Visualizing Treewidth

Alvin Chiu
University of California, Irvine, CA, USA

Thomas Depian
TU Wien, Austria

David Eppstein
University of California, Irvine, CA, USA

Michael T. Goodrich
University of California, Irvine, CA, USA

Martin Nöllenburg
TU Wien, Austria

Abstract

A *witness drawing* of a graph is a visualization that clearly shows a given property of a graph. We study and implement various drawing paradigms for witness drawings to clearly show that graphs have bounded pathwidth or treewidth. Our approach draws the tree decomposition or path decomposition as a tree of bags, with induced subgraphs shown in each bag, and with "tracks" for each graph vertex connecting its copies in multiple bags. Within bags, we optimize the vertex layout to avoid crossings of edges and tracks. We implement a visualization prototype for crossing minimization using dynamic programming for graphs of small width and heuristic approaches for graphs of larger width. We introduce a taxonomy of drawing styles, which render the subgraph for each bag as an arc diagram with one or two pages or as a circular layout with straight-line edges, and we render tracks either with straight lines or with orbital-radial paths.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Dynamic programming; Human-centered computing → Graph drawings

Keywords and phrases Graph drawing, witness drawings, pathwidth, treewidth

Digital Object Identifier 10.4230/LIPIcs.GD.2025.17

Related Version *Full Version*: `https://arxiv.org/abs/2508.19935` [16]

Supplementary Material *Software*: `https://doi.org/10.17605/OSF.IO/QFZ5V` [17]

Funding *Thomas Depian*: Supported by Vienna Science and Technology Fund (WWTF) [10.47379/ICT22029].
David Eppstein: Supported by NSF grant 2212129.
Michael T. Goodrich: Supported by NSF grant 2212129.
Martin Nöllenburg: Supported by Vienna Science and Technology Fund (WWTF) [10.47379/ICT22029].

1 Introduction

Work in graph drawing can be often viewed as studying paradigms for visualizing a graph, G, to clearly illustrate one or more properties of G while also optimizing one or more quality criteria, such as area or number of edge crossings; see, e.g., [19, 21, 26, 35, 53]. This viewpoint is related to the concept of a *visual proof*, which is a proof given as a visual representation called a *witness* (or visual certificate); see, e.g., [31, 46–48]. Ideally, a visual proof should be clear and concise: the property being proven should be immediately discernible simply by examining the witness. A prominent example is a crossing-free drawing as a witness for planarity. In this paper, we are interested in visualizing treewidth.

A *tree decomposition* of a graph, $G = (V, E)$, is a tree, T, with nodes, $V_1, V_2, \ldots, V_k$, which are called *bags*, where

1. Each V_i is a subset of V and $\bigcup_{i=1}^{k} V_i = V$.
2. If $v \in V_i$ and $v \in V_j$, then v is in each bag in the (unique) path in T from V_i to V_j.
3. For each edge $(v, w) \in E$ there is a bag that contains both v and w.

The *width* of a tree decomposition is one less than the size of its largest bag, and the *treewidth* of a graph is the smallest width of a tree decomposition. The *pathwidth* of a graph is the width of a smallest-width tree decomposition whose tree is a path. See, e.g., [13, 23, 24, 27, 37, 43] for more details regarding these topics, including applications in graph drawing and results that many NP-hard optimization problems can be solved in polynomial time for graphs with bounded treewidth or pathwidth.

We are interested in paradigms for witness drawings of graphs to certify their bounded pathwidth or treewidth. Our approaches are inspired by an example visual proof from Wikipedia that a graph has treewidth 3; see Figure 1. In this illustration, each bag is drawn as a disk with its member vertices drawn as points inside the disk. In addition, for each vertex, $v \in V$, in this drawing, there is a set of curves, each of which we call a *track*, that connect the different copies of v in the bags of the tree decomposition, i.e., the *support* of v in T. The set of tracks in T for a vertex $v \in V$ in such a drawing will form a subtree in T, which will always be a path if T is itself a path. Note, however, that even if the support for every vertex is a path, this does not imply that T is a path, e.g., as shown in Figure 1.

To provide a witness drawing for the pathwidth or treewidth of a graph, $G = (V, E)$, we add one more requirement to our visualization, which is missing from the visualization shown in Figure 1, namely *information faithfulness*, i.e., that the drawing represents the (entire) graph G and its decomposition T. In particular, we require that for each edge, $(v, w) \in E$, we must draw (v, w) as a curve inside each bag V_i such that $v, w \in V_i$ in the given tree decomposition, T, of G. Still, we are interested in such witness drawings that minimize edge and track crossings, as in Figure 1.

To see why this requirement is important, consider a graph G with pathwidth at most ω. Its *interval-width* is bounded by $\omega + 1$, i.e., G is a subgraph of an interval graph G' whose largest clique has size at most $\omega + 1$ [18]. Since interval graphs can be represented as closed intervals on the real line, such a representation of G' can be seen as a certificate that G has bounded pathwidth. However, we argue that this is not an information faithful witness drawing as it shows G' rather than G and, in particular, does not allow a reconstruction of G from the (interval) drawing of G'.

Related Work. Mehlhorn *et al.* [48] and McConnell *et al.* [46] introduce the paradigm of *certifying algorithms*, which output their result as well as a concise proof, called a "witness", that shows that the algorithm produced this output correctly. Subsequent to this work, there has been considerable work on certifying algorithms, including their inclusion in the LEDA and CGAL systems [34]. For example, additional work for witness drawings in graph drawing include work on visualizing proximity properties, such as for Delaunay graphs, Gabriel graphs, and rectangle graphs; see, e.g., [2–5, 39–42].

In GD 2024, the *GraphTrials* framework was introduced, which involves a visual proof for a graph property that can be used in an interactive proof modeled after a bench trial before a judge where a prover establishes that a graph has a given property based on its visualization [31]. Although the authors did not include pathwidth or treewidth in their GraphTrials framework, we feel that our approach to producing witness drawings for graphs with bounded pathwidth or treewidth nevertheless fits into their GraphTrials framework. In

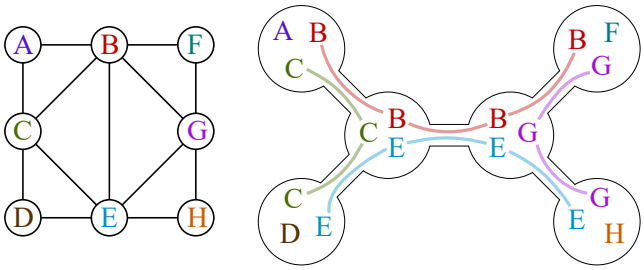

Figure 1 A witness drawing of a graph with treewidth 3. Tracks are shown color coded to illustrate the support for each vertex in the graph. Note that this visualization does not draw graph edges inside the bags of the tree decomposition. Public domain images by David Eppstein [28].

particular, in addition to faithfully representing the provided information, we argue that our created witness drawings contain all the necessary information to efficiently certify that the graph at hand has bounded pathwidth or treewidth. Moreover, such drawings can facilitate the formation of a mental model for potential judges. Thus, we believe that the drawings satisfy the required properties for visual certificates as defined in the framework [31].

We are interested in the natural optimization criterion of minimizing edge crossings in a witness drawing for a graph with bounded pathwidth or treewidth, including track-track crossings, edge-edge crossings, and track-edge crossings. In particular, few track-track crossings can aid in verifying that our decomposition satisfies Property 2, i.e., that the bags containing the vertex v induce a connected subgraph of T, for all $v \in V$. Furthermore, minimizing edge-edge and track-edge crossings improves the visual quality of the witness drawing and can thus support the verification of the (remaining) properties. Although we are not aware of any prior work on this criterion, minimizing track-track crossings is related to minimizing crossings in storyline drawings [22,33,38,54,55] and metro maps [1,6,10,11,30,50–52]. Minimization of edge-edge crossings within the bags is related to crossing minimization in linear and circular graph layouts [7,9,25,32,36,44,56].

Multiple past works include individual visualizations of tree decompositions. These include drawings of a tree with each bag shown as a set of vertices within each tree node, separate from any drawing of the graph [13,43], and drawings of a graph overlaid by bags drawn as regions that surround or connect the vertices [29]. Separately from their applications in treewidth and structural graph theory, tree decompositions with bags of non-constant size have also been used in other ways: for instance, SPQR decompositions are, essentially, tree decompositions of adhesion 2, whose bags induce 3-connected subgraphs, and these have often been visualized as a tree of 3-connected components [20,49]. However, we are unaware of past works studying visualizations of tree decompositions in any systematic way.

Our Results. In this paper, we systematize and implement witness drawings for graphs with bounded treewidth or pathwidth. Given a graph, $G = (V, E)$, our approach is to draw the vertices in each bag of a tree decomposition, T, of G as points in a disk, either in a circular layout or a linear layout. For each vertex, $v \in V$, we connect each pair of copies of v in adjacent bags in T with a curve we call a *track*, ideally with all the tracks color coded with the same color (the color for v). For each bag, V_i, in T, we draw the edges of the subgraph of G induced by V_i. In the case of a linear layout of vertices, we draw this subgraph as an arc diagram [56] (either as one-page or two-page book drawing) and in the case of a circular layout of vertices, we draw this subgraph as a circular layout with straight-line edges [9]. Our optimization goal is to minimize the total number of edge crossings, including track-track

crossings, edge-edge crossings, and track-edge crossings. See Figure 2 for three different witness drawing styles. We design and implement fixed-parameter-tractable linear-time crossing minimization algorithms for graphs with bounded pathwidth or treewidth.

Missing details for statements marked by ★ can be found in the full version [16].

