{4k+3})$. However, then Reduction Rule 6.9 is applicable for $H = G[A_1 \cup A_2 \cup A_3]$ where

$$A_1 = \{x \in V_1 \colon \sigma_1(x_1) \leq \sigma_2(x) \leq \sigma_1(x_{4k+3})\}$$

$$A_2 = \{y \in V_2 \colon \sigma_2(y_1) \leq \sigma_2(y) \leq \sigma_2(y_{4k+3})\},$$

and

$$A_3 = \{z \in V_3 \colon \sigma_3(z_1) \leq \sigma_3(z) \leq \sigma_3(s_{4k+3})\}$$

with $u_1 = x_1$, $v_1 = y_1$, $w_1 = z_1$, $u_2 = x_{4k+3}$, $v_2 = y_{4k+3}$, and $w_2 = z_{4k+3}$. This contradicts the assumption that the rule is not applicable. Thus, $\ell \leq 8k+5$.

By Lemma 30, we obtain that

$$|[u, v]| \leq 4(8k+6)(k+3)(4k+2)(2k+3)[4(2k+3)(4k+2)+1].$$

Because the number of maximal safe intervals is at most $2k+1$, we obtain that at most $4(8k+6)(2k+1)(k+3)(4k+2)(2k+3)[4(2k+3)(4k+2)+1]$ vertices of V_2 are in the maximal safe intervals. Since the only vertices that are not in a safe intervals are the vertices of $D \cap V_2$ and $|D \cap V_2| \leq 2k$, we conclude that (4) holds.

To conclude the proof of the lemma, recall that for every $u \in V_2$, $|N_G(u) \cap (V_i \setminus D)| \leq k+3$ for $i \in \{1,3\}$ by Observation 16. Therefore, $|(V_1 \cup V_2) \setminus D| \leq 2(k+3)|V_2|$. Recall that also that $|D \cap (V_1 \cup V_3)| \leq 2k$. Then

$$\begin{aligned}|V(G)| \leq& (2k+7)|V_2| + 2k\\ \leq& (2k+7)[4(8k+6)(2k+1)(k+3)(4k+2)(2k+3)[4(2k+3)(4k+2)+1]+2k]+2k\\ \leq& 2^{15}(k+2)^8.\end{aligned}$$

□

Now we state the final rule.

Reduction Rule 6.10. If $|V(G)| > 2^{15}(k+2)^8$ or $|E(G)| > 2^{16}(k+2)^8 + k$ then return a trivial no-instance and stop.

Lemma 32. *Reduction Rule 6.9 is safe.*

Proof. If $|V(G)| > 2^{15}(k+2)^8$ then (G, V_1, V_2, V_3, k) is a no-instance of 3-Layer Crossing Minimization by Lemma 31. Suppose that $|V(G)| \leq 2^{15}(k+2)^8$. If (G, V_1, V_2, V_3, k) is a yes-instance then there is a set S of at most k edges such that the graph G' obtained from G by the deletion of the edges of S is planar. Because G' is bipartite with the bipartition $(V_2, V_1 \cup V_3)$, we have that $|E(G')| \leq 2|V(G')| - 4 \leq 2^{16}(k+2)^8 - 4$. Thus, if $|E(G)| > 2^{16}(k+2)^8 + k$ then (G_1, G_2, G_3, k) is a no-instance. □

This completes the description of the kernelization algorithm. The correctness follows from Lemma 22–Lemma 32.

6.2 Running time evaluation

It is easy to show that each of the reduction rules can be executed in polynomial time implying that the total running time of the kernelization algorithm is polynomial. However, a naive implementation may give a huge dependence on n. For example, for the direct application of Reduction Rule 6.9, we can consider all 6-tuples $(u_1, v_1, w_1, u_2, v_2, w_2)$ of vertices and then consider all separations (A, B) with $A \cap B = \{u_1, v_1, w_1, u_2, v_2, w_2\}$ where $G[A]$ is connected and conditions (i)–(iv) are fulfilled. It is straightforward to verify (i) and (ii), then condition (iii) is verified by making use of Proposition 1 in linear time, and to verify (iv) and find paths, we can use the standard maximum flow algorithm of Ford and Fulkerson [24]. However, this leads to $\mathcal{O}(n^8)$ time. We show that with some additional work, we can achieve running time which is linear in n. To obtain this, we show that we can replace the exhaustive application of each rule whenever possible by a restricted number of calls in such a way that the total running time used by the rule is linear in n. Also, we implement some additional steps in the algorithm allowing us to rule out no-instances. We give them here instead of Section 6.1 because they do not influence the kernel size and their only purpose is to speed up computations.

Recall that at the beginning of the algorithm, we use Proposition 1, to check in time $\mathcal{O}(n)$ whether G admits a 3-layer drawing respecting (V_1, V_2, V_3) without crossings. If this is the case, we immediately return a trivial yes-instance and stop. We also stop and output a trivial no-instance if $k = 0$ and G does not admit such a drawing. Further, we note that we can bound the number of the edges in the input instance (G, V_1, V_2, V_3, k). In the same way as in the proof of Lemma 32, we observe that if (G, V_1, V_2, V_3, k) is a yes-instance then there is a set S of at most k edges such that the graph G' obtained from G by the deletion of the edges of S is planar. Then because G' is bipartite, $|E(G')| \leq 2|V(G')| - 4$. Thus, if $|E(G)| \geq |V(G)| + k - 3$, we conclude that (G, V_1, V_2, V_3, k) is a no-instance and return a trivial no-instance. If $n \leq 2^{15}(k+2)^8$ then we can simply return the input instance. This means that it can be assumed that G has no drawing

respecting (V_1, V_2, V_3) without crossings, $k \geq 1$, $|E(G)| \leq n + k - 3$, and $n > 2^{15}(k+2)^8$. In particular, we have that $|E(G)| \leq 2n$, that is, the size of the input graph is linear in n.

To apply Reduction Rule 6.1, we use standard tools [32] to compute the connected components of G in $\mathcal{O}(n)$ time. Then for each connected component H, we use Proposition 1 to decide whether H a 3-layer drawing respecting $(V_1 \cap V(H), V_2 \cap V(H), V_3 \cap V(H))$ without crossings and delete the vertices of H if this holds. By Proposition 1, we have that this can be done in $\mathcal{O}(n)$ time. Notice that the subsequent reduction rules do not create new connected components and do not turn any component into a graph that could be drawn without crossings. Thus, Reduction Rule 6.1 will not be called again afterward.

For Reduction Rule 6.2, we observe that the inclusion maximal sets of pendent vertices $P \subseteq V_1$ or $P \subseteq V_3$ that have the same neighbors in V_2 can be found in linear time. Then Reduction Rule 6.2 can be applied in the total $\mathcal{O}(n)$ time for all such subsets. Similarly, to apply Reduction Rule 6.3, we find inclusion maximal subsets of vertices $Q \subseteq V_2$ of degree two that have the same unique neighbors in V_1 and V_2. This can be done in $\mathcal{O}(n)$ time either directly or by observing that these sets compose modules of G and the modules can be found in linear time [9]. Note that Reduction Rule 6.3–Reduction Rule 6.8 do not create new pendent vertices. However, new pendent vertices may appear after applying Reduction Rule 6.9. This possibility will be considered in the analysis of Reduction Rule 6.9. Similarly, to apply Reduction Rule 6.3, we find inclusion maximal subsets of vertices $Q \subseteq V_2$ of degree two that have the same unique neighbors in V_1 and V_2. This can be done in $\mathcal{O}(n)$ time either directly or by observing that these sets compose modules of G and the modules can be found in linear time [9]. Notice that Reduction Rule 6.7–Reduction Rule 6.9 can create new vertices in V_2 of degree two with the neighbors in V_1 and V_3. To remedy this for Reduction Rule 6.7 and Reduction Rule 6.8, we simply repeat the described procedure applying Reduction Rule 6.3 after the exhaustive application of Reduction Rule 6.5 and Reduction Rule 6.7 and Reduction Rule 6.8. Since this is done two times, the total running time for Reduction Rule 6.3 is still linear. For Reduction Rule 6.9, we incorporate the application of Reduction Rule 6.3 in the analysis of the rule.

The analysis of the subsequent rules is more complicated and we do it in separate statements.

Lemma 33. *Reduction Rule 6.4 can be exhaustively applied in $\mathcal{O}(n)$ time.*

Proof. We show how to apply Reduction Rule 6.4 for nice connected components H of $G - u$ for vertices $u \in V_1$. The case $u \in V_3$ is symmetric. Recall that a connected component H of $G - u$ is nice if (i) $V(H) \subseteq V_2 \cup V_3$ and (ii) H admits a 2-layer drawing without crossings respecting either $(V_2 \cap V(H), V_3 \cap V(H))$. Because we are interested in H with $V(H) \subseteq V_2 \cup V_3$, we consider the graph $G' = G[V_2 \cup V_3]$ and list all connected components by standard algorithms [32]. Simultaneously, we compute the number of vertices in each component. This can be done in $\mathcal{O}(n)$ time. Then among all these components, we find the components H such that (i) H has a 2-layer drawing, that is, H is a caterpillar by Observation 1, and (ii) H has a unique neighbor in V_1; when we find such a neighbor u of H, we assign H to u. Finally, for all $u \in V_1$, if we have $s \geq 2k + 2$ components H assigned to the same vertex u then we delete $s - (2k + 1)$ components starting with trivial ones. This also can be done in linear time and we conclude that the total running time for Reduction Rule 6.4 is $\mathcal{O}(n)$. This concludes the proof. □

We note that Reduction Rule 6.5–Reduction Rule 6.8 cannot create new nice components for some $u \in V_1 \cup V_3$ but this may happen after applying Reduction Rule 6.9. We deal with this issue through the analysis of Reduction Rule 6.9.

The following easy observation is useful for the next rules.

Observation 17. If either $G[V_1 \cup V_2]$ or $G[V_2 \cup V_3]$ contains a cycle with at least $2k + 4$ vertices then (G, V_1, V_2, V_3, k) is a no-instance of 3-Layer Crossing Minimization.

Proof. Assume that C is a cycle in $G[V_1 \cup V_2]$ with at least $2k + 4$ vertices and consider arbitrary 3-layer drawing $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ of G.

Suppose that k is even. Let u be the first vertex of $V(C) \cap V_1$ in σ_1 and let v be the last vertex of $V(C) \cap V_2$ in σ_2. Because C is a cycle, there are two internally disjoint (u,v)-paths P and Q such that one of them, say, P has at least $k+3$ edges. Then Q crosses every edge of P except, possibly, the first and the last edge. Thus, the total number of crossings is at least $k+1$.

If k is odd, we let u and v be the first and the last vertices, respectively, of $V(C) \cap V_1$ in σ_1. We again have internally disjoint (u,v)-path P and Q is C such that P has at least $k+3$ edges. Then Q crosses at least $k+1$ edges of P.

In both cases, the number of crossings is at least $k+1$ implying that (G, V_1, V_2, V_3, k) is a no-instance. This concludes the proof. □

Lemma 34. *There is an algorithm running in $\mathcal{O}(n)$ time that either exhaustively applies Reduction Rule 6.5 and Reduction Rule 6.6 or correctly concludes that (G, V_1, V_2, V_3, k) is a no-instance.*

Proof. We show the lemma for Reduction Rule 6.5 as the proof for Reduction Rule 6.6 is symmetric. The idea is to find maximal subgraphs H satisfying conditions (i)–(iv) of the rule and apply the rule to them simultaneously.

Suppose that Reduction Rule 6.5 can be applied for $H = G[A]$ with the corresponding vertices u_1, u_2, v_1, v_2. Because H admits a 2-layer drawing $\sigma_H = (\sigma_1^H, \sigma_2^H)$ without crossings respecting $(V_1 \cap V(H), V_2 \cap V(H))$ such that (a) for every $w \in V_1$, $\sigma_1^H(u_1) \leq \sigma_1^H(w) \leq \sigma_1^H(u_2)$ and (b) for every $w \in V_2$, $\sigma_2^H(v_1) \leq \sigma_2^H(w) \leq \sigma_2^H(v_2)$, we have that H is a caterpillar by Observation 1 with a backbone P that has one end-vertex in $\{u_1, v_1\}$, the other in $\{u_2, v_2\}$, and does not contain u_1v_1, u_2v_2; we call such a backbone *proper*. Since H has a matching of size $4k+3$, $|V(P)| \geq 4k+3$. Notice that each inner vertex w of P has no neighbors in V_3 and is adjacent to at most two vertices with non-pendent neighbors.

We use these properties to identify the backbones of caterpillars for which the rule will be applied. More precisely, we aim to find inclusion maximal proper backbones and we say that H is *backbone maximal* if the proper backbone of H is inclusion maximal. Notice that G can contain distinct H with the same proper backbone P. In particular, while one endpoint of u_1v_1 and u_2v_2 is in P, we can choose the second endpoint in a different way. However, for any possible choice of these vertices, the edges u_1v_1 and u_2v_2 are incident to the same vertices of P. This implies the following property for any two graphs H and H' with the same proper backbone P with boundary vertices u_1, u_2, v_1, v_2 and u_1', u_2', v_1', v_2', respectively: M is a matching in H with $u_1v_1, u_2v_2 \in M$ if and only if $M' = (M \setminus \{u_1v_1, u_2v_2\}) \cup \{u_1'v_1', u_2'v_2'\}$ is a matching in H' with $u_1'v_1', u_2'v_2' \in M'$. Thus, it is sufficient to find the inclusion maximal proper backbones and restrict the application of Reduction Rule 6.5 to the graphs H constructed for them.

We find the sets $U_i \subseteq V_i$ of the vertices that have two non-pendent neighbors for $i \in \{1,2\}$ where each vertex of U_2 has no neighbors in V_3. Then we construct $G[U_1 \cup U_2]$ and find all connected components of size at least $4k+1$ in linear time using the depth-first search [32]. Each connected component is either a cycle or a path. If there is a component C that is a cycle with at least $4k+1$ vertices, C has at least $4k+2 \geq 2k+4$ vertices by bipartiteness. By Observation 17, we conclude that (G, V_1, V_2, V_3, k) is a no-instance and stop. From now on, we assume that this is not the case and the family $\mathcal{P}$ of connected components of $G[U_1 \cup U_2]$ consists of paths with at least $4k+1$ vertices.

