# Untangling Tanglegrams: Comparing Trees by their Drawings *

Balaji Venkatachalam[1], Jim Apple[1], Katherine St. John[2], Dan Gusfield[1]

[1] Department of Computer Science, UC Davis
{balaji, apple, gusfield}@cs.ucdavis.edu
[2] Department of Mathematics and Computer Science, Lehman College, and the Graduate Center, City University of New York. stjohn@lehman.cuny.edu

**Abstract.** A tanglegram is a pair of trees on the same set of leaves with matching leaves in the two trees joined by an edge. Tanglegrams are widely used in biology – to compare evolutionary histories of host and parasite species and to analyze genes of species in the same geographical area. We consider optimizations problems in tanglegram drawings. We show a linear time algorithm to decide if a tanglegram admits a planar embedding by a reduction to the planar graph drawing problem. This problem was considered by Fernau, Kauffman and Poths. (*FSTTCS 2005*). Our reduction method provides a simpler proof and helps to solve a conjecture they posed, showing a fixed-parameter tractable algorithm for minimizing the number of crossings over all $d$-ary trees.
For the case where one tree is fixed, we show an $O(n \log n)$ algorithm to determine the drawing of the second tree that minimizes the number of crossings. This improves the bound from earlier methods. We introduce a new optimization criteria using Spearman's footrule optimization and give an $O(n^2)$ algorithm.
We also show integer programming formulations to quickly obtain tanglegram drawings that minimize the two optimization metrics discussed. We prove lower bounds on the maximum gap between the optimal solution and the heuristic of Dwyer and Schreiber (*Austral. Symp. on Info. Vis. 2004*) to minimize crossings.

## 1 Introduction

Determining the evolutionary history, or phylogeny, of a set of species is an important problem in biology. Often represented as trees, phylogenies are used for determining ancestral species, designing vaccines, and drug discovery. [26]. The popular criteria to reconstruct an optimal tree – maximum parsimony and likelihood – are NP-hard [12, 23], so heuristic methods (i.e. [16, 25]) are used that can yield many possible trees. Comparing these trees, as well as those generated on multiple genes, or for co-evolving species, is a necessary task for data analysis [21].

A visual way to compare two trees is via a tanglegram which shows the spatial relationship among the leaves. Roughly, a tanglegram consists of two trees with additional edges linking pairs of corresponding leaves (see Sect. 2). Tanglegrams are widely used in biology, including, to compare evolutionary histories of host and parasite species

* This research was partially supported by NSF grants SEI-BIO 0513910, SEI-SBE 0513660, CCF-0515378, and IIS-0803564.

and to analyze genes of species in the same geographical area [9, 28]. The number of edge crossings in tanglegrams serves as a good measure to the extent of *horizontal gene transfer*, which has been inferred by viewing single layouts of tanglegrams [5, pg. 204-206]. Drawings with fewer crossings or with matching leaves close together are more useful in biological analysis. We focus on two natural measures of complexity that are used for comparing permutations: the crossing number (or Kendall-$\tau$) and Spearman's footrule distance [6]. The former measures the number of times edges between the leaves cross, and the latter, the sum of the distances between leaf pairs. These are widely used, including, in ranking search results on the web and in voting systems [10, 7]. We focus on the complexity of these ranking problems and give efficient algorithms for drawing tanglegrams.

Crossing minimization in tanglegrams has parallels to crossing minimization in graphs [13, 18]. Computing the minimum number of crossings in a graph is NP-complete [13]. However it can be verified in linear time that a graph has a planar drawing (with zero crossings) [15, 24]. Analogously, crossing minimization in tanglegrams is NP-complete, while the special case of planarity can be decided in linear time. Fernau et al. [11] showed this by a reduction to the upward flow problem [2]. Independently, Lozano et al. [20] showed a simple dynamic programming based solution that gives a planar drawing in $O(n^2)$ time.

In recent work, Buchin et al. [4] showed approximation results and a fixed parameter tractable algorithm for complete tanglegrams (where every leaf has the same depth). We do not use this restriction and the results in this paper hold for arbitrary trees. Nöllenburg et al. show some experimental results [4] and discuss an integer quadratic program for the crossing problem [22]. Bansal et al. [1] define a *generalized tanglegram* to allow multiple edges between leaves in the two trees.

*Our results:* The case where only one tree is mutable is called the *one-tree crossing minimization (OTCM)* problem and has been studied for balanced trees in [8]. For arbitrary trees Fernau et al. [11] showed an $O(n \log^2 n)$ solution, while Bansal et al. [1] show an $O(n \log^2 n / \log \log n)$ solution. We provide an algorithm that improves the time bound to $O(n \log n)$ (Sect. 3.1).