2 Preliminaries

For an integer $p \geq 1$, we use $[p]$ as a shorthand for the set $\{1, 2, \ldots, p\}$. Let $G = (V, E)$ be a graph with vertex set V and edge set E. All considered graphs are simple and undirected. Throughout this paper, we use $n := |V|$ and $m := |E|$ to refer to the number of vertices and edges of G, respectively. For a tree (or path) decomposition T of G with bags $V_1, V_2, \ldots, V_k$, we let ω_T, or simply ω if T is clear from the context, denote its width. Recall $\omega_T := \max_{i \in [k]} |V_i| - 1$. We use $\overline{\omega} := \omega + 1$ as a shorthand for the maximum number of vertices in a bag. Without loss of generality, we root the tree T at an arbitrary bag. We use *decomposition* as a synonym for path and tree decompositions. Furthermore, we assume that the decomposition T consists of $k = \mathcal{O}(n)$ bags and that each bag of T has degree at most three. It is well-known that every graph of treewidth ω admits a certifying tree decomposition with $k = \mathcal{O}(n)$ fulfilling this property [14, 45]. In particular, in *nice tree decompositions*, which have found attention in the design of parameterized (dynamic programming) algorithms [18], every bag has degree at most three. With slight abuse of notation, we write $V_i \in T$ to indicate that V_i is a bag of T. Furthermore, for a bag $V_i \in T$, we let $G_i := G[V_i]$ and $E_i := E(G_i)$ denote the subgraph of G induced by V_i and its edge set, respectively.

3 A Taxonomy of Width-Witness Drawing Styles

Before we describe how to compute a witness drawing for a decomposition T of G, we examine the different witness drawing styles illustrated in Figure 2. In all of them, each bag $V_i \in T$ is represented as a disk D_i of uniform radius. We consider three different drawing styles that differ in how they arrange vertices inside each bag and how they draw the tracks of the support for each vertex. We use parenthesized letters to refer to each style.

Linear Drawings (L). In a *linear drawing*, as in Figure 2b, we arrange the vertices along a vertical line inside a disk and draw the edges as arcs to the left or right of the vertices. This drawing style is inspired by one- and two-page book drawings [36, 56]. The tracks for each support are straight-line edges connecting the respective vertices. We use L1 and L2 to differentiate between one- and two-page book drawings where needed. In an L1-drawing, all arcs, must be either to the right or to the left of the vertices, but this can change between different disks. In an L2-drawing, the arcs can be on both sides of the vertices.

Circular Drawings (C). Figure 2c shows a *circular drawing* of T. There, we arrange the vertices V_i along a circle inside the disk D_i and draw the edges E_i as straight-line chords of said circle. Combinatorially speaking, this is equivalent to a one-page book drawing of G_i [12]. As for linear drawings, we draw tracks as straight-line edges.

Orbital Drawings (O). Our last drawing style, of *orbital drawings*, is inspired by the edge routing in previous drawings with circular vertex placements [8, 15]. It is illustrated in Figure 2d and uses again a circular arrangement of the vertices. However, we now subdivide

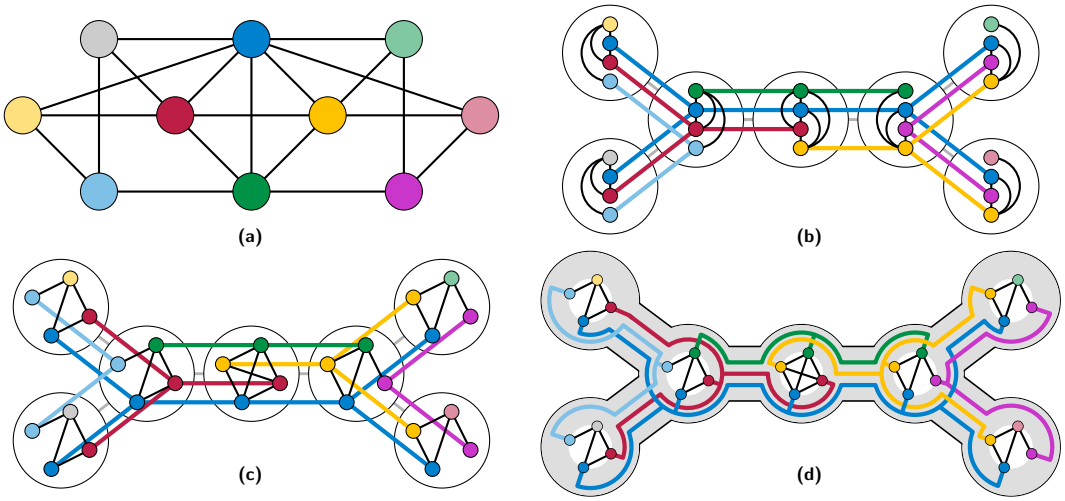

Figure 2 Witness drawings for a graph, G, with treewidth 3. A standard drawing of G is shown in **(a)**. We show in **(b)** a witness drawing of a tree decomposition, in which the subgraph for each bag is drawn as an arc diagram. The two lower witness drawings show the subgraph for each bag in a circular layout with straight-line edges. In **(c)**, the tracks connecting instances of the same vertex are drawn as straight segments, while in **(d)** the track edges are routed in orbits surrounding the bag vertices. The track edges are colored to match the vertices they represent. The drawing in **(b)** has 6 track-track crossings, no edge-edge crossings, and 6 track-edge crossings. The drawing in **(c)** has 4 track-track crossings, 1 edge-edge crossing, and 13 track-edge crossings. The drawing in **(d)** has 12 track-track crossings, 1 edge-edge crossing, and no track-edge crossings.

the disk D_i along the vertices V_i into an inner and outer part: In the inside, we draw the edges E_i as straight-line chords of the respective circle. The outside, however, the so-called *track-routing area*, is reserved for tracks. These are aligned as orbits and are not allowed to leave the track-routing area. We allow them to be routed clockwise or counterclockwise around D_i. This drawing style has no track-edge crossings and all track-track crossings occur inside the track-routing area. While this can result in a cleaner visualization, forcing tracks to stay inside the track-routing area may clutter drawings of decompositions of larger width.

In each drawing style, we evenly distribute the vertices within each disk. For linear drawings the vertex permutation uniquely defines each vertex's position inside the disk, but for the other two drawing styles this still leaves a free rotation angle. To that end, we let $0 \leq \alpha < 2\pi/\overline{\omega}$ denote the starting angle of the first vertex inside each disk, where $\alpha = 0$ corresponds to a placement at twelve o'clock (Figure 3a). We choose a single α across all disks and we select it so that no interior of a track crosses a vertex independent of the order of the vertices inside each disk. For each bag of a tree decomposition, we evenly distribute its children, if there are any, to the right of its disk (Figure 3b).

A witness drawing Γ of a given decomposition T of G consists of a drawing of (i) the tree (or path) T, (ii) each of its bags in one of the above-described drawing styles, and (iii) the support for each vertex of G. Depending on the drawing style, we call Γ an L-, C-, or O-drawing of T, or simply drawing, if the style is clear from the context. The drawing Γ might contain crossings, and we differentiate between the following three crossing types: If two tracks cross, we call it a track-track crossing (or simply t/t-crossing), if a track and an edge in a disk cross, it is a track-edge (t/e) crossing, and if two edges of G cross inside a disk, we call it an edge-edge (e/e) crossing; recall Figure 2. If neither of these crossings exist,

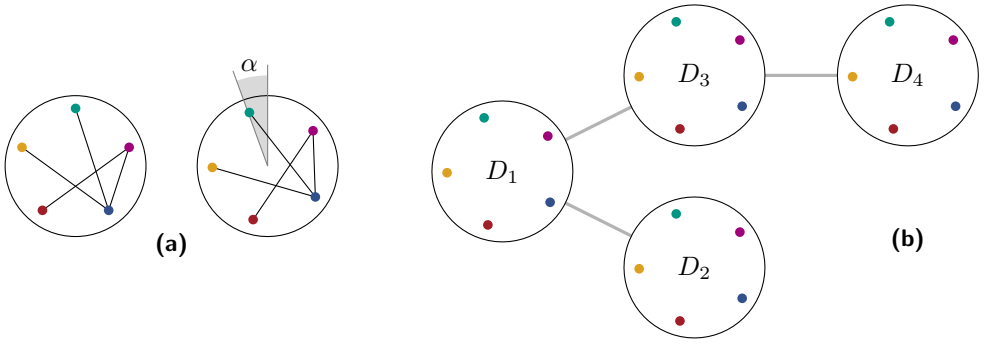

Figure 3 (a) The same circular arrangement of V_i can result in different vertex placements in the disk D_i depending on the angle α: left: $\alpha = 0$, right: $\alpha = 20°$. **(b)** Left-to-right drawing of the tree T with four bags. We omit visualizing edges and tracks to maintain readability.

we call Γ *planar*. Note that we do not count crossings with edges of T. We treat t/t-, t/e-, and e/e-crossings equally throughout the remainder of the paper. However, our algorithms can be readily adapted to weight different types of crossings differently.

4 Linear Drawings

We start our study by considering linear drawings, which visualize each G_i as a book drawing with one or two pages and draw the tracks as straight lines. Observe that the drawing of G_i is uniquely defined by the linear order $\prec_i$ of V_i and the function $\sigma_i \colon E_i \to \{\ell, r\}$, also called *page assignment*, that assigns each edge to the left or right page, respectively. In this section, we present two algorithms that compute, for a given decomposition T of constant width ω, a crossing-minimal L-drawing in linear time. Minimizing the crossings of an L-drawing is tightly interwoven with minimizing the e/e-crossings in the book drawing of each G_i, $i \in [k]$. The known hardness of the latter problem [44] allows us to show also hardness for the former problem, which justifies our focus on constant-width decompositions.

▶ **Theorem 4.1 (★).** *Deciding if a path decomposition T has an L1-drawing with at most c crossings is* NP*-hard. The same holds true for L2-, C-, and O-drawings.*

Since a path decomposition is a restricted form of a tree decomposition, Theorem 4.1 also extends to the latter type of decompositions.

4.1 Linear Drawings for Path Decompositions

We now focus on the more restrictive setting, where we want to compute a witness drawing of a path decomposition T of G of width ω. Throughout this section, we let the bags of T be ordered as they appear on the path T and assume V_1 to be the leftmost bag.

