Minimum Height Drawings of Ordered Trees in Polynomial Time: Homotopy Height of Tree Duals

Tim Ophelders
Department of Information and Computing Science, Utrecht University, The Netherlands
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Salman Parsa
Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT, USA

Abstract

We consider drawings of graphs in the plane in which vertices are assigned distinct points in the plane and edges are drawn as simple curves connecting the vertices and such that the edges intersect only at their common endpoints. There is an intuitive quality measure for drawings of a graph that measures the height of a drawing $\phi\colon G \hookrightarrow \mathbb{R}^2$ as follows. For a vertical line ℓ in $\mathbb{R}^2$, let the height of ℓ be the cardinality of the set $\ell \cap \phi(G)$. The height of a drawing of G is the maximum height over all vertical lines. In this paper, instead of abstract graphs, we fix a drawing and consider plane graphs. In other words, we are looking for a homeomorphism of the plane that minimizes the height of the resulting drawing. This problem is equivalent to the homotopy height problem in the plane, and the homotopic Fréchet distance problem. These problems were recently shown to lie in NP, but no polynomial-time algorithm or NP-hardness proof has been found since their formulation in 2009. We present the first polynomial-time algorithm for drawing trees with optimal height. This corresponds to a polynomial-time algorithm for the homotopy height where the triangulation has only one vertex (that is, a set of loops incident to a single vertex), so that its dual is a tree.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Graph drawing, homotopy height

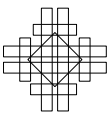
Digital Object Identifier 10.4230/LIPIcs.SoCG.2022.55

Related Version *Full Version*: `https://arxiv.org/abs/2203.08364`

Funding *Tim Ophelders*: This author was supported by the Dutch Research Council (NWO) under project no. VI.Veni.212.260.
Salman Parsa: This author was funded in part by the SLU Research Institute and by NSF grant CCF-1614562.

1 Introduction

A tree T is called an *ordered tree* if for each vertex, a fixed cyclic ordering of its incident edges is given. Let T be an ordered tree and let $f\colon |T| \to \mathbb{R}^2$ be a drawing of the tree, that is, a continuous injection from the underlying topological space of the tree to the plane, in which the clockwise order of edges around each vertex is as prescribed. Any ordered tree can be recovered from any of its drawings up to degree 2 nodes. Any two drawings of the same ordered tree can be obtained from one another using an orientation-preserving homeomorphism of the plane. We are interested in drawings that minimize the height in the following sense. Given a drawing ϕ and a vertical line ℓ, the *height* of the line ℓ is defined as $H(\ell) := |\phi(T) \cap \ell|$. That is, the number of times that the line ℓ intersects the drawing, where vertical segments count as infinitely many intersections. The problem of drawing a tree T with optimal height asks for a drawing $\phi\colon |T| \to \mathbb{R}^2$ that minimizes the maximum height over all vertical lines. We call such a drawing an *optimal height drawing*. We emphasize that our drawings are not necessarily straight-line. In fact, there exist instances for which any



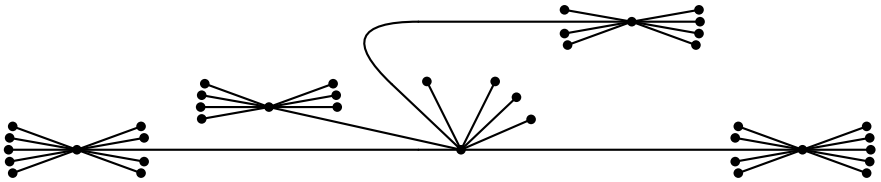


Figure 1 A bend is necessary in any drawing with height 5.

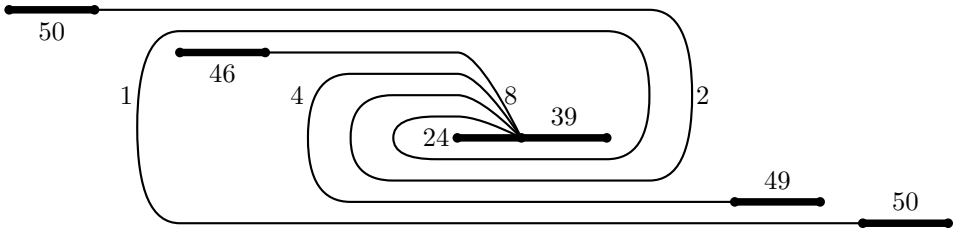


Figure 2 Spirals (e.g. the edge with weight 1) may be necessary to draw weighted trees optimally.

optimal drawing requires a bend in some edge. An example is given in Figure 1. One can check that any optimal drawing of this tree requires a bend in some edge. Although we will consider only unweighted trees, the definition of height naturally extends to edge-weighted graphs. Already in the case of weighted trees with only one vertex of degree at least three, an optimal drawing might even require an edge to form a spiral. Figure 2 depicts an instance whose optimal drawing requires a spiral according to a computer-assisted enumeration of its drawings. We do not know whether unweighted trees also require spiraling edges.

The optimal height drawing of graphs is related to two significant classes of problems in computer science, and in particular, computational geometry and topology. If, instead of ordered trees, we take (unordered) trees and allow edges to cross in the output drawing, we obtain the classical min-cut linear arrangement problem. This problem is well-studied [7, 11, 14] and Yannakakis [15] presented an $O(n \log n)$ time algorithm for drawing trees with optimal height in this sense. Of course, optimal drawings with straight-line edges always exist in this setting. On the other hand, it is known that the graph version as well as the weighted tree version [13] of the same problem is NP-hard. Since the trees corresponding to the reduction can be drawn optimally without self-intersection, it follows that optimal height drawing of unordered weighted trees is also NP-hard. All the mentioned problems lie in NP.