For both end-vertices of each path $P \in \mathcal{P}$, we do the following. By the construction, each end-vertex of P has a neighbor of degree two that is not in P. If these neighbors are the same we obtain that $G[V_1 \cup V_2]$ has a cycle with at least $4k+2 \geq 2k+4$ vertices. Then by Observation 17, (G, V_1, V_2, V_3, k) is a no-instance and we stop. Otherwise, we construct P' from P by appending these neighbors. Thus, we obtain the family $\mathcal{P}'$ of all potential edge-disjoint backbones of caterpillars H for which Reduction Rule 6.5 can be applied and, furthermore, the construction of $\mathcal{P}'$ can be done in linear time. Notice that the paths in $\mathcal{P}'$ are not necessarily proper backbones because the first and the last edges may be not in a proper backbone. However, each inclusion maximal proper backbone is included in some path of the family.

For each $P \in \mathcal{P}'$, we construct H_P together with vertices $u_1, u_2 \in V_1$ and $v_1, v_2 \in V_2$ such that P is a backbone of H_P. Initially, we set $H_P := P$. Let x and y be the end-vertices of P.

(i) For each $z \in V(P) \setminus \{x, y\}$, find the pendent vertices $z' \in V_1 \cup V_2$ adjacent to z and add z' and zz' to H_P.

(ii) If x is adjacent in G to a vertex $z \in (V_1 \cup V_2) \setminus V(P)$ then add z and xz to H_P. Otherwise, select z to be the neighbor of x in P and set $P := P - x$. Then set $\{u_1, v_1\} := \{x, z\}$.

(iii) If y is adjacent in G to a vertex $z \in (V_1 \cup V_2) \setminus V(P)$ then add z and yz to H_P. Otherwise, select z to be the neighbor of y in P and set $P := P - y$. Then set $\{u_2, v_2\} := \{y, z\}$.

Notice that it may happen that $\{u_1, v_1\} \cap \{u_2, v_2\} \neq \emptyset$ if either z selected in (ii) equals y, or z selected in (iii) equals x, or the vertices z selected in (ii) and (iii) are the same. However, in all these cases, we obtain that $G[V_1 \cup V_2]$ has a cycle with at least $4k + 2 \geq 2k + 4$ vertices. Then we conclude that (G, V_1, V_2, V_3, k) is a no-instance and we stop. Otherwise, H_P with u_1, u_2, v_1, v_2 is a backbone maximal subgraph of G for an inclusion maximal backbone P with the properties that

(i) H_P is a connected graph such that for $A = V(H_P) \subseteq V_1 \cup V_2$ and $B = (V(G) \setminus V(H_P)) \cup \{u_1, u_2, v_1, v_2\}$, (A, B) is a separation of G with $A \cap B = \{u_1, u_2, v_1, v_2\}$, and $u_1, u_2 \in V_1$, and $v_1, v_2 \in V_2$,

(ii) $u_1v_1, u_2v_2 \in E(G)$,

(iii) H_P has a 2-layer drawing $\sigma_H = (\sigma_1^H, \sigma_2^H)$ without crossings respecting $(V_1 \cap V(H_P), V_2 \cap V(H_P))$, where σ_i^H is an order of $V_i \cap V(H_P)$ for $i \in \{1, 2\}$, such that (a) for every $w \in V_1$, $\sigma_1^H(u_1) \leq \sigma_1^H(w) \leq \sigma_1^H(u_2)$ and (b) for every $w \in V_2$, $\sigma_2^H(v_1) \leq \sigma_2^H(w) \leq \sigma_2^H(v_2)$,

(iv) $|V(H_P)| \geq 4k + 1$.

Furthermore, the construction of H_P for all $P \in \mathcal{P}'$ can be done in linear time.

Let $P \in \mathcal{P}'$. We check whether H_P has a matching M with at least $4k + 3$ edges containing u_1v_1 and u_2v_2 and if such a matching exists we pick M in such a way that would allow us to delete the maximum number of vertices. We use the fact that H_P is a caterpillar and the 2-layer drawing of H_P is unique by Observation 1 up to the reversals of the orders and permutation of pendent vertices in the orders. We find the 2-layer drawing $\sigma_H = (\sigma_1^H, \sigma_2^H)$ without crossings with $\sigma_1^H(u_1) < \sigma_1^H(u_2)$ and $\sigma_2^H(v_1) < \sigma_2^H(v_2)$. Notice that the vertices of $V(P) \cap V_1$ and $V(P) \cap V_2$ are ordered in the path order and for every matching M, at least one of the endpoints of of each edge in P. We find M by the following greedy procedure. We set $M := \{x_1y_1\}$ where $x_1 = u_1$ and $y_1 = v_1$. Then we try to add $2k + 1$ edges x_iy_i with $x_i \in V_1 \cap V(H_P)$ and $y_i \in V_2 \cap V(H_P)$ for $i \in \{2, \ldots, 2k + 2\}$. We greedily select first with respect to σ_1^H vertex x_i that is not incident to any edge of the already constructed matching that has a neighbor that is not saturated in M. Then among the unsaturated neighbors of x_i, we select y_i which is first with respect to σ_2^H. If we succeed with the first $2k + 2$ edges we proceed with selecting $2k + 1$ from the other side. We add $x_{4k+3}y_{4k+3}$ with $x_{4k+3} = u_2$ and $y_{4k+3} = v_2$ if possible. Then we add $2k$ edges $x_{4k+3-i}y_{4k+3-i}$ for $i \in \{1, \ldots 2k\}$ following the reversals of σ_1^H and σ_2^H. If this greedy choice gives a matching M of size $4k + 3$ we apply Reduction Rule 6.5. By the choice of M, we have that the rule would be applied at most once for each H_P. Because σ_H can be constructed in linear time by Observation 1 and the greedy procedure can be done in linear time, we conclude that the overall running time is $\mathcal{O}(n)$.

Finally, we note that because of the choice of the edges added by the rule, the rule cannot be applied a second time for the graph obtained from each H_P. This completes the proof. □

We use similar arguments to deal with Reduction Rule 6.7 and Reduction Rule 6.8.

Lemma 35. *There is an algorithm running in $\mathcal{O}(n)$ time that either exhaustively applies Reduction Rule 6.7 and Reduction Rule 6.8 or correctly concludes that (G, V_1, V_2, V_3, k) is a no-instance.*

Proof. By symmetry, it is sufficient to prove the lemma for Reduction Rule 6.7. As in the previous lemma, we find inclusion maximal subgraphs H satisfying conditions (i)–(v) of the rule and apply the rule for them simultaneously.

Suppose that Reduction Rule 6.7 can be applied for $H = G[A]$ with the corresponding vertices u_1, u_2, v_1, v_2, w. Since H admits a 2-layer drawing $\sigma_H = (\sigma_1^H, \sigma_2^H)$ without crossings respecting $(V_1 \cap V(H), V_2 \cap V(H))$ such that (a) for every $x \in V_1$, $\sigma_1^H(u_1) \leq \sigma_1^H(x) \leq \sigma_1^H(u_2)$ and (b) for every $x \in V_2$, $\sigma_2^H(v_1) \leq \sigma_2^H(x) \leq \sigma_2^H(v_2)$, H is a caterpillar by Observation 1 with a backbone P that has one end-vertex in $\{u_1, v_1\}$, the other in $\{u_2, v_2\}$, and does not contain u_1v_1, u_2v_2. As before, we call such a backbone *proper*. We find the potential inclusion maximal backbones and construct H with them. Again, we say that H is *backbone maximal* if the proper backbone of H is inclusion maximal. Note that since H has a matching of size $4k+3$, $|V(P)| \geq 4k+3$.

Consider $G' = G[V_1 \cup V_2]$. For $i \in \{1, 2\}$, we find the set $U_i \subseteq V_i$ of vertices that have two non-pendent neighbors in G'. We construct $G[U_1 \cup U_2]$ and find all connected components of size at least $4k+1$ in linear time using the depth-first search [32]. Each connected component is either a cycle or a path. If there is a component C that is a cycle with at least $4k+1$ vertices, then because G' is bipartite, C has at least $4k+2 \geq 2k+4$ vertices and (G, V_1, V_2, V_3, k) is a no-instance by Observation 17. In this case, we stop. Otherwise, each connected component is a path. For such a path P, each end-vertex of P has a neighbor of degree two that is not in P by definition. If these neighbors are the same, we conclude that (G, V_1, V_2, V_3, k) is a no-instance by Observation 17 and stop. Assume that this is not the case. We modify each P by appending the corresponding neighbors of the end-vertices. This way we obtain in $\mathcal{O}(n)$ time the family of edge-disjoint path $\mathcal{P}$ with the property that every inclusion maximal proper backbone is a subpath of one of the paths of the family.

Up to now, our analysis did not include the vertices of V_3. Recall that we are looking for H together with u_1, u_2, v_1, v_2, w such that H is separated by $\{u_1, u_2, v_1, v_2, w\}$. To find inclusion maximal backbones, we consider subpaths of the paths of $\mathcal{P}$.

Consider $P \in \mathcal{P}$. For a vertex $v \in V(P)$, let $R(v)$ be the sets of pendent neighbors of $v \notin V(P)$ in G'. Note that these sets can be constructed in linear time. Let $\{s_0, \ldots, s_\ell\} = V(P) \cap V_2$ and assume that the vertices are numbered in the path order. For $i \in \{1, \ldots, \ell\}$, denote by t_i the common neighbor of s_{i-1} and s_i in P. We find inclusion maximal (s_i, s_j)-subpaths Q_P of P for $0 \leq i \leq j \leq \ell$ such that there is $w \in V_3$ that

(i) there is $z \in \{s_i\} \cup R(t_{i+1})$ such that w is the unique neighbor of z in V_3,

(ii) there is $z \in \{s_j\} \cup R(t_j)$ such that w is the unique neighbor of z in V_3,

(iii) for any $z \in \{s_i, \ldots, s_j\} \cup \bigcup_{p=i+1}^{j} R(t_p)$, z either has no neighbors in V_3 or is adjacent to w.

All these paths Q_P for all $P \in \mathcal{P}$ can be constructed in $\mathcal{O}(n)$ time by tracing each path P in the path order. For each $P \in \mathcal{P}$, we find the paths Q_P with at least $4k+1$ vertices each and denote by $\mathcal{Q}_P$ the family of all these paths. We use the path from $\mathcal{Q}_P$ for finding the inclusion maximal proper backbones. However, in some cases, we have to these paths.

Consider a path $Q \in \mathcal{Q}_P$ for some $P \in \mathcal{P}$ and assume that Q is the (s_i, s_j)-path for $0 \leq i < j \leq \ell$. We construct Q' together with special vertices v_1 and v_2 as follows.

Suppose that $i > 0$ and there is $i' \in \{0, \ldots, i-1\}$ such that there exists $v_1 \in \{s_{i'}\} \cup R(t_{i'+1})$ adjacent to w (and some other vertex of V_3) but for any $i'' \in \{i'+1, i-1\}$, the vertices of $\{s_{i''}\} \cup R(t_{i''+1})$ are not adjacent to any vertex of V_3. Then we extend Q by adding v_1, $v_1t_{i'+1}$ and the $(t_{i'+1}, s_i)$ subpath of P. Assume now that this is not the case and for each $i' \in \{0, i-1\}$, the vertices of $\{s_{i'}\} \cup R(t_{i'+1})$ are not adjacent to any vertex of V_3. Then if the first vertex z of P is in V_1 and there is $v_1 \in V_2 \setminus V(P)$ adjacent to both w and z, we add v_1, v_1z, and the (z, s_i)-subpath.

In all other cases, we set v_1 to be the vertex of $\{s_i\} \cup R(t_{i+1})$ adjacent to w, delete s_i, and add v_1 and v_1t_{i+1}.

Symmetrically, if $j < \ell$ and there is $j' \in \{j+1, \ldots, \ell\}$ such that there exists $v_2 \in \{s_{j'}\} \cup R(t_{j'})$ adjacent to w (and some other vertex of V_3) but for any $j'' \in \{j+1, j'-1\}$, the vertices of $\{s_{j'}\} \cup R(t_{j''})$ are not adjacent to any vertex of V_3 then we extend Q by adding v_2, $v_2t_{j''}$ and the $(t_{i''}, s_j)$ subpath of P. Furthermore, if this is not the case but for each $j' \in \{j+1, \ell\}$, the vertices of $\{s_{j'}\} \cup R(t_{j'})$ are not adjacent to any vertex of V_3 and it holds that the last vertex z of P is in V_1 and there is $v_2 \in V_2 \setminus V(P)$ adjacent to both w and z, we add v_2, v_2z, and the (s_j, z)-subpath. In all other cases, we set v_2 to be the vertex of $\{s_j\} \cup R(t_j)$ adjacent to w, delete s_j, and add v_2 and v_2t_j.

Observe that by the above construction, it may happen that $v_1 = v_2$. However, in this case, we obtain a cycle in G' with at least $4k+2$ vertices and by Observation 17, conclude that (G, V_1, V_2, V_3, k) is a no-instance. From now on, we assume that this is not the case and we obtained a path. We denote it by Q'. Observe that all such paths can be constructed from the paths $Q \in \mathcal{Q}_P$ in linear time by tracing P.

For given Q and Q' together with two special vertices v_1 and v_2, we construct H_Q together with u_1, u_2, v_1, v_2 as follows. Initially, $H_Q := Q'$.

- For each internal vertex z of Q' which is an internal vertex of Q, we add z and the vertices of $R(z)$ together with incident edges.
- If there is a vertex of $V_1 \setminus V(Q')$ adjacent to v_1 then select u_1 be such a vertex. Otherwise, set u_1 be the neighbor of v_1 in Q'.
- If there is a vertex of $V_1 \setminus V(Q')$ adjacent to v_2 then select u_2 be such a vertex. Otherwise, set u_2 be the neighbor of v_2 in Q'.

If we obtain that $u_1 = u_2$ then by by Observation 17, (G, V_1, V_2, V_3, k) is a no-instance and we stop. Otherwise, H_Q is a caterpillar with the properties:

(i) H_Q is a connected graph such that for $A = V(H_Q) \cup \{w\}$ and $B = (V(G) \setminus A) \cup \{u_1, u_2, v_1, v_2, w\}$, (A, B) is a separation of G with $A \cap B = \{u_1, u_2, v_1, v_2, w\}$, $A \cap V_3 = \{w_3\}$, $u_1, u_2 \in V_1$, and $v_1, v_2 \in V_2$.

(ii) $u_1v_1, u_2v_2, v_1w, v_2w \in E(G)$,

(iii) H_Q has a 2-layer drawing $\sigma_H = (\sigma_1^H, \sigma_2^H)$ without crossings respecting $(V_1 \cap V(H_Q), V_2 \cap V(H_Q))$, where σ_i^H is an order of $V_i \cap V(H_Q)$ for $i \in \{1, 2\}$, such that (a) for every $x \in V_1 \cap V(H_Q)$, $\sigma_1^H(u_1) \leq \sigma_1^H(x) \leq \sigma_1^H(u_2)$ and (b) for every $x \in V_2 \cap V(H)$, $\sigma_2^H(v_1) \leq \sigma_2^H(x) \leq \sigma_2^H(v_2)$,

(iv) $|V(H_Q)| \geq 4k+1$.