All of our algorithms, including those for C- and O-drawings, are based on the insight that, for each bag $V_i \in T$, the number of crossings involving edges and/or tracks incident to V_i is determined by the drawings of G_i in the disk D_i and the drawings of the induced subgraphs of neighboring bags. This enables us to employ a dynamic programming (DP) algorithm which processes the path decomposition from left to right and computes a crossing-minimal drawing for the first i bags, for every $i \in [k]$. Since the framework of our algorithms is identical across all drawing styles (and decomposition types), we first describe its general structure and later discuss how it can be adapted to specific drawing styles and tree decompositions.

A General DP Algorithm. We let k denote the number of bags in T and recall $\overline{\omega} = \omega + 1$. Let Γ_i be a drawing of the graph G_i in the disk D_i for each $i \in [k]$. Let $\mathcal{G}_i$ denote the set of all possible drawings Γ_i of G_i. All drawing styles admit a combinatorial representation of Γ_i, which implies that $\mathcal{G}_i$ is finite for each $i \in [k]$. Let $\Delta(\Gamma_i)$ denote a drawing of the tracks inside the disk D_i, given a drawing Γ_i of G_i, and let $\mathcal{D}(\Gamma_i)$ be the set of all such drawings. L- and C-drawings represent the tracks as straight lines. Therefore, $\Delta(\Gamma_i)$ is only relevant for O-drawings and we leave it undefined for the other two drawing styles. For this drawing style, we can again represent $\Delta(\Gamma_i)$ combinatorially, ensuring that $\mathcal{D}(\Gamma_i)$ is finite. We define $\mathcal{G}_{\max} \coloneqq \max_{i\in[k]} |\mathcal{G}_i|$ and $\mathcal{D}_{\max} \coloneqq \max_{\Gamma_i\in\mathcal{G}_i, i\in[k]} |\mathcal{D}(\Gamma_i)|$ as the maximum number of drawings of $G_1, \ldots, G_k$ and their corresponding track drawings, respectively.

In our algorithm, we maintain an $\mathcal{O}(k \cdot \mathcal{G}_{\max} \cdot \mathcal{D}_{\max})$-size table C, where $C[i, \Gamma_i, \Delta(\Gamma_i)]$ stores the minimum number of crossings of a drawing for the bags $V_1, V_2, \ldots, V_i$, where Γ_i is the drawing of G_i in D_i and $\Delta(\Gamma_i)$ the drawing of its incident tracks. We call $S_i = (i, \Gamma_i, \Delta(\Gamma_i))$ a *state* of our DP. To compute $C[S_i]$, we need to count all e/e-crossings in Γ_i involving edges from E_i, and all t/e-, and t/t-crossings involving also edges or tracks incident to vertices from V_{i-1} for a given state S_{i-1}, if the bag V_{i-1} exists. Let $\mathrm{cr}_{\mathsf{e/e}}(\cdot)$, $\mathrm{cr}_{\mathsf{t/e}}(\cdot,\cdot)$, and $\mathrm{cr}_{\mathsf{t/t}}(\cdot,\cdot)$ denote these number of crossings, respectively. Note that the exact definition of $\mathrm{cr}_{\mathsf{e/e}}(\cdot)$, $\mathrm{cr}_{\mathsf{t/e}}(\cdot,\cdot)$, and $\mathrm{cr}_{\mathsf{t/t}}(\cdot,\cdot)$ depends on the desired drawing style. For $i > 1$, the number of crossings $\mathrm{cr}(S_i, S_{i-1})$ for two states $S_i = (i, \Gamma_i, \Delta(\Gamma_i))$ and $S_{i-1} = (i-1, \Gamma_{i-1}, \Delta(\Gamma_{i-1}))$ equals the sum of $\mathrm{cr}_{\mathsf{e/e}}(S_i)$, $\mathrm{cr}_{\mathsf{t/e}}(S_i, S_{i-1})$, and $\mathrm{cr}_{\mathsf{t/t}}(S_i, S_{i-1})$. We can now relate the different states in C to each other as follows, where $S_i = (i, \Gamma_i, \Delta(\Gamma_i))$, $i \in [k]$, $\Gamma_i \in \mathcal{G}_i$, and $\Delta(\Gamma_i) \in \mathcal{D}(\Gamma_i)$:

$$C[S_i] = \begin{cases} \mathrm{cr}_{\mathsf{e/e}}(S_i) & \text{if } i = 1 \\ \min\limits_{\substack{S_{i-1} \,=\, (i-1,\Gamma_{i-1},\Delta(\Gamma_{i-1})),\\ \Gamma_{i-1}\in\mathcal{G}_{i-1},\ \Delta(\Gamma_{i-1})\in\mathcal{D}(\Gamma_{i-1})}} C[S_{i-1}] + \mathrm{cr}(S_i, S_{i-1}) & \text{otherwise} \end{cases} \tag{1}$$

Using induction, we can show that Equation (1) captures the minimum number of crossings.

▶ **Lemma 4.2 (★).** *Let T be a path decomposition of G with k bags. For each $i \in [k]$, drawings $\Gamma_i \in \mathcal{G}_i$ and $\Delta(\Gamma_i) \in \mathcal{D}(\Gamma_i)$, the table entry $C[i, \Gamma_i, \Delta(\Gamma_i)]$ equals the minimum number of crossings of a witness drawing for the bags $V_1, \ldots, V_i$ with the drawings Γ_i and $\Delta(\Gamma_i)$ for G_i and its incident tracks, respectively.*

With Lemma 4.2 at hand, we now have all key ingredients for our DP. It remains to define for each drawing style the representations of Γ_i and $\Delta(\Gamma_i)$, if needed, and to implement the crossing counting functions $\mathrm{cr}_{\mathsf{e/e}}(\cdot)$, $\mathrm{cr}_{\mathsf{t/e}}(\cdot,\cdot)$, and $\mathrm{cr}_{\mathsf{t/t}}(\cdot,\cdot)$. This will be the main task for the remainder of this and the upcoming sections.

Two-Page Linear Drawings. We now aim to compute a crossing-minimal L2-drawing of a path decomposition T. Recall that a two-page book drawing Γ_i of $G_i = (V_i, E_i)$ is uniquely defined by the linear order $\prec_i$ on V_i and the page assignment $\sigma_i \colon E_i \to \{\ell, r\}$. Thus, we set $\Gamma_i = \langle \prec_i, \sigma_i \rangle$. As the tracks are drawn using straight lines between the vertices in the respective disks, it is sufficient to store only the drawing for each G_i, $i \in [k]$, in our DP table C. As $\mathcal{G}_{\max} = \mathcal{O}(\overline{\omega}! \cdot 2^{\overline{\omega}^2})$ holds, the size of the table C is in $\mathcal{O}(k \cdot \overline{\omega}! \cdot 2^{\overline{\omega}^2})$ and it only remains to implement the crossing counting functions.

For e/e-crossings, we observe that $\mathrm{cr}_{\mathsf{e/e}}(\cdot)$ equals the number of edges with alternating endpoints in $\prec_i$ but assigned in σ_i to the same page; see Figure 4a and Equation (2).

$$\mathrm{cr}_{\mathsf{e/e}}(S_i) \coloneqq |\{(u,v),(a,b) \in E_i \mid \sigma((u,v)) = \sigma((a,b)), u \prec_i a \prec_i v \prec_i b\}| \tag{2}$$

To determine the number of t/e-crossings, it is sufficient to observe that an edge $(u, v) \in E_{i-1}$ with $u \prec_{i-1} v$ crosses with a track for the vertex $w \in V_{i-1} \cap V_i$ if and only if $u \prec_{i-1} w \prec_{i-1} v$ and $\sigma_{i-1}((u, v)) = r$ as visualized in Figure 4b. A symmetric observation can be made for edges $(u, v) \in E_i$ with $\sigma_i((u, v)) = \ell$, yielding the following function:

$$\begin{aligned} \mathrm{cr}_{\mathsf{t/e}}(S_i, S_{i-1}) \coloneqq & \sum_{\substack{(u,v)\in E_{i-1} \\ \sigma_{i-1}((u,v))=r}} |\{w \in V_{i-1} \cap V_i \mid u \prec_{i-1} w \prec_{i-1} v\}| \\ & + \sum_{\substack{(u,v)\in E_i \\ \sigma_i((u,v))=\ell}} |\{w \in V_{i-1} \cap V_i \mid u \prec_i w \prec_i v\}| \end{aligned} \tag{3}$$

This leaves us with counting the t/t-crossings, for which it is sufficient to count the number of inversions between $\prec_{i-1}$ and $\prec_i$; see Figure 4c and Equation (4).

$$\mathrm{cr}_{\mathsf{t/t}}(S_i, S_{i-1}) \coloneqq |\{u, v \in V_{i-1} \cap V_i \mid u \prec_{i-1} v, v \prec_i u\}| \tag{4}$$

Combining this with the recurrence relation from Equation (1), we obtain the following:

▶ **Theorem 4.3.** *Let T be a path decomposition of G of width ω. We can compute a crossing-minimal L2-drawing of T in $\mathcal{O}(n \cdot (\overline{\omega}!)^2 \cdot 4^{\overline{\omega}^2} \cdot \overline{\omega}^4)$ time.*

Proof. We employ our DP algorithm storing the above-described states to compute a crossing-minimal L2-drawing of T. Recall that the table C has size $\mathcal{O}(k \cdot \overline{\omega}! \cdot 2^{\overline{\omega}^2})$, where $k = \mathcal{O}(n)$ denotes the number of bags in T. To compute a single state for some $i \in [k]$, we have to access up to $\mathcal{O}(\overline{\omega}! \cdot 2^{\overline{\omega}^2})$ states for $i - 1$. A closer analysis of Equations (2)–(4) reveals that we can evaluate the function $\mathrm{cr}(\cdot, \cdot)$ in $\mathcal{O}(\overline{\omega}^4)$ time. Thus, we can compute the minimum number of crossings for a given state in $\mathcal{O}(\overline{\omega}! \cdot 2^{\overline{\omega}^2} \cdot \overline{\omega}^4)$ time. Overall, this amounts to $\mathcal{O}(k \cdot (\overline{\omega}!)^2 \cdot 4^{\overline{\omega}^2} \cdot \overline{\omega}^4)$ time to fill the entire table C. The minimum number of crossings can be obtained by taking the minimum over all $C[k, \cdot, \cdot]$. We can use standard backtracking techniques to compute the crossing-minimal L2-drawing of T. The correctness of our algorithm follows directly from the correctness of our recurrence relation, i.e., Equation (1), established in Lemma 4.2. ◀

