Simultaneous Drawing of Layered Trees

Julia Katheder[1][0000-0002-7545-0730],
Stephen G. Kobourov[2][0000-0002-0477-2724],
Axel Kuckuk[1][0000-0002-5070-3412],
Maximilian Pfister[1][0000-0002-7203-0669], and
Johannes Zink[3][0000-0002-7398-718X]

[1] Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Tübingen, Germany
firstname.lastname@uni-tuebingen.de
[2] Department of Computer Science, University of Arizona, Tucson, USA
lastname@cs.arizona.edu
[3] Institut für Informatik, Universität Würzburg, Würzburg, Germany
lastname@informatik.uni-wuerzburg.de

Abstract. We study the crossing-minimization problem in a layered graph drawing of planar-embedded rooted trees whose leaves have a given total order on the first layer, which adheres to the embedding of each individual tree. The task is then to permute the vertices on the other layers (respecting the given tree embeddings) in order to minimize the number of crossings. While this problem is known to be NP-hard for multiple trees even on just two layers, we describe a dynamic program running in polynomial time for the restricted case of two trees. If there are more than two trees, we restrict the number of layers to three, which allows for a reduction to a shortest-path problem. This way, we achieve XP-time in the number of trees.

Keywords: layered drawing · tree drawing · crossing-minimization · dynamic program · XP-algorithm

1 Introduction

Visualizing hierarchical structures as directed trees is essential for many applications, from software engineering [2] to medical ontologies [1] and phylogenetics in biology [12]. Phylogenetic trees in particular can serve as an example to illustrate the challenges of working with hierarchical structures, as they are inferred from large amounts of data using various computational methods [19] and need to be analyzed and checked for plausibility using domain knowledge [9]. From a human perspective, visual representations are needed for this purpose. Most available techniques focus on the visualization of a single tree [7]. However, certain tasks may require working with multiple, possible interrelated trees, such as the comparison of trees [9,11] or analyzing ambiguous lineages [13]. Graham and Kennedy [7] provide a survey for drawing multiple trees in this context.

While there are many different visualization styles for trees (see an overview by Schulz [15]), directed node-link diagrams are the standard. The most common approach to visualize a directed graph as a node-link diagram is the layered drawing approach due to Sugiyama et al. [17]. After assigning vertices to layers, the next step is to permute the vertices on each layer such that the number of crossings is minimized, as crossings negatively affect the readability of a graph drawing [14,18]. However, this problem turns out to be hard even when restricting the number of layers or the type of graphs. For example, if the number of layers is restricted to two, crossing minimization remains NP-hard for general graphs [6], even if the permutation on one layer is fixed [5], known as the one sided crossing minimization (OSCM) problem. However, it is known that OSCM is fixed-parameter tractable in the number of crossings, which has first been shown by Dujmovic and Whitesides [4]. For the special case of a single tree on two layers, OSCM can be solved in polynomial time [8] and in the case that both layers are variable, the problem can be reduced to the minimum linear arrangement problem [16], which is polynomial-time solvable [3]. For an arbitrary number of layers, the problem is still NP-hard even for trees [8], however, the obtained trees in the reduction [8] are not drawn upward in the direction from the leaves to a root vertex (and we do not see an obvious way to adjust their construction). With respect to forests, the general case where $k \in O(n)$ is known to be NP-hard [10] even for $\ell = 2$ layers and trees of maximum degree 4.

Our Contributions. We consider the crossing-minimization problem for an n-vertex forest of k trees whose vertices are assigned to ℓ layers such that all leaves are on the first layer in a fixed total order and the vertices on each of the other layers need to be permuted. In other words, the task is to draw k layered rooted trees whose leaves may interleave simultaneously, while minimizing the number of crossings.

We show that the case of $k = 2$ trees is polynomial-time solvable for arbitrary ℓ using a dynamic program (see Sec. 3). Furthermore, we describe an XP-algorithm[4] in the number k of trees modeling the solution space by a k-dimensional grid graph for $\ell = 3$ layers. Our result generalizes to planar graphs under certain conditions (see Sec. 4). We conclude with the open case of $k \geq 3$ and $\ell \geq 4$ (see Sec. 5).

2 Preliminaries

Let F be a given forest of k disjoint rooted trees $T_1, \ldots, T_k$ directed towards the roots such that all vertices except for the roots have out-degree 1. For an integer $\ell \geq 2$, let an assignment of the vertices to ℓ layers be given, such that each tree T_i is drawn upward, i.e., for any directed edge $(u, v) \in T_i$, we have that the layer of u, denoted by $L(u)$, is strictly less than $L(v)$. This implies that

[4] XP is a parameterized running-time class and an XP-algorithm has a running time in $O(|I|^{f(k)})$, where $|I|$ is the size of the instance, f a computable function, and k the parameter. Note that every FPT-algorithm is an XP-algorithm but not vice versa.

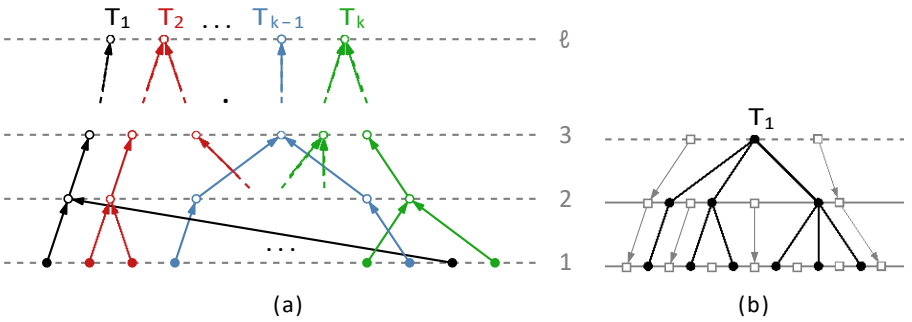


Fig. 1: (a) Upward drawing of k disjoint directed rooted trees $T_1, \dots T_k$ on ℓ layers. As indicated by the filled vertices, the total order $<_1$ of layer 1 is given, while the total orders $<_2, \dots, <_\ell$ need to be determined. In the following figures, we drop the arrowheads and assume an upward direction. (b) Illustration of positions (gray boxes) with respect to T_1 and their respective ideal positions indicated by a directed gray arrow from each position p to its ideal position $p^{\square}$.

if $L(u) = 1$, u is a leaf of T_i. The other way around, we also require that for any leaf v, $L(v) = 1$. Note that the roots of the trees can be placed on different layers, while layer ℓ hosts the root of every tree with height exactly ℓ. We refer to the set of vertices of T_i on layer j as $V_j(T_i)$ and we define the set of all vertices on layer j as $V_j(F) = V_j(T_1) \dot\cup \cdots \dot\cup V_j(T_k)$.