Previous work on tanglegrams is limited to crossing minimization. We borrow Spearman's footrule distance function to use as an optimization criterion here. We show an $O(n^2)$ solution for the one-tree fixed case (Sect. 3.2).

We provide a method that has a simple intuition and allows us to use well studied solutions of graph drawing problems. Further, it leads to a simple fixed parameter tractable (FPT) algorithm. We show a linear time algorithm for planarity testing by a reduction to the planar graph drawing problem (Sect. 4.1). We can also use the fixed parameter algorithm for minimizing crossing numbers in graphs [18] to improve the running time of the FPT algorithm of Fernau et al. [11] for crossing minimization in binary trees and answer their conjecture for $d$-ary trees for $d > 2$ (Sect. 4.2). For the praxis of tanglegram drawing, we show integer programming formulations to obtain tanglegram drawings that minimize the two optimization metrics discussed (Sect. 5). We also show a lower bound on the worst case behavior of the heuristic of [8].

## 2 Preliminaries

We define tanglegrams and their drawings following [9, 11]: Let $L(T)$ denote the leaves of a tree $T$ . A linear order $<$ on $L(T)$ is called **suitable** if T can be embedded into the plane such that $L(T)$ is mapped onto a straight line in the order given by $<$. A **tanglegram** $(T_1, T_2; M)$ is given by a pair of rooted binary trees $(T_1, T_2)$ with perfect matching $M \subseteq L(T_1) \times L(T_2)$. In this paper we consider trees with $n$ leaves labeled $[n] = \{1, \ldots, n\}$, with $M$ matching leaves with identical labels.

A **drawing** of $(T_1, T_2; M)$ is given by two suitable linear orders $<_1$ and $<_2$ on $L(T_1)$ and $L(T_2)$, respectively. We call a drawing **proper** if it is **realized** by planar embeddings of $T_1$ and $T_2$ such that:

1. $L(T_1)$ and $L(T_2)$ lie on two parallel lines $L_1$ and $L_2$
2. All nodes of $T_i$ lie within the half-plane bounded by $L_{3-i}$ not containing $L_i$
3. Every node is farther from the line than its children.

Let $cr(T_1, T_2, M, <_1, <_2)$ denote the number of crossings in the drawing of $(T_1, T_2; M)$ given by linear orders $<_1$ and $<_2$. Note that by the definition only matching edges may cross and that the number of crossings is independent of the chosen realization. It is easy to see that a pair of edges cross at most once.

We consider two optimization criteria for drawing a tanglegram. The first is minimizing the number of crossings in the drawing, that is, for a given tanglegram $(T_1, T_2; M)$, we want

$$\min_{<_1, <_2} cr(T_1, T_2, M, <_1, <_2) \ .$$

Since the crossings can be changed by flipping the children at an internal node, the problem is to determine the order of the children at each internal node that minimizes the number of crossings.

The second criterion is based on the distance between the leaves in the orderings. Given a drawing $(T_1, T_2, M, <_1, <_2)$, let $\pi_i$ be the permutation on the leaves induced by $<_i$, $i = 1, 2$, $\pi_i : L(T_i) \to [n]$. Then, Spearman's **footrule distance** [3, 6] is given by

$$d_{foot}(\pi_1, \pi_2) = \sum_{i \in [n]} |\pi_1(i) - \pi_2(i)| \ .$$

Again, the optimization problem is to obtain the drawing that minimizes the distance.

Let $d$ be a distance measure on tanglegram drawings. We define $d(T_1, T_2, M, <_1, \cdot)$ to be the minimal value of $d(T_1, T_2, M, <_1, <_2)$ for all suitable linear orders $<_2$ on $L(T_2)$. Similarly $d(T_1, T_2, M, \cdot, \cdot)$ is defined to be the minimal value of $d(T_1, T_2, M, <_1, <_2)$ for all suitable linear orders, $<_1$ and $<_2$ on $L(T_1)$ and $L(T_2)$, respectively. We define the following two natural problems for crossings in tanglegrams:

**One-Tree Crossing Minimization (OTCM)**
INSTANCE: A tanglegram $(T_1, T_2; M)$ with suitable linear linear order, $<_1$ on $L(T_1)$.
RESULT: A $<_2$ with $cr(T_1, T_2, M, <_1, <_2)$ minimal.

**Two-Tree Crossing Minimization (TTCM)**
INSTANCE: A tanglegram $(T_1, T_2; M)$ and parameter $k$.
QUESTION: Is $cr(T_1, T_2, M, \cdot, \cdot) \leq k$?

One- and two-tree footrule distance minimization problems are defined analogously.

## 3 One-Tree Optimization Problems

For one-tree minimization problems, we assume, w.l.o.g, that the all tree labels are in $[n]$, that $M$ is the identity matching, and that $<_1$ is simply $<_{[n]}$.