With the use of weights, we can prioritize some types of crossings over others. Furthermore, by restricting the DP table C and relation from Equation (1), we can generalize the above-presented DP. For example, to compute an L1-drawing, we can remove the page assignment

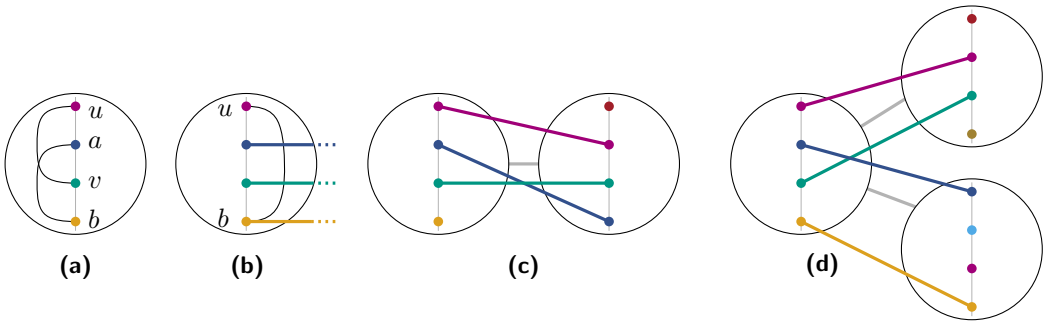

Figure 4 Illustration of different types of crossing: **(a)** The e/e-crossing between the two edges (u, v) and (a, b) is enforced by the relative order of their endpoints. **(b)** The edge (u, b) crosses the blue and green tracks because the respective vertices lie between u and b. **(c)** The blue and green tracks cause a t/t-crossing because the relative order among the respective endpoints is inversed. **(d)** A "criss-cross" t/t-crossing between blue and green that can occur in tree decompositions.

from our definition of a drawing Γ_i of G_i. Furthermore, it could be desired to enforce a consistent linear order across different bags, i.e., forbid t/t-crossings. To that end, we can consider in Equation (1) only those linear orders $\prec'$ that are consistent with $\prec$, i.e., where $\prec|_{V_{i-1}} = \prec'|_{V_i}$. Below, we summarize the effects on the running time.

▶ **Corollary 4.4.** *Let T be a path decomposition of G of width ω. We can compute a crossing-minimal*

- *L1-drawing of T in $\mathcal{O}(n \cdot (\overline{\omega}!)^2 \cdot \overline{\omega}^4)$ time.*
- *L2-drawing of T without t/t-crossings in $\mathcal{O}(n \cdot (\overline{\omega}!)^2 \cdot 4^{\overline{\omega}^2} \cdot \overline{\omega}^4)$ time.*

4.2 Linear Drawings for Tree Decompositions

Similarly, for graphs of bounded treewidth, we can obtain the minimum number of total crossings for L-drawings via DP. Given a tree decomposition T of G, we orient T "left-to-right", where the root bag is leftmost and children are to the right of the parent. We then obtain a right-to-left ordering of the tree's bags, $V_k, V_{k-1}, \ldots, V_1$, where V_1 is the root bag.

DP for Treewidth. Our DP formulation in this problem uses the same table as for path decompositions, but a given bag may now have more than one previous bag that can affect its crossings. We assume that the tree decomposition T only has bags of degree up to three. As such, any bag has at most one parent (due to the "left-to-right" orientation) and at most two children. We consider three cases to calculate $C[S_i]$ for a bag V_i, based on its in-degree $\deg^-(V_i) \in \{0, 1, 2\}$. The cases where $\deg^-(V_i)$ equal zero or one are similar to the two cases presented in the DP for pathwidth, but when $\deg^-(V_i) = 2$ the crossing function $\mathrm{cr}(\cdot,\cdot,\cdot)$ depends on the current state and its two child states. In this case, the current bag has two children belonging to its in-degree neighboring set $N^-(V_i)$ whose embedding (which child is above the other) is decided by the DP. Our DP table has the following recurrence relation:

$$C[S_i] = \begin{cases} \mathrm{cr}_{\mathsf{e}/\mathsf{e}}(S_i) & \text{if } \deg^-(V_i) = 0 \\ \min\limits_{\substack{S_x \,=\, (x, \Gamma_x, \Delta(\Gamma_x)) \\ V_x \in N^-(V_i)}} C[S_x] + \mathrm{cr}(S_i, S_x) & \text{if } \deg^-(V_i) = 1 \\ \min\limits_{\substack{(S_x, S_y) \\ V_x, V_y \in N^-(V_i), x \neq y}} C[S_x] + C[S_y] + \mathrm{cr}(S_i, S_x, S_y) & \text{if } \deg^-(V_i) = 2 \end{cases} \tag{5}$$

Two-Page Linear Drawings. The number of e/e-crossings does not change within the parent bag, so that remains the same as in Equation (2). The number of t/e-crossings can be treated independently for both children $V_x, V_y \in N^-(V_i)$, as there are no such crossings between the children. So Equation (6) is similar to Equation (3) except that we sum over both children.

$$\mathrm{cr}_{\mathsf{t}/\mathsf{e}}(S_i, S_x, S_y) := \sum_{i' \in \{x,y\}} \left(\sum_{\substack{(u,v) \in E_{i'} \\ \sigma_{i'}((u,v)) = \ell}} |\{w \in V_{i'} \cap V_i \mid u \prec_{i'} w \prec_{i'} v\}| \right.$$
$$\left. + \sum_{\substack{(u,v) \in E_i \\ \sigma_i((u,v)) = r}} |\{w \in V_{i'} \cap V_i \mid u \prec_i w \prec_i v\}| \right) \tag{6}$$

For t/t-crossings, we must handle t/t-crossings between children. Without loss of generality, let the child V_x that comes first in the input be above child V_y. Then such t/t-crossings will occur when two tracks from the children "criss-cross", where a track from the top child goes

below the track from the bottom child as in Figure 4d, captured by Equation (7). Then we add the t/t-crossings between parent and child as in Equation (4), but for both children now:

$$\mathrm{cr}_{\mathsf{t/t}}(S_i, S_x, S_y) \coloneqq |\{(u,v) \in (V_x \cap V_i) \times (V_y \cap V_i) \mid v \prec_i u\}| \tag{7}$$

$$+ \sum_{i' \in \{x,y\}} |\{u, v \in V_{i'} \cap V_i \mid u \prec_{i'} v, v \prec_i u\}| \tag{8}$$

A similar analysis to Theorem 4.3 and Corollary 4.4 yields the following results for L-drawings:

▶ **Theorem 4.5.** *Let T be a tree decomposition of G of width ω. We can compute a crossing-minimal L2-drawing of T in $\mathcal{O}(n \cdot (\overline{\omega}!)^3 \cdot 8^{\overline{\omega}^2} \cdot \overline{\omega}^4)$ time.*

Proof. The table C has the same size $\mathcal{O}(k \cdot \overline{\omega}! \cdot 2^{\overline{\omega}^2})$, with $k = \mathcal{O}(n)$, and computing $\mathrm{cr}(\cdot,\cdot,\cdot)$ still takes $\mathcal{O}(\overline{\omega}^4)$ time as in the pathwidth case. However, computing a single state must now check all possible pairings of its two child states, yielding $\mathcal{O}((\overline{\omega}! \cdot 2^{\overline{\omega}^2})^2)$ states in the worst case. So, altogether, the running time will be in $\mathcal{O}(n \cdot (\overline{\omega}!)^3 \cdot 8^{\overline{\omega}^2} \cdot \overline{\omega}^4)$. ◀

▶ **Corollary 4.6.** *Let T be a tree decomposition of G of width ω. We can compute a crossing-minimal L1-drawing of T in $\mathcal{O}(n \cdot (\overline{\omega}!)^3 \cdot \overline{\omega}^4)$.*

Finally, we note that our DP algorithm can be extended to decompositions T with bags of arbitrary degree by generalizing Equations (5)–(7). In particular, for Equation (7), we must account for t/t-crossings between every pair of children of V_i. As stated in Equation (5), we consider all possible combinations of states for V_i's children, and the number of such combinations grows with their number. Moreover, observe that the embedding of these children affects the number of t/t-crossings, and the number of potential embeddings is itself exponential in the degree d of V_i. Therefore, allowing bags with non-constant degree in T introduces an exponential factor in terms of d in the running time of our DP algorithm.

5 Drawing Styles with Circular Vertex Placements

In this section, we discuss how to adapt our DP algorithm presented in Section 4 to compute crossing-minimal C- and O-drawings.

Circular Drawings. Circular drawings arrange the vertices in every disk D_i, $i \in [k]$, on a circle and draw the edges E_i as straight lines. Thus, the drawing Γ_i of G_i is uniquely defined by the placement of the vertices V_i in D_i. Since they are evenly distributed on a circle, it suffices to store in Γ_i for each bag $V_i \in T$, the counterclockwise order $\prec_i$ of the vertices V_i as they appear in D_i, starting at the twelve o'clock position. Recall that the angle parameter α specifies the starting angle of the first vertex in the order $\prec_i$ in the disk D_i. As the tracks are again straight lines, we do not need to store $\Delta(\Gamma_i)$, reducing the size of C to $\mathcal{O}(k \cdot \overline{\omega}!)$.

When filling the table C, we recall that the recurrence relations from Equations (1) and (5) include the number of e/e-, t/e-, and t/t-crossings involving the vertices of a bag V_i, $i \in [k]$, and those of its adjacent bags (if they exist). The number of e/e-crossings corresponds, due to the equivalence with one-page book drawings, to the number of edge pairs with alternating endpoints in $\prec_i$ and is, therefore, purely combinatorial in this setting. In contrast, the t/e-, and t/t-crossings are geometric in nature and depend on the concrete positions of the vertices within the disks. Observe that these positions depend on α and we remark that the choice of α influences the number of observed crossings; see also Figure 5. Consequently, we equip the crossing counting functions $\mathrm{cr}_{\mathsf{t/e}}(\cdot,\cdot,\cdot)$ and $\mathrm{cr}_{\mathsf{t/t}}(\cdot,\cdot,\cdot)$ with the parameter α. Altogether, this is sufficient to compute crossing-minimal C-drawings.