The optimal height graph drawing problem also shows up as a special case of an important open problem in computational geometry and topology called the homotopy height problem [2, 3, 5, 6, 10]. In this context, a homotopy corresponds to a one-parameter family of curves γ_i ($i \in [0, 1]$) that sweeps a surface in a continuous way, where γ_0 and γ_1 are part of the input. Roughly speaking, the homotopy height problem considers a surface homeomorphic to a sphere, disk, or annulus, endowed with a metric, and asks for a homotopy of curves that sweeps the surface in such a way that the longest curve γ_i is as short as possible. For a perfectly round sphere, the homotopy height is the length of its equator. For the purpose of computation, discrete versions of the problem have been considered, where the surface is endowed with a cellular decomposition, and the lengths of curves are measured by the number of intersections with cell boundaries. Each curve in general position with the cellular decomposition can be represented as a walk on the dual graph of the decomposition. The vertices of the dual graph are represented geometrically as representative points of cells, and edges of the dual graph correspond to pieces of cell boundaries shared by two cells. As a curve

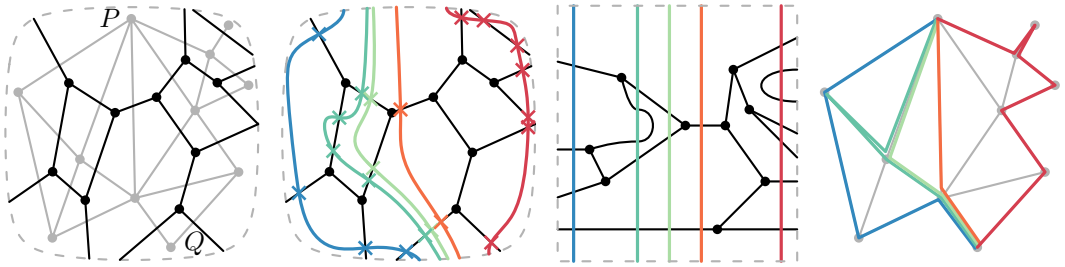


Figure 3 Left: a cellular decomposition of a disk (black) and its dual. Middle: some curves of a homotopy whose curves start at P and end at Q, and a homeomorphism of the disk that sends the curves to vertical lines. Right: the corresponding walks in the dual graph.

sweeps over the surface, it can sweep over vertices of the dual graph (resulting in a face flip), or create or remove pairs of intersections with edges of the embedded graph (resulting in a spike or unspike). Figure 3 illustrates a dual graph (of a cellular decomposition) with vertices P and Q, and a homotopy through curves γ_i connecting P to Q. If the cell decomposition contains exactly one vertex, then its dual is a tree and the problem of homotopy height becomes equivalent to drawing trees with optimal height (In this case, the starting and ending curves are nested circles in the unbounded face so that the curves sweep an annulus).

Although homotopy height admits an efficient $O(\log n)$-approximation algorithm [10], its exact computation appears to be very challenging. In fact, it was only recently shown to lie in the complexity class NP [5] in the setting of edge-weighted graphs. If the curves at the start and end of a homotopy are disjoint and shortest curves, it is known that there exists an optimal homotopy that sweeps the surface in a monotone fashion [4]. Homotopy height is closely related to other important graph parameters [2].

The duality relation between graph drawings and homotopy height is depicted in Figure 3.

In this paper instead of graphs we consider plane trees or ordered trees. We present the first polynomial-time algorithm for the optimal height drawing of unit weight plane trees. Our results give a polynomial algorithm for the homotopy height of unit-weight one-vertex (multi-)graphs. This might point to the possibility that the problem for the general graphs is also polynomial. However, already in our restricted setting, the algorithm is quite involved and does not have a clear extension to general graphs.

Although our notion of height has frequently been studied in recent years, there exist related parameters of graph drawings that also quantify some notion of height [1, 2, 12].

2 Background and terminology

2.1 Drawings and local disks

Drawings. Formally we work with *plane trees* instead of ordered trees. This is just some reasonable, e.g. piecewise-linear, drawing $g\colon T \to \mathbb{R}^2$ of a finite tree T in the Euclidean plane. This plane drawing is fixed once and for all for any ordered tree T and respects the given ordering around each vertex. In order to distinguish the Euclidean plane containing the drawing g we use the symbol Π for this plane, so that $g(T) \subset \Pi$.

Convention. With a slight abuse of notation, we will not distinguish between T and its embedding $g(T) \subset \Pi$ in the plane. We use the words edge and path for edges and simple curves on T exclusively, and reserve the word curve for curves in the drawing plane (the plane to which Π is mapped).

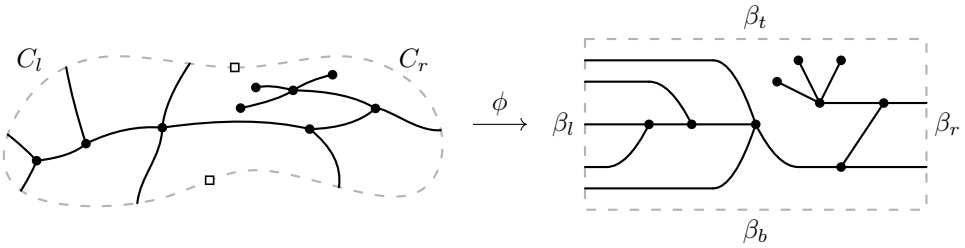


Figure 4 A local disk and a drawing.

A *drawing* ϕ of a tree T, is a continuous injective function mapping Π into $\mathbb{R}^2$. We consider only drawings ϕ in which the image of every edge e is piecewise-linear, and such that every vertical line intersects the drawing in a finite number of points. It is not difficult to see that this restriction does not affect the optimal height of the drawing. In our figures, for aesthetic purposes, we often draw edges as smooth curves.

Let $E = E(T)$, $V = V(T)$ denote the set of edges and vertices of T. We always denote the number of vertices by n. By $H(\phi)$ we denote the *height* of the drawing ϕ. That is, the maximum number of points of the drawing on a vertical line.

Local disks. Let $D \subset \Pi(T)$ be a topological disk in the plane in which T is drawn. We denote the boundary of D by ∂D. Let $T_D = T \cap D$ and assume T_D is connected. We say that an edge $e \in E$ is a *boundary edge* of D if $e \cap \partial D \neq \emptyset$. We call an edge *internal* if it lies in the interior of D. We denote by $B(D)$ the set of boundary edges of D. Let $\partial D = C_l \cup C_r$ where $C_l \cap C_r = \{p_N, p_S\}$ is a set of two points, where none is in T. Intuitively, we think of C_l and C_r as the left and right boundary of D. This "partition" of ∂D divides the set of boundary edges $B(D)$ into *left* and *right boundary edges* $B(D) = B_L(D) \sqcup B_R(D)$. We call $(D, B_L(D), B_R(D))$ a *local disk*.

A *drawing of a local disk* (D, B_L, B_R) is a homeomorphism $\phi\colon D \to Q$ onto a rectangle Q with edges $(\beta_l, \beta_t, \beta_r, \beta_d)$, such that under ϕ, the boundary edges in B_L intersect β_l, and those in B_R intersect β_r and $\phi(T_D) \cap (\beta_t \cup \beta_r) = \emptyset$. See Figure 4. Note that we can select a local disk whose interior contains the whole tree, such that $T_D = T$ and there are no boundary edges. The height of the left (right) boundary in any drawing is the number of left (right) boundary edges of the local disk, and we call this number as the *left (right) boundary height*. When the two boundary heights are equal we simply say boundary height.