We further require that the total order $<_1$ of layer 1 (i.e., the order of all leaves) is given as part of the input, with the additional restriction that $<_1$ induces a planar embedding E_i with respect to each individual tree T_i, that is, there exists an ordering of the (non-leaf) vertices of T_i such that no two edges of T_i cross, see Fig. 1a for an illustration. Since the leaves of each T_i are all on layer 1, the embedding E_i is unique and implies a total ordering $<^i$ of the vertices of T_i on every layer $j \in \{2, \dots, \ell\}$. Therefore, we henceforth assume that $V_j(T_i)$ appears in the corresponding vertex order $<^i$, and if we combine all $<^i$ for $i \in \{1, \dots, k\}$, we obtain a partial order, which we call $\preceq_j$.

The task is to find a total order $<_j$ of $V_j(F)$ extending the partial order $\preceq_j$ for each $j \in \{2, \dots, \ell\}$ such that the total number of pairwise edge crossings implied by a corresponding straight-line realization of F is minimized.

We restrict the notion of an upward drawing even further since we require that for any directed edge $(u, v) \in T_i$, we have that $L(u) + 1 = L(v)$. If our input does not fulfill this requirement, this can be achieved by subdividing edges which span several layers (as commonly done, e.g., in the Sugiyama framework). In the following, we assume that n is the number of vertices after subdivision and let $n_1, \dots, n_k$ be the number of vertices of $T_1, \dots, T_k$, respectively. Furthermore, we denote the number of vertices of tree T_i on layer j by $n_{i|j} = |V_j(T_i)|$.

3 Two Trees on Arbitrarily Many Layers

In this section, we assume that we are given a forest $F = \{T_1, T_2\}$ with embeddings E_1 and E_2. We fix the drawing of T_1 according to E_1 and the only remaining task is to add the non-leaf vertices of T_2 in the order prescribed by E_2 such that the number of crossings is minimized. To this end, we describe a dynamic programming approach, which leads to the following theorem.

Theorem 1. *Let F be an n-vertex layered forest of two rooted trees, where all leaves are assigned to layer 1 and have a fixed order, which prescribe a planar embedding of each tree individually. We can compute a drawing of F where each tree is drawn in the prescribed planar embedding with the minimum number of crossings in $O(n^3)$ time.*

Proof. As stated before, we fix the drawing of T_1 according to E_1. Hence it remains to prove that our dynamic program embeds T_2 according to E_2, which we do in Lemma 1. In Lemma 2, we show that the resulting drawing has the minimum number of crossings. This proves the correct behavior of our algorithm. In Lemma 3, we also show the runtime bound of $O(n^3)$. □

Description of the Dynamic Program. Consider some layer $j \in \{2, \dots, \ell\}$ and index the vertices in $V_j(T_1)$ according E_1 from left to right by $1, \dots, n_{1|j}$. In a complete drawing, we define, for a vertex v of T_2, its position p on layer j with respect to the index of the closest vertex of T_1 to the left of v. If there is no such vertex, we set $p = 0$. Let $C_v = \{c_1, c_2, \dots, c_{\text{indeg}(v)}\}$ be the ordered set of children of v in T_2, which lie on layer $j - 1$, where $\text{indeg}(v)$ is the in-degree of v.

For our dynamic program, we define a function o, which has as first parameter a vertex v of T_2 and as second parameter a position p on layer $L(v)$. The value of o shall describe the number of crossings in an optimal partial solution for the drawing of the subtree of T_2 rooted at v and placed at position p. As usual in a dynamic program, we compute a function value once and then save it in a lookup table. Additionally, we save the recursive dependencies that led to a value to reconstruct a drawing in the end. If $j \geq 3$, we define o as follows.

$$o[v, p] = \sum_{i=1}^{|C_v|} \min_{q \in \{0, \dots, n_{1|j-1}\}} o[c_i, q] + \chi_{j-1}(q, p)$$

where $\chi_x(y, z)$ is a crossing function describing the number of crossings an edge between layers x and $x + 1$ admits if its source is arranged at position y (of layer x) and its target is arranged at position z (of layer $x + 1$). If for some c_i, there is more than one position for q resulting in a minimum value of $o[v, p]$, we choose the position q that maximizes $\chi_{j-1}(q, p)$.

For the recursive function o, we add a terminating formulation for the vertices on layer $j = 2$. Recall that for the leaves on layer 1, there is a total order $<_1$ given. Hence, for a vertex $v \in V_2(T_2)$, the position of each child of v is fixed,

leading to the following simplified formulation of o, where p_{c_i} is the given position of leaf c_i.

$$o[v, p] = \sum_{i=1}^{|c_v|} \chi_1(p_{c_i}, p)$$

To compute the value o^* of the dynamic program as a whole, we take the minimum of all values of o for the root r_2 of T_2:

$$o^* = \min_{p \in \{0, \ldots, n_{1|L(r_2)}\}} o[r_2, p].$$

We return a drawing corresponding to o^*, i.e., we specify for each vertex v of T_2 its position with respect to T_1 when computing o^*. Finally, for vertices of T_2 having the same position, we arrange them in the order given by E_2.