Furthermore, the construction of all H_Q for $Q \in \mathcal{Q}_P$ and $P \in \mathcal{P}$ can be done in $\mathcal{O}(n)$.

We apply Reduction Rule 6.7 for the constructed H_Q given together with the vertices u_1, u_2, v_1, v_2, w. Similarly to the proof of Lemma 34, we greedily find a matching M with at least $4k+3$ to ensure that the rule would delete the maximum number of vertices if the required matching exists. We use Observation 1 to find the unique (up to the permutations of pendent vertices) 2-layer drawing $\sigma_H = (\sigma_1^H, \sigma_2^H)$ without crossings with $\sigma_1^H(u_1) < \sigma_1^H(u_2)$ and $\sigma_2^H(v_1) < \sigma_2^H(v_2)$. Recall that w is adjacent to v_1 and v_2. We set $M := \{x_1y_1\}$ where $x_1 = u_1$ and $y_1 = v_1$. Then we try to add $2k+1$ edges x_iy_i with $x_i \in V_1 \cap V(H_Q)$ and $y_i \in V_2 \cap V(H_Q) \cap N_G(w)$ for $i \in \{2, \ldots, 2k+2\}$. We greedily select first with respect to σ_2^H vertex $y_i \in N_G(w)$ that is not incident to any edge of the already constructed matching that has a neighbor that is not saturated in M. Then among the unsaturated neighbors of y_i, we select x_i which is first with respect to σ_1^H. If we succeed with the first $2k+2$ edges we proceed with selecting $2k+1$ from the other side. We add $x_{4k+3}y_{4k+3}$ with

$x_{4k+3} = u_2$ and $y_{4k+3} = v_2$ if possible. Then we add $2k$ edges $x_{4k+3-i}y_{4k+3-i}$ for $i \in \{1, \dots 2k\}$ following the reversals of σ_1^H and σ_2^H. If this greedy choice gives a matching M of size $4k+3$ we apply Reduction Rule 6.7. By the choice of M, we have that the rule would be applied at most once for each H_Q. Because σ_H can be constructed in linear time by Observation 1 and the greedy procedure can be done in linear time, we conclude that the overall running time is $\mathcal{O}(n)$.

Notice that the rule does not create new paths of length two with their end-vertices in V_1 and V_3. Therefore, we cannot get a possibility to apply Reduction Rule 6.7 again. Also, after applying Reduction Rule 6.8 we do not get new possibilities to apply Reduction Rule 6.7. This completes the proof. □

We proved that Reduction Rule 6.1–Reduction Rule 6.8 can be exhaustively applied in $\mathcal{O}(n)$ time. However, for Reduction Rule 6.5–Reduction Rule 6.8, our analysis heavily relied on Observation 1. This cannot be done for Reduction Rule 6.9 and with this rule, we show that it can be implemented in $k^{\mathcal{O}(1)} \cdot n$ time. We use the following auxiliary results.

Suppose that Reduction Rule 6.9 can be applied for H with the corresponding vertices $u_1, v_1, w_1, u_2, v_2, w_2$ satisfying properties (i)–(iv) of the rule. In particular, H has a family of vertex disjoint paths $P = \{x_1y_1z_1, \dots, x_{4k+3}y_{4k+3}z_{4k+3}\}$ of size $4k+3$ with $u_1v_1w_1, u_2v_2w_2 \in P$ and it holds that $x_1 = u_1$, $y_1 = v_1$, $z_1 = w_1$, $x_{4k+3} = u_2$, $y_{4k+3} = v_2$, $z_{4k+3} = w_2$, $\sigma_1^H(x_1) < \cdots < \sigma_1^H(x_{4k+3})$, $\sigma_2^H(y_1) < \cdots < \sigma_2^H(y_{4k+3})$, and $\sigma_3^H(z_1) < \cdots < \sigma_3^H(z_{4k+3})$. We say that the vertex y_{2k+2} is a *center* of H.

Lemma 36. *Suppose that Reduction Rule 6.1–Reduction Rule 6.8 are not applicable for (G, V_1, V_2, V_3, k) and suppose that Reduction Rule 6.9 can be applied for H centered in $v \in V_2$ where H is an inclusion minimal subgraph with this property. Then $|V(H)| \leq 2^{13}(k+2)^7$ and all vertices of H are at distance at most $2^{11}(k+2)^6$ from v.*

Proof. Suppose that Reduction Rule 6.9 can be applied for H centered in v where H is chosen to be inclusion minimal. Consider the corresponding vertices u_1, v_1, w_1 and u_2, v_2, w_2, the 3-layer drawing $\sigma_H = (\sigma_1^H, \sigma_2^H, \sigma_2^H)$ without crossings respecting $(V_1 \cap V(H), V_2 \cap V(H), V_3 \cap V(H))$, and the family $P = \{x_1y_1z_1, \dots, x_{4k+3}y_{4k+3}z_{4k+3}\}$ of vertex disjoint paths of size $4k+3$ with $u_1v_1w_1, u_2v_2w_2 \in P$ and such that $x_1 = u_1$, $y_1 = v_1$, $z_1 = w_1$, $x_{4k+3} = u_2$, $y_{4k+3} = v_2$, $z_{4k+3} = w_2$, $\sigma_1^H(x_1) < \cdots < \sigma_1^H(x_{4k+3})$, $\sigma_2^H(y_1) < \cdots < \sigma_2^H(y_{4k+3})$, and $\sigma_3^H(z_1) < \cdots < \sigma_3^H(z_{4k+3})$. Recall that $v = y_{2k+2}$.

Applying Lemma 30 for H and σ, we obtain that

$$|[y_1, y_\ell]| \leq 4(4k+2)(k+3)(4k+2)(2k+3)[4(2k+3)(4k+2)+1].$$

Using Observation 16, we obtain that

$$\begin{aligned}|V(H)| \leq& |V(H) \cap V_2| + 2(k+3)|V(H) \cap V_2| = (2k+7)|V(H) \cap V_2| \\ \leq& 4(2k+7)(4k+2)(k+3)(4k+2)(2k+3)[4(2k+3)(4k+2)+1] \\ \leq& 2^{13}(k+2)^7.\end{aligned}$$

Similarly, we obtain that

$$[y_1, y_{2k+2}], [y_{2k+2}, y_{4k+3}] \leq 4(2k+1)(k+3)(4k+2)(2k+3)[4(2k+3)(4k+2)+1]$$

and, therefore, each vertex of H is at the distance at most

$$8(2k+1)(k+3)(4k+2)(2k+3)[4(2k+3)(4k+2)+1] \leq 2^{11}(k+2)^6$$

from v. This concludes the proof. □

Lemma 36 gives the following idea for the implementation of Reduction Rule 6.9. We consider vertices v of V_2 that have neighbors in both V_1 and V_2, that is, they potentially could be centers for H. Then we perform the breadth-first search from v on depth $\mathcal{O}(k^7)$ and apply the rule in the ball. However, for this, we have to guarantee that the ball has bounded size. We obtain such a bound in the next lemma.

Lemma 37. *Suppose that Reduction Rule 6.1–Reduction Rule 6.8 are not applicable for a yes-instance (G, V_1, V_2, V_3, k). Then for any $v \in V_2$ and any $\ell \geq 1$, the number of vertices of G at distance at most ℓ from v is at most $2^{12}(\ell + 2k + 2)(k + 2)^6$.*

Proof. Let $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ be a 3-layer drawing of G with at most k crossings. Let D be the set of endpoints of crossing edges. Recall that $|D \cap V_2| \leq 2k$ and $|D \cap (V_1 \cup V_3)| \leq 2k$. Consider $v \in V_2$ and let $r \in V_2$ be the last vertex with respect to σ_2 at distance at most ℓ from v. We prove the following claim.

Claim 6.2. $|[v, r]| \leq 4(\ell + 2k + 2)(k + 3)(4k + 2)(2k + 3)[4(2k + 3)(4k + 2) + 1] + 2k$

Proof. Let $R = \{y_1, \ldots, y_s\} \subseteq [v, r]$ be an inclusion maximal set of vertices is safe subintervals of $[v, r]$ such that each vertex of R has neighbors in both V_1 and V_3 and all these neighbors are distinct, and assume that $\sigma_2(y_1) < \cdots < \sigma_2(y_s)$. Notice that $s \leq \ell + 1$. Let $I_1, \ldots, I_p$ be the maximal safe subintervals of $[v, r]$ containing the vertices of R and let s_i be the number of vertices of R in I_i for $i \in \{1, \ldots, p\}$. Note that $p \leq 2k + 1$. By Lemma 30,

$$|I_i| \leq 4(s_i + 1)(k + 3)(4k + 2)(2k + 3)[4(2k + 3)(4k + 2) + 1],$$

for every $i \in \{1, \ldots, p\}$ and

$$\begin{aligned}\sum_{i=1}^{p} |I_i| \leq& 4(s + p)(k + 3)(4k + 2)(2k + 3)[4(2k + 3)(4k + 2) + 1]\\ \leq& 4(\ell + 2k + 2)(k + 3)(4k + 2)(2k + 3)[4(2k + 3)(4k + 2) + 1].\end{aligned}$$

Then

$$|[v, r]| \leq 4(\ell + 2k + 2)(k + 3)(4k + 2)(2k + 3)[4(2k + 3)(4k + 2) + 1] + 2k$$

as $|D \cap V_2| \leq 2k$. This proves the claim. □

Now let $r' \in V_2$ be the first vertex with respect to σ_2 at distance at most ℓ from v. By symmetry, we have that $|[r', v]|] \leq 4(\ell + 2k + 2)(k + 3)(4k + 2)(2k + 3)[4(2k + 3)(4k + 2) + 1] + 2k$ and conclude that

$$|[r', r]|] \leq 8(\ell + 2k + 2)(k + 3)(4k + 2)(2k + 3)[4(2k + 3)(4k + 2) + 1] + 2k$$

To prove the lemma, we observe that every vertex at distance at most ℓ from v is in $N_G[[r', r]]$. By Observation 16 and because $|D \cap (V_1 \cup V_2)| \leq k$,

$$\begin{aligned}|N_G[[r', r]]| \leq& |[r', r]| + 2(k + 3)|[r', r]| + 2k = (2k + 7)|[r', r]| + 2k\\ \leq& (2k + 7)[8(\ell + 2k + 2)(k + 3)(4k + 2)(2k + 3)[4(2k + 3)(4k + 2) + 1] + 2k] + 2k\\ \leq& 2^{12}(\ell + 2k + 2)(k + 2)^6.\end{aligned}$$

This proves the lemma. □

Lemma 38. *There is an algorithm running in $k^{\mathcal{O}(1)} \cdot n$ time that either exhaustively applies Reduction Rule 6.9 or correctly concludes that (G, V_1, V_2, V_3, k) is a no-instance. Furthermore, the algorithms simultaneously apply Reduction Rule 6.2–Reduction Rule 6.4 and Reduction Rule 6.7 and Reduction Rule 6.8 if they could be used after applying Reduction Rule 6.9.*

Proof. Let (G, V_1, V_2, V_3, k) be an instance of 3-Layer Crossing Minimization such that Reduction Rule 6.1–Reduction Rule 6.8 cannot be applied. We consider all vertices $v \in V_2$ having neighbors in both V_1 and V_3 and try to apply Reduction Rule 6.9 for H centered in v. For this, we do the following.

We perform BFS starting from v for the depth upper bounded by $2^{11}(k+2)^6$. If the implementation of BFS, we count the number of vertices visited by the algorithm. If BFS visits more than $2^{12}(2^{11}(k+2)^6+2k+2)(k+2)^6$ vertices in some step then we conclude that (G, V_1, V_2, V_3, k) is a no-instance by Lemma 37. We stop BFS and return the answer. Suppose that this is not the case and BFS produces the set of all vertices U at distance at most $2^{11}(k+2)^6$ from v whose size is at most $2^{12}(2^{11}(k+2)^6+2k+2)(k+2)^6$.

By Lemma 36, if Reduction Rule 6.9 can be applied for some H centered in v then such a graph can be found in $G' = G[W]$. Because $|W| \leq 2^{12}(2^{11}(k+2)^6+2k+2)(k+2)^6$, we can in $k^{\mathcal{O}(1)}$ time find H together with the vertices $u_1, v_2, w_1, u_2, v_2, w_2$, the 3-layer drawing $\sigma_H = (\sigma_1^H, \sigma_2^H, \sigma_3^H)$ of H, and the family of vertex disjoint path $P = \{x_1y_1z_1, \ldots, x_{4k+3}y_{4k+3}z_{4k+3}\}$ of size $4k+3$ satisfying conditions (i)–(iv) of the rule with $v = y_{2k+2}$ if such a graph H exists. For example, we can do it in a naive way as follows.[23] We can consider all 6-tuples $(u_1, v_1, w_1, u_2, v_2, w_2)$ of vertices and then consider all separations (A, B) G' with $A \cap B = \{u_1, v_1, w_1, u_2, v_2, w_2\}$ where $G[A]$ is connected and conditions (i)–(iv) are fulfilled. It is straightforward to verify (i) and (ii), condition(iii) is verified by making use of Proposition 1 in linear time, and to verify (iv), we can additionally guess x_{2i+2} and z_{2k+2} and use the standard maximum flow algorithm of Ford and Fulkerson [24] to find other paths in P. If the algorithm finds H, we apply Reduction Rule 6.9.

Suppose that the rule was applied and let H' be the graph obtained from H. Then it can happen that some previous rules would be applicable for the obtained instance. However, applying these rules is restricted by H' and can be performed locally.

First, note that z_{2k+1} or z_{2k+3} may become pendent and, potentially, Reduction Rule 6.2 could be applied for them and their neighbors y_{2k+1} and y_{2k+3}. Since the neighbors of y_{2k+1} and y_{2k+3} are in H' and Reduction Rule 6.2 could not be applied for them before, we can apply the rule in $\mathcal{O}(k)$ time. Similarly, for $i \in \{2k+1, 2k+3\}$, the vertex y_i may become a vertex of degree two adjacent only to x_i and z_i. Then Reduction Rule 6.3 may be applicable and we can apply it in $\mathcal{O}(k)$ time. Further, we can obtain a new nice component for $u = x_{2k+1}$ or $u = x_{2k+3}$. Observe that all nice components of $G - u$ have their vertices in H. More precisely, for $u = x_{2k+1}$, it holds that for each $y \in V_2$ in a nice component, $\sigma_2(y_{2k}) < \sigma_2(x) < \sigma_2(x_{2k+2})$ and for each $z \in V_3$ in a nice component, $\sigma_2(z_{2k}) < \sigma_3(z) < \sigma_3(3_{2k+2})$. For $u = x_{2k+3}$, the structure is symmetric. Thus, Reduction Rule 6.4 can be applied in polynomial in k time in H' (in fact, in $\mathcal{O}(k^2)$ time). Notice that because Reduction Rule 6.9 does not create paths of length two with their end-vertices in V_1 and V_2, the rule does not create new possibilities to apply Reduction Rule 6.7 and Reduction Rule 6.8. Therefore, the total clean-up time is polynomial in k.