The move sequence of a drawing. Consider sweeping a vertical line over a drawing of T (or the interior of a local disk). The sweep line encounters three types of events: left bends (points interior to edges of T whose x-coordinate in the drawing is locally minimal), right bends (symmetric to left bends), and vertices. We will refer to these events as *moves*, and the corresponding point of T as its *location* (i.e. the vertex corresponding to a vertex move, or the point interior to the edge corresponding to the bend move). We assume that all bends and vertex moves occur at distinct x-coordinates, and refer to the left-to-right sequence of moves of a drawing as its *move sequence*.

2.2 Cuts and shortcuts

Let D be a local disk. By a *cut* in the local disk (D, B_L, B_R) we mean the sequence of edges crossed by a curve that connects p_N to p_S, where p_N and p_S are some two points giving rise to the local disk (D, B_L, B_R) (an edge might repeat consecutively in the sequence). Some

times we refer to the curve itself as a cut. Note that the same local disk can be defined with many such pair of points but this choice is not important. The *length* (or *height*) of a cut is the number of edges in it (counted with repetition), or the number of intersections of the curve with the tree T_D. A cut C is a *shortcut* if its length is smallest over all cuts of D. For the proof of the following lemma we refer to [5, Lemma 4.2].

▶ **Lemma 1** (Pausing at a shortcut). *Let D be a local disk, $\phi\colon D \to Q$ a drawing and C a shortcut in D. There is a drawing ϕ' of height less than or equal to the height of ϕ in which there is a vertical line defining the cut C. Moreover, vertical lines of ϕ that are disjoint from C are unaffected and appear in ϕ'.*

We say that the drawing ϕ can *pause* at the shortcut C, resulting in the drawing ϕ'. When a cut C is vertical in an optimal drawing and each sub-disk cut by C contains a connected part of T_D, then C subdivides the problem into two sub-problems whose optimal drawings can easily be merged to form an optimal drawing of the original disk.

3 Overview of the algorithm

Our main result is an algorithm for computing optimal drawings of plane trees. This algorithm is a dynamic program which in a high level works as follows. Each cell of the dynamic programming table represents a local disk and stores the optimal height of that disk (or an optimal drawing, if an optimal drawing is to be computed). The local disks represented by the cells are of two special types: spine disks and skew spine disks (defined in Section 6). These disks essentially are local disks that cannot be cut by shortcuts. Row m of the table stores all spine or skew spine disks with exactly m interior vertices. For $m > 1$, row m of the table is built using the information in lower rows in two phases. The first phase constructs all possible m-vertex spine and skew spine disks. The second phase computes the height of an optimal height drawing for each of the computed disks of row m (or computes a drawing, if the the drawing is needed). The computed optimal height (or optimal drawing) will be stored again in the table. The base of the table consists of spine or skew spine disks with a single interior vertex. The possibilities for the decomposition of a single spine disk or skew spine disk into such disks with fewer vertices are shown in Figures 10 and 11. With this description, a final optimal drawing consists of drawings in Figures 10 and 11 nested inside each other, and where the deepest level is a single vertex disk. A trapezoid (skew spine disk) will fit into a trapezoid and a rectangle into a rectangle (spine disk).

There are two ingredients in the proof of correctness. First, we show in Proposition 6 that any (sufficiently general) drawing can be turned, without increasing the height, into a drawing that has a hierarchical structure. The root of this structure tree is a spine disk containing the whole tree (with zero boundary edges). The nodes of the structure tree are (skew) spine disks. Each node is cut essentially into a collection of sub-disks, using shortcuts that are made vertical via pausing. These sub-disks are (skew) spine disks that form the children of the node. Each node, has one of polynomially many possibilities for the decomposition, depicted in Figures 10 and 11. The spine disks corresponding to leaves of any structure tree are single-vertex local disks and thus trivial to draw optimally. In brief, any drawing can be turned into one which has a tree structure of spine and skew spine disks without increasing the height.

The second ingredient of the proof of correctness is a proof that there exists some optimal drawing such that a super-set of all the (skew) spine disks in its tree structure can be enumerated in polynomial time. For this purpose, we define a quality measure for a drawing. To rule out pathological drawings and simplify our arguments, we need to consider simplified

drawings which are the result of applying simplification moves of Figure 6. We also consider balanced drawings, which are ones where the height of the lines on both sides of (and very close to) any vertex differ by at most one. Among all the optimal drawings, we take the drawing which is simplified, balanced, and maximizes our quality measure. Lemma 7 asserts that such an optimized drawing has itself a tree structure of (skew) spine disks. The tree structure of a drawing which optimizes a slightly stronger measure, namely the secondary quality, is called a fat structure. Proposition 10 characterizes the spine disks that can appear in a fat structure. This description allows us to easily enumerate all possible spine disks that can appear in a fat structure in polynomial time and only store the (skew) spine disks in our table that conform to this characterization. This will result in a polynomial-sized table and hence a polynomial algorithm. Omitted proofs can be found in the full version.

4 Simplifying the drawings

Let ϕ be a drawing of a local disk D and $T = T_D$. We label the left (resp. right) bends of ϕ as either *stuck* or not, depending on whether the bend encloses the next (resp. previous) move. Figure 5 illustrates the two possible reasons for a right bend to be stuck. Two consecutive moves of a drawing may admit a simplification (of the drawing) that replaces the two sequences by a simpler sequence of moves without increasing the height of the drawing. For each of these simplifications, either the first move is a non-stuck left bend, or the second move is a non-stuck right bend. We explain the types of simplifications involving a non-stuck right bend (see Figure 6), the types involving a non-stuck left bend are symmetric. As mentioned, the second move is a non-stuck right bend, so we distinguish cases based on the first move of the pair.

Stuck slide. In this case, the first move is a stuck right bend. The non-stuck right bend does not enclose the stuck right bend (otherwise it would also be stuck). Exchanging the order of the two bends ensures that neither of the resulting bends are stuck.

Bend-bend (resp. vertex-bend) separation. The first move is a left bend (resp. vertex) that is not connected to the right bend. We exchange the moves, reducing the height of the line in between.

Bend-bend cancellation. The first move is a left bend that is connected to the right bend. We replace the bends by an x-monotone curve, reducing the number of bends.

Vertex-bend cancellation. The first move is a vertex that is connected to the right bend. We replace the bend by an x-monotone curve, reducing the number of bends. We call a vertex-bend cancellation *strong* if the simplification does not decrease the absolute difference between the number of edges incident to the left and right of the vertex.

We say that a drawing ϕ is *strongly simplified* if no simplification move is possible, and *simplified* if only strong simplification moves are possible.