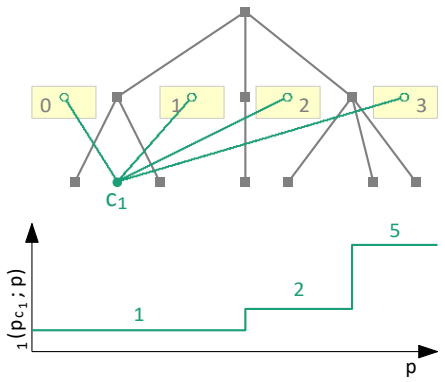
Correctness. Next, we prove the correct behavior of our dynamic program by showing that T_2 is embedded according to E_2 (Lemma 1) and by showing that the resulting drawing has the minimum number of crossings (Lemma 2). Mainly because Lemma 1 is rather intricate to prove, we next introduce some more notation and concepts, for which we show four claims that lead to the proofs of these lemmas.

A key observation is that for a position p on layer j, there is precisely one ideal position p^* on layer $j-1$ such that $\chi_{j-1}(p^*, p) = 0$ and for two positions p, q with $p < q$ on layer j, the ideal positions p^*, q^* on layer $j-1$ appear in the same order, i.e., $p^* < q^*$. (Imagine going down the gap of E_1 where p is located as illustrated in Fig. 1b.) In Claim 1, we formalize another observation regarding ideal positions. Essentially, the claim says that the further the endpoints of an edge are away from a pair of position and ideal position, the more crossings occur. For simplicity, we assume henceforth that each of the functions o and χ_j returns ∞ for parameters outside of its domain.

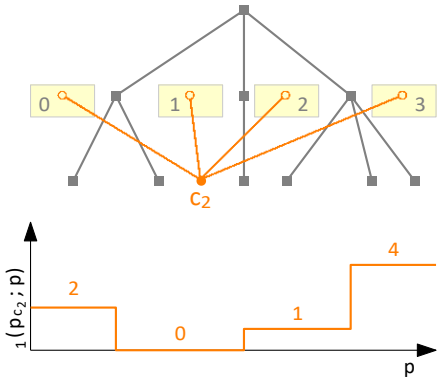
Claim 1. On a layer $j \in \{2, \ldots, L(r_1)\}$, let $p \in \{0, \ldots, n_{1|j}\}$ be a position and let $p^* \in \{0, \ldots, n_{1|j-1}\}$ be the ideal position of p on layer $j-1$. For any $x \in \mathbb{N}_0$, it holds that $\chi_{j-1}(p^* \pm x, p) \leq x$ and $\chi_{j-1}(p^*, p \pm (x+1)) > \chi_{j-1}(p^*, p \pm x) \geq x$.

Proof. Consider an edge (u, v) of T_2 with its endpoints being placed at p^* and p. We know that $\chi_{j-1}(p^*, p) = 0$. Now, for every position that we move u (v, resp.) to the left or right of p^* (of p), we change sides with a vertex w of T_1. Because w has exactly (at least) one incident edge going upwards (downwards) to its parent (a child) that we have not crossed before, the number of crossings increases by exactly (at least) 1. □

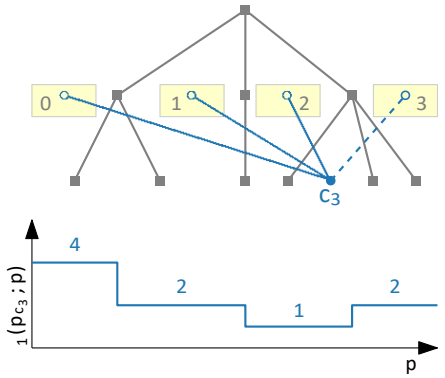
For a vertex $v \in V_j(T_2)$ on a layer j, we define $P_{opt}(v)$ as the set of every position p where o[v, p] is minimum. We analyze the properties of P_{opt} in Claim 2.



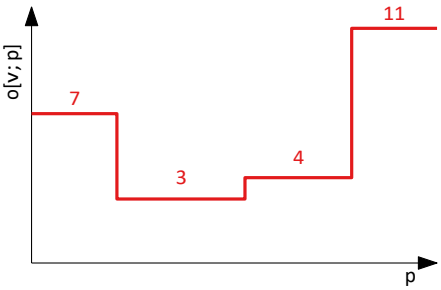
(a) Crossings of the edge (c_1, v) dependent on the position p of v.



(b) Crossings of the edge (c_2, v) dependent on the position p of v.



(c) Crossings of the edge (c_3, v) dependent on the position p of v.



(d) Sum of the three crossing functions gives $o[v, p]$.

Fig. 2: Example of a vertex v of $V_2(T_2)$ having three children c_1, c_2, c_3, where the position p_c of a child c and the position p of v determine the value of $o[v, p]$. Here, we perceive χ and o as functions dependent on p.

Claim 2. For every layer $j \in \{2, \ldots, L(r_2)\}$, let $v_1, v_2, \ldots, v_{n_{2|j}}$ be the vertices in $V_j(T_2)$ in the order of E_2. It holds that $\min P_{opt}(v_1) \leq \min P_{opt}(v_2) \leq \ldots \leq \min P_{opt}(v_{n_{2|j}})$ and $\max P_{opt}(v_1) \leq \ldots \leq \max P_{opt}(v_{n_{2|j}})$.

Further, for every $v \in V_j(T_2)$, $P_{opt}(v)$ is an interval of natural numbers and, for any $x \in N_0$, $o[v, \min P_{opt}(v) - (x+1)] > o[v, \min P_{opt}(v) - x] \geq x$ and $o[v, \max P_{opt}(v) + (x+1)] > o[v, \max P_{opt}(v) + x] \geq x$.