Notice that for each $v \in V_2$, we execute the above procedure only once. If we modify the graph and, in particular, make v nonadjacent to any vertex of V_3 then we make it impossible to apply the rule for v. If we decide that v cannot serve as a center then because Reduction Rule 6.9 does not create new paths of length two with their end-vertices in V_1 and V_2, v cannot be a center of some H obtained after applying the rule for some other centers. This means that the total running time is $k^{\mathcal{O}(1)} \cdot n$ and concludes the proof. □

The algorithms from Lemma 34, Lemma 35, and Lemma 38 either exhaustively apply the corresponding rules or correctly report that the input instance is a no-instance of 3-Layer Crossing Minimization. In the latter case, we immediately stop and return a trivial no-instance. Otherwise, summarizing the time of initial preprocessing, our observations about Reduction Rule 6.1–Reduction Rule 6.3, Lemma 33–Lemma 35, and Lemma 38, we conclude that the total running

[23] This can be done slightly faster using Observation 16 to guess u_1, w_1, u_2, w_2 for given v_1 and v_2 and then using the fact that the pathwidth of H is at most 3 if H admits a 3-layer drawing.

time of our kernelization algorithm is $k^{\mathcal{O}(1)} \cdot n$. This completes the proof of Theorem 4.

7 Lower bounds

In this section, we show computational lower bounds stated in Theorem 5 and Theorem 3. Both bounds are proved via constructing a polynomial parameter transformation of Disjoint Factors introduced by Bodlaender, Thomassé, and Yeo [4].

Given a string s over an alphabet Σ, a *factor* is a nontrivial substring of s whose first and last symbols are the same. Then Disjoint Factors asks, given a string s over an alphabet Σ of size k, whether there are k disjoint factors with distinct first (and last) symbols. It was proved in [4] that Disjoint Factors has no polynomial kernel when parameterized by k unless NP $\subseteq$ coNP /poly. Besides this, the authors proved that Disjoint Factors is NP-complete. The proof was done by a reduction from the 3-SAT problem. We stress that, given an instance of 3-SAT with n variables and m clauses, the reduction constructs an instance of Disjoint Factors with the alphabet of size $\mathcal{O}(n+m)$. Thus, the NP-completeness proof implies a computational lower bound up to ETH which we need. We summarize these results in the following proposition.

Proposition 4 ([4])**.** Disjoint Factors *does not admit a polynomial kernel when parameterized by k unless* NP $\subseteq$ coNP /poly*. Furthermore, the problem cannot be solved in $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ time on strings of length n unless ETH fails.*

Our lower bounds for h-Layer Crossing Minimization are proved by almost the same reductions. However, note that to prove Theorem 5, it is sufficient to obtain any polynomial parameter transformation. For the ETH lower bound, we have to be more careful and avoid blowing up the parameter, and to do it, we use an extra layer in the drawing. That is the reason why the kernalization lower bound is established for 4-Layer Crossing Minimization and the ETH bound is obtained for 5-Layer Crossing Minimization.

We need the following observation.

Observation 18. Given an alphabet Σ of size k, the symbols of Σ can be encoded by distinct binary strings of length $2\lceil \log k \rceil + 2$ such that each binary string s encoding a symbol contains an equal number of zeros and ones and the reversal of s is distinct from any other string. Furthermore, such an encoding can be constructed in polynomial time.

Proof. To see this, note that $\binom{2\lceil \log k \rceil}{\lceil \log k \rceil} \geq 2^{\lceil \log k \rceil}$, that is, k symbols can be encoded by distinct strings of length $2\lceil \log k \rceil$ with exactly $\lceil \log k \rceil$ zeros and the same number of ones. Then to prevent the reversal of a string to be equal to another string, we can add zero in the beginning and append one in the end. □

Suppose that (s, k) is an instance of Disjoint Factors where s is a string over $\Sigma = \{s_1, \ldots, s_k\}$ of size $k \geq 2$. Using Observation 18, we assume that the symbols of Σ are encoded by binary strings of length $2\ell = 2\lceil \log k \rceil + 2$ such that the strings and their reversals are distinct and each symbol is encoded by a string containing equal number ℓ of zeros and ones. In the following subsection, we will describe the basic gadgets used in our reductions.

7.1 Basic gadgets

First, we construct gadgets that are used to encode zeros and ones in the encoding of the symbols. The vertices of these gadgets are placed in the sets V_1, V_2, V_3 of the 4 or 5-partite graphs constructed by the reductions.

For zero, we construct the graph Z and the "complementing" graph $\widehat{Z}$ as follows (see Figure 28).

- Construction of Z:

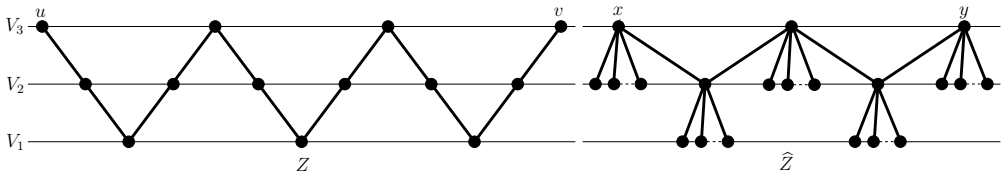

Figure 28: Construction of Z and $\widehat{Z}$.

- construct two *root* vertices u and v and join them by a (u, v)-path $w_0 \cdots w_{12}$,
- put w_2, w_6, w_{10} in V_1, $w_1, w_3, w_5, w_7, w_9, w_{11}$ in V_2, and w_0, w_4, w_8, w_{12} in V_3.

- Construction of $\widehat{Z}$:
  - construct two *root* vertices x and y and join them by a (x, y)-path $w_0w_1w_2w_3w_4$,
  - put w_1, w_3 in V_2 and w_0, w_2, w_4 in V_3,
  - for $z \in \{w_1, w_3\}$, construct a set of $p = 40k^3$ vertices, make them adjacent to z, and put then in V_1,
  - for $z \in \{w_0, w_2, w_4\}$, construct a set of p vertices, make them adjacent to z, and put then in V_2.

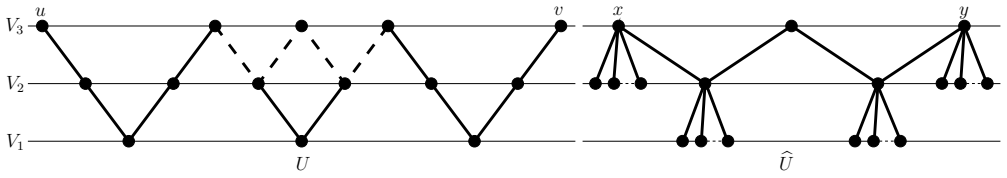

Figure 29: Construction of U and $\widehat{U}$; the V_3-no-gap path is shown by dashed lines.

For one, the graph U and the "complementing" graph $\widehat{U}$ are constructed as follows (see Figure 29).

- Construction of U:
  - construct two *root* vertices u and v and join them by a (u, v)-path $w_0 \cdots w_{12}$,
  - put w_2, w_6, w_{10} in V_1, $w_1, w_3, w_5, w_7, w_9, w_{11}$ in V_2, and w_0, w_4, w_8, w_{12} in V_3,
  - construct a vertex w, make it adjacent to w_5 and w_7, and put it in V_3.

- Construction of $\widehat{U}$:
  - construct two *root* vertices x and y and join them by a (x, y)-path $w_0w_1w_2w_3w_4$,
  - put w_1, w_3 in V_2 and w_0, w_2, w_4 in V_3,
  - for $z \in \{w_1, w_3\}$, construct a set of p vertices, make them adjacent to z, and put then in V_1,
  - for $z \in \{w_0, w_4\}$, construct a set of p vertices, make them adjacent to z, and put then in V_2.

The choice of p will be explained later; now we just observe that p is bigger than the value of the parameter in the constructed instances of 4-Layer Crossing Minimization and 5-Layer Crossing Minimization. The unique (up to the reversals of the orders and permutations of pendent vertices) 3-layer drawings of Z, $\widehat{Z}$, U, and $\widehat{U}$ without crossings are shown in Figure 28 and

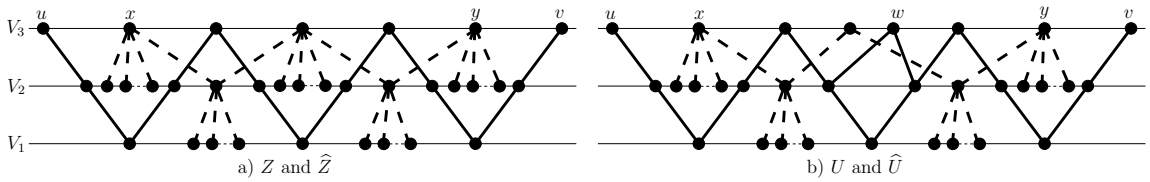

Figure 30: 3-layer drawing of (a) Z and $\widehat{Z}$ and (b) U and $\widehat{U}$; the edges of $\widehat{Z}$ and $\widehat{U}$ are shown by dashed lines.

Figure 29; we call these drawings *canonical*. To give an intuition of how these gadgets are going to be used in the reductions, we show in Figure 30 types of 3-layer drawings that will occur.

Our next aim is the construction of gadgets for encoding the symbols $s_1, \ldots, s_k$ of Σ. They are constructed by making use of the above gadgets for zeros and ones and two auxiliary gadgets F and C that are constructed as follows.

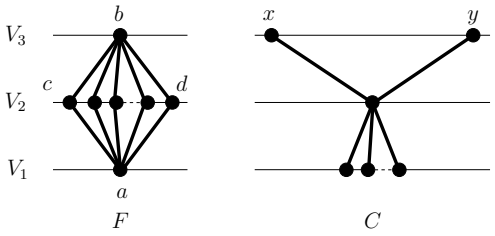

Figure 31: Construction of F and C.

- Construction of the *filling* gadget F:
  - construct two vertices a and b, and put a and b in V_1 and V_3, respectively.
  - construct a set X of p vertices, select two vertices $c, d \in X$, make the vertices of X adjacent to a and b, and include X in V_2,
- Construction of the *connection* gadget C:
  - construct two *root* vertices x and y and put them in V_3,
  - construct a vertex z, make it adjacent to x and y, and put z in V_2,
  - construct a set of p vertices, make them adjacent to z, and put them in V_1.

To encode s_i for $i \in \{1, \ldots, k\}$, we assume that $s_i = \delta_1 \ldots \delta_{2\ell}$ where $\delta_j \in \{0, 1\}$ for each $j \in \{1, \ldots, 2\ell\}$, that is, we identify the symbols and their binary encodings. Then we construct two graphs S_i and $\widehat{S}_i$ by making using the gadgets constructed above. Informally, S_i is constructed by concatenating copies of Z and U constructed for each $j \in \{1, \ldots, 2\ell\}$ and adding two "boundary" copies of F. Similarly, $\widehat{S}_i$ is constructed by connecting copies of $\widehat{Z}$ and $\widehat{U}$ but these copies are connected via copies of C. The construction of S_i and $\widehat{S}_i$ for $s_i = 0101$ is shown in Figure 32.

- Construction of S_i:
  - construct a copy of F,
  - if $\delta_1 = 0$ then construct a copy of Z and construct a copy of U if $\delta_1 = 1$, then identify the root u of the gadget with the vertex b of F and the vertex adjacent to u with d,
  - for each $j \in \{2, \ldots, 2\ell\}$, construct a copy of Z if $\delta = 0$ and construct a copy of U if $\delta_1 = 1$, then identify the root u of the gadget with the root v of the gadget constructed for δ_{j-1},

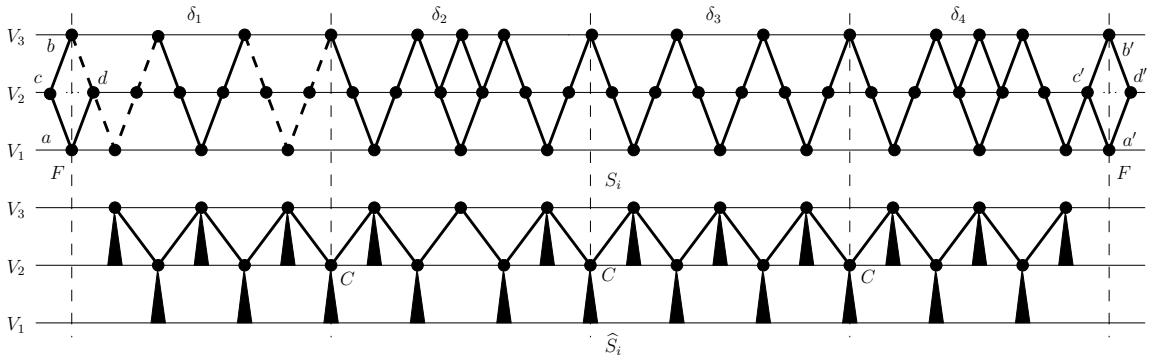

Figure 32: Construction of S_i and $\widehat{S}_i$ for $s_i = 0101$; black triangles are used to show sets of pendent vertices of size p in $\widehat{S}_i$, and examples of V_3-gap paths in S_i are shown by thick dashed lines.

  - construct a copy of F, then identify the vertex b with the root v of the gadget constructed for $\delta_{2\ell}$ and identify the vertex c with the vertex adjacent to v.

- Construction of $\widehat{S}_i$:
  - if $\delta_1 = 0$ then construct a copy of $\widehat{Z}$ and construct a copy of $\widehat{U}$ if $\delta_1 = 1$,
  - for each $j \in \{2, \ldots, 2\ell\}$, construct a copy of C and then construct a copy of Z if $\delta = 0$ and construct a copy of U if $\delta_1 = 1$, identify the root x of the copy of C with the root y of the gadget constructed for δ_{j-1} and identify the root y of C with the root of the new copy of either $\widehat{Z}$ or $\widehat{U}$.