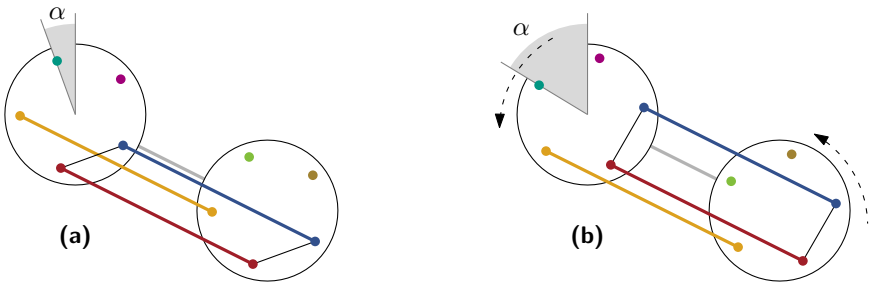

Figure 5 The observed crossings not only depend on the order of the vertices in the disks but also on the choice of α (also indicated with the dashed arrows): The t/e-crossing with the yellow track in **(a)** is not present in **(b)** although we use the same linear orders in both visualizations.

Orbital Drawings. Orbital drawings extend circular drawings by routing the tracks as orbits around the vertices; recall Figure 2d. Since tracks must remain within the track-routing area, every O-drawing is free of t/e-crossings. However, unlike in the other two drawing styles, the drawing of tracks within a disk is no longer uniquely determined by the placement of the vertices. In particular, tracks can be assigned to different orbits and may orbit the drawing Γ_i of G_i either clockwise or counterclockwise. Thus, while it still suffices to store a linear order $\prec_i$ of V_i for Γ_i, representing $\Delta(\Gamma_i)$ now requires additional care.

We model $\Delta(\Gamma_i)$ with two parts. First, to model the assignment of tracks to orbits, we introduce an *orbit assignment function* $\lambda_i \colon V_i \to [\overline{\omega}]$, where $\lambda_i(v)$ specifies the orbit used for the tracks of vertex v, numbered from the center of the disk D_i outward; see Figure 6. Two tracks for different vertices $u, v \in V_i$ may share the same orbit if they do not intersect. Otherwise, we require $\lambda_i(u) \neq \lambda_i(v)$ to guarantee that no two tracks for different vertices share the same orbit simultaneously. Second, to model the direction in which the tracks orbit, i.e., clockwise or counterclockwise, we consider the (up to two) children V_x and V_y and the parent V_q of the bag V_i in the tree decomposition T, if they exist. We define a *direction function* $\delta_i \colon V_i \times \{x, y, q\} \to \{\mathrm{cw}, \mathrm{ccw}\}$ which, for each vertex $v \in V_i$, captures the direction (clockwise (cw) or counterclockwise (ccw)) of the track from V_i to each adjacent bag. Figure 6 illustrates the semantic of δ_i and underlines the importance of storing this information separately for each adjacent bag. Thus, for a given drawing Γ_i of G_i, we set $\Delta(\Gamma_i) = (\lambda_i, \delta_i)$. This increases the size of the DP table C to $\mathcal{O}(k \cdot \overline{\omega}! \cdot (8\overline{\omega})^{\overline{\omega}})$.

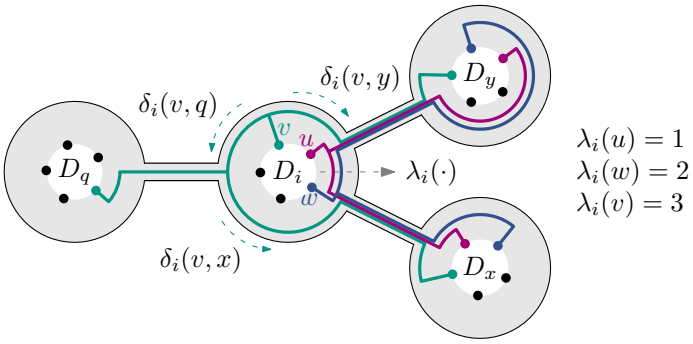

Figure 6 Visualization of a (partial) O-drawing and the information that we store in the table C. The track routing area is indicated in gray. Black vertices represent arbitrary other vertices. In this drawing, we have $\lambda_i(u) = 1$, $\lambda_i(w) = 2$, $\lambda_i(v) = 3$, $\delta_i(v, y) = \mathrm{cw}$, and $\delta_i(v, q) = \delta_i(v, x) = \mathrm{ccw}$. Note that these orientations are required to minimize the t/t-crossings involving the tracks for vertex v.

Regarding the computation of the number of crossings, we observe that the function $\mathrm{cr}_{\mathsf{e}/\mathsf{e}}(\cdot)$ from C-drawings can be reused, and that t/e-crossings do not occur in this drawing style. t/t-crossings involving a track for a vertex $v \in V_i$ occur in the following two cases. First, if v also appears in an adjacent bag, and a track for some vertex $u \in V_i$ with $\lambda_i(u) < \lambda_i(v)$ passes by the position of v in Γ_i, then these two tracks cross: The track for v must cross the one of u to reach its orbit; see Figure 7a. Second, for each adjacent bag V_j with $v \in V_j$, we count the number of vertices $u \in V_i$ such that a track for u blocks the path from D_i to D_j, see Figure 7b. In both cases, the crossings can be determined from the starting angle α, the order of the children, and the stored information, i.e., Γ_i and $\Delta(\Gamma_i)$, for the disk D_i of V_i and the disks of its adjacent bags. Hence, the function $\mathrm{cr}_{\mathsf{t}/\mathsf{t}}(\cdot,\cdot,\cdot)$ still only depends on the states for V_i and its child bags. Finally, since tracks to occurrences of the same vertex $v \in V_i$ in different bags can overlap, we must avoid double-counting in such cases; see again Figure 7a. We summarize below the findings of this section. Recall that $k = \mathcal{O}(n)$ holds.

▶ **Theorem 5.1 (★).** *Let T be a decomposition of G of width ω. We can compute a crossing-minimal*

- *C-drawing of T in $\mathcal{O}(n \cdot (\overline{\omega}!)^2 \cdot \overline{\omega}^4)$ time if T is a path decomposition.*
- *C-drawing of T in $\mathcal{O}(n \cdot (\overline{\omega}!)^3 \cdot \overline{\omega}^4)$ time if T is a tree decomposition.*
- *O-drawing of T in $\mathcal{O}(n \cdot (\overline{\omega}!)^2 \cdot (4\overline{\omega})^{2\overline{\omega}} \cdot \overline{\omega}^4)$ time if T is a path decomposition.*
- *O-drawing of T in $\mathcal{O}(n \cdot (\overline{\omega}!)^3 \cdot (8\overline{\omega})^{3\overline{\omega}} \cdot \overline{\omega}^4)$ time if T is a tree decomposition.*

6 Heuristic Approaches to Compute Linear Witness Drawings

A proof-of-concept implementation of our DP from Section 4.2 for L2-drawings of tree decompositions required over 8 minutes to terminate even for small decompositions of width $\omega = 3$ with four bags. This is no surprise given the running time bounds established in Theorem 4.5. For tree decompositions of larger width, this became too large in practice with a single state needing to check up to $(5! \cdot 2^{5^2})^2 \approx 10^{19}$ possible configurations for $\omega = 4$.

Consequently, we do not want to restrict our attention to exact, but slow, algorithms, but also explore heuristics to efficiently compute drawings with a small number of crossings, but without any formal guarantee on optimality. In this section, we describe three heuristics for L2-drawings. We evaluate their performance in Section 7 and provide in Figures 8 and 9 sample drawings computed by them.

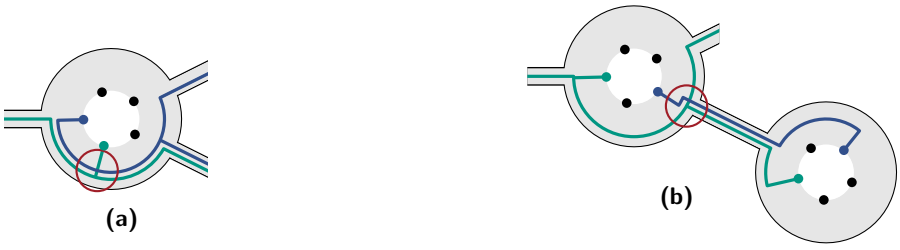

Figure 7 The two cases how t/t-crossings in O-drawings can arise: In **(a)**, the blue tracks orbit inside the green ones, thus causing a crossing when they pass the green vertex. t/t-crossings are highlighted with the red circles. The crossing in **(b)** occurs because the green track blocks the visibility to the lower-right disk for the blue track. Observe that there is only one t/t-crossing in **(a)**, although there is technically one blue track from the upper- and one from the lower-right disk.

Global Book Drawing Heuristic. Since we visualize in an L2-drawing each graph G_i, $i \in k$, as a two-page book drawing of G_i, our first heuristic computes such a drawing $\langle \prec, \sigma \rangle$ for the entire graph G. After that, we set for each disk D_i $\prec_i = \prec|_{V_i}$ and $\sigma_i(e) = \sigma(e)$ for every edge $e \in E_i$, i.e., we project the drawing $\langle \prec, \sigma \rangle$ down into each bag. If T is a tree decomposition, we additionally perform a bottom-up traversal of T, choosing for each internal bag V_i the embedding of its children that yields the fewer t/e- and t/t-crossings locally.