Proof. We show this claim by induction over the layers $j = 2, 3, \ldots$

For $j = 2$ and every $v \in V_j(T_2)$, the children of v have fixed positions and, therefore, $o[v, p]$ only depends on the number of crossings induced by the position $p \in \{0, \ldots, n_{1|j}\}$; see Fig. 2 for an example. We next show that $P_{opt}(v)$ is an interval. Observe that $o[v, p] = \sum_{i=1}^{|C_v|} \chi_1(p_{c_i}, p)$ is a sum of discrete functions (with variable p) where each admits its minimum value for one or two neighboring values of p and apart from at most two values at or around this minimum, all of these functions increase or decrease by the same amount if we add or subtract 1 to p, which follows by the argument presented in the proof of Claim 1. (These functions here are weakly unimodal, i.e., they have a global minimum and they increase monotonously when moving away from that minimum.) Now to find the positions where that sum is minimum, we traverse the values of its domain: we start with $p = 0$. If we increase p by one, then all functions that have not yet reached its minimum decrease, while the functions that had already reached their minimum increase by the same amount. Hence, this sum is minimum in the interval of the domain that has the minima of the single crossing functions equally distributed on the left and on the right side. Furthermore, for each position further to the left or right, the sum increases by at least one. It remains to show that the minima and maxima of $P_{opt}(v_1), P_{opt}(v_2), \ldots, P_{opt}(v_{n_{2|j}})$ increase monotonously. Since the children of the vertices on layer 2 (i.e., the leaves) are ordered, the minima and maxima of all crossing functions are ordered and so are the minima and maxima of the sums.

Now consider $j > 2$. Again $o[v, p]$ is a sum, but now we add, for every child c of v and a position q, $o[c, q]$ and $\chi_{j-1}(q, p)$. The sum of minima is again a sum of unimodal functions with similar properties as before: the $\chi_{j-1}(q, p)$ summands increase and decrease around their minimum as before, while the $o[c, q]$ summands increase and decrease before and after their minimum at least as much as a $\chi_{j-1}(q, p)$ summand due to the induction hypothesis. (They behave like a weighted $\chi_{j-1}(q, p)$ summand.) Hence, if we sum them up, we apply a weighted version of the previous argument to obtain the properties stated in the claim. In particular, the minima and maxima of $P_{opt}(v_1), P_{opt}(v_2), \ldots, P_{opt}(v_{n_{2|j}})$ increase monotonously since the minima and maxima of P_{opt} of the children on layer $j-1$ are ordered by the induction hypothesis. □

For a vertex u on a layer $j-1$, we define the natural position $p_{nat}(u, p)$ of u with respect to the position p of its parent vertex on layer j as

$$p_{nat}(u, p) = \begin{cases} p, & \text{if } p \in P_{opt}(u) \\ \max P_{opt}(u), & \text{if } p > \max P_{opt}(u) \\ \min P_{opt}(u), & \text{if } p < \min P_{opt}(u). \end{cases}$$

In Claim 3, we describe, for a vertex v, the behavior of the natural positions of v's children and their relationship to o[v, p].

Claim 3. For a vertex $v \in V_j(T_2)$ on a layer j, let $c_1, \ldots, c_{|C_v|}$ be the children of v. For any position $p \in \{0, \ldots, n_{1|j}\}$, it holds that $p_{nat}(c_1, p) \leq \ldots \leq p_{nat}(c_{|C_v|}, p)$ and $o[v, p] = \sum_{i=1}^{|C_v|}(o[c_i, p_{nat}(c_i, p)] + \chi_{j-1}(p_{nat}(c_i, p), p))$.

Proof. We partition the children of v into three groups: if for a child c_i (where $i \in \{1, \ldots, |C_v|\}$), $p \in P_{opt}(c_i)$, we set $q_i = p$. If for a child c_i, $p > \max P_{opt}(c_i)$, we set $q_i = \max P_{opt}(c_i)$, and, symmetrically, if $p < \min P_{opt}(c_i)$, we set $q_i = \min P_{opt}(c_i)$. By Claim 2, we observe that $q_1 \leq \ldots \leq q_{|C_v|}$. Since $q_i = p_{nat}(c_i, p)$, this proves the first part of the claim.

Now for the second part, if $o[v, p] = \sum_{i=1}^{|C_v|}(o[c_i, q_i] + \chi_{j-1}(q_i, p))$, then for some i, $o[c_i, q_i] + \chi_{j-1}(q_i, p) > \min_{q' \in \{0,\ldots,n_{1|j-1}\}}(o[c_i, q'] + \chi_{j-1}(q', p))$. Let $\hat{q} \in \{0, \ldots, n_{1|j-1}\}$ be a position such that $o[c_i, q_i] + \chi_{j-1}(q_i, p) > o[c_i, \hat{q}] + \chi_{j-1}(\hat{q}, p)$. Since we know $o[c_i, q_i] \leq o[c_i, \hat{q}]$, it follows that $\chi_{j-1}(q_i, p) > \chi_{j-1}(\hat{q}, p)$.

First note that $p \notin P_{opt}(c_i)$ because otherwise $\chi_{j-1}(q_i, p) < \chi_{j-1}(\hat{q}, p)$. It follows that q_i is the minimum or maximum position of $P_{opt}(c_i)$ – assume w.l.o.g. that $q_i = \max P_{opt}(c_i)$. Then, $q_i < \hat{q}$ (and hence $\hat{q} \notin P_{opt}(c_i)$) because otherwise again $\chi_{j-1}(q_i, p) < \chi_{j-1}(\hat{q}, p)$.

We distinguish two cases. The first case is $q_i < \hat{q} \leq p$. By Claim 1, we know that $\chi_{j-1}(q_i, p) - \chi_{j-1}(\hat{q}, p) = \hat{q} - q_i$. By Claim 2, we know that $o[c_i, \hat{q}] - o[c_i, q_i] \geq \hat{q} - q_i$. Hence, we have $o[c_i, \hat{q}] - o[c_i, q_i] \geq \chi_{j-1}(q_i, p) - \chi_{j-1}(\hat{q}, p)$, which we can reformulate as $o[c_i, q_i] + \chi_{j-1}(q_i, p) \leq o[c_i, \hat{q}] + \chi_{j-1}(\hat{q}, p)$, which contradicts our initial assumption.

The second case is $q_i < p < \hat{q}$. Now we have $\chi_{j-1}(q_i, p) = p - q_i$ and $\chi_{j-1}(\hat{q}, p) = \hat{q} - p$. If we add up these two equations, we get $\chi_{j-1}(q_i, p) + \chi_{j-1}(\hat{q}, p) = \hat{q} - q_i$. As we still have $o[c_i, \hat{q}] - o[c_i, q_i] \geq \hat{q} - q_i$, we get $o[c_i, q_i] + \chi_{j-1}(q_i, p) \leq o[c_i, \hat{q}] - \chi_{j-1}(\hat{q}, p)$, which, of course, also contradicts our initial assumption. □

Now in the last claim, which is Claim 4, we directly investigate the positions that are chosen by our dynamic program as the positions of the children of a vertex – they turn out to be the natural positions.