### 3.1 One-Tree Crossing Minimization

We give an algorithm for the one-tree crossing minimization with running time $O(n \log n)$. As in [8, 11], we exploit the optimal substructure property of the problem and recursively work on the subtrees. Our results are due to novel use of efficient data structures to maintain lists of the subtrees' leaves. To calculate the optimal layout at any internal node, $v$, we analyze the child subtrees to calculate which of the two available layouts is better. This is sufficient since:

**Lemma 1.** *Let $<_2$ be an optimal suitable linear order on $L(T_2)$. Then for every subtree, $S$, of $T_2$, $<_2$ is an optimal suitable linear order for $L(S)$.*

*Proof.* Assume not. Then there is some $<_B$ for $S$ with fewer crossings. Define a new ordering, $<_N$ on $L(T_2)$, using $<_B$:

$$x <_N y \iff \begin{cases} x <_B y \text{ if } x, y \in L(S) \\ x <_2 y \text{ otherwise} \end{cases}$$

By construction, $cr(T_1, T_2, M, <_1, <_N) < cr(T_1, T_2, M, <_1, \cdot)$, contradicting the optimality of $<_2$. □

**Theorem 2.** *OTCM can be solved in $O(n \log n)$ time.*

*Proof.* Any suitable order on $L(T_2)$ can be constructed by choosing, for each non-leaf node in $T_2$, one of the two possible orders of its children. At each node, we chose an ordering recursively, starting from nodes closest to the line $L_2$.

For each internal node, we not only decide the optimal order for its children, we also construct a **2-3 finger tree**, an ordered search tree with fast split and append operations [17]. The finger trees at siblings will be used to decide the ordering for their shared parent.

The base case is for our induction is simply the leaves. These require no layout decision, and can be made into a singleton finger tree of size 1 in constant time [17].

At every internal node $v$ we construct a finger tree holding the leaf labels of its descendents, ordered by $<_1$. Since $v$ is farther from $L_2$ than either of its children, induction allows us to assume each child already has a finger tree associated with it. The method for constructing a finger tree and layout choice at $v$ is shown in Algorithm 1.

The algorithm takes as input two finger trees $p$ and $q$ corresponding to the two child nodes ($\mathrm{node}(p)$ and $\mathrm{node}(q)$). The trees are merged according to the usual merge procedure on finger trees: for each $q_h \in q$, split what remains of $p$ using $q_h$ as a pivot. The left partition is saved, and the right partition is split again with the next $q_h \in q$. Once $q$ is empty, the partitions of $p$ are concatenated, interspersing the $q_h$'s where they were used as pivots. Algorithm 1 returns the merged finger tree as $result$.

**Algorithm 1** Container merging for the minimal-crossing single-tree problem. The inputs $p$, $q$, and the output $result$ are finger trees sorted according to $<_1$. If the output $count < maxCrossings/2$, then the child node corresponding to $p$ should precede the child node corresponding to $q$ to minimize crossings.

1: $count \leftarrow 0$
2: $result \leftarrow \langle\rangle$
3: $maxCrossings \leftarrow |p||q|$
4: **while** $|q| > 0$ **do**
5: $(q_h, q) \leftarrow \mathrm{head/tail}(q)$ // pops the first element of $q$
6: $(r, p) \leftarrow \mathrm{split}(p, q_h)$ // splits $p$, removes elements less than $q_h$ into $r$
7: $result \leftarrow result +\!\!+ r$ // appends elements smaller than $q_h$.
8: $result \leftarrow result +\!\!+ \langle p_h \rangle$
9: $count \leftarrow count + |p|$ // the number of crossings for $q_h$
10: **end while**
11: $result \leftarrow result +\!\!+ q$
12: **return** $(count < maxCrossings/2, result)$

To determine an optimal layout, our algorithm modifies the merge algorithm to count the number of crossings at each pivot. If $\mathrm{node}(p)$ is realized preceding $\mathrm{node}(q)$, the number of new crossings induced is $|\{(x, y) : x \in p, y \in q, y <_1 x\}| = \sum_{y \in q} |\mathrm{split}_2(p, y)|$, where $\mathrm{split}_2(p, y)$ is the right partition of $p$ using $y$ as the pivot. Our algorithm calculates this in the variable $count$.

We now must determine if the number of crossings is lower with $\mathrm{node}(p)$ preceding or following $\mathrm{node}(q)$. Since the every pair of nodes not crossing in one layout of of $(\mathrm{node}(p), \mathrm{node}(q))$ is crossing in the other, their sum is the total number of possible crossing pairs: $|p||q|$. A layout with no greater than $\frac{|p||q|}{2}$ crossings is therefore optimal. Algorithm 1 returns $\mathrm{True}$ if $\mathrm{node}(p)$ should precede $\mathrm{node}(q)$.