Since computing a crossing-minimal one-page book drawing for a graph is an NP-hard problem [44], we already employ a heuristic for this step. Klawitter, Mchedlidze, and Nöllenburg [36] performed an extensive experimentally evaluation on book drawing heuristics. In their conclusion, they recommended the heuristic `conGreedy+`, which places one vertex after the other on the spine, selecting the next vertex based on the number of already placed neighbors, and extends the spine order and page assignment greedily. While for two-page book drawings of k-trees, which are maximal graphs of treewidth k, it was slightly outperformed by other heuristics, it performed best overall and in particular for graphs with a sub-quadratic number of edges. The heuristic has a running time of $\mathcal{O}(m^2 n)$ [36] and we call it `Global conGreedy+`.

Local Book Drawing Heuristic. Computing a book drawing for the whole graph G and then applying it to each disk D_i, $i \in k$, has two main disadvantages. On the one hand, we usually do not visualize the entire graph G in a single bag, but several induced subgraphs G_i separately. Therefore, the drawing $\langle \prec_i, \sigma_i \rangle$ for G_i might contain unnecessary crossings. On the other hand, the above-described heuristic is, apart from the embedding step, unaware of t/e- and t/t-crossings. This, in particular, leads to situations where swapping the page assignment of one edge $e \in E_i$ or the entire edge set E_i can dramatically reduce the number of t/e-crossings. The latter occurs especially at the root and leave bags of T.

Therefore, we now employ the algorithm `conGreedy+` in each bag V_i separately during a top-down traversal of T, thus computing for each G_i a book drawing from scratch. Since the drawing for the parent of a bag V_i has been determined, we can take the potential t/t- and t/e-crossings into account. More precisely, when selecting the spine position of a vertex $v \in V_i$ and the page assignment for its incident edges in G_i we also compute the number of t/e- and t/t-crossings that this causes based on the drawing of the parent, similar to Equations (6) and (7). Furthermore, we forward this information to its children when considering their two possible embeddings if necessary. We call this heuristic `Local conGreedy+` and, since the size of each G_i is bounded, its running time is $\mathcal{O}(n \cdot \overline{\omega}^5)$.

Local Search. In addition to the two above-mentioned heuristics, we also perform a local search to reduce the number of crossings. Given an L2-drawing of T, we traverse T bottom up. For each bag $V_i \in T$, we perform one or several of the following movements.

Vertex-Swap: Take two vertices $u, v \in V_i$ and swap their position on the spine $\prec_i$.
Edge-Swap: Take two edges $e_1, e_2 \in E_i$ with $\sigma_i(e_1) \neq \sigma_i(e_2)$ and swap their page assignment.
Edge-Flip: Take an edge $e \in E_i$ and assign it to the other page.
Embedding-Flip: Flip the embedding of the subtree V_i, i.e., the order of its children.

For a given bag $V_i \in T$, we perform a hill-climbing approach and exhaustively apply above movements until no further improvement can be made in the drawing $\langle \prec_i, \sigma_i \rangle$. Then, we proceed to the next bag $V_j \in T$. Note that when evaluating the number of crossings, we keep the drawings in the remaining bags fixed. Thus, after a bottom-up traversal of T, we perform a second traversal, this time top-down. Afterwards, or after a predetermined period of time, we return the best solution found so far.

7 Experimental Evaluation

We implemented the dynamic program for L2-crossings as described in Section 4.2 as well as all of the heuristics presented in Section 6 in `Python 3.6.9`. All algorithms run on a system with Intel Xeon E5-2640 v4 10-core processors at 2.40 GHz. We set hard limits for memory and time of 96 GB and fifteen minutes, respectively, but memory was not a limitation for our algorithms. If we improve a solution from a heuristic using local search, we indicate this using "`& LS`". Also the local search has a maximum running time of fifteen minutes. The computed drawings are visualized using `d3.js` on an independent system and this final step is not considered here. The instances are based on a public repository containing graphs extracted from "Sage" together with a tree decomposition of them.[1] Out of the 150 graphs, 55 of them were equipped with a (tree or path) decomposition T where each bag, i.e., node of the decomposition T, has degree at most three in T and we used these in our experiments. We have no information on how these decompositions were computed or if they are optimal. The width of the decompositions ranges between one and 53 and they have between one and 143 bags. For 60% of the instances, both parameters were simultaneously below ten. The graphs have between four and 143 vertices and between six and 1,008 edges, with a mean density, i.e., $2m/(n(n-1))$, of 0.37.

Computed Drawings. We provide in Figures 8 and 9 sample drawings of two graphs together with the witness drawings computed by our different algorithms. Generally, we can see that the DP and `Global conGreedy+` produce drawings with a more consistent arrangement of the vertices on the individual spines. Comparing Figures 8b and f, we can observe that the obtained drawing from `Global conGreedy+ & LS` is nearly identical to the one of the DP, confirming the observed low crossing numbers. The consistency across different spines yields also a visually more pleasing drawing, as we can see in Figure 9. We provide in the full version [16] more sample drawings computed by our algorithms and analyze below their running time and the quality of the produced drawings.

Running Times, Tradeoffs, and the Influence of the Width. All heuristics terminated on every instance within fifteen minutes. The DP could solve only thirteen instances within the time limit. To evaluate the crossing minimization performance of the heuristics on a larger set of instances, we ran the DP with a time limit of six hours, after which 21 of the 55 instances, all of width at most four, could be solved optimally. For the remaining 34 instances, `Global conGreedy+` obtained four, `Local conGreedy+` three, `Global conGreedy+ & LS` 18, and `Local conGreedy+ & LS` 20 times a solution with the fewest crossings.

Figure 10 shows the relative number of crossings, compared to the best-performing algorithm, i.e., the DP where available and one of the heuristics otherwise, and running time for each algorithm and instance. We group instances by width and sort them within each group based on the number of bags. In the full version [16], we provide a filtered version of Figure 10, focusing on instances for which the DP terminated within six hours. Regarding the running time, we can observe in Figure 10a that all heuristics without the local search step terminated within a second on every instance except the largest ones. As

[1] The original dataset is available at `https://github.com/freetdi/named-graphs`. The filtered dataset, source code, and evaluation code can be found on OSF [17]. The application can also be accessed online at `https://www.ac.tuwien.ac.at/projects/visualizing-treewidth/`.

[2] A drawing of the graph can be found online: `https://en.wikipedia.org/wiki/Brinkmann_graph`.

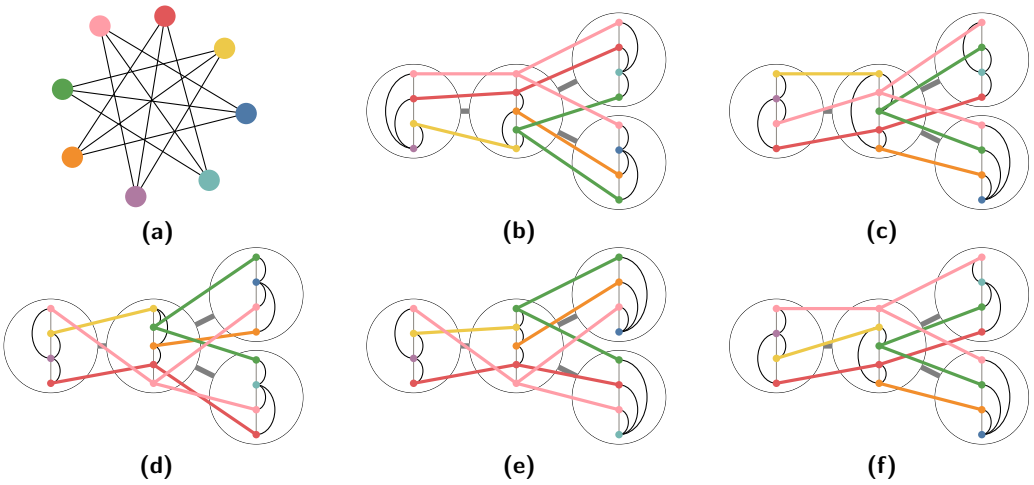

Figure 8 Comparison of the drawings computed by our algorithms for the decomposition of the *Wagner* graph; we indicate the number of crossings in the parenthesis. **(a)** Force-based drawing of the graph, **(b)** DP (3), **(c)** `Global conGreedy+` (8), **(d)** `Local conGreedy+` (9), **(e)** `Local conGreedy+ & LS` (5), and **(f)** `Global conGreedy+ & LS` (4).

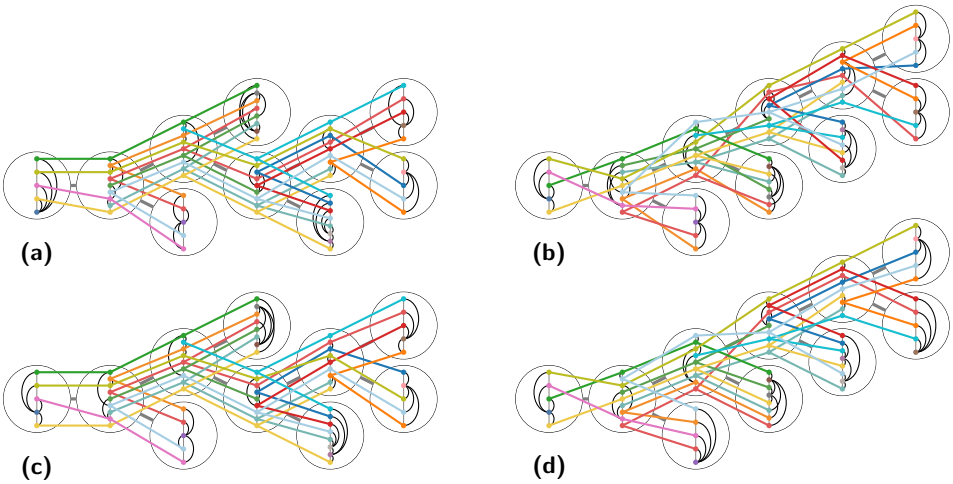

Figure 9 Comparison of the drawings computed by our heuristics for the *Brinkmann* graph[2]; we indicate the number of crossings in the parenthesis. **(a)** `Global conGreedy+` (113), **(b)** `Local conGreedy+` (105), **(c)** `Global conGreedy+ & LS` (64), and **(d)** `Local conGreedy+ & LS` (72).