Claim 4. For a vertex $v \in V_j(T_2)$ on a layer j and a position $p \in \{0, \ldots, n_{1|j}\}$, the dynamic program selects $p_{nat}(c_1, p), \ldots, p_{nat}(c_{|C_v|}, p)$ as the positions of v's children $c_1, \ldots, c_{|C_v|}$ on layer $j - 1$.

Proof. Recall that, for any $i \in \{1, \ldots, |C_v|\}$, if there is more than one position for c_i resulting in a minimum value of o[v, p], the position of q with the maximum value of $\chi_{j-1}(q, p)$ is used as a tie-breaker rule. If $p_{nat}(c_i, p) = p$, then p is the only position of c_i that can lead to a minimum value of o[v, p] and our claim is true.

Now, due to symmetry, we assume w.l.o.g. that $p_{nat}(c_i, p) = \max P_{opt}(c_i)$. Let $p' = p_{nat}(c_i, p)$ be a position of c_i yielding a minimum value of o[v, p]. The position p' cannot lie within $P_{opt}(c_i)$ by Claim 1 since this would result in a larger

number of crossings, while $o[c_i, p'] = o[c_i, p_{nat}(c_i, p)]$. Hence, $p_{nat}(c_i, p) < p'$. By Claim 2, $o[c_i, p'] - o[c_i, p_{nat}(c_i, p)] \geq p' - p_{nat}(c_i, p)$. This means, that, for each position further to the right of $p_{nat}(c_i, p)$, the value of the dynamic program for c_i increases by at least one, while the number of crossings according to the function χ_{j-1} increases by exactly one (see Claim 1). Thus, $p_{nat}(c_i, p)$ is one (of possibly several) position(s) of c_i admitting a minimum value of $o[v, p]$. If p' also admits a minimum value of $o[v, p]$, but $o[c_i, p'] > o[c_i, p_{nat}(c_i, p)]$, it follows that $\chi_{j-1}(p', p) < \chi_{j-1}(p_{nat}(c_i, p), p)$. Hence, due to the tie-breaker rule, our algorithm would have selected $p_{nat}(c_i, p)$ instead of p'. □

Now we have gathered everything to establish the key lemma of this section.

Lemma 1. The drawing of T_2 is embedded according to E_2.

Proof. We prove that the vertices of T_2 are ordered according to $<_2^j$ by induction on layer j, starting with the layer of the root r of T_2. On layer $j = L(r_2)$, there is only r, which, of course, cannot contradict $<_2^j$.

Let $j < L(r_2)$ and $v_1, v_2, \ldots v_{n_{2|j+1}}$ be the vertices on layer $j+1$. Then, by Claims 3 and 4, the children C_{v_i} on layer j have increasing positions respect-ing $<_2^j$ for every $i \in \{1, \ldots, n_{2|j+1}\}$ and the edges to these children do not cross. It remains to show that no pair of edges between layers j and $j+1$ without a common endpoint cross. By our induction hypothesis, for two vertices $v_i, v_{i'}$ on layer $j+1$ with $i < i'$, the position p_i of v_i is not greater than the position $p_{i'}$ of $v_{i'}$. By Claim 1, it follows for the ideal positions of p_i and $p_{i'}$ that $p_i^{\square} \leq p_{i'}^{\square}$. By Claim 2, the min and max values of P_{opt} of the vertices on layer j are monotonically increasing. Hence, by Claim 4 and the definition of p_{nat}, there are no crossings between edges with target v_i and edges with target $v_{i'}$, as this would contradict $p_i^{\square} \leq p_{i'}^{\square}$. Hence, the edges between layer j and $j+1$ are planar, which concludes the induction step. □

After we have now shown that the dynamic program yields a valid solution, i.e., a drawing where both trees are internally crossing-free, it remains to prove that the number of crossings between T_1 and T_2 is minimum.

Lemma 2. The number of crossings in the computed drawing is minimum.

Proof. We show by induction over the layers $j = 2, 3, \ldots$ that for a vertex v and a position p, $o[v, p]$ is the minimum number of crossings induced by (T_1 and) the subtree of T_2 rooted at v, which we call T_v, across all drawings of T_v when we place v at position p. For $j = 2$, this is clear as we just sum up the number of crossings induced by the edges to the leaves.

Let $j \geq 3$. By Claim 4, we know that the dynamic program has selected the positions $p_{nat}(c_1, p) \leq \ldots \leq p_{nat}(c_{|C_v|}, p)$ for the children $c_1, \ldots, c_{|C_v|}$ of v. By our induction hypothesis, we know that, for each $c \in C_v$, $o[c, p_{nat}(c, p)]$ corresponds to a drawing of T_c at position $p_{nat}(c, p)$ with the minimum number of crossings. We add up the number of crossings between layer $j-1$ and j, and by the formulation of the dynamic program, we know that this is again minimum across all positions of c.

□

Running Time. It remains to analyze the running time of our dynamic program.

Lemma 3. The running time of our algorithm is in $O(n_1^2 \cdot n_2) \subseteq O(n^3)$.

Proof. For a vertex v of $V_j(T_2)$ and a position $p \in \{0, \dots, n_{1|j}\}$, we can compute $o[v, p]$ by finding, for each child $c \in C_v$ and a position q in a subset of $\{0, \dots, n_{1|j-1}\}$, the minimum of $o[c, q] + \chi_{j-1}(q, p)$.

The number of children over all steps is in $O(n_2)$ as T_2 is a tree and the number of positions is in $O(n_1)$. We can pre-compute and store all values $\chi_j(q, p)$ in $O(n_1^2)$ time. We have $O(n_1 n_2)$ entries of $o[v, p]$, which we can compute in overall $O(n_1^2 n_2) \subseteq O(n^3)$ time. The optimal root placement can be found in linear time. For the backtracking when constructing the final drawing, we simply store for each entry $o[v, p]$ a pointer to the entries it is based on. □

4 Multiple Trees on Three Layers

In this section, we consider the case that we are given a forest $F = \{T_1, \dots, T_k\}$ of k trees spanning (at most) three layers each, and we show the following result.