We say that ϕ is *balanced* if for any vertex v, the heights of the vertical lines immediately to the left and right of v are equal if the degree of v is even, and differ by 1 if the degree of v is odd. Balanced drawings will be useful for our algorithms. However, strong vertex-bend cancellations may make vertices less balanced.

Figure 5 A right bend (marked) stuck around a vertex (left) or stuck around a bend (right). The bold line represents a bundle of arbitrarily many edges incident to the vertex.

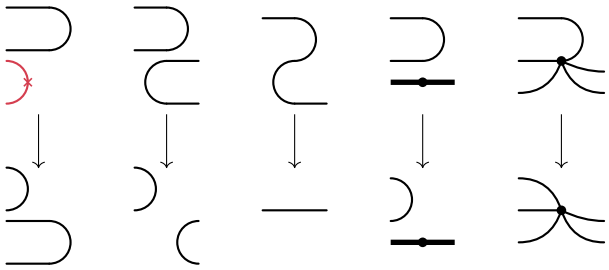


Figure 6 Left to right: stuck slide, bend-bend separation, bend-bend cancellation, vertex-bend separation, (strong) vertex-bend cancellation.

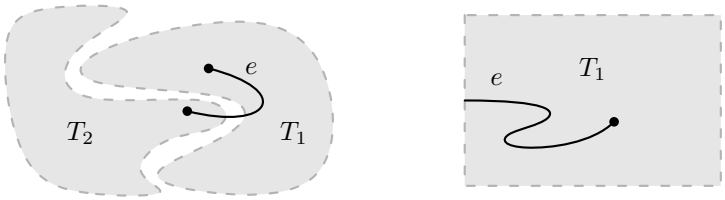


Figure 7 Two local disks (containing sub-trees T_1 and T_2) with a single boundary edge e (left). An exposed drawing of the sub-tree T_1 with anchor edge e (right).

▶ **Lemma 2.** *If there is a drawing ϕ of height H of a local disk D, then there exists a balanced simplified drawing of D of height at most H with a bounded number of moves.*

▶ **Lemma 3.** *Any simplified drawing of height H of a local disk D with n vertices has at most $(H+1)n$ moves if $n > 0$, and at most H moves if $n = 0$.*

▶ **Observation 4.** *Let ϕ be a balanced drawing of the local disk D. Then applying all possible non-strong simplifying moves to h keeps the drawing balanced.*

5 Bubbling a sub-tree

Let e be an edge of a tree T. There are two sub-trees T_1, T_2 of T that result from removing e. For $i \in \{1, 2\}$, we call the rooted trees $T_i = T_i(e)$, with the root chosen to be the endpoint of e in T_i, the *rooted sub-trees anchored via the edge* e and the edge e the *anchor edge* of the rooted tree T_i. We call the endpoint of e which is not the root of T_i the *anchor vertex* of T_i. The *exposed height* of the sub-tree $T_i \cup \{e\}$, denoted $eH(T_i, e)$, is the height of the optimal height drawing of a local disk containing T_i in its interior and such that the anchor edge e is the single boundary edge, see Figure 7. We call such a drawing of T_i an *exposed drawing* of the sub-tree T_i with respect to e.

Let P be a simple path (possibly of length[1] 0) in T. The *neighborhood* of P, denoted $N(P)$, is the subset of vertices of T not in P which are connected in T to some vertex of P by an edge. We say that a rooted sub-tree $T'' \subset T$ is a sub-tree *anchored at* P if $V(T'') \subset V(T) - V(P)$ and the root of T'' is in $N(P)$. It follows that, for any P, the edge-set of T is partitioned into three sets, the edges of P, the edges of sub-trees anchored at P, and the anchor edges incident to P.

[1] The length of a path is the number of its edges.

We call an exposed drawing of a sub-tree T' in a drawing of T *a bubble* if the strip of the plane containing this exposed drawing contains no moves of the rest of the drawing. In other words, we can compress the exposed drawing of T' into a drawing inside an arbitrary small bubble, without affecting the height of the drawing of T.

Let ϕ be a drawing of the tree T and let $T' \subset T$ be a sub-tree anchored using an edge e to a vertex v. We say that ϕ' is obtained by *bubbling the sub-tree* T' *at the point* $t \in \mathbb{R}^2$ if ϕ' is such that i) $H(\phi') \leq H(\phi)$, ii) T' is drawn in a bubble with e the boundary edge and t being the point of e on the boundary, iii) the drawing is changed only over T' and the edge e, and iv) the ϕ'-image of e is contained in ϕ-image of $T' \cup e$. See Figure 8 for an example, where $t = t_r$. One of our main observations is that bubbling is always possible at a suitable t.

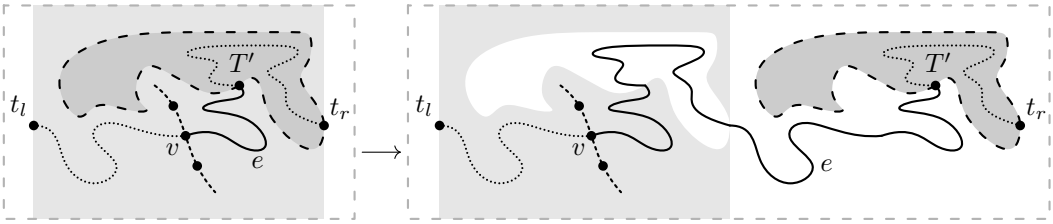


Figure 8 Bubbling the sub-tree T', t_l and t_r are locations of extreme moves in the given drawing.

▶ **Lemma 5.** *Let T be a tree and h be a drawing of T. Let T' a sub-tree anchored at a vertex v. Let t_l and t_r be the points of T which have the smallest and the largest x-coordinate, respectively. We can assume these points are unique. If T' contains exactly one of t_l and t_r, then T' can be bubbled at that point.*

6 Spine disks

Let D be a local disk and $B(D)$ the set of its boundary edges. Recall that the boundary edges of a local disk are divided into left boundary edges, $B_L(D)$, and right boundary edges, $B_R(D)$. Let $e_l \in B_L(D)$ and $e_r \in B_R(D)$. We say that e_l and e_r are *opposite* one another if they are incident to the same vertex in the interior of D. We call a local disk a *spine disk* with *spine* P, if all of the following hold:

1. P is a simple path in T_D, such that every boundary edge is incident to a vertex of P, and P is interior-disjoint from boundary edges of T_D.
2. There is a bijection $\alpha\colon B_L(D) \to B_R(D)$ such that each e is opposite $\alpha(e)$.
3. If P has at least two vertices, there are boundary edges incident to its extremal vertices.