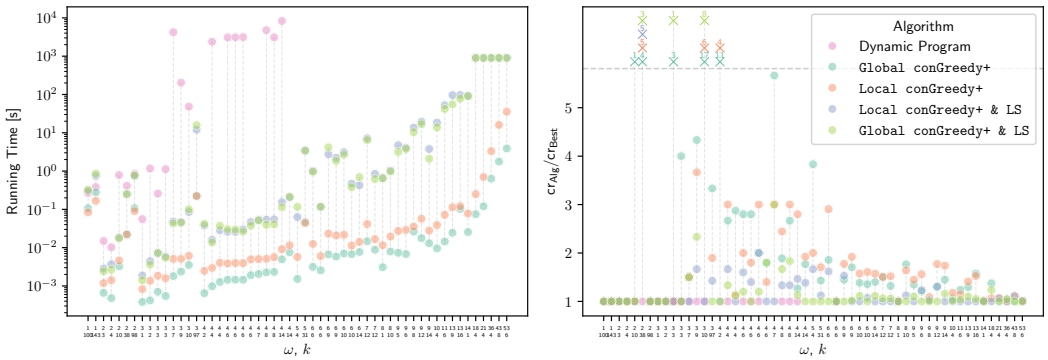

Figure 10 (a) Wall-clock running time ($\log_{10}$-scale) of the heuristics and **(b)** their optimality-ratio compared to the best found solution for different values of ω and number of bags k. We group instances based on ω, sort each group by k, and indicate on the x-axis the value of ω and k in the first and second row, respectively. Vertical lines connect data points for one instance.

expected, `Global conGreedy+` is the fasted heuristic, since it has to compute only one book drawing for the entire graph. Regarding the optimality-ratio, Figure 10b shows that for instances with larger width, `Global conGreedy+` seems to work slightly better than `Local conGreedy+`. We believe that computing a book drawing for each G_i individually could yield situations where the obtained spine orders yield many `t`/`t`-crossings. In particular, we suspect that spine orders that "locally" admit few crossings could "globally" introduce, for example, unexpected "criss-cross" `t`/`t`-crossings (recall Figure 4d). We see the strength of our DP for decompositions of width two or three. For larger width, the high running time becomes impractical. However, small-width instances are often simple enough that all heuristics can solve them to optimality in a few milliseconds. Interestingly, for instances with very large treewidth, the benefit of local search drastically decreases: Despite performing movement operations for the entire fifteen minutes, the obtained solution is, if at all, only negligibly better then the corresponding heuristic without the local search. However, for instances of moderate width, local search considerably improved the initial solution.

8 Concluding Remarks and Future Directions

In this paper, we have initiated the study of visualizing tree and path decompositions of graphs to visually certify their treewidth and pathwidth, respectively. Our experimental study shows that for decompositions of width at most three, we can compute exact crossing-minimal drawings in a few seconds. For decompositions of medium to larger width, the heuristics combined with a local search step seem to be the preferred method. We see the computation of a crossing-minimal drawing where we do not restrict the placement of the child bags or the starting angle inside the disk as a challenging open algorithmic problem. Furthermore, the taxonomy presented Section 3 can be seen as a first step towards exploring the design space for witness drawings of graphs with bounded pathwidth or treewidth. In particular, future work should explore further drawing styles for (i) the graph G_i inside each bag V_i, (ii) tracks between the bags, (iii) the underlying tree (or path) T, and possibilities (iv) to distinguish different vertices other than color. On top of that, a comprehensive design study can also consider (v) the option to not visualize an edge uv in every bag V_i with $u, v \in V_i$ but only

in one such bag, (vi) the algorithmic and visual impact of the structure of the underlying tree/path T, e.g., its number of bags, their maximum degree, or the "niceness" of T, and (vii) further optimization criteria other than the (unweighted) number of crossings.

Additionally, we believe that the exploration of the design space should go hand in hand with the development of further exact and heuristic algorithms and an empirical evaluation of the drawing styles proposed in this article or extracted from the (extended) taxonomy, e.g., with a user study. Finally, designing and computing suitable witness drawings for other width parameters of graphs is an interesting, independent direction for future work.

References

1 Evmorfia N. Argyriou, Michael A. Bekos, Michael Kaufmann, and Antonios Symvonis. On metro-line crossing minimization. *Journal of Graph Algorithms and Applications*, 14(1):75–96, 2010. doi:10.7155/jgaa.00199.

2 Boris Aronov, Muriel Dulieu, and Ferran Hurtado. Witness (delaunay) graphs. *Computational Geometry*, 44(6):329–344, 2011. doi:10.1016/j.comgeo.2011.01.001.

3 Boris Aronov, Muriel Dulieu, and Ferran Hurtado. Witness rectangle graphs. In Frank Dehne, John Iacono, and Jörg-Rüdiger Sack, editors, *Proc 12th International Symposium on Algorithms and Data Structures (WADS'11)*, volume 6844 of *Lecture Notes in Computer Science*, pages 73–85. Springer, 2011. doi:10.1007/978-3-642-22300-6_7.

4 Boris Aronov, Muriel Dulieu, and Ferran Hurtado. Witness gabriel graphs. *Computational Geometry*, 46(7):894–908, 2013. doi:10.1016/j.comgeo.2011.06.004.

5 Boris Aronov, Muriel Dulieu, and Ferran Hurtado. Witness rectangle graphs. *Graphs and Combinatorics*, 30(4):827–846, 2014. doi:10.1007/s00373-013-1316-x.

6 Matthew Asquith, Joachim Gudmundsson, and Damian Merrick. An ILP for the metro-line crossing problem. In James Harland and Prabhu Manyem, editors, *Proc. 14th Computing: The Australasian Theory Symposium (CATS'08)*, volume 77 of *CRPIT*, pages 49–56. Australian Computer Society, 2008. URL: http://crpit.scem.westernsydney.edu.au/abstracts/CRPITV77Asquith.html.

7 Michael J. Bannister and David Eppstein. Crossing minimization for 1-page and 2-page drawings of graphs with bounded treewidth. In Christian Duncan and Antonios Symvonis, editors, *Proc. 22nd International Symposium on Graph Drawing and Network Visualization (GD'14)*, volume 8871 of *Lecture Notes in Computer Science*, pages 210–221. Springer, 2014. doi:10.1007/978-3-662-45803-7_18.

8 Lukas Barth, Benjamin Niedermann, Ignaz Rutter, and Matthias Wolf. A topology-shape-metrics framework for ortho-radial graph drawing. *Discrete & Computational Geometry*, 70(4):1292–1355, 2023. doi:10.1007/s00454-023-00593-y.

9 Michael Baur and Ulrik Brandes. Crossing reduction in circular layouts. In Juraj Hromkovič, Manfred Nagl, and Bernhard Westfechtel, editors, *Proc. 30th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'04)*, volume 3353 of *LNCS*, pages 332–343. Springer Berlin Heidelberg, 2004. doi:10.1007/978-3-540-30559-0_28.

10 Michael A. Bekos, Michael Kaufmann, Katerina Potika, and Antonios Symvonis. Line crossing minimization on metro maps. In Seok-Hee Hong, Takao Nishizeki, and Wu Quan, editors, *Proc. 15th International Symposium on Graph Drawing and Network Visualization (GD'07)*, volume 4875 of *Lecture Notes in Computer Science*, pages 231–242. Springer, 2007. doi:10.1007/978-3-540-77537-9_24.

11 Marc Benkert, Martin Nöllenburg, Takeaki Uno, and Alexander Wolff. Minimizing intra-edge crossings in wiring diagrams and public transportation maps. In Michael Kaufmann and Dorothea Wagner, editors, *Proc. 14th International Symposium on Graph Drawing and Network Visualization (GD'06)*, volume 4372 of *Lecture Notes in Computer Science*, pages 270–281. Springer, 2006. doi:10.1007/978-3-540-70904-6_27.

12 Frank Bernhart and Paul C. Kainen. The book thickness of a graph. *Journal of Combinatorial Theory, Series B*, 27(3):320–331, 1979. doi:10.1016/0095-8956(79)90021-2.

13 Hans L. Bodlaender. Treewidth: Characterizations, applications, and computations. In Fedor V. Fomin, editor, *Proc. 32nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG'06)*, volume 4271 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006. doi:10.1007/11917496_1.

14 Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21(2):358–402, 1996. doi:10.1006/jagm.1996.0049.

15 Annika Bonerath, Martin Nöllenburg, Soeren Terziadis, Markus Wallinger, and Jules Wulms. Boundary labeling in a circular orbit. In Stefan Felsner and Karsten Klein, editors, *Proc. 32nd International Symposium on Graph Drawing and Network Visualization (GD'24)*, volume 320 of *LIPIcs*, pages 22:1–22:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.GD.2024.22.

16 Alvin Chiu, Thomas Depian, David Eppstein, Michael T. Goodrich, and Martin Nöllenburg. Visualizing treewidth, 2025. doi:10.48550/arXiv.2508.19935.

17 Alvin Chiu, Thomas Depian, David Eppstein, Michael T. Goodrich, and Martin Nöllenburg. Visualizing treewidth: Supplementary material, 2025. Source Code; Last accessed: 2025-08-18. doi:10.17605/OSF.IO/QFZ5V.

18 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

19 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

20 Giuseppe Di Battista and Roberto Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*, 15(4):302–318, 1996. doi:10.1007/s004539900017.

21 Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. A survey on graph drawing beyond planarity. *ACM Computing Surveys*, 52(1):4:1–4:37, 2019. doi:10.1145/3301281.

22 Alexander Dobler, Michael Jünger, Paul J. Jünger, Julian Meffert, Petra Mutzel, and Martin Nöllenburg. Revisiting ILP models for exact crossing minimization in storyline drawings. In Stefan Felsner and Karsten Klein, editors, *Proc. 32nd International Symposium on Graph Drawing and Network Visualization (GD'24)*, volume 320 of *LIPIcs*, pages 31:1–31:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.GD.2024.31.

23 Vida Dujmović, David Eppstein, and David R. Wood. Structure of graphs with locally restricted crossings. *SIAM Journal on Discrete Mathematics*, 31(2):805–824, 2017. doi:10.1137/16M1062879.