Theorem 2. Let F be an n-vertex layered forest of k rooted trees on three layers, where all leaves are assigned to layer 1 and have a fixed order, which prescribes a planar embedding of each tree individually. We can compute a drawing of F where each tree is drawn in the prescribed planar embedding with the minimum number of crossings in $O(n^k)$ time.

The first property we use to prove Theorem 2 is that the order of roots on layer 3 is fixed, similar to the order of the leaves on layer 1. We can assume this because there are only up to k roots on layer 3, with at most $k!$ ways to arrange them. We simply consider each permutation of the roots on layer 3 individually, and henceforth assume that both total orders $<_1$ and $<_3$ are given, fixing the roots and leaves, and the only remaining task is to compute $<_2$ of the vertices on layer 2 while maintaining their partial order $\subseteq_2$. Note that if any tree has its root on layer 2, we treat this root like the other vertices of layer 2.

As in Sec. 3, we use the notion of positions and crossing functions, however, we slightly adjust their definitions to better suit the setting of this section. Let σ be a permutation of $V_2(F)$ indexed by $1, 2, \dots$ and respecting the partial order $\subseteq_2$. For $i \in \{1, \dots, k\}$ and some vertex $v \in V_2(F) \setminus V_2(T_i)$, we denote the position (starting at 0) of v within the subsequence of σ consisting of the vertices $V_2(T_i) \cap \{v\}$ by p_i^v.[5] Note that, in a drawing using σ as $<_2$, we can charge every crossing to precisely two vertices of $V_2(F)$ as any crossing occurs between two edges that have two distinct endpoints on layer 2. Now observe that for a vertex $v \in V_2(T_j)$, where $j \in \{1, \dots, k\}$, the number of crossings charged to v with respect to σ depends only on p_i^v for each $i \in \{1, \dots, k\} \setminus \{j\}$. Therefore, we introduce the crossing function $\chi_i^v(p_i^v)$ returning the resulting number of crossings

[5] This is a generalization of the positions introduced in Sec. 3 where all positions were relative to (the given embedding of) T_1.

when we insert v at a position $p \in \{0, \dots, n_{i|2}\}$ into the planar embedding of T_i. The number $\chi_\sigma(v)$ of crossings charged to v when using permutation σ is then $\chi_\sigma(v) = \sum_{i \in \{1,\dots,k\} \setminus \{j\}} \chi_i^v(p_i^v)$ and the total number $\chi(\sigma)$ of crossings when using permutation σ is then $\chi(\sigma) = \sum_{v \in V_2(F)} \chi_\sigma(v)/2$.

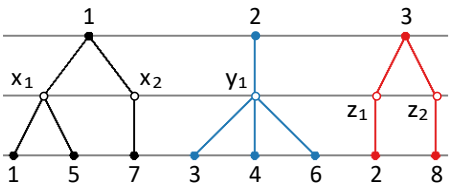
Lemma 4. For all combinations of $i \in \{1, \dots, k\}$, $v \in V_2(F) \setminus V_2(T_i)$, and $p \in \{0, \dots, n_{i|2}\}$, we can compute every value $\chi_i^v(p)$ in a total of $O(n^2)$ time.

Proof. First save, for every $v \in V_2(F)$, the star S_v induced by v and v's neighbors in total $O(n)$ time. Now for a fixed $i \in \{1, \dots, k\}$, consider the given planar embedding E_i of T_i. Also fix $v \in V_2(F) \setminus V_2(T_i)$ and compute $\chi_i^v(0)$ by checking, for every pair of edges of S_v and T_i, if there is a crossing if v is the leftmost vertex on layer 2. Then for $p = 1, \dots, n_{i|2}$, update $\chi_i^v(p-1)$ to $\chi_i^v(p)$ by checking each pair of edges from S_v and the star around the p-th vertex of $V_2^i(T_i)$. Over all of these steps, all vertices, and all trees, every pair of edges is considered at most four times, which yields a running time in $O(n^2)$. □

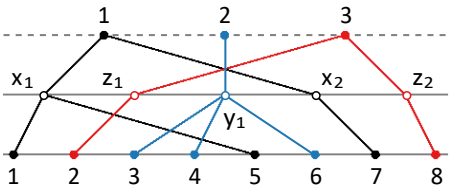
Reduction to a Shortest-Path Problem. We now construct a weighted directed acyclic st-graph H (see Fig. 3c) whose st-paths represent precisely all total orders of $V_2(F)$ that respect the vertex orders $<_2^1, \dots, <_2^k$ given for each tree by its prescribed planar embedding (see Fig. 3a). Moreover, for an st-path π representing a total order σ of $V_2(F)$, the weight of π is twice the number of crossings induced by σ. We let H be the k-dimensional grid graph of side lengths $n_{1|2} \times \cdots \times n_{k|2}$ directed from one corner to an opposite corner. More precisely, H has the node set $\{(x_1, \dots, x_k) \mid x_1 \in \{0, \dots, n_{1|2}\}, \dots, x_k \in \{0, \dots, n_{k|2}\}\}$ and there is a directed edge from $(x_1, \dots, x_k)$ to $(y_1, \dots, y_k)$ if $x_j + 1 = y_j$ for exactly one $j \in \{0, \dots, k\}$ and $x_i = y_i$ otherwise. Observe that, within H, $(0, \dots, 0)$ is the unique source and $(n_{1|2}, \dots, n_{k|2})$ is the unique sink, which we denote by s and t, respectively. We let an edge e from $(x_1, \dots, x_k)$ to $(y_1, \dots, y_k)$ where $x_j + 1 = y_j$ represent (i) taking the y_j-th vertex of $V_2(T_j)$, to which we refer as v next, (ii) after having taken x_i vertices of $V_2(T_i)$ for each $i \in \{1, \dots, k\}$. Thus, we let the weight w_e of e in H be the number of crossings charged to v in this situation, that is, $w_e = \sum_{i \in \{1,\dots,k\} \setminus \{j\}} \chi^v(x_i)$.