If there are no boundary edges, we let P be an arbitrary vertex, so that P is always defined. Therefore a local disk that contains all of the input tree T and P chosen to be any vertex is a spine disk.

A *skew spine disk* is a spine disk to which a new boundary edge incident to some vertex of P is added. It follows that the height of one boundary line of any drawing of a skew spine disk is one more than the height of the other boundary line. We call a (skew) spine disk a *vertex disk* if there is a single vertex in its interior. Figure 9 shows an optimal drawing of a tree and a "decomposition" of the drawing into spine (rectangle) and skew spine disks (trapezoids). Note that a skew spine disk with a single boundary edge is a bubble. All skew spine disks in Figure 9 are bubbles.

6.1 Spine decomposition

We introduce some terminology before stating one of our main propositions. Let D be a local disk and let C be a collection of disjoint shortcuts (combinatorially distinct from the left and the right boundary lines) in D, and let C cut the disk D into disks $D_1, \ldots, D_m$. According to Lemma 1, an optimal drawing ϕ of D can be obtained by gluing optimal drawings ϕ_i of the D_i, $i = 1, \ldots, m$. Then we say ϕ is obtained by *merging* the drawings $\phi_1, \ldots, \phi_m$. In our schematics, we draw a rectangle for a spine disk and a trapezoid for a skew disk. The shorter side of the trapezoid has one less boundary edge than the long side. A vertex inside a rectangle or a trapezoid indicates a vertex disk. A thick black line is a collection of parallel lines whose number is indicated. A *pipe* in a drawing bounded by two vertical lines l and r is a subpath of an edge of T drawn as an x-monotone curve between these lines. Observe that a pipe can always be drawn as a straight line connecting the lines l and r.

If D is a vertex spine disk or vertex skew spine disk then there is a trivial, straight-line, optimal drawing of D. These disks form the building blocks of our drawings. The following proposition shows how more complicated (skew) spine disks can be decomposed into less complicated ones and eventually into vertex (spine) disks.

▶ **Proposition 6** (Spine Decomposition). *Let D be a spine (resp. skew spine) local disk with $b \geq 0$ boundary edges on one side and b (resp. $b+1$) boundary edges on the other side. If D is not a vertex disk and D has a drawing of height H, then D has a drawing of height at most H that can be decomposed as one of the cases of Figure 10 (resp. 11), up to horizontal and vertical reflection. In these drawings m, a_i, c_j are non-negative integers.*

6.2 Structure tree

Recursive applications of Proposition 6 to an optimal-height drawing of a spine or a skew spine disk, for instance one containing all of T, result in an optimal drawing that has a hierarchical structure. Any node in the hierarchy is a spine or skew spine disk, and a node is decomposed into its children using one of the possibilities of Proposition 6. The leaves of the hierarchy are vertex disks. We call this hierarchy a *structure tree* of the optimal drawing. We call a drawing which has such a structure tree a *structured drawing*. For instance the drawings output by Proposition 6 are structured drawings.

7 Optimizing the optimal-height drawings

In this section, we first define the quality of drawings. We then consider those optimal drawings that maximize this quality measure.

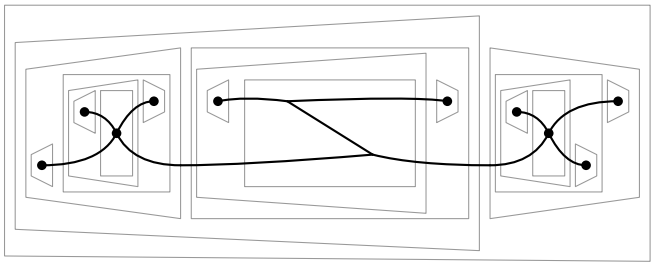


Figure 9 Spine and skew spine disks.

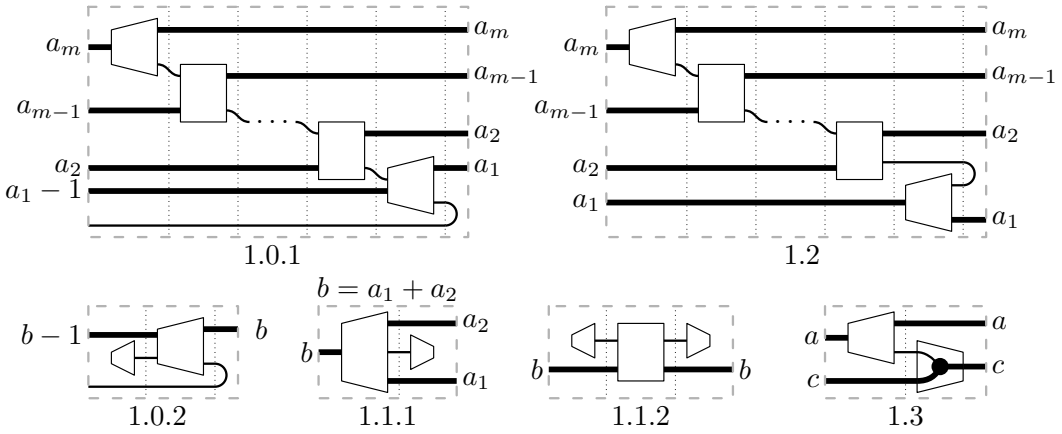


Figure 10 Decomposition of spine disks. Thick lines indicate bundles of parallel edges. The number of parallel edges in bundles are indicated by the labels on the sides. The values a_i can be 0. Rectangles indicate spine disks and trapezoids indicate skew spine disks. A black dot indicates a vertex disk.