*Complexity* Kaplan and Tarjan [17] describe $\mathrm{split}$ and append ($+\!\!+$) operations on 2-3 finger trees. The operation $(t_L, t_R) \leftarrow \mathrm{split}(t, v)$ takes $O(\log(\min(|t_L|, |t_R|)))$ time, and $t_1 +\!\!+ t_2$ takes $O(\log(\min(|t_1|, |t_2|)))$ time. Therefore, the head/tail split on line 5 and append on line 8 take only $O(1)$ time. The values $|p|$ and $|q|$ can be computed in $O(1)$ time as shown by Hinze and Patterson [14], where they maintain the trees with size information.

The call to $\mathrm{split}$ in line 6 takes time proportional to the logarithm of the smaller of $\{|r|, |p \backslash r|\}$. Taking $d_i$ as the size of $r$ in the loop iteration when $q_h$ is the $i$th element in $q$, the total time taken in line 6 is no more than $\sum_{i=0}^{|q|} \alpha \log d_i$ where $\sum_{i=0}^{|q|} d_i \leq p$. This applies to lines 7 and 11 as well; these lines append to $result$ all of $p$, in $q + 1$ pieces.

The total complexity is bounded by the shared complexity of lines 6, 7 and 11. Since the sum of the logarithms is maximized when all the $d_i$s are equal [17], the complexity is thus $O\left(\sum_{i \leq |q|} \log d_i\right) = O\left(|q| \log\left(\frac{|p|}{|q|}\right)\right)$. The total time to calculate the optimal layout at a node with $n$ descendant leaves is given by the recurrence: $T(n) = T(l) + T(r) + O\left(l \log\left(\frac{r}{l}\right)\right)$, where $l, r$ are the number of leaves in the left and right subtrees,

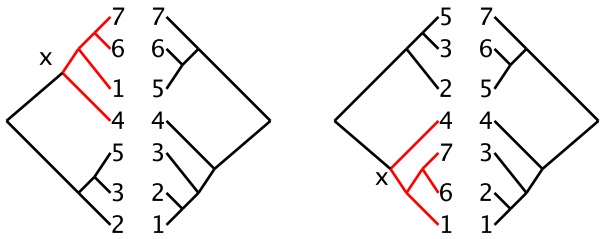


**Fig. 1.** One-tree footrule distance minimization with respect to the identity permutation $1, \ldots, 7$. Consider the configuration of the subtree rooted at $x$. In the left figure, the configuration of this subtree is optimal and contributes 4 to the overall distance. If the same layout were to be used at position 1, shown in the right figure, it would contribute 8 to the footrule distance value. However, the optimal configuration at that position has footrule distance 4 as shown in the right figure.

and $l + r = n$. Using induction, assume that $T(m) \in O(m \log m)$ for all $m < n$.

$$T(n) = O(l \log l) + O(r \log r) + O\left(l \log\left(\frac{r}{l}\right)\right) = O((l + r) \log r) = O(n \log n) \quad .\square$$

### 3.2 One-Tree Distance Minimization

The crossings minimization problem has the optimal substructure property, i.e., a configuration that minimizes the number of crossings of a subtree is also a configuration that minimizes the crossings in any optimal solution. Therefore, once the value of an optimal configuration of a subtree is computed, we can reuse the configuration irrespective of where the subtree appears in the final layout of the solution. However, in the footrule distance minimization problem, the optimal configuration of a subtree depends on the position of the subtree. An optimal configuration for one position need not be an optimal solution for all positions. See Fig. 1 for an example.

Nonetheless, we can find an $O(n^2)$ algorithm using dynamic programming. For a leaf labeled $i$ at position $j$, the footrule distance is $|i - j|$. Consider an internal node $v$ with children $u$ and $w$ with $c_1$ leaves and $c_2$ leaves in the two subtrees, respectively. The optimal solution for the subtree rooted at $v$ with the leaves starting at position $i$, $D(v, i)$, is obtained by either drawing $u$ on the left with leaves from $i$ through $i + c_1 - 1$ and $w$ on the right with leaves from $i + c_1$ through $i + c_1 + c_2 - 1$, or in the opposite order. We choose the ordering that minimizes the value.

$$D(v, i) = min\{D(u, i) + D(w, i + c_1), D(w, i) + D(u, i + c_2)\} \quad .$$

The optimal solution for the tree is $D(root, 1)$. The correctness of the algorithm is straightforward. The algorithm can be run in $O(n^2)$ time, since there are $n - 1$ internal nodes and for each node we do a constant amount of computation in at most $n$ positions.