24 Vida Dujmović, Pat Morin, and David R. Wood. Layout of graphs with bounded tree-width. *SIAM Journal on Computing*, 34(3):553–579, 2005. doi:10.1137/S0097539702416141.

25 Vida Dujmović and David R. Wood. On linear layouts of graphs. *Discrete Mathematics & Theoretical Computer Science*, 6(2):339–358, 2004. doi:10.46298/DMTCS.317.

26 John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gordon Woodhull. Graphviz - open source graph drawing tools. In Petra Mutzel, Michael Jünger, and Sebastian Leipert, editors, *Proc. 9th International Symposium on Graph Drawing and Network Visualization (GD'01)*, volume 2265 of *Lecture Notes in Computer Science*, pages 483–484. Springer, 2001. doi:10.1007/3-540-45848-4_57.

27 David Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27(3):275–291, 2000. doi:10.1007/s004530010020.

28 David Eppstein. Tree decomposition.svg via Wikimedia Commons, 2010. Last accessed: 2025-08-14. URL: https://upload.wikimedia.org/wikipedia/commons/a/a7/Tree_decomposition.svg.

29 David Eppstein. What is ... treewidth? *Notices of the AMS*, 72(2):172–175, 2025. doi:10.1090/noti3043.

30 Martin Fink and Sergey Pupyrev. Metro-line crossing minimization: Hardness, approximations, and tractable cases. In Stephen Wismath and Alexander Wolff, editors, *Proc. 21st International Symposium on Graph Drawing and Network Visualization (GD'13)*, volume 8242 of *Lecture Notes in Computer Science*, pages 328–339. Springer, 2013. doi:10.1007/978-3-319-03841-4_29.

31 Henry Förster, Felix Klesen, Tim Dwyer, Peter Eades, Seok-Hee Hong, Stephen G. Kobourov, Giuseppe Liotta, Kazuo Misue, Fabrizio Montecchiani, Alexander Pastukhov, and Falk Schreiber. GraphTrials: Visual proofs of graph properties. In Stefan Felsner and Karsten Klein, editors, *Proc. 32nd International Symposium on Graph Drawing and Network Visualization (GD'24)*, volume 320 of *LIPIcs*, pages 16:1–16:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.GD.2024.16.

32 Emden R. Gansner and Yehuda Koren. Improved circular layouts. In Michael Kaufmann and Dorothea Wagner, editors, *Proc. 14th International Symposium on Graph Drawing and Network Visualization (GD'06)*, volume 4372 of *Lecture Notes in Computer Science*, pages 386–398. Springer, 2006. doi:10.1007/978-3-540-70904-6_37.

33 Martin Gronemann, Michael Jünger, Frauke Liers, and Francesco Mambelli. Crossing minimization in storyline visualization. In Yifan Hu and Martin Nöllenburg, editors, *Proc. 24th International Symposium on Graph Drawing and Network Visualization (GD'16)*, volume 9801 of *Lecture Notes in Computer Science*, pages 367–381. Springer, 2016. doi:10.1007/978-3-319-50106-2_29.

34 Michael Hoffmann, Lutz Kettner, and Stefan Näher. Two computational geometry libraries: LEDA and CGAL. In *Handbook of Discrete and Computational Geometry*, pages 1799–1831. Chapman and Hall/CRC, 2017.

35 Michael Jünger and Petra Mutzel. *Graph Drawing Software*. Springer, 2012.

36 Jonathan Klawitter, Tamara Mchedlidze, and Martin Nöllenburg. Experimental evaluation of book drawing algorithms. In Fabrizio Frati and Kwan-Liu Ma, editors, *Proc. 25th International Symposium on Graph Drawing and Network Visualization (GD'17)*, volume 10692 of *Lecture Notes in Computer Science*, pages 224–238. Springer, 2017. doi:10.1007/978-3-319-73915-1_19.

37 Ephraim Korach and Nir Solel. Tree-width, path-widt, and cutwidth. *Discrete Applied Mathematics*, 43(1):97–101, 1993. doi:10.1016/0166-218X(93)90171-J.

38 Irina Kostitsyna, Martin Nöllenburg, Valentin Polishchuk, André Schulz, and Darren Strash. On minimizing crossings in storyline visualizations. In Emilio Di Giacomo and Anna Lubiw, editors, *Proc. 23rd International Symposium on Graph Drawing and Network Visualization (GD'15)*, volume 9411 of *Lecture Notes in Computer Science*, pages 192–198. Springer, 2015. doi:10.1007/978-3-319-27261-0_16.

39 Dieter Kratsch, Ross M. McConnell, Kurt Mehlhorn, and Jeremy P. Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. In *Proc. 14th ACM-SIAM Symposium on Discrete Algorithms (SODA'03)*, pages 158–167. ACM/SIAM, 2003. URL: http://dl.acm.org/citation.cfm?id=644108.644137.

40 Dieter Kratsch, Ross M. McConnell, Kurt Mehlhorn, and Jeremy P. Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. *SIAM Journal on Computing*, 36(2):326–353, 2006. doi:10.1137/S0097539703437855.

41 William J. Lenhart and Giuseppe Liotta. Mutual witness Gabriel drawings of complete bipartite graphs. *Theoretical Computer Science*, 974:114115, 2023. doi:10.1016/j.tcs.2023.114115.

42 William J. Lenhart and Giuseppe Liotta. Mutual witness gabriel drawings of complete bipartite graphs. In Patrizio Angelini and Reinhard von Hanxleden, editors, *Proc. 30th Proc. 21st International Symposium on Graph Drawing and Network Visualization (GD'22)*, volume 13764 of *Lecture Notes in Computer Science*, pages 25–39. Springer, 2023. doi:10.1007/978-3-031-22203-0_3.

43 Silviu Maniu, Pierre Senellart, and Suraj Jog. An experimental study of the treewidth of real-world graph data. In Pablo Barceló and Marco Calautti, editors, *Proc. 22nd International*

Conference on Database Theory (ICDT'19), volume 127 of *LIPIcs*, pages 12:1–12:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.ICDT.2019.12.

44 Sumio Masuda, Toshinobu Kashiwabara, Kazuo Nakajima, and Toshio Fujisawa. On the NP-completeness of a computer network layout problem. In *Proc. IEEE Intl. Symposium on Circuits and Systems*, pages 292–295, 1987.

45 Jiři Matoušek and Robin Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics*, 108(1–3):343–364, 1992. doi:10.1016/0012-365x(92)90687-b.

46 Ross M. McConnell, Kurt Mehlhorn, Stefan Näher, and Pascal Schweitzer. Certifying algorithms. *Computer Science Review*, 5(2):119–161, 2011. doi:10.1016/j.cosrev.2010.09.009.

47 Kurt Mehlhorn, Stefan Näher, Thomas Schilz, Stefan Schirra, Michael Seel, Raimund Seidel, and Christian Uhrig. Checking geometric programs or verification of geometric structures. In Sue Whitesides, editor, *Proc. 12th International Symposium on Computational Geometry (SoCG'96)*, SCG '96, pages 159–165. ACM, 1996. doi:10.1145/237218.237344.

48 Kurt Mehlhorn, Stefan Näher, Michael Seel, Raimund Seidel, Thomas Schilz, Stefan Schirra, and Christian Uhrig. Checking geometric programs or verification of geometric structures. *Computational Geometry*, 12(1-2):85–103, 1999. doi:10.1016/S0925-7721(98)00036-4.

49 Petra Mutzel. The SPQR-tree data structure in graph drawing. In Jos C. M. Baeten, Jan Karel Lenstra, Joachim Parrow, and Gerhard J. Woeginger, editors, *Proc. 30th International Colloquium on Automata, Languages and Programming (ICALP'03)*, volume 2719 of *Lecture Notes in Computer Science*, pages 34–46. Springer, 2003. doi:10.1007/3-540-45061-0_4.

50 Martin Nöllenburg. An improved algorithm for the metro-line crossing minimization problem. In David Eppstein and Emden R. Gansner, editors, *Proc. 17th International Symposium on Graph Drawing and Network Visualization (GD'09)*, volume 5849 of *Lecture Notes in Computer Science*, pages 381–392. Springer, 2009. doi:10.1007/978-3-642-11805-0_36.

51 Sergey Pupyrev, Lev Nachmanson, Sergey Bereg, and Alexander E. Holroyd. Edge routing with ordered bundles. In Marc J. van Kreveld and Bettina Speckmann, editors, *Proc. 19th International Symposium on Graph Drawing and Network Visualization (GD'11)*, volume 7034 of *Lecture Notes in Computer Science*, pages 136–147. Springer, 2012. doi:10.1007/978-3-642-25878-7_14.

52 Sergey Pupyrev, Lev Nachmanson, Sergey Bereg, and Alexander E. Holroyd. Edge routing with ordered bundles. *Computational Geometry*, 52:18–33, 2016. doi:10.1016/j.comgeo.2015.10.005.

53 Roberto Tamassia, editor. *Handbook of Graph Drawing and Visualization*. Chapman and Hall/CRC, 2013.

54 Thomas C. van Dijk, Martin Fink, Norbert Fischer, Fabian Lipp, Peter Markfelder, Alexander Ravsky, Subhash Suri, and Alexander Wolff. Block crossings in storyline visualizations. *Journal of Graph Algorithms and Applications*, 21(5):873–913, 2017. doi:10.7155/jgaa.00443.

55 Thomas C. van Dijk, Fabian Lipp, Peter Markfelder, and Alexander Wolff. Computing storyline visualizations with few block crossings. In Fabrizio Frati and Kwan-Liu Ma, editors, *Proc. 25th International Symposium on Graph Drawing and Network Visualization (GD'17)*, volume 10692 of *Lecture Notes in Computer Science*, pages 365–378. Springer, 2017. doi:10.1007/978-3-319-73915-1_29.

56 Martin Wattenberg. Arc diagrams: Visualizing structure in strings. In Pak Chung Wong and Keith Andrews, editors, *Proc. 8th IEEE Information Visualization Conference (InfoVis'02)*, pages 110–116. IEEE Computer Society, 2002. doi:10.1109/INFVIS.2002.1173155.