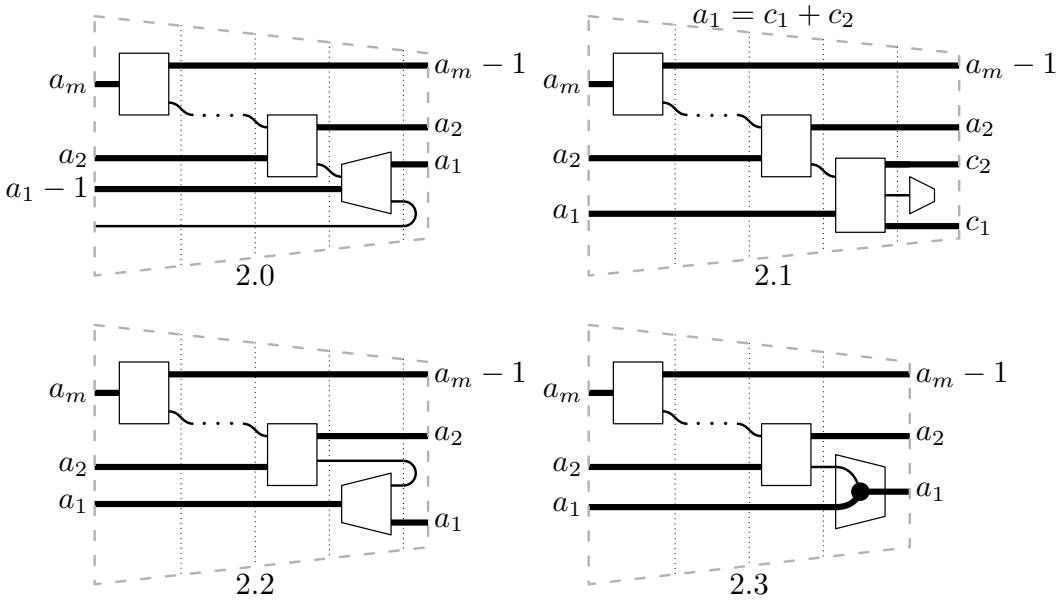


Figure 11 Decomposition of skew spine disks.

7.1 Quality of a drawing

Let ϕ be any drawing of a local disk D. We denote by $\Lambda' = \Lambda'(\phi) = \{\lambda_i\}$ the set of combinatorially distinct vertical lines in the plane (that lie in general position with the drawing). That is, the strip S_i bounded by λ_i and λ_{i+1}, after removing pipes, is either: i) a vertex move, that is, contains a single vertex and no bends, or ii) contains a single bend and no vertices. Such a set of vertical lines can be chosen for any drawing h in general position. Consider a strip S_{ij} bounded by λ_i and λ_j. If S_{ij} contains only pipes we remove λ_i or λ_j (whichever is to the right of the other) and all the lines in between form Λ', and repeat this operation. Let $\Lambda = \Lambda(\phi)$ be the remaining set of vertical lines. We again consider strips S_{ij} bounded by λ_i and λ_j in Λ. If after removing pipes from the strip S_{ij} it becomes a bubble, spine disk, skew spine disk, or bend, we respectively say that S_{ij} is a bubble, spine disk, skew spine disk or bend. Recall that a bubble is a special type of skew spine disk with only one boundary edge in one side.

Note that bubbles are either disjoint or nested and therefore give rise to a hierarchical structure. We say that a vertex v is of *depth* d if it is contained in exactly d bubbles. That is, there are exactly d strips, bounded by the lines of Λ, that contain the vertex and that are bubbles. We define the *depth* of a bubble and a (skew) spine disk analogously to depth of a vertex to be the number of bubbles that properly contain them.

We say that a line $\lambda \in \Lambda(\phi)$ is of depth i if it is contained in exactly i strips which are bubbles, and in none of them it is a boundary. For instance, lines of depth 0 do not cut any bubble. Let $\Lambda_i = \Lambda_i(\phi)$ denote the set of lines of depth i. Let $\Lambda_{i,j} = \Lambda_{i,j}(\phi) \subset \Lambda_i$ be the set of lines of height j and depth i, and let $\delta_{i,j}(\phi) = |\Lambda_{i,j}|$ be the number of lines of depth i and height j. Note that $\delta_{i,j}(\phi)$ can be 0. Let $\Delta_i(\phi)$ be the sequence $(\delta_{i,0}(\phi), \delta_{i,1}(\phi), \ldots)$, and define the *quality* of the drawing ϕ as

$$Q(\phi) = (\Delta_0(\phi), \Delta_1(\phi), \ldots).$$

For two drawings ϕ and ϕ', we compare their qualities $Q(\phi)$ and $Q(\phi')$ lexicographically, where we also compare the sequences $\Delta_i(\phi)$ to $\Delta_i(\phi')$ lexicographically. Specifically, a drawing of maximum quality maximizes the depth sequences Δ_i from left to right. That is, we are interested in the drawings where the sequence Δ_0 is maximized, and among these the sequences where Δ_1 is maximized, and so on. We emphasize that maximizing the quality does not necessarily minimize the height of the drawing. Instead, we merely use the quality measure to reduce the search space for minimum height drawings.

There remains still some arbitrariness in optimal drawings with maximum quality. For instance, a star with $2k$ leaves and a central vertex can be drawn with optimal height and with maximum quality in an exponentially many different ways, giving rise to exponentially many spine disks, by changing the order of the vertices. We get rid of these choices using the notion of secondary quality to be defined in Section 7.3.

By Lemma 2, there exists an optimal simplified and balanced drawing of any local disk D, and by Lemma 3, $(H+1)n$ is an upper bound on the complexities of simplified drawings with height H. Therefore, the set of quality sequences of the set of all optimal, balanced and simplified drawings of a local disk is non-empty. Moreover, each quality sequence for such a drawing consists of at most $H(H+1)n$ terms $\delta_{i,j}$, since the depth is at most the number of lines in Λ and each depth-sequence Δ_i contains at most $H = O(n)$ different height values.

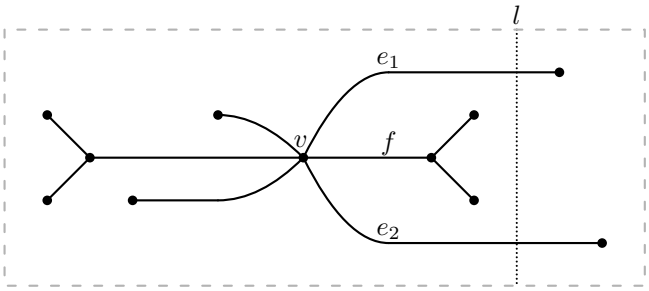


Figure 12 The edge f is sandwiched between e_1 and e_2 with respect to l.

7.2 Properties of drawings with maximum quality

Since the dynamic program only constructs structured drawings, we need to argue that the maximum quality drawing is structured.

▶ **Lemma 7.** *Let ϕ be a simplified, balanced drawing of a spine disk D that has maximum quality $Q(\phi)$ over all drawings with the same height as ϕ. Then ϕ is a structured drawing.*

Recall that for a path P the set of anchor edges, $A(P)$, is the set of edges which have exactly one endpoint on the path P. Also, the set of anchor edges of a spine disk D, $A(D)$, is the set of anchor edges of the spine path of D. Let D be a (balanced) spine disk with $2b$ boundary edges and $e \in A(D)$ be an anchor edge of D. Let $H(D)$ denote, as always, the optimal height of the disk D and $eH(T', e')$ denote the optimal exposed height of a sub-tree T' with respect to the edge e'. Also recall that the sub-tree T_e anchored by e is the sub-tree rooted at the endpoint of e which is not in the spine of the disk D. We say e is *light* (with respect to D) if the exposed height of the sub-tree T_e satisfies $eH(T_e, e) \leq H(D) - b + 1$.