The unique (up to the reversals of the orders and permutations of twin vertices) 3-layer drawings of each S_i and $\widehat{S}_i$ without crossings are shown in Figure 32; these drawings are obtained from the canonical drawing of the gadgets Z, $\widehat{Z}$, U, and $\widehat{U}$, and we call such drawings of S_i and $\widehat{S}_i$ *canonical*. We use a', b', c', and d' to denote the vertices a, b, c, and d, respectively, of the second copy of F in S_i.

For $i, j \in \{1, \ldots, 2\ell\}$, we say that a 3-layer drawing $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ of the disjoint union of S_i and $\widehat{S}_j$ is *restrictive* if for every vertex $w \in V_3 \cap V(\widehat{S}_j)$, $\sigma_3(b) < \sigma_3(w) < \sigma_3(b')$ where b and b', respectively, are vertices of the copies of F in $V_3 \cap V(S_i)$. The properties of the restrictive drawings are summarized in the following lemma.

Lemma 39. *Let $i, j \in \{1, \ldots, 2\ell\}$. Then the following holds for the disjoint union H of S_i and $\widehat{S}_j$:*

(i) Every restrictive 3-layer drawing of H has at least $14\ell - 2$ crossings.

(ii) A restrictive 3-layer drawing of H with exactly $14\ell - 2$ crossings exists if and only if $i = j$.

(iii) If $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ is a restrictive 3-layer drawing of H with $14\ell - 2$ crossings then the induced drawings of S_i and $\widehat{S}_j$ are canonical, and, furthermore, if $\sigma_3(b) < \sigma_3(b')$ then at least one vertex of $V_3 \cap V(\widehat{S}_j)$ is drawn immediately after b and at least one vertex is drawn immediately before b', and if $\sigma_3(b') < \sigma_3(b)$ then at least one vertex of $V_3 \cap V(\widehat{S}_j)$ is drawn immediately after b' and at least one vertex is drawn immediately before b.

Proof. Let $i, j \in \{1, \ldots, 2\ell\}$ and consider $H = S_i \uplus \widehat{S}_j$.

First, we observe that if $i = j$ then a restrictive 3-layer drawing with exactly $14\ell - 2$ crossings exists. We consider the canonical 3-layer drawing $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ of S_i and then insert the vertices of $\widehat{S}_i$ between the vertices of S_i as it is shown in Figure 33 following the idea demonstrated in Figure 30. By the direct computations, we have that the number of crossings for this drawing is exactly $14\ell - 2$ because s_i has length 2ℓ and contains exactly ℓ ones.

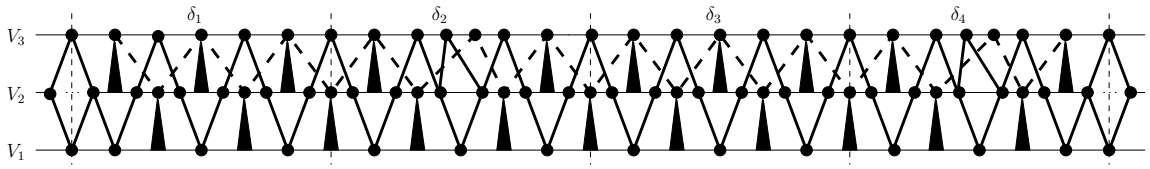

Figure 33: The restrictive 3-layer drawing of $H = S_i \uplus \widehat{S}_i$ constructed for $s_i = 0101$; black triangles are used to show sets of pendent vertices of size p in $\widehat{S}_i$, the edges of $\widehat{S}_i$ are shown by thick dashed lines.

Now we prove that for any restricted 3-layer drawing of H, the number of crossings is at least $14\ell - 2$ and the number of crossings is at least $14\ell - 1$ if $i \neq j$.

Consider a restrictive 3-layer drawing $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ of H whose number of crossing is minimum. If the number of crossings is at least $p = 40k^3 > 14\ell - 2$ then the claim holds. Assume that the number of crossings is at most $p - 1$.

By the definition of restrictive drawings, for every vertex $w \in V_3 \cap V(\widehat{S}_j)$, $\sigma_3(b) < \sigma_3(w) < \sigma_3(b')$. Because F has p (a, b)-paths of length two, we additionally obtain that for every vertex $w \in V_1 \cap V(\widehat{S}_j)$, $\sigma_1(a) < \sigma_1(w) < \sigma_1(a')$. Otherwise, because $\widehat{S}_j$ is connected, each of the paths in one of the copies of F in S_i would have a crossing edge.

For $t \in \{2, 3\}$, we say that the vertices of $\widehat{S}_j$ in V_t incident to sets of p pendent vertices are *V_t-big* and we say that a vertex of degree two in $V_3 \cap V(\widehat{S}_j)$ is *V_3-small*. We need the following observation.

Claim 7.1. *Let u be a pendent vertex adjacent to a V_t-big vertex v for some $t \in \{2, 3\}$. Then uv does not cross any edge in σ.*

Proof of Claim 7.1. Because v is a V_t-big vertex, v is a adjacent to p pendent vertices U. Since the number of crossings is at most $p - 1$, there is $w \in U$ such that wv does not cross any edge. If uv is crossing some other edge, we can modify the drawing by putting u immediately after w. This would decrease the number of crossings contradicting the choice of σ. This proves the claim. □

Given an induced path $P = w_1w_2w_3w_4w_5$ on five vertices in S_i, we say that P is a *V_3-gap path* if (i) $w_1, w_5 \in V_3$, (ii) $w_3 \in V_1$, and (iii) $d_{S_i}(w_2) = 2$ or $d_{S_i}(w_4) = 2$ (see Figure 32). Let $\sigma' = (\sigma_1, \sigma'_2, \sigma'_3)$ be the 3-layer drawing of S_i induced by σ. For a V_3-gap path $P = w_1w_2w_3w_4w_5$, it is said that P is drawn *properly* if (i) either $\sigma'_3(w_1) < \sigma'_3(w_5)$ and $\sigma'_2(w_2) < \sigma'_2(w_4)$ or, symmetrically, $\sigma'_3(w_5) < \sigma'_3(w_1)$ and $\sigma'_2(w_4) < \sigma'_2(w_2)$, and (ii) one may insert two adjacent additional vertices $u \in V_2$ and $v \in V_3$ in such a way that either u is between w_2 and w_4 in σ'_2 and v is between w_1 and w'_5 in σ'_3 and uv does not cross any edge in the drawing of S_i with respect to σ'.

Claim 7.1 and the construction of S_i imply that for every V_3-big vertex v there is a properly drawn V_3-gap path $P = w_1w_2w_3w_4w_5$ such that either $\sigma_3(w_1) < \sigma_3(v) < \sigma_3(v_5)$ or, symmetrically, $\sigma_3(w_5) < \sigma_3(v) < \sigma_3(v_1)$. We say that v is *drawn in the P-gap*. It is also easy to make the following observation.

Claim 7.2. *If v is a V_3-big vertex of $\widehat{S}_j$ then each edge vx where x is not pendent crosses at least one edge of S_i.*

Proof of Claim 7.2. We have that v is drawn in the P-gap for some properly drawn V_3-gap path $P = w_1w_2w_3w_4w_5$. We assume without loss of generality that $\sigma_3(w_1) < \sigma_3(v) < \sigma_3(w_5)$ and $\sigma_2(w_2) < \sigma_2(w_4)$. Notice that x is a V_2-big vertex. Then by Claim 7.1, either $\sigma_2(x) < \sigma_2(w_2)$ or $\sigma_2(w_4) < \sigma_2(x)$. In both cases, vx crosses at least one edge of S_i. This proves the claim. □

Now we show that V_3-big vertices are arranged in σ in a specific way.

Claim 7.3. *There are no two distinct V_3-big vertices u and v that are drawn in the same P-gap.*

Proof of Claim 7.3. Assume that V_3-big vertices u and v are drawn in the same P-gap for a V_3-gap path $P = w_1w_2w_3w_4w_5$ with $\sigma_3(w_1) < \sigma_3(u) < \sigma_3(v) < \sigma_3(w_5)$. Notice that $\widehat{S}_j$ has a (u,v)-path Q that contains a V_2-big vertex w. By Claim 7.1, we have that either $\sigma_2(w) < \sigma_2(w_2)$ or $\sigma_2(w_4) < \sigma_2(w)$. In the first case, the edges of Q would cross each edge ux for a pendent vertex x adjacent to u, and Q would cross every edge vx for a pendent vertex x adjacent to v. In both cases, we obtain a contradiction with Claim 7.1. This proves the claim. □

Notice that $\widehat{S}_j$ has exactly 5ℓ V_3-big vertices and S_i has exactly the same number of V_3-gap paths. Then Claim 7.3 implies every V_3-gap path is drawn properly and for each V_3-gap path P, there is a V_3-big vertex drawn in the P-gap.

We say that three distinct vertices $u_1, u_2, u_3 \in V_3 \cap V(\widehat{S}_j)$ are *consecutive* if the unique (u_1, u_3)-path in $\widehat{S}_j$ contains u_2.

Claim 7.4. *Suppose that $u_1, u_2, u_3 \in V_3 \cap V(\widehat{S}_j)$ are consecutive vertices and u_1 and u_3 are V_3-big. Then either $\sigma_3(u_1) < \sigma_3(u_2) < \sigma_3(u_3)$ or, symmetrically, $\sigma_3(u_3) < \sigma_3(u_2) < \sigma_3(u_1)$.*

Proof of Claim 7.4. Assume without loss of generality that $\sigma_3(u_1) < \sigma_3(u_3)$. We show that $\sigma_3(u_1) < \sigma_3(u_2) < \sigma_3(u_3)$. For the sake of contradiction, assume that $\sigma_3(u_2) < \sigma_3(u_1)$. Then consider the unique (u_2, u_3)-path Q in $\widehat{S}_j$. Because u_1, u_2, u_3 are consecutive, Q avoids u_1. Then we have that edges Q cross every u_1x for a pendent vertex x incident to u_1 contradicting Claim 7.1. The case $\sigma_3(u_3) < \sigma_3(u_2)$ is symmetric. This concludes the proof. □

By Claim 7.4, we immediately obtain that the 3-layer drawing of $\widehat{S}_j$ induced by σ is canonical.

Now we consider V_3-small vertices, that is, vertices of $\widehat{S}_j$ of degree two. Following the previous notation, we say that v is *drawn in the P-gap* for a V_3-gap $P = w_1w_2w_3w_4w_5$ if either $\sigma_3(w_1) < \sigma_3(v) < \sigma_3(w_5)$ or $\sigma_3(w_5) < \sigma_3(v) < \sigma_3(w_1)$. In exactly the same way as in Claim 7.2, we have the following claim.

Claim 7.5. *If v is a V_3-small vertex of $\widehat{S}_j$ drawn in the P-gap for a V_3-gap path P then each edge vx where x is not pendent crosses at least one edge of S_i.*

We say that $P = w_1w_2w_3w_4w_5$ is a *V_3-no-gap path* if P is a induced path in one of the gadgets U such that (i) $w_1, w_3, w_5 \in V_3$ and (ii) $w_2, w_4 \in V_2$ (see Figure 29). A V_3-small vertex v is in the P-gap for a V_3-no-gap path P if there are $u_1, u_2 \in \{w_1, w_3, w_5\}$ such that $\sigma_3(u_1) < \sigma_3(v) < \sigma_3(u_2)$. We show the following bound on the number of crossings. Notice that by the construction, for a V_3-small vertex v, there are two distinct V_3-big vertices $x, y \in V_3 \cap V(\widehat{S}_j)$ at distance two from v.

Claim 7.6. *Let v be a V_3-small vertex of $\widehat{S}_j$ drawn in the P-gap for a V_3-no gap-path P and let $x, y \in V_3 \cap V(\widehat{S}_j)$ be two distinct V_3-big vertices at distance two from v. Then the edges of the (x,y)-path in $\widehat{S}_j$ cross at least 6 edges of S_i.*

Proof of Claim 7.6. By Claim 7.4, we can assume without loss of generality that $\sigma_3(x) < \sigma_3(v) < \sigma_3(y)$. Let $Q = w_1w_2w_3w_4w_5$ and $Q' = w'_1w'_2w'_3w'_4w'_4$ be properly drawn V_3-gap paths such that x and y are drawn in the Q and Q'-gaps, respectively. We assume without loss of generality that $\sigma_3(w_1) < \sigma_3(x) < \sigma_3(w_5)$ and $\sigma_3(w'_1) < \sigma_3(y) < \sigma_3(w'_5)$. By Claim 7.3, Q and Q' are distinct V_3-gap paths. Because of Claim 7.4 and the observation that every V_3-gap path is drawn properly and for each V_3-gap path P, there is a V_3-big vertex drawn in the P-gap, we obtain that $P = u_1u_2u_3u_4u_5$ for $u_1 = w_5$ and $u_5 = w'_1$. Let $R = w_4u_1u_2u_3u_4u_5w'_2$. This path has 6 edges and each edge of R crosses at least one edge of the (x,y)-path in $\widehat{S}_j$. This proves the claim. □

Now we obtain the lower bound for the number of crossings if every V_3-small vertex is drawn in the P-gap for a V_3-no-gap path.

Claim 7.7. *If every V_3-small vertex is drawn in the P-gap for a V_3-no-gap path P then the total number of crossings is at least $14\ell - 2$. Furthermore, if the number of crossings is exactly $14\ell - 2$ then the induced drawings of S_i and $\widehat{S}_j$ are canonical.*

Proof of Claim 7.7. Recall that $\widehat{S}_j$ has exactly 5ℓ V_3-big vertices and S_i has exactly the same number of V_3-gap paths. Also, $\widehat{S}_j$ has exactly ℓ V_3-small vertices and S_i has ℓ V_3-no-gap paths. Notice that v_3-small vertices are at a distance at least 6 from each other. Consider the edges of (x_v, y_v)-paths R_v in $\widehat{S}_j$ for V_3-small vertices v such that $x_v, y_v \in V_3$ are distinct vertices at distance two from v. By Claim 7.6, these edges cross at least 6ℓ edges of S_i. Furthermore, $\widehat{S}_j$ has $8\ell - 2$ edges xy not included in any path R_v such that $x \in V_3$ and $y \in V_2$ is not pendent. By Claim 7.2, these edges cross at least $8\ell - 2$ edges of S_i. Summarizing, we obtain that the edges of $\widehat{S}_j$ cross at least $14\ell - 2$ edges of S_i. We already noticed that the 3-layer drawing of $\widehat{S}_j$ induced by σ is canonical by Claim 7.4. For S_i, we observe that If the number of crossings is exactly $14\ell - 2$ then there are no crossings between the edges of S_i. Thus, the induced drawing of S_i is canonical. □

Next, we get the lower bound if there is a V_3-small vertex that is not drawn in the P-gap for a V_3-no-gap path.