## 4 A Reduction Method for Two-tree Crossing Problems

When both leaf orderings in a tanglegram are allowed to change, the complexity of crossing minimization increases greatly. While the case where one tree is fixed (OTCM)

is solvable in polynomial time, the TTCM problem, where the ordering of both trees is mutable, is NP-hard [11]. However, the special case of checking if a tanglegram has a drawing with zero crossings (planarity testing) can be solved in linear time [11].

### 4.1 Two-tree Tanglegram Planarity

We first define a natural extension of tanglegrams for planarity testing.

**Definition 3.** *An* augmented *tanglegram is a tanglegram with the roots of the two trees joined by an edge. This edge is called the* augmented edge.

**Lemma 4.** *A tanglegram has a proper drawing with zero crossings iff the augmented tanglegram has a planar drawing.*

*Proof.* The "if" direction of the lemma is straightforward. For the other direction, consider a planar drawing of the augmented tanglegram. If the drawing of the augmented tanglegram is proper, removing the augmenting edge gives us a proper planar drawing. If the drawing of the augmented tanglegram is not proper, we need to show a way to rearrange the edges of this drawing to produce a proper drawing.

To do so, first contract the internal edges of the two trees except for the two edges out of each root. During the contracting process, shown in Fig. 2, no new planar regions are produced. Regions that are bounded between the internal edges of one tree, the edges connecting the leaves, and the internal edges of the other tree vanish when the internal edges are contracted (see Fig. 2). We call the resulting graph the *reduced graph* and label the root and its two children $r_1, u_1, v_1$, respectively, in one tree and $r_2, u_2, v_2$ in the other.

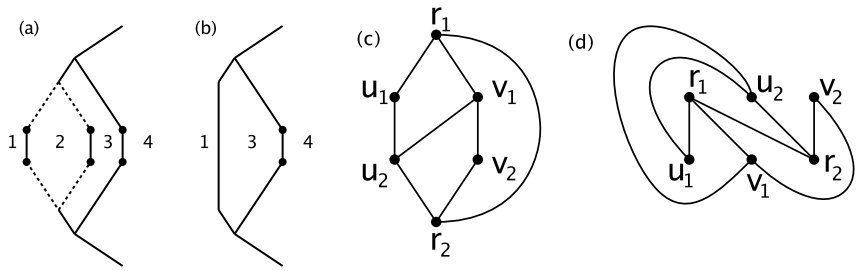


**Fig. 2.** (a) & (b): Contraction process: After contracting the dashed internal edges, planar region 2 vanishes. The new edge can be thought of as containing the region 2 within it, and is called a super-edge. (c) & (d): Avoiding $K_{3,3}$ minor: There are at most 3 edges between pairs $(u_1, v_1)$ and $(u_2, v_2)$. (d) is not proper. The edges can be rearranged to form (c).

There are four possible edges between $\{u_1, v_1\}$ and $\{u_2, v_2\}$. We call these edges between the two trees *super-edges*. Each of these super-edges represents the union (merger) of some of the regions. We claim that at most three of these edges exist. If

all four edges existed, then together with the augmented edge $(r_1, r_2)$ they would form $K_{3,3}$ (see Fig. 2) contradicting the planarity of the original drawing.

Without loss of generality, let the three super-edges be $(u_1, u_2)$, $(v_1, v_2)$ and $(u_2, v_1)$. Any drawing on the reduced graph with these edges can be redrawn to a proper drawing of the reduced graph (as the example in Fig. 2). Rearranging the super edges is equivalent to rearranging of the edges and the regions of the original graph. Now expanding the edges in the reverse order of contraction gives us a proper drawing for the tanglegram. □

The idea of the proof can be extended to get an algorithm that generates a proper drawing by setting a convention for left and right children and remembering the left and right order on the children during edge contraction. The order of some edges might be reversed in rearranging the super-edges. Finally, the order information is used recursively during the edge expansion to obtain a proper drawing.

**Theorem 5.** *Deciding if a tanglegram admits a planar drawing can be done in linear time.*

*Proof.* Apply the linear planar graph drawing algorithm [15, 24] to the augmented tanglegram. Follow the previous lemma to obtain a proper drawing. □

### 4.2 Fixed Parameter Tractability of TTCM

The two-tree problem (TTCM), when restricted to binary trees, is fixed parameter tractable with parameter $k$, the number of crossings, as shown by Fernau et al. [11]. Their proof relies on the trees being binary and achieves the result through a complicated analysis of quadruples of leaves. They conjecture difficulty for $d$-ary trees for $d > 2$. We use our reduction method to utilize the elegant work of Kawarabayashi and Reed [18]. We give a simple proof that resolves the conjecture of [11] that TTCM is fixed parameter tractable over the class of all finite trees:

**Theorem 6.** *TTCM is fixed parameter tractable over the class of all finite trees with parameter, $k$, the number of crossings. The algorithm takes time quadratic in $n$.*

As in the planar drawing problem, we create an augmented tanglegram and use the FPT algorithm of crossing minimization in graphs from [18]. Like in the planarity case we want to disallow crossings with internal edges. To achieve this, we add $n$ duplicate edges around each internal edge and the augmented edge. If two internal edges cross, there will be $n^2$ crossings, which is more than the number of crossings in the sought proper drawing. Similarly, anything but the proper drawing of the edges connecting the leaves will increase the number of crossings. This ensures proper drawing. The proof of Theorem 6 is in [27].

## 5 Integer Programming solutions

Integer Linear Programming (ILP) is one of the standard approaches to obtain fast solutions for hard problems as they provide provably optimal solutions. Though the runtime

is not polynomially bounded, they are fast in many practical settings, and are often better than provably efficient methods. We describe ILP formulations for the two-tree optimization problems considered in this paper.

### 5.1 Crossing minimization

The formulation for crossing minimization is based on the following intuition: if the leaf $i$ is to the left of leaf $j$ in both of the trees, then the edges connecting the $i$'s and the $j$'s do not cross. The edges cross if there is an inversion in the order.

To realize this, we introduce binary variables $x_{i,j}$ for all leaf pairs $(i, j)$ such that $i < j$. $x_{i,j}$ is set to 1 iff $i$ appears before $j$ in the linear order. For every internal node $k$ we introduce a variable $y_k$. Let $c_1$ and $c_2$ be the two children of $k$. $y_k = 1$ if $c_1, c_2$ are to the left and right, respectively, and $y_k = 0$ otherwise. For leaves $i$ in the subtree below $c_1$ and $j$ in the subtree below $c_2$, if $i < j$ then $x_{i,j} = 1 \iff y_k = 1$, so $x_{i,j} = y_k$. If $j < i$ then $y_k = 1 - x_{i,j}$. Analogously, for the second tree we define these constraints over variables $x'_{i,j}$ and $y'_k$.

If $i$ is to the left (or right) of $j$ in the drawing of both trees in the tanglegram, then there is no crossing. $i$ and $j$ cross only when the order is reversed. That is, $i, j$ cross iff $x_{i,j} \neq x'_{i,j}$. We let $z_{i,j} = x_{i,j} \oplus x'_{i,j}$. We can rewrite the XOR as linear inequalities. The objective function for minimizing the number of crossings is therefore $\min \sum_{i<j} z_{i,j}$.

### 5.2 Distance minimization

We describe two different formulations for the distance minimization problem. The first formulation is based on the dynamic programming idea used in the one-tree distance minimization problem. The second uses the simple fact that the the order of its children in an internal node determines the relation between the leaves in the two subtrees.

*Dynamic programming version* For a vertex $k$ we set a binary variable $y_{k,p} = 1$ when the subtree beneath it is placed starting at position $p$. For instance, $y_{root,1} = 1$ always. If $k$ is an internal node, let $i$ and $j$ be the its children with $l$ and $r$ leaves in the subtrees below them.

$y_{k,p} = 1$ implies that node $i$ is placed at position $p$ or $p + r$. This implication is written by the inequality $y_{i,p} + y_{i,p+r} \geq y_{k,p}$. Similarly $y_{j,p} + y_{j,p+l} \geq y_{k,p}$. Both $i$ and $j$ cannot be the left (or right) child of $k$ simultaneously, so $y_{j,p} + y_{i,p} \leq 1$.

Every leaf must occur exactly once. For every leaf $l$, therefore, $\sum_{r\in[n]} y_{l,r} = 1$. Every position must have exactly one leaf, so $\forall r \in [n]$, $\sum_{l\in\text{leaves}} y_{l,r} = 1$. We use variables $y'$ and similar inequalities for the second tree.

Binary variables $z_{l,r,r'} = 1$ only when the leaf $l$ is present at positions $r, r'$ in the two trees respectively. $z_{l,r,r'}$ contributes $|r - r'|$ to the distance value. Therefore, the objective function is $\min \sum_{\text{leaf } l} \sum_{r\in[n]} \sum_{r'\in[n]} |r - r'| z_{l,r,r'}$.