Clearly, any st-path π in H has (unweighted) length n_2. If we traverse π, we can think of layer 2 as being empty when we start at s, and then, for each edge of π, we take the corresponding vertex of V_2 and add it to layer 2. Since edge weights equal the number of crossings the corresponding vertices would induce in this situation, finding a lightest st-path in H means finding a crossing-minimal total order of layer 2 (see Fig. 3). By constructing H (using Lemma 4 to compute the edge weights) and searching for an st-path of minimum weight, we obtain an XP-algorithm in k; see Theorem 2, which we formally prove next.

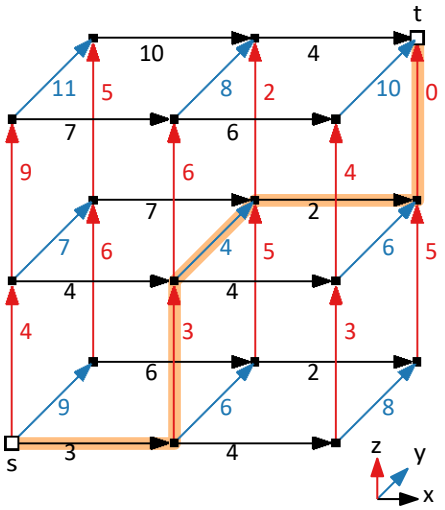
Proof (of Theorem 2). For F, we fix each order of layer 3 once and compute the corresponding k-dimensional grid graph H. We first argue that the st-paths of H represent precisely the possible total orders of $V_2(F)$ and their weights are



(a) We are given three embedded trees on three layers together with a total order of the leaves and the roots (numbers on the top and the bottom side).



(b) Drawing of the trees from (a) with the minimum number of crossings (six pairwise crossings) where the orders of the leaves and the roots are given. The total order of the vertices on layer 2, the middle layer, corresponds to the shortest st-path highlighted in orange in (c).



(c) Grid graph H with edge weights whose st-paths represent precisely the (allowed) total orders of the vertices on layer 2 of the forest shown in (a). The width in x-dimension is two and represents first choosing vertex x_1 and then vertex x_2 of the first tree. Similarly, the y- and z-dimension represent the vertices of the second and third tree, respectively. The st-path highlighted in orange is the lightest path with weight 12.

Fig. 3: Reducing the problem of finding a layered drawing of k trees on three layers with the minimum number of crossings, where the leaves and the roots are fixed, to a shortest-path problem in a weighted k-dimensional grid graph.

twice the number of crossings in the corresponding drawing of F. Thereafter, we argue about the running time.

Since we have a k-dimensional grid graph, any st-path in H traverses $n_{1|2}$ edges in the first dimension, $n_{2|2}$ edges in the second dimension, etc. We can interleave the edges of different dimensions arbitrarily to obtain different paths. Hence, each path is equivalent to exactly one total order σ extending the partial order $\preceq_2$, which is given by $<^1_2, \ldots, <^k_2$. The number of crossings of a drawing only depends on σ. Every crossing occurs between two edges being incident to precisely two distinct vertices of $V_2(F)$. We charge the crossing to these two vertices. Hence, we can add up the numbers of crossings charged to each vertex v (i.e., $\chi_\sigma(v)$) and divide the sum by two. These numbers of crossings charged to the vertices are by definition the edge weights of H. Therefore, each minimum-

weight st-path in H is equivalent to a minimum-crossing drawing of F for the given order of leaves and roots.

It remains to argue about the running time. The number of directed edges of H is upper-bounded by

$$E(H) = \sum_{j=1}^{k} n_{j|2} \prod_{i \in \{1,\dots,k\}\setminus\{j\}} n_{i|2} + 1 \leq k \prod_{i=1}^{k} n_{i|2} + 1 \leq k \left(\frac{n}{k}\right)^k$$

To compute the weight of each such edge, we sum up $k-1$ values of $\chi_i^v(p)$, which we have pre-computed in $O(n^2)$ time using Lemma 4. Therefore, we can construct H including the assignment of edge weights in $O(k^2(n/k)^k)$ time and we can find a minimum-weight path in H in $O(k(n/k)^k)$ time using topological sorting. Recall that we construct a graph H for at most $k!$ permutations of the roots on layer 3. For the final minimum-crossing drawing, we use the permutation of $V_2(F)$ and the permutation of $V_3(F)$ that correspond to the lightest minimum-weight path in any H. Hence, the total running time is in $O(k!k^2(n/k)^k) \subseteq O(k^3 \cdot (k-1) \cdot \ldots \cdot 2 \cdot 1/k^k \cdot n^k) \subseteq O(n^k)$. □

Finally, we remark that our XP-algorithm from Theorem 2 can be generalized in two ways.

Remark 1. By definition, $\preceq_2$ has only constraints between vertices of the same tree. We can extend $\preceq_2$ by (arbitrarily many) constraints between vertices of different trees and Theorem 2 still holds. This is because we can easily adjust our reduction: say x is the i-th vertex on layer 2 of the first tree, y is the j-th vertex on layer 2 of the second tree, and let the constraint $x \preceq_2 y$ be given. Then, in H, we set the weight of every edge representing x and lying in the y-dimension at a position $\geq j$ to ∞. Symmetrically, we set the weight of every edge representing y and lying in the x-dimension at a position $< i$ to ∞. This way, we prevent that a lightest path chooses an edge representing y before it chooses an edge representing x. If and only if there is an st-path with non-infinity weight in H, there is a valid arrangement of the vertices on layer 2.

Remark 2. Requiring trees on the three layers is a stronger restriction than actually needed. For our reduction, we only use the property that the vertex order on layer 3 is fixed, which we achieve by trying all permutations. For this approach, it sufices if layer 3 is sparse. Hence, our result also holds for k planar-embedded graphs provided that on layer 3, there are $O(k)$ vertices. Moreover, for planar-embedded graphs and an arbitrary number of vertices on layer 3, Theorem 2 holds as well if the total order of vertices on layer 3 is prescribed.