Claim 7.8. *If there is V_3-small vertex v that is not drawn in the P-gap for a V_3-no-gap path P then the total number of crossings is at least 14ℓ.*

Proof of Claim 7.8. Denote by $t \geq 1$ the number of V_3-small vertices that are not drawn in the P-gap for a V_3-no-gap paths P. Consider a V_3-no-gap path $P = u_1u_2u_3u_4u_5$ such that no V_3-small vertex is drawn in the P-gap. Let $Q = w_1w_2w_3w_4w_5$ and $Q' = w'_1w'_2w'_3w'_4w'_5$ be V_3-gap paths such that $u_1 = w_5$ and $u_5 = w'_1$. Recall that Claim 7.3, there are two V_3-big vertices u and u' that are drawn in the Q and Q'-gaps, respectively. Also, we can assume that $\sigma_3(w_1) < \sigma_3(w_5), \sigma_3(u_3), \sigma_3(w'_1)$ and $\sigma_3(w_5), \sigma_3(u_3), \sigma_3(w'_1) < \sigma_3(w'_5)$. We consider two cases.

First, assume that u and u' are at distance two in $\widehat{S}_j$. Then there is a V_2-big vertex v that is adjacent to u and u'. By Claim 7.1, we have that either $\sigma_2(w_4) < \sigma_2(v) < \sigma_2(u_2), \sigma_2(u_4), \sigma_2(w'_2)$ or, symmetrically, $\sigma_2(w_4), \sigma_2(u_2), \sigma_2(u_4) < \sigma_2(v) < \sigma_2(w'_4)$. These cases are symmetric and we assume that $\sigma_2(w_4) < \sigma_2(v) < \sigma_2(u_2), \sigma_2(u_4), \sigma_2(w'_2)$. Then the edge vu' crosses every edge of the path $u_1u_2u_3u_4u_5w'_2$. Thus, vu' crosses at least 5 edges. Notice also that u' and v are at a distance of at least two from any small V_2-vertex of $\widehat{S}_j$.

Next, we suppose that u and u' are at a distance of at least four. Then by Claim 7.3 and Claim 7.4, u and u' are at distance four and there is a V_3-small vertex x at distance two from both u and u'. We have that x is not drawn in the P-gap. Using Claim 7.4, we conclude that x is either drawn in the Q-gap or Q'-gap. By symmetry, we assume that x is drawn in the Q-gap. Then $\sigma_3(u) < \sigma_3(x) < \sigma_3(u_1), \sigma_3(u_3), \sigma_3(u_5)$. By the construction of $\widehat{S}_j$, there is a V_2-big vertex v that is adjacent to x and u'. By Claim 7.1, we have that either $\sigma_2(w_4) < \sigma_2(v) < \sigma_2(u_2), \sigma_2(u_4), \sigma_2(w'_2)$ or $\sigma_2(w_4), \sigma_2(u_2), \sigma_2(u_4) < \sigma_2(v) < \sigma_2(w'_4)$. In the first case, we have that vu' crosses every edge of the path $u_1u_2u_3u_4u_5w'_2$. Notice that u' and v are at a distance at least two from any small V_2-vertex of $\widehat{S}_j$ distinct from x. If $\sigma_2(w_4), \sigma_2(u_2), \sigma_2(u_4) < \sigma_2(v) < \sigma_2(w'_4)$ then, by same arguments, we have that xv crosses every edge of $w_4u_1u_2u_3u_4u_5$ and x and v are at distance at least two from any small V_2-vertex of $\widehat{S}_j$ distinct from x.

We conclude that in both cases, there is an edge $xy \in E(\widehat{S}_j)$ with $x \in V_2$ and $y \in V_3$ such that xy crosses at least 5 edges of S_i and both x and y are at distance at least two from any small V_3-vertex that is drawn in the R-gap for a V_3-no-gap path R.

For a V_3-small vertex v that is drawn in the P-gap for some V_3-gap path P, denote by R_v the (x_v, y_v)-paths in $\widehat{S}_j$ such that $x_v, y_v \in V_3$ are distinct vertices at distance two from v. There are $\ell - t$ such paths for v that are not drawn in the Q-gap for a V_3-no-gap path Q. By Claim 7.6, these edges cross at least $6(\ell - t)$ edges of S_i. Further, there a set L of $8\ell + 4t - 2$ edges xy of $\widehat{S}_j$ that not included in any path R_v such that $x \in V_3$ and $y \in V_2$ is not pendent. By Claim 7.2 and

Claim 7.5 each of the edges of A crosses at least one edge of S_i. Also, we proved that there are t edges $xy \in A$ such that xy crosses at least 5 edges of S_i. Then the total number of crossings is at least $6(\ell - t) + 8\ell + 4t - 2 + 4t = 14\ell + 2t - 2 \geq 14\ell$. This concludes the proof. □

Combining Claim 7.7 and Claim 7.8, we obtain that the number of crossing for the optimum restrictive 3-layer drawing σ of $H = S_i \uplus \widehat{S}_j$ is at least $14\ell - 2$. This proves claim (i) of the lemma.

To show (ii), we have to argue that the number of crossings is $14\ell - 2$ if and only if $i = j$. If $i = j$ then recall that we already demonstrated a restrictive 3-layer drawing with $14\ell - 2$ crossings. Thus, it remains to prove that if the number of crossings is $14\ell - 2$ then $i = j$.

Assume that the number of crossings for σ is $14\ell - 2$. Because of Claim 7.8, every V_3-small vertex is drawn in the P-gap for some V_3-no-gap path P. By Claim 7.7, we have that the induced drawings of S_i and $\widehat{S}_j$ are canonical. Recall that S_i was constructed for the binary string encoding of the symbol s_i and $\widehat{S}_j$ was constructed for the encoding of s_j. By Claim 7.3 and Claim 7.4 and the observation that $\widehat{S}_j$ has exactly 5ℓ V_3-big vertices and S_i has exactly the same number of V_3-gap paths, we conclude that either the encodings of s_i and s_j are the same or the reversal of the encoding of s_i coincides with the encoding of s_j. However, the considered binary encoding of Σ has the properties that the strings encoding distinct symbols are distinct, and the reversal of each string is distinct from any other strings in the encoding. Therefore, $s_i = s_j$ and $i = j$.

To obtain (iii), we combine Claim 7.3, Claim 7.7, Claim 7.8, and the observation that for each V_3-gap path P, there is a V_3-big vertex drawn in the P-gap. This concludes the proof of the lemma. □

7.2 Main step

Given the gadgets S_i and $\widehat{S}_i$ constructed for the symbols $s_1, \ldots, s_k$ of Σ and their properties summarized in Lemma 39, we proceed with reducing the instance (s, k) of Disjoint Factors. In this subsection, we explain the main step. Here, we deal with 4-layer embeddings. We remind that $\ell = \lceil \log k \rceil + 1$ and $p = 40k^3$.

Let $s = r_1 \ldots r_n$ where $r_i \in \Sigma$. We construct the graphs R and $\widehat{R}$ encoding the instance (s, k) of Disjoint Factors.

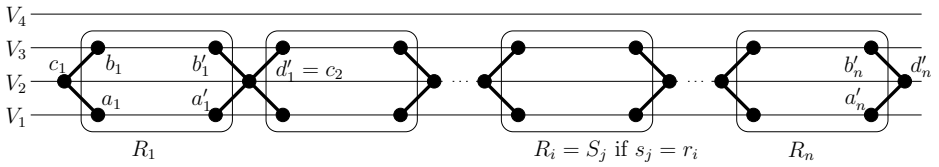

Figure 34: Construction of R.

- Construction of R (see Figure 34):
  - for every $i \in \{1, \ldots, n\}$, construct a copy R_i of S_j for $s_j = r_i$ and denote by $a_i, b_i, c_i, d_i, a'_i, b'_i, c'_i, d'_i$ the respective vertices $a, b, c, d, a', b', c', d'$ of S_j.
  - concatenate these copies by merging the vertex d'_{i-1} of R_{i-1} with the vertex c_i of R_i for $i \in \{2, \ldots, n\}$.
- Construction of $\widehat{R}$ (see):
  - for every $i \in \{1, \ldots, k\}$, construct two copies $\widehat{S}^1_i$ and $\widehat{S}^2_i$ of $\widehat{S}_i$,
  - for every $i \in \{1, \ldots, k\}$, construct a vertex v_i and make it adjacent to every vertex of $V_3 \cap (V(\widehat{S}^1_i) \cup V(\widehat{S}^2_i))$,
  - put the vertices $v_1, \ldots, v_k$ in V_4.

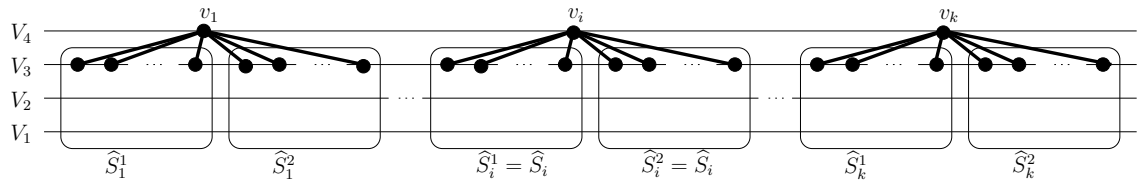

Figure 35: Construction of $\widehat{R}$.

Notice that $V_4 \cap V(R) = \emptyset$.

We say that a 4-layer drawing $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ of the disjoint union of R and $\widehat{R}$ is *restrictive* if for every vertex $w \in V_3 \cap V(\widehat{R})$, $\sigma_3(b_1) < \sigma_3(w) < \sigma_3(b'_n)$. We prove the following lemma that is crucial for the reduction.

Lemma 40. *The instance (s, k) of* Disjoint Factors *is a yes-instance of the problem if and only if there is a restrictive 4-layer drawing of the disjoint union of R and $\widehat{R}$ with at most $2k(14\ell - 2)$ crossings.*

Proof. Suppose that (s, k) is a yes-instance of Disjoint Factors. Then the string $s = r_1 \dots r_n$ has k disjoints nontrivial substrings $r_{i_1} \dots r_{j_1}, \dots, r_{i_k} \dots r_{j_k}$ such that the symbols r_{i_t} and r_{j_t} are the same for $t \in \{1, \dots, k\}$. We assume without loss of generality that $r_{i_t} = r_{j_t} = s_t$ for each $t \in \{1, \dots, k\}$ and it holds that $j_{t-1} < i_t$ for all $t \in \{2, \dots, k\}$. We construct the 4-layer drawing $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ of $H = R \uplus \widehat{R}$ as follows.

- Construct canonical 3-layer drawings of each R_i for $i \in \{1, \dots, n\}$ such that c_i is drawn before d'_i.
- Concatenate these drawings by merging the vertex d'_{i-1} of R_{i-1} with the vertex c_i of R_i for $i \in \{2, \dots, n\}$ as it is shown in Figure 34.
- For each $t \in \{1, \dots, k\}$, draw $\widehat{S}_t^1$ in such a way that the vertices of $V_1 \cap V(\widehat{S}_t^1)$ are drawn between a_{i_t} and a'_{i_t}, the vertices of $V_2 \cap V(\widehat{S}_t^1)$ are drawn between c_{i_t} and d'_{i_t}, the vertices of $V_3 \cap V(\widehat{S}_t^1)$ are drawn between b_{i_t} and b'_{i_t}, and the 3-layer drawing of $R_{i_t} \uplus \widehat{S}_t^1$ has $14\ell - 2$ crossings using Lemma 39.
- For each $t \in \{1, \dots, k\}$, draw $\widehat{S}_t^2$ in such a way that the vertices of $V_1 \cap V(\widehat{S}_t^2)$ are drawn between a_{j_t} and a'_{j_t}, the vertices of $V_2 \cap V(\widehat{S}_t^2)$ are drawn between c_{j_t} and d'_{j_t}, the vertices of $V_3 \cap V(\widehat{S}_t^2)$ are drawn between b_{j_t} and b'_{j_t}, and the 3-layer drawing of $R_{j_t} \uplus \widehat{S}_t^2$ has $14\ell - 2$ crossings using Lemma 39.
- Define σ_4 by setting $\sigma_4(v_1) < \dots < \sigma_4(v_k)$.

By the assumptions that $r_{i_t} = r_{j_t} = s_t$ for each $t \in \{1, \dots, k\}$ and $j_{t-1} < i_t$ for all $t \in \{2, \dots, k\}$, we have that the edges xy with $x \in V_3$ and $y \in V_4$ have no crossings. Then the total number of crossings is $2k(14\ell - 2)$.

For the opposite direction, assume that $H = R \uplus \widehat{R}$ admits a restrictive 4-layer drawing $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ with at most $2k(14\ell - 2)$ crossings.

Consider arbitrary $i \in \{1, \dots, n\}$. Recall that R_i, that is a copy of S_j for $s_j = r_i$, contains two disjoint copies of filling gadgets F. We denote the copy containing a_i, b_i by F_i and the copy containing a'_i, b'_i by F'_i. We have that $a_i \in V_1$, $b_i \in V_3$, and F_i contains p (a_i, b_i)-paths of length two. Because $p = 40k^3 > 2k(14(k+1) - 2) \geq \ell$, there is a (a_i, b_i)-path P_i in F_i such that the edges of P_i do not cross any edge. By the same arguments, there is a (a'_i, b'_i) path P'_i of length two such that the edges of P'_i do not cross any edge.

Because R is connected, we conclude that either $\sigma_1(a_1) < \sigma_1(a'_1) < \sigma_1(a_2) < \dots \sigma_1(a_n) < \sigma_1(a'_n)$ and $\sigma_3(b_1) < \sigma_3(b'_1) < \sigma_3(b_2) < \dots \sigma_3(b_n) < \sigma_3(b'_n)$ or $\sigma_1(a'_n) < \sigma_1(a_n) < \sigma_1(a'_{n-1}) < \dots \sigma_1(a'_1) <$

$\sigma_1(a_1)$ and $\sigma_3(b'_n) < \sigma_3(b_n) < \sigma_3(b'_{n-1}) < \ldots \sigma_3(b'_1) < \sigma_3(b_1)$. Because σ is restrictive, $\sigma_1(a_1) < \sigma_1(a'_1) < \sigma_1(a_2) < \ldots \sigma_1(a_n) < \sigma_1(a'_n)$ and $\sigma_3(b_1) < \sigma_3(b'_1) < \sigma_3(b_2) < \ldots \sigma_3(b_n) < \sigma_3(b'_n)$.