*Distance version* Consider an internal node $i$ with $m$ leaves in its subtree and let its two children be $c_1, c_2$. Let $j, k$ be leaves in subtrees $c_1, c_2$ respectively. Let $x_j$ denote the position of leaf $j$ in the linear order, $[n]$. Introduce a binary variable $y_i$ for each internal

**Table 1.** Running time of ILP solutions: average time, in secs, is averaged over 30 runs.

| Crossing Problems | | | Distance problems | | | | |
|---|---|---|---|---|---|---|---|
| Input size | Crossing | | Input Size | Distance | | Dynamic Programming | |
| | Time | variance | | Time | variance | Time | variance |
| 10 | 0.02 | 0.01 | 6 | 0.12 | 0.04 | 0.41 | 0.25 |
| 20 | 0.32 | 0.17 | 10 | 16.87 | 19.21 | 36.34 | 18.69 |
| 30 | 2.03 | 0.54 | 11 | 75.93 | 110.80 | 99.04 | 56.06 |
| 40 | 7.79 | 1.7 | 12 | 182.10 | 245.75 | 324.36 | 211.48 |
| 50 | 20.87 | 3.64 | 15 | 781.88 | 1171.95 | 8663.02 | 6208.82 |

node $i$ to model the choice of $c_1$ or $c_2$ being the left child. $y_i = 1$ when $c_1$ is the left child (and $j$ is to the left of $k$). The opposite is implied by $y_i = 0$.

$$y_i = 1 \iff -(m-1) \leq x_j - x_k \leq -1 \quad (1)$$

$$y_i = 0 \iff 1 \leq x_j - x_k \leq m-1 \quad (2)$$

These implications are written as the following inequalities: $x_j - x_k + 1 \leq m(1-y)$ and $x_j - x_k + my \geq 1$.

Next we need to ensure that all leaves $1 \leq x_j \leq n$ and all $x_j$'s are unique. The uniqueness constraints can be written in a number of ways. We model them as a matching problem. It has been observed in the ILP literature that the vertices of the matching polytope are all lattice points and therefore the ILP software need not apply further reduction techniques [19]. As usual, we define similar inequalities on variables $x'_i$ and $y'_i$ for similar constraints on the second tree. Finally, the optimization criterion is $\min \sum_i |x_i - x'_i|$.

### 5.3 Timing

To generate a random tree we take a random subset of $[n]$. This is the set of leaves on the left subtree of the root. The rest of the elements are leaves of the right subtree. We recurse on these subsets to generate the random tree. We take two such trees to form a random tanglegram.

We executed the ILP formulations of the problem using CPLEX-10 on a Pentium IV 3 GHz dual-core desktop machine with 2GB of RAM. The data shown in Table 1 are obtained by averaging the running time over thirty runs each for problems of various data sizes. The crossing minimization problem is very fast. The distance version is slower in comparison. It is relatively fast for small datasets. The distance version is about three times faster than the dynamic programming version. We see in our examples that most of the executions run in about less than half of the reported mean time. There are about 10% of the cases that take much longer, leading to increased variance. In most of these cases CPLEX obtains the optimal solution quickly or finds a solution very close to optimal solution very soon, but takes much longer to make minor improvements or to ensure there is no better solution.

## 6 Dwyer and Schreiber's seesaw heuristic

Though [4] shows that, assuming the Unique Games Conjecture, there is no constant-factor approximation algorithm for TTCM, Dwyer and Schreiber [8] present a heuristic for $n$ tree crossing minimization that iteratively solves OTCM for each tree. The idea is to fix $<_2$, then solve OTCM on $T_1$, then fix $<_1$ and solve OTCM on $T_2$. They found that this yielded a good solution after ten or fewer iterations. We call this "seesawing".

**Theorem 7.** *For any $N$, there is an $n > N$, and a tanglegram drawing of size $n$ for which the optimal drawing produced by seesawing has $\Omega(n^2)$ more crossings than an optimal drawing.*

We call a drawing that can't be improved by seesawing *seesaw-optimal*. We prove the theorem (in [27]) by finding one tanglegram that has a seesaw-optimal drawing that is inferior to its optimal drawing. By iteratively replacing the leaves with copies of the drawing, we create a chain of seesaw-optimal drawings with a quadratically increasing number of crossings, while the optimal crossing number stays small. From this we describe planar tanglegrams of arbitrarily large size and seesaw-optimal drawings with $\Omega(n^2)$ crossings.

## 7 Conclusion and Open Problems

We have shown several significantly faster algorithms for tanglegram drawing, including for planar, k-crossing, and one-tree optimization problems. We have also introduced the footrule distance metric for tanglegrams and shown an efficient one-tree drawing algorithm. We conjecture that the two-tree distance minimization problem is NP-complete. Future work includes improving drawing heuristics for tanglegrams with the distance metric. Our ILP solution for the crossing metric is efficient, but the ILP solution for the distance problem is slower and may perhaps be improved. It also remains to explore the seesaw method for the distance heuristic, though we have shown it can be larger than the optimal solution by $\Omega(n^2)$ in the crossing case. For the one-tree problem, though distance between permutations can be computed in linear time (while counting crossings takes $\Omega(n \log n)$), distance seems the harder measure to optimize.