The significance of light edges is that if we know a boundary edge e of D is light then given any drawing of D we can redraw the sub-tree T_e near the boundary of D in a small bubble without increasing the height, since the maximum height over the bubble would be $eH(T_e, e) + b - 1$ which would be at most $H(D)$.

▶ **Lemma 8.** *Let D be a spine disk of depth d in a drawing ϕ with maximum quality $Q(\phi)$. Let $e \in A(D)$ be a light edge and T_e the sub-tree anchored by e. If e is a boundary edge then T_e is drawn in a bubble of depth d with e as the single boundary edge. Moreover, the strip between the bubble of T_e and the disk D is a sequence of bubbles of depth d anchored at the spine of D, or bends.*

7.3 Breaking ties while respecting the orders

In our arguments we will use a tie-breaking mechanism to decide between optimal drawings which all have maximum quality. We first define a perturbation of the original heights.

Let v be a vertex in D and let e_1, e_2 and f be edges incident to v. Let l be a vertical line. We say that f is *sandwiched* between e_1 and e_2 with respect to l if f does not intersect l but e_1 and e_2 intersect l, and f lies in the resulting bigon, see Figure 12. We add a small ϵ ($0 < \epsilon \ll 1$) for every sandwiched edge with respect to l to the height of l. The resulting value is called the *perturbed height* of l.

Consider the set Λ of lines of a given drawing as defined above and let $W(\phi) = (w_1, w_2, \ldots)$ be the sequence of perturbed heights of lines in Λ, sorted in a non-decreasing order. Let ϕ_1 and ϕ_2 be two drawings with maximum quality. We say that ϕ_1 has a better *secondary quality* than ϕ_2 if the sequence $W(\phi_1)$ is lexicographically smaller than $W(\phi_2)$.

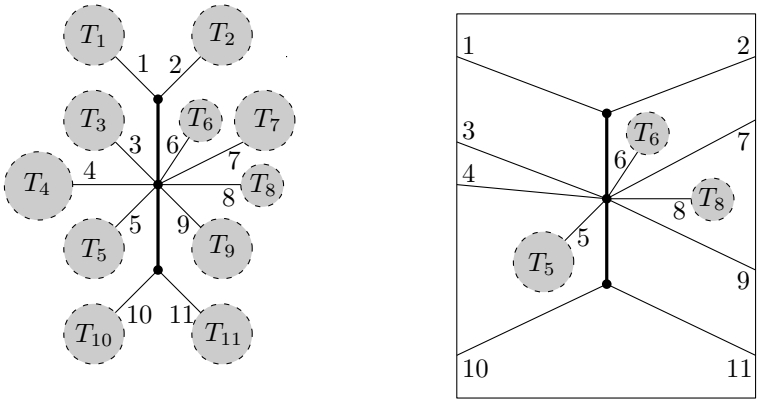


Figure 13 Left: the path P (thick edges), anchor edges $A(P)$ numbered 1 to 11, and the anchored sub-trees. Right: A spine disk with spine path P and $b = 3$. Proposition 10 states the following. 1) Edges 5, 6, 8 are light. 2) Among the six sets $\{1\}$, $\{2\}$, $\{3,4\}$, $\{7,9\}$, $\{10\}$, $\{11\}$ at most one can contain a light edge. Assume it is $\{7,9\}$. 3) If $eH(T_7) = eH(T_9) = H(D) - 3$ then $eH(T_8) \neq H(D) - 3$. If $eH(T_7) < H - 3$ and $eH(T_9) < H - 3$, then $eH(T_8) \geq H - 3$.

7.4 Fat structures

Let D be a local disk. Let ϕ be an optimal, simplified and balanced drawing such that $Q(\phi)$ is maximal among such drawings and also its secondary quality is the best possible. By Lemma 7, such a drawing has a structure tree. We call the resulting structure a *fat structure*[2] for D. It follows that any drawing which is optimal, simplified and balanced has to have worse or equal quality (or equal quality and equal or worse secondary quality).

The proof of the following is straightforward.

▶ **Lemma 9.** *Let ϕ be a drawing of local disk D with a fat structure. Then for every local disk D', corresponding to a node in the structure tree, the restriction of the structure to D' is a fat structure.*

The following lemma allows us to enumerate the spine disks which are possible in a fat structure. Refer to Figure 13 for an example.

▶ **Proposition 10** (Characterization of Spine Disks in Fat Structures)**.** *Let P be a path in the tree and let D be a spine disk with spine path P, such that D is a node in a fat structure. Let $b > 0$ be the number of left (equivalently right) boundary edges of D.*

1. *Every edge $e \in A(P)$ that lies entirely in the interior of D is light.*
2. *All light boundary edges of D are incident to a single vertex v, and intersect the same (left or right) boundary.*
3. *For $\eta \geq 0$, let $E(\eta) \subset A(P)$ be the set of anchor edges of P, incident to v, for which the exposed height of the sub-tree anchored by that edge is $H(D) - b + 1 - \eta$. Then, for $\eta = 0$, if any edge e in $E(0)$ is not a boundary edge, then e is not sandwiched, with respect to the boundary lines, between two edges of $E(0)$ that are boundary edges. Moreover, if any edge e of $E(\geq 1) := \bigcup_{j \geq 1} E(j)$ is not a boundary edge, then e is not sandwiched between two edges of $E(\geq 1)$ that are boundary edges.*

[2] The name comes from the fact that the bubbles in a minimal drawing tend to contain a maximal part of the tree.

We remark that the secondary quality is needed only in the proof of the second part of statement **3**. That is, the rest of proposition holds for drawings with maximum quality.

8 The dynamic program

We describe the algorithm for computing the optimal height of an input drawing. Modifying the dynamic program to compute an actual optimal height drawing is standard.

We think of row m of the dynamic programming table as containing (the description) of those spine and skew spine disks that have exactly m vertices in their interior and satisfy Proposition 10, together with their optimal heights. For $m = 1$, i.e. the first row, we must consider the (skew) spine disks with exactly one vertex in their interior. Since we are interested in balanced drawings, we know that each vertex v of even degree defines $O(d(v))$ distinct spine disks, where $d(v)$ is the degree of the vertex v. These are given by all the $O(d(v))$ possible balanced partitions of the edges incident to v into left and right edges, maintaining the order around v. The optimal drawings are trivial. Similarly, vertices with odd degree determine $O(d(v))$ distinct skew spine disks.