Furthermore, because all the paths P_i and P'_i have no crossing edges and the gadgets $\widehat{S}_t$ are connected, we have that for every $t \in \{1, \ldots, k\}$ and every $q \in \{1, 2\}$, it holds that

- either there is $i \in \{1, \ldots, n\}$ such that $\sigma_3(b_i) < \sigma_3(w) < \sigma_3(b'_i)$ for each $w \in V_3 \cap V(\widehat{S}^q_t)$,
- or there is $i \in \{2, \ldots, n\}$ such that $\sigma_3(b'_i) < \sigma_3(w) < \sigma_3(b_{i-1})$ for each $w \in V_3 \cap V(\widehat{S}^q_t)$.

However, in the second case, the paths $a'_{i-1}d_{i-1}a_i$ and $b'_{i-1}d_{i-1}b_i$ would cross every edge of $\widehat{S}^q_t$ contradicting that the number of crossings is at most $2k(14\ell - 2)$ because $\widehat{S}^q_t$ contains more than p edges. Thus, for every $t \in \{1, \ldots, k\}$ and every $q \in \{1, 2\}$, we have that there is $i \in \{1, \ldots, n\}$ such that $\sigma_3(b_i) < \sigma_3(w) < \sigma_3(b'_i)$ for each $w \in V_3 \cap V(\widehat{S}^q_t)$. Then by Lemma 39(i), we have that the induced drawing of $R_i \uplus \widehat{S}^q_t$ has at least $14\ell - 2$ crossings. Because the total number of crossings is at most $2k(14\ell - 2)$, we conclude that the induced drawing of $R_i \uplus \widehat{S}^q_t$ has exactly $14\ell - 2$ crossings. By Lemma 39(ii), this means that R_i is a copy of S_t. Note also that the edges $xy \in E(\widehat{R})$ with $x \in V_3$ and $y \in V_4$ do not cross each other.

Suppose that there is $i \in \{1, \ldots, n\}$ such that R_i is a copy of S_j and it holds that $\sigma_3(b_i) < \sigma_3(w) < \sigma_3(b'_i)$ for all $w \in V_3 \cap (V(\widehat{S}^1_j) \cup V(\widehat{S}^2_j)$. Then by Lemma 39(iii), we have that there are crossings of edges of $\widehat{S}^1_j$ and $\widehat{S}^2_j$. This means that the total number of crossings is at least $2k(14\ell - 2) + 1$. Therefore, for each $t \in \{1, \ldots, k\}$, there are distinct $i_t, j_t \in \{1, \ldots, n\}$ such that R_{i_t} and R_{j_t} are copies of S_t and it holds that (i) $\sigma_3(b_{i_t}) < \sigma_3(w) < \sigma_3(b'_{i_t})$ for all $w \in V_3 \cap V(\widehat{S}^1_t)$ and (ii) $\sigma_3(b_{j_t}) < \sigma_3(w) < \sigma_3(b'_{j_t})$ for all $w \in V_3 \cap V(\widehat{S}^2_t)$. By symmetry, we assume that $i_t < j_t$ for each $t \in \{1, \ldots, k\}$.

Consider the substrings $r_{i_t} \ldots r_{j_t}$ of s for $t \in \{1, \ldots, k\}$. Because R_{i_t} and R_{j_t} are copies of S_t, we have that $r_{i_t} = r_{j_t} = s_t$, that is, each $r_{i_t} \ldots r_{j_t}$ is a factor starting and ending with the symbol s_t.

Consider distinct $t, t' \in \{1, \ldots, k\}$ and assume, by symmetry, that $\sigma_4(v_t) < \sigma_4(v_{t'})$. For each $x \in V_3 \cap V(\widehat{S}^2_t)$ and each $y \in V_3 \cap V(\widehat{S}^1_t)$, the edges $v_t x$ and $v_{t'} y$ do not cross. Thus, $j_t < i_{t'}$. This means that $r_{i_t} \ldots r_{j_t}$ and $r_{i_{t'}} \ldots r_{j_{t'}}$. Then the substrings $r_{i_t} \ldots r_{j_t}$ of s for $t \in \{1, \ldots, k\}$ are disjoint factors of s starting with distinct symbols. We conclude that (s, k) is a yes-instance of Disjoint Factors. This finishes the proof. □

7.3 Proof of Theorems 5 and 3

Now we are ready to prove Theorem 5 and Theorem 3. Recall that in our base reduction in Section 7.2, we use restrictive 4-layer drawings of the disjoint union of R and $\widehat{R}$ (see Lemma 40). To prove our main results, we have to get rid of this constraint. We again remind that $\ell = \lceil \log k \rceil + 1$ and $p = 40k^3$. We start with Theorem 5 which we restate.

Theorem 5. *For any $h \geq 4$, h-Layer Crossing Minimization does not admit a polynomial kernel unless* NP $\subseteq$ coNP /poly.

Proof. Clearly, it is sufficient to prove the theorem for $h = 4$. The theorem is proved by constructing a polynomial parameter transformation of Disjoint Factors. Let (s, k) be an instance of Disjoint Factors. We construct the instance $(G, V_1, V_2, V_3, V_4, k')$ of 4-Layer Crossing Minimization. The graph G and V_1, V_2, V_3, V_4 are constructed as follows (see Figure 36).

- Construct R and $\widehat{R}$ as explained in Section 7.2.
- Construct a set L of p vertices, put them in V_4, and make them adjacent to the vertex b_1 of R.

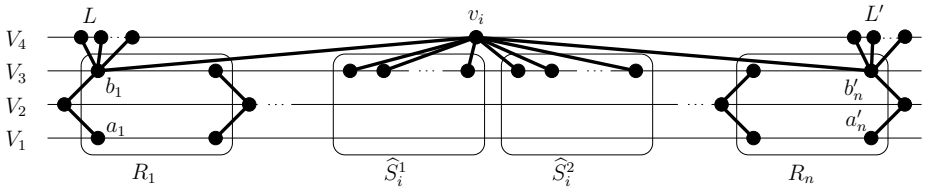

Figure 36: Construction of G.

- Construct a set L' of p vertices, put them in V_4, and make them adjacent to the vertex b'_n of R.
- For each $i \in \{1, \ldots, k\}$, make the vertex v_i adjacent to b_1 and b'_n.

We set $k' = 2k(14\ell - 2) + \frac{1}{2}k(k-1) + k(k-1)(12\ell + 1)$.

We claim that (s, k) is a yes-instance of Disjoint Factors if and only if $(G, V_1, V_2, V_3, V_4, k')$ is a yes-instance of 4-Layer Crossing Minimization.

Suppose that (s, k) is a yes-instance of Disjoint Factors. Then by Lemma 40, there is a restrictive 4-layer drawing $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ of the disjoint union of R and $\widehat{R}$ with at most $2k(14\ell - 2)$ crossings. We construct the 4-layer drawing $\sigma' = (\sigma_1, \sigma_2, \sigma_3, \sigma'_4)$ of G by the following modification of σ_4: we insert the vertices of L before the vertices $v_1, \ldots, v_k$ and we put the vertices of L' after the vertices $v_1, \ldots, v_k$. Notice that the paths $b_1 v_i b'_n$ and $b_1 v_j b'_n$ for distinct $i, j \in \{1, \ldots, k\}$ have exactly one pair of crossing edges. Therefore, the total number of crossings of such paths is $\binom{k}{2}$. Also, for all distinct $i, j \in \{1, \ldots, k\}$, the path $b_1 v_i b'_n$ crosses every edge $v_j x$ for $x \in V_3 \cap (V(\widehat{S}_j^1) \cup V(\widehat{S}_j^2))$. The edges $b_1 y$ for $y \in L$ and $b'_n z$ for $z \in L'$ do not cross any edge. Therefore, the total number of crossings in σ' is at most $k' = 2k(14\ell - 2) + \frac{1}{2}k(k-1) + k(k-1)(12\ell + 1)$.

For the opposite direction, assume that $(G, V_1, V_2, V_3, V_4, k')$ is a yes-instance of 4-Layer Crossing Minimization, that is, there is a 4-layer drawing $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ with at most k' crossings. Consider (a_1, x)-paths of length three in G for $x \in L$. By the construction, G has p edge-disjoint paths of this type. We have that $p = 40k^3 > 2k(14(k+1) - 2) + \frac{1}{2}k(k-1) + k(k-1)(12(k+1)+1) \geq k'$ if $k \geq 2$.[24] Therefore, there is $x \in L$ such that the edges of the (a_1, x)-path P of length three in G do not cross any edge in σ. By the same arguments, there is $x' \in L'$ such that the edges of the (a'_n, x')-path P' of length three in G have no crossings. By symmetry, we can assume without loss of generality that $\sigma_1(a_1) < \sigma_1(a'_n)$ and $\sigma_3(b_1) < \sigma_3(b'_n)$. Then because G is connected, we have that for every vertex $w \in V_3 \cap V(\widehat{R})$, $\sigma_3(b_1) < \sigma_3(w) < \sigma_3(b_n)'$. Thus, σ induces a restrictive 4-layer drawing of the disjoint union of R and $\widehat{R}$. Observe that for any choice of σ_4, the paths $b_1 v_i b'_n$ and $b_1 v_j b'_n$ for distinct $i, j \in \{1, \ldots, k\}$ have exactly one pair of crossing edges and the total number of crossings with the edges of these paths is $\binom{k}{2}$. Because σ is restrictive, we also have that for all distinct $i, j \in \{1, \ldots, k\}$, the path $b_1 v_i b'_n$ crosses every edge $v_j x$ for $x \in V_3 \cap (V(\widehat{S}_j^1) \cup V(\widehat{S}_j^2))$. Therefore, the total number of crossings of this type is at least $k(k-1)(12\ell+1)$. It follows that the disjoint union of R and $\widehat{R}$ has at most $k' - \left(\frac{1}{2}k(k-1) + k(k-1)(12\ell+1)\right) \leq 2k(14\ell - 2)$ crossings. Since the drawing induced by σ is restrictive, Lemma 40 implies that (s, k) is a yes-instance of Disjoint Factors.

It is straightforward to verify that given (s, k), the construction of G can be done in polynomial time. We also have that $k' = \mathcal{O}(k^2 \log k)$. Therefore, the reduction is a polynomial parameter transformation. Then using Proposition 4, we obtain that 4-Layer Crossing Minimization does not admit a polynomial kernel unless NP $\subseteq$ coNP /poly. This concludes the proof. □

In the proof of Theorem 5, we constructed a polynomial parameter transformation of Disjoint Factors where the parameter in the obtained instance of 4-Layer Crossing Minimization is $k' = \mathcal{O}(k^2 \log k)$. This is sufficient for a kernelization lower bound but k' is too big to give a

[24] The value of $p = 40k^3$ was chosen to be big enough to satisfy this inequality.

reasonable lower bound for the running time up to ETH. Hence, we use a slightly different reduction in the proof of Theorem 3 which we restate.

Theorem 3. *For any $h \geq 5$, h-Layer Crossing Minimization cannot be solved by an algorithm running in $2^{o(k/\log k)} \cdot n^{\mathcal{O}(1)}$ time unless ETH fails.*

Proof. Is sufficient to prove the theorem for $h = 5$. We again reduce Disjoint Factors and for an instance (s, k) of Disjoint Factors, we construct the instance $(G, V_1, V_2, V_3, V_4, V_5, k')$ of 5-Layer Crossing Minimization. The graph G and V_1, V_2, V_3, V_4, V_5 are constructed as follows (see Figure 37).

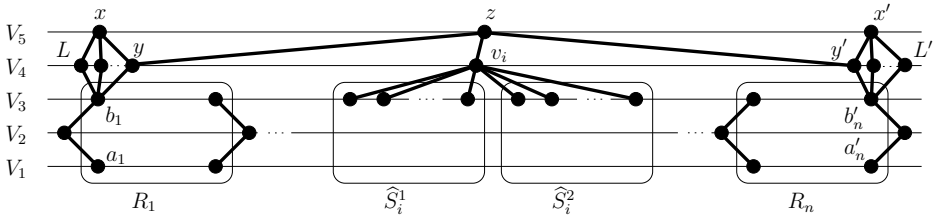

Figure 37: Construction of G.

- Construct R and $\widehat{R}$ as explained in Section 7.2.
- Construct two vertices x and x' and put them in V_5.
- Construct a set L of p vertices, put them in V_4, and make them adjacent to x and the vertex b_1 of R.
- Construct a set L' of p vertices, put them in V_4, and make them adjacent to x' and the vertex b'_n of R.
- Select vertices $y \in L$ and $y' \in L'$.
- Construct a vertex z, put it in V_5, and make it adjacent to y, y', and $v_i \in V(\widehat{R})$ for all $i \in \{1, \ldots, k\}$.

We set $k' = 2k(14\ell - 2)$.

We prove that (s, k) is a yes-instance of Disjoint Factors if and only if $(G, V_1, V_2, V_3, V_4, V_5, k')$ is a yes-instance of 5-Layer Crossing Minimization.

Let (s, k) be a yes-instance of Disjoint Factors. Then by Lemma 40, there is a restrictive 4-layer drawing $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ of the disjoint union of R and $\widehat{R}$ with at most $2k(14\ell - 2)$ crossings. We construct the 5-layer drawing $\sigma' = (\sigma_1, \sigma_2, \sigma_3, \sigma'_4, \sigma_5)$ of G:

- set $\sigma_5(x) < \sigma_5(z) < \sigma_5(x')$,
- construct σ'_4 by modifying σ_4:
  - insert the vertices of L before $v_1, \ldots, v_k$ in such a way that y is the last vertex of L in the order,
  - insert the vertices of L after $v_1, \ldots, v_k$ in such a way that y' is the first vertex of L in the order.

Notice that in the obtained 5-layer drawing all crossings are in the disjoint union of R and $\widehat{R}$. Thus, the total number of crossings is at most $2k(14\ell - 2)$ and $(G, V_1, V_2, V_3, V_4, V_5, k')$ is a yes-instance of 5-Layer Crossing Minimization.

For the opposite direction, assume that $(G, V_1, V_2, V_3, V_4, k')$ is a yes-instance of 5-Layer Crossing Minimization, that is, there is a 5-layer drawing $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$ with at most k' crossings. Consider (a_1, x)-paths of length four in G and note that G has p edge disjoint paths of this type. Since $p = 40k^4 > 2k(14(k+1)-2) \geq 2k(14\ell - 2)$, there is an (a_1, x)-path P of length four in G such that its edges do not cross any edge. Symmetrically, there is an (a'_n, x')-path of length four in G with edges without crossings. Then we can assume without loss of generality that $\sigma_1(a_1) < \sigma_1(a'_n)$ and $\sigma_3(b_1) < \sigma_3(b'_n)$. Since G is connected, for every vertex $w \in V_3 \cap V(\widehat{R})$, $\sigma_3(b_1) < \sigma_3(w) < \sigma_3(b'_n)$. Therefore, σ induces a restrictive 4-layer drawing of the disjoint union of R and $\widehat{R}$. Because the total number of crossings is at most $2k(14\ell - 2)$, Lemma 40 implies that (s, k) is a yes-instance of Disjoint Factors.