## References


[1] M. S. Bansal, W.-C. Chang, O. Eulenstein, and D. Fernández-Baca. Generalized binary tanglegrams: Algorithms and applications. In *BiCoB*, 2009.

[2] P. Bertolazzi, G. D. Battista, C. Mannino, and R. Tamassia. Optimal upward planarity testing of single-source digraphs. *SIAM J. Comput.*, 27(1):132–169, 1998.

[3] T. C. Biedl, F.-J. Brandenburg, and X. Deng. Crossings and permutations. In *Graph Drawing*, pages 1–12, 2005.

[4] K. Buchin, M. Buchin, J. Byrka, M. Nöllenburg, Y. Okamoto, R. I. Silveira, and A. Wolff. Drawing (complete) binary tanglegrams: Hardness, approximation, fixed-parameter tractability. In *Graph Drawing*. Springer-Verlag, 2008.

[5] A. Burt and R. Trivers. *Genes in Conflict*. Belknap Harvard Press, 2006.

[6] P. Diaconis and R. L. Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(2):262–268, 1977.
[7] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW*, pages 613–622, 2001.
[8] T. Dwyer and F. Schreiber. Optimal leaf ordering for two and a half dimensional phylogenetic tree visualisation. In *Australasian Symp. on Info. Vis.*, pages 109–115, 2004.
[9] R. D. M. P. (Ed.). *Tangled Trees: Phylogeny, Cospeciation, and Coevolution*. University Of Chicago Press, 2002.
[10] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists. In *SODA*, pages 28–36, 2003.
[11] H. Fernau, M. Kaufmann, and M. Poths. Comparing trees via crossing minimization. In *FSTTCS*, pages 457–469, 2005.
[12] L. R. Foulds and R. L. Graham. The Steiner problem in phylogeny is NP-complete. *Adv. in Appl. Math.*, 3(1):43–49, 1982.
[13] M. Garey and D. S. Johnson. Crossing number is np-complete. *SIAM Journal on Algebraic and Discrete Methods*, 4:312–316, 1983.
[14] R. Hinze and R. Paterson. Finger trees: A simple general-purpose data structure. *Journal of Functional Programming*, 16(2):197–217, 2006.
[15] J. E. Hopcroft and R. E. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.
[16] J. P. Huelsenbeck and F. Ronquist. Mrbayes: Bayesian inference of phylogeny, 2001.
[17] H. Kaplan and R. E. Tarjan. Purely functional representations of catenable sorted lists. In *STOC '96*, pages 202–211. ACM, 1996.
[18] K. Kawarabayashi and B. Reed. Computing crossing number in linear time. In *STOC*, pages 382–390, 2007.
[19] J. Lee. All-different polytopes. *Journal of Combin. Optim.*, 6(3):335–352, 2002.
[20] A. Lozano, R. Y. Pinter, O. Rokhlenko, G. Valiente, and M. Ziv-Ukelson. Seeded tree alignment and planar tanglegram layout. In *WABI*, pages 98–110, 2007.
[21] D. M.Hillis, T. Heath, and K. S. John. Analysis and visualization of tree space. *Systematic Biology*, 3:471–482, 2005.
[22] M. Nöllenburg, D. Holten, M. Völker, and A. Wolff. Drawing binary tanglegrams: An experimental evaluation. In *ALENEX*, pages 106–119. SIAM, 2009.
[23] S. Roch. A short proof that phylogenetic tree reconstruction by maximum likelihood is hard. *IEEE/ACM Trans. Comp. Biol. and Bioinf.*, 3(1):92–94, 2006.
[24] W. K. Shih and W.-L. Hsu. A new planarity test. *Theor. Comput. Sci.*, 223(1-2):179–191, 1999.
[25] D. Swofford. *PAUP*. Phylogenetic Analysis Using Parsimony (*and Other Methods). Version 4*. Sinauer Associates, Sunderland, Massachusetts, 2002.
[26] D. L. Swofford, G. J. Olsen, P. J. Waddell, and D. M. Hillis. Phylogenetic inference. In *Molecular Systematics, Second Edition*, pages 407–514. Sinauer, 1996.
[27] B. Venkatachalam, J. Apple, K. S. John, and D. Gusfield. Untangling tanglegrams: Comparing trees by their drawings. Technical Report CSE-2009-1, UC Davis, CS, 2009.
[28] W. N. Wan Zainon and P. Calder. Visualising phylogenetic trees. In W. Piekarski, editor, *Seventh Australasian User Interface Conference (AUIC2006)*, volume 50 of *CRPIT*, pages 145–152, Hobart, Australia, 2006. ACS.