5 Conclusion and Open Problems

In this work, we approach the problem of crossing minimization of layered rooted trees from two directions. First, by describing a cubic-time dynamic program in

Theorem 1, keeping the number of trees k small, namely $k = 2$, while allowing an arbitrary number of layers. Inversely, our second result stated in Theorem 2 is an XP-time algorithm for an arbitrary number of trees, restricted to only three layers. Hence, there is a gap between these two results, which has not yet been explored and naturally raises the following open problem. Going one step further, is the case $k = 3$ trees and $\ell = 4$ layers polynomial-time solvable, and if so, for which k and ℓ does it become hard? Moreover, we pose the question of improving the complexity class for the case of $\ell = 3$ and $k > 2$, namely, can we solve the case of three layers in FPT-time in the number k of trees? Alternatively this may be proved to be W[1]-hard. Lastly, note that in our setting, we require that every tree preserves its given planar embedding (imposed by the order of its leaves). It is not clear, whether there exists a solution with less crossings without this restriction, although our current believe is that, in any minimum-crossing solution, all of them are drawn planar.

Acknowledgments. We thank the organizers of the workshop GNV 2022 in Heiligkreuztal for the fruitful atmosphere where some of the ideas of this paper arose. We also thank the anonymous reviewers for their helpful feedback.

References

1. Olivier Bodenreider. The unified medical language system (UMLS): Integrating biomedical terminology. Nucleic Acids Research, 32(suppl_1):267–270, 2004. doi: 10.1093/nar/gkh061.
2. Sudarshan S. Chawathe, Anand Rajaraman, Hector Garcia-Molina, and Jennifer Widom. Change detection in hierarchically structured information. ACM SIGMOD Record, 25(2):493–504, 1996. doi:10.1145/235968.233366.
3. Fan R. K. Chung. On optimal linear arrangements of trees. Computers and Mathematics with Applications, 10(1):43–60, 1984. doi:10.1016/0898-1221(84) 90085-3.
4. Vida Dujmovic and Sue Whitesides. An eficient fixed parameter tractable algorithm for 1-sided crossing minimization. Algorithmica, 40(1):15–31, 2004. doi: 10.1007/S00453-004-1093-2.
5. Peter Eades and Nicholas C. Wormald. Edge crossings in drawings of bipartite graphs. Algorithmica, 11(4):379–403, 1994. doi:10.1007/bf01187020.
6. Michael R. Garey and David S. Johnson. Crossing number is NP-complete. SIAM Journal on Algebraic Discrete Methods, 4(3):312–316, 1983. doi:10.1137/0604033.
7. Martin Graham and Jessie Kennedy. A survey of multiple tree visualisation. Information Visualization, 9(4):235–252, 2010. doi:10.1057/ivs.2009.29.
8. Martin Harrigan and Patrick Healy. k-level crossing minimization is NP-hard for trees. In Naoki Katoh and Amit Kumar, editors, Proc. 5th International Workshop on Algorithms and Computation (WALCOM'11), volume 6552 of Lecture Notes in Computer Science, pages 70–76. Springer, 2011. doi:10.1007/ 978-3-642-19094-0_9.
9. Zipeng Liu, Shing Hei Zhan, and Tamara Munzner. Aggregated dendrograms for visual comparison between many phylogenetic trees. IEEE transactions on visualization and computer graphics, 26(9):2732–2747, 2019. doi:10.1109/tvcg. 2019.2898186.

10. Xavier Muñoz, Walter Unger, and Imrich Vrto. One sided crossing minimization is NP-hard for sparse graphs. In Petra Mutzel, Michael Jünger, and Sebastian Leipert, editors, Proc. 9th International Symposium on Graph Drawing (GD'01), volume 2265 of Lecture Notes in Computer Science, pages 115–123. Springer, 2001. doi:10.1007/3-540-45848-4_10.
11. Tamara Munzner, François Guimbretiere, Serdar Tasiran, Li Zhang, and Yunhong Zhou. Treejuxtaposer: scalable tree comparison using Focus+Context with guaranteed visibility. ACM Transactions on Graphics, 22(3):453–462, 2003. doi:10.1145/1201775.882291.
12. Georgios A Pavlopoulos, Theodoros G Soldatos, Adriano Barbosa-Silva, and Reinhard Schneider. A reference guide for tree analysis and visualization. BioData mining, 3(1):1–24, 2010. doi:10.1186/1756-0381-3-1.
13. Pere Puigbò, Yuri I Wolf, and Eugene V Koonin. Search for a 'tree of life' in the thicket of the phylogenetic forest. Journal of Biology, 8(6):1–17, 2009. doi:10.1186/jbiol159.
14. Helen C. Purchase, David A. Carrington, and Jo-Anne Allder. Empirical evaluation of aesthetics-based graph layout. Empirical Software Engineering, 7(3):233–255, 2002. doi:10.1023/A:1016344215610.
15. Hans-Jörg Schulz. Treevis.net: A tree visualization reference. IEEE Computer Graphics and Applications, 31(6):11–15, 2011. doi:10.1109/mcg.2011.103.
16. Farhad Shahrokhi, Ondrej Sýkora, László A. Székely, and Imrich Vrto. On bipartite drawings and the linear arrangement problem. SIAM Journal on Computing, 30(6):1773–1789, 2000. doi:10.1137/S0097539797331671.
17. Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. Methods for visual understanding of hierarchical system structures. IEEE Transactions on Systems, Man, and Cybernetics, 11(2):109–125, 1981. doi:10.1109/TSMC.1981.4308636.
18. Colin Ware, Helen C. Purchase, Linda Colpoys, and Matthew McGill. Cognitive measurements of graph aesthetics. Information Visualization, 1(2):103–110, 2002. doi:10.1057/palgrave.ivs.9500013.
19. Ziheng Yang and Bruce Rannala. Molecular phylogenetics: principles and practice. Nature reviews genetics, 13(5):303–314, 2012. doi:10.1038/nrg3186.