Suppose that there is an algorithm $\mathcal{A}$ solving 5-Layer Crossing Minimization in $2^{o(k/\log k)} \cdot n^{\mathcal{O}(1)}$ time on n-vertex graphs. Then we can solve Disjoint Factors using our reduction. Given an instance (s, k) of Disjoint Factors, we use the reduction to construct an equivalent instance of 5-Layer Crossing Minimization. This can be done in polynomial time and the value of the parameter is $\mathcal{O}(k \log k)$. Then we apply $\mathcal{A}$ to the obtained instance. The total running time is $2^{o(k \log k/\log(k \log k))} \cdot |s|^{\mathcal{O}(1)}$. Since $\log k \leq \log(k \log k)$, the algorithm runs in $2^{o(k)} \cdot |s|^{\mathcal{O}(1)}$ time. However, by Proposition 4, this refutes ETH. This completes the proof. □

8 Conclusion

We conclude by stating some open problems. We proved in Theorem 1 and Theorem 2 that h-Layer Crossing Minimization can be solved in $2^{\mathcal{O}(k^\alpha \log k)} \cdot n^{\mathcal{O}(1)}$ time if $h = 2, 3$ with $\alpha = 1/2, 2/3$ respectively, and we demonstrated in Theorem 3 that it is unlikely that for any fixed the problem could be solved in $2^{o(k/\log k)} \cdot n^{\mathcal{O}(1)}$ time for any $h \geq 5$. The most intriguing question is whether 4-Layer Crossing Minimization can be solved in $2^{\mathcal{O}(k^{1-\varepsilon} \operatorname{poly} \log k)} \cdot n^{\mathcal{O}(1)}$ time for some positive $\varepsilon < 1$. Note also that even for $h \geq 5$, our computation lower bound does not exclude the existence of an algorithm with a subexponential in k running time. Can h-Layer Crossing Minimization be solved in $2^{\mathcal{O}(k/\log k)} \cdot n^{\mathcal{O}(1)}$ time for $h \geq 4$? In fact, we are not aware even of whether there is a $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ algorithm for $h \geq 4$. In particular, the algorithm of Dujmović et al. [14] is not single-exponential. Finally, is it possible to improve our results for $h = 2, 3$? Our subexponential algorithms for $h = 2, 3$ run in $2^{\mathcal{O}(k^{1/2} \log k)} \cdot n^{\mathcal{O}(1)}$ and $2^{\mathcal{O}(k^{2/3} \log k)} \cdot n^{\mathcal{O}(1)}$ time, respectively. Is it possible to shave off the logarithmic factor from the exponent of the FPT term? More strongly, for $h = 3$, can we obtain an $2^{\mathcal{O}(k^{1/2} \log k)} \cdot n^{\mathcal{O}(1)}$ time algorithm? The size of the kernel Theorem 4 for $h = 3$ is $\mathcal{O}(k^8)$ and the constant hidden in the big $\mathcal{O}$-notation is huge. We believe that by a more accurate analysis, we can decrease the constant. However, it seems that to substantially decrease the dependence on k, more sophisticated algorithmic tools are needed. We note that the quadratic kernel for 2-Layer Crossing Minimization by Kobayashi et al. [43] heavily relies on the structure of graphs admitting 2-layer drawings without crossings (see Observation 1), and for $h = 3$, the structure becomes a great deal more complicated. Is it possible to obtain a polynomial kernel of "moderate" size for $h = 3$, say, quadratic or cubic?

References

[1] N. Alon, D. Lokshtanov, and S. Saurabh, *Fast FAST*, in Proceedings of the 36th International Colloquium of Automata, Languages and Programming (ICALP), vol. 5555 of Lecture Notes in Comput. Sci., Springer, 2009, pp. 49–58. 4

[2] P. Ashok, F. V. Fomin, S. Kolay, S. Saurabh, and M. Zehavi, *Exact algorithms for terrain guarding*, ACM Trans. Algorithms, 14 (2018), pp. 25:1–25:20. 5

[3] V. Blazej, B. Klemz, F. Klesen, M. D. Sieper, A. Wolff, and J. Zink, *Constrained and ordered level planarity parameterized by the number of levels*, in 40th International Sympo-

sium on Computational Geometry, SoCG 2024, June 11-14, 2024, Athens, Greece, W. Mulzer and J. M. Phillips, eds., vol. 293 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, pp. 20:1–20:16. 18, 22, 42

[4] H. L. Bodlaender, S. Thomassé, and A. Yeo, *Kernel bounds for disjoint cycles and disjoint paths*, Theor. Comput. Sci., 412 (2011), pp. 4570–4578. 15, 96

[5] G. Brückner and I. Rutter, *Partial and constrained level planarity*, in Proceedings of the 28h Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2017, pp. 2000–2011. 6

[6] M.-J. Carpano, *Automatic display of hierarchized graphs for computer-aided decision analysis*, IEEE Transactions on Systems, Man, and Cybernetics, 10 (1980), pp. 705–715. 3

[7] T. Catarci, *The assignment heuristic for crossing reduction*, IEEE Transactions on Systems, Man, and Cybernetics, 25 (1995), pp. 515–521. 3

[8] J. Chuzhoy and Z. Tan, *A subpolynomial approximation algorithm for graph crossing number in low-degree graphs*, in Proceedings of the 54th Annual ACM Symposium on Theory of Computing, STOC, 2022, p. TBA. 6

[9] D. Corneil, M. Habib, C. Paul, and M. Tedder, *A recursive linear time modular decomposition algorithm via lexbfs*, 2024. 88

[10] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*, Springer, 2015. 16

[11] G. Di Battista and E. Nardelli, *Hierarchies and planarity theory*, IEEE Transactions on systems, man, and cybernetics, 18 (1988), pp. 1035–1046. 6

[12] R. Diestel, *Graph Theory, 4th Edition*, vol. 173 of Graduate texts in mathematics, Springer, 2012. 15

[13] 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, *A fixed-parameter approach to 2-layer planarization*, Algorithmica, 45 (2006), pp. 159–182. 6

[14] V. Dujmović, M. R. Fellows, M. Kitching, G. Liotta, C. McCartin, N. Nishimura, P. Ragde, F. Rosamond, S. Whitesides, and D. R. Wood, *On the parameterized complexity of layered graph drawing*, Algorithmica, 52 (2008), pp. 267–292. 3, 4, 6, 17, 108

[15] V. Dujmović, H. Fernau, and M. Kaufmann, *Fixed parameter algorithms for one-sided crossing minimization revisited*, Journal of Discrete Algorithms, 6 (2008), pp. 313–323. 3, 4

[16] V. Dujmović and S. Whitesides, *An efficient fixed parameter tractable algorithm for 1-sided crossing minimization*, Algorithmica, 40 (2004), pp. 15–31. 4

[17] P. Eades, B. D. McKay, and N. C. Wormald, *On an edge crossing problem*, in Proceedings of the 9th Australian Computer Science Conference, Australian National University, 1986, pp. 327–334. 17

[18] P. Eades and N. C. Wormald, *Edge crossings in drawings of bipartite graphs*, Algorithmica, 11 (1994), pp. 379–403. 3

[19] U. Feige, *Faster FAST (Feedback Arc Set in Tournaments)*, CoRR, abs/0911.5094 (2009). 4

[20] H. Fernau, *Two-layer planarization: Improving on parameterized algorithmics*, Journal of Graph Algorithms and Applications, 9 (2005), pp. 205–238. 6

[21] H. Fernau, F. V. Fomin, D. Lokshtanov, M. Mnich, G. Philip, and S. Saurabh, *Social choice meets graph drawing: How to get subexponential time algorithms for ranking and drawing problems*, Tsinghua Science and Technology, 19 (2014), pp. 374–386. 4

[22] F. V. Fomin, D. Lokshtanov, S. Saurabh, and M. Zehavi, *Kernelization: theory of parameterized preprocessing*, Cambridge University Press, 2019. 16

[23] F. V. Fomin and M. Pilipczuk, *Subexponential parameterized algorithm for computing the cutwidth of a semi-complete digraph*, in Proceedings of the 21st Annual European Symposium on Algorithms (ESA), LNCS, vol. 8125 of Lecture Notes in Comput. Sci., Springer, 2013, pp. 505–516. 4

[24] L. R. Ford, Jr. and D. R. Fulkerson, *Maximal flow through a network*, Canadian J. Math., 8 (1956), pp. 399–404. 87, 95

[25] R. Fulek, M. J. Pelsmajer, M. Schaefer, and D. Štefankovič, *Hanani–Tutte, monotone drawings, and level-planarity*, Thirty essays on geometric graph theory, (2013), pp. 263–287. 6

[26] R. Ganian, H. Müller, S. Ordyniak, G. Paesani, and M. Rychlicki, *A tight subexponential-time algorithm for two-page book embedding*, in 51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia, K. Bringmann, M. Grohe, G. Puppis, and O. Svensson, eds., vol. 297 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, pp. 68:1–68:18. 3

[27] M. R. Garey and D. S. Johnson, *Crossing number is NP-complete*, SIAM Journal on Algebraic Discrete Methods, 4 (1983), pp. 312–316. 3, 4

[28] M. Grohe, *Computing crossing numbers in quadratic time*, Journal of Computer and System Sciences, 68 (2004), pp. 285–302. 6

[29] T. Hamm and P. Hliněný, *Parameterised partially-predrawn crossing number*, in 38th International Symposium on Computational Geometry, SoCG 2022, June 7-10, 2022, Berlin, Germany, X. Goaoc and M. Kerber, eds., vol. 224 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, pp. 46:1–46:15. 3

[30] F. Harary and A. Schwenk, *Trees with hamiltonian square*, Mathematika, 18 (1971), pp. 138–140. 4

[31] L. S. Heath and A. L. Rosenberg, *Laying out graphs using queues*, SIAM Journal on Computing, 21 (1992), pp. 927–958. 3

[32] J. E. Hopcroft and R. E. Tarjan, *Efficient algorithms for graph manipulation [H] (algorithm 447)*, Commun. ACM, 16 (1973), pp. 372–378. 18, 88, 89, 91

[33] R. Impagliazzo and R. Paturi, *Complexity of k-sat*, in Proceedings of the 14th Annual IEEE Conference on Computational Complexity, Atlanta, Georgia, USA, May 4-6, 1999, IEEE Computer Society, 1999, pp. 237–240. 4, 17

[34] R. Impagliazzo, R. Paturi, and F. Zane, *Which problems have strongly exponential complexity?*, J. Comput. Syst. Sci., 63 (2001), pp. 512–530. 4, 17

[35] M. Jünger, S. Leipert, and P. Mutzel, *Level planarity testing in linear time*, in In Proceedings of the 6th International Symposium Graph Drawing (GD), Springer, 1998, pp. 224–237. 6

[36] M. Jünger, S. Leipert, and P. Mutzel, *Level planarity testing in linear time*, in Graph Drawing, 6th International Symposium, GD'98, Montréal, Canada, August 1998, Proceedings, S. Whitesides, ed., vol. 1547 of Lecture Notes in Computer Science, Springer, 1998, pp. 224–237. 17, 18

[37] M. Jünger and P. Mutzel, *2-layer straightline crossing minimization: Performance of exact and heuristic algorithms*, in Graph algorithms and applications i, World Scientific, 2002, pp. 3–27. 3

[38] ——, *Graph drawing software*, Springer Science & Business Media, 2012. 3

[39] M. Karpinski and W. Schudy, *Faster algorithms for Feedback Arc Set Tournament, Kemeny Rank Aggregation and Betweenness Tournament*, in Proceedings of the 21st International Symposium on Algorithms and Computation (ISAAC), vol. 6506 of Lecture Notes in Comput. Sci., Springer, 2010, pp. 3–14. 4

[40] K. Kawarabayashi, *Planarity allowing few error vertices in linear time*, in Proceedings of the 50th Annual Symposium on Foundations of Computer Science (FOCS), IEEE, 2009, pp. 639–648. 5

[41] P. N. Klein and D. Marx, *A subexponential parameterized algorithm for subset TSP on planar graphs*, in Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014, C. Chekuri, ed., SIAM, 2014, pp. 1812–1830. 5

[42] B. Klemz and G. Rote, *Ordered level planarity and its relationship to geodesic planarity, bi-monotonicity, and variations of level planarity*, ACM Transactions on Algorithms (TALG), 15 (2019), pp. 1–25. 6

[43] Y. Kobayashi, H. Maruta, Y. Nakae, and H. Tamaki, *A linear edge kernel for two-layer crossing minimization*, Theor. Comput. Sci., 554 (2014), pp. 74–81. 4, 5, 8, 18, 108

[44] Y. Kobayashi and H. Tamaki, *A fast and simple subexponential fixed parameter algorithm for one-sided crossing minimization*, Algorithmica, 72 (2015), pp. 778–790. 4

[45] Y. Kobayashi and H. Tamaki, *A faster fixed parameter algorithm for two-layer crossing minimization*, Information Processing Letters, 116 (2016), pp. 547–549. 4

[46] D. Lokshtanov, F. Panolan, S. Saurabh, R. Sharma, J. Xue, and M. Zehavi, *Crossing number in slightly superexponential time (extended abstract)*, in Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025, Y. Azar and D. Panigrahi, eds., SIAM, 2025, pp. 1412–1424. 3

[47] J. Pach and G. Tóth, *Thirteen problems on crossing numbers*, Geombinatorics, 9 (2000), pp. 194–207. 6

[48] M. Schaefer, *The graph crossing number and its variants: A survey*, The Electronic Journal of Combinatorics, (2012), pp. DS21–Sep. 6

[49] ——, *Crossing numbers of graphs*, CRC Press, 2018. 6

[50] F. Shahrokhi, O. Sýkora, L. A. Székely, and I. Vrťo, *On bipartite drawings and the linear arrangement problem*, SIAM J. Comput., 30 (2001), pp. 1773–1789. 6

[51] K. Sugiyama, S. Tagawa, and M. Toda, *Methods for visual understanding of hierarchical system structures*, IEEE Transactions on Systems, Man, and Cybernetics, 11 (1981), pp. 109–125. 3, 4, 6

[52] R. Tamassia, *Handbook of graph drawing and visualization*, CRC press, 2013. 3, 6

[53] T. Tantau, *Graph drawing in tikz*, Journal of Graph Algorithms and Applications, 17 (2013), pp. 495–513. 3

[54] M. Zehavi, *Parameterized analysis and crossing minimization problems*, Comput. Sci. Rev., 45 (2022), p. 100490. 4