Assume that we have populated the table up to row $m-1$. The algorithm first computes all spine and skew spine disks with m vertices that satisfy Proposition 10. If $m = n$ we take the spine disk containing all the tree. If we are computing skew spine disks we take all the bubbles with m vertices (there are at most $2(n-1)$ bubbles). The rest of the (skew) spine disks are computed as follows. We determine only spine disks, and this also determines all possible skew spine disks. This is because a skew spine disk is the result of changing one non-boundary anchor edge of a spine disk into a boundary edge.

If we know the exposed heights of anchored sub-trees, then Proposition 10 implies that a spine disk is determined uniquely given the following parameters. We also indicate an upper bound on the number of possibilities for each of them.

1. The spine path P: $O(n^2)$ possibilities.
2. The boundary height b: $O(n)$ possibilities.
3. The height H: $O(n)$ possibilities.
4. A partition of $A(P)$ into cyclically contiguous subsequences $A_L(P)$ and $A_R(P)$: $O(n^2)$ possibilities.
5. The vertex v to which light boundary edges are incident: $O(n)$ possibilities.
6. Two consecutive sequences of edges around v: one for $E(0)$ and the other for $E(\geq 1)$: $O(n^4)$ possibilities.

Therefore, there are polynomially many possible values for all these parameters, namely $O(n^{11})$. A particular set of values may or may not define a valid spine disk, and a more careful analysis may result in asymptotically fewer relevant values. It is straightforward to compute the disk (if any) that corresponds to a particular set of values for the parameters.

Let p be the number of vertices in (some choice for) the spine path P. If the sub-tree anchored by an edge e has more than $m-p$ vertices then e has to be a boundary edge. Otherwise, the exposed height of the anchored sub-tree can be read from the table since it has at most $m-1$ vertices and is a bubble. Thus, we can determine which set of parameters determine a spine disk with m internal vertices. Additional details can be found in the full version of the paper.

After determining possible (skew) spine disks, we compute their optimal heights. This can be done by considering the polynomially many different ways that a (skew) spine disk can be decomposed into (skew) spine disks of smaller complexity (i.e vertices and edges), given in Figures 10 and 11. For a given (skew) spine disk D, the number of possibilities is

determined by the number of its possible first and last moves. These moves are either bends or vertices which are determined by an edge or a vertex of the tree, respectively. It follows that the total number of possibilities is polynomial. We consider all of the polynomially many decompositions of D. For a given decomposition, we read the optimal heights of their inner disks from the table. These heights can be used to derive the height of a drawing of D. The optimal height of D is the minimum such value over all of its decompositions. For more details to this part of the algorithm, and an exception to the general description above, we refer to the full version of the paper, where we prove the following.

▶ **Theorem 11.** *Let D be a (skew) spine disk. There is a polynomial-time algorithm for drawing D with optimal height.*

9 Discussion

We have presented the first polynomial-time algorithm for drawing plane trees with optimal height. The case of weighted plane trees remains open. Moreover, the setting of unweighted graphs remains open, but is believed to be NP-hard by some. However, we believe that a polynomial time algorithm may exist even in this setting.

If the graph setting turns out to be NP-hard, then the situation resembles that of the (non-embedded) min-cut linear arrangement problem, which has a polynomial time algorithm for unweighted trees [15], but is NP-hard for graphs [9, 8].

There are other interesting problems around the complexity and properties of optimal height drawings that might help in finding faster algorithms. As one such property, we conjecture that for unweighted trees there always exists an optimal drawing without spiraling edges. A spiral on an edge is depicted in Figure 2.

References

1. Hugo A Akitaya, Maarten Löffler, and Irene Parada. How to fit a tree in a box. In *Proceedings of the 26th International Symposium on Graph Drawing and Network Visualization (GD 2018)*, pages 361–367. Springer, 2018.
2. Therese Biedl, Erin Wolf Chambers, David Eppstein, Arnaud de Mesmay, and Tim Ophelders. Homotopy height, grid-major height and graph-drawing height. In *Proceedings of the 27th Graph Drawing and Network Visualization (GD 2019)*, pages 468–481. Springer, 2019.
3. Benjamin Burton, Erin Chambers, Marc van Kreveld, Wouter Meulemans, Tim Ophelders, and Bettina Speckmann. Computing optimal homotopies over a spiked plane with polygonal boundary. In *Proceddings of the 25th Annual European Symposium on Algorithms (ESA)*, 2017.
4. Erin Wolf Chambers, Gregory R Chambers, Arnaud de Mesmay, Tim Ophelders, and Regina Rotman. Constructing monotone homotopies and sweepouts. *arXiv preprint*, 2017. `arXiv:1704.06175`.
5. Erin Wolf Chambers, Arnaud de Mesmay, and Tim Ophelders. On the complexity of optimal homotopies. In *Proceedings of the 29th Annual Symposium on Discrete Algorithms (SODA 2018)*, pages 1121–1134, 2018.
6. Erin Wolf Chambers and David Letscher. On the height of a homotopy. In *Canadian Conference on Computational Geometry (CCCG)*, volume 9, pages 103–106, 2009.
7. Moon-Jung Chung, Fillia Makedon, Ivan Hal Sudborough, and Jonathan Turner. Polynomial time algorithms for the min cut problem on degree restricted trees. *SIAM Journal on Computing*, 14(1):158–177, 1985.
8. Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. Freeman San Francisco, 1979.

9 F Gavril. Some NP-complete problems on graphs. In *Proc. Conf. on Inform. Sci. and Systems, 1977*, pages 91–95, 1977.

10 Sariel Har-Peled, Amir Nayyeri, Mohammad Salavatipour, and Anastasios Sidiropoulos. How to walk your dog in the mountains with no magic leash. *Discrete & Computational Geometry*, 55(1):39–73, 2016.

11 Thomas Lengauer. Upper and lower bounds on the complexity of the min-cut linear arrangement problem on trees. *SIAM Journal on Algebraic Discrete Methods*, 3(1):99–113, 1982.

12 Debajyoti Mondal, Muhammad Jawaherul Alam, and Md. Saidur Rahman. Minimum-layer drawings of trees. In Naoki Katoh and Amit Kumar, editors, *WALCOM: Algorithms and Computation*, pages 221–232. Springer, 2011.

13 B. Monien and I.H. Sudborough. Min cut is NP-complete for edge weighted trees. *Theoretical Computer Science*, 58(1):209–229, 1988.

14 Yossi Shiloach. A minimum linear arrangement algorithm for undirected trees. *SIAM Journal on Computing*, 8(1):15–32, 1979.

15 Mihalis Yannakakis. A polynomial algorithm for the min-cut linear arrangement of trees. *Journal of the ACM*, 32(4):950–988, 1985.