Untangling Tanglegrams: Comparing Trees by their Drawings *

Balaji Venkatachalam[†], Jim Apple[†], Katherine St. John[‡], Dan Gusfield[†]

December 24, 2009

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 also studied by Fernau, Kauffman and Poths (*FSTTCS 2005*). A similar reduction to a graph crossing problem also helps to solve an open problem 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 criterion using Spearman's footrule distance and give an $O(n^2)$ algorithm.

We also show integer programming formulations to quickly obtain tanglegram drawings that minimize the two optimization measures 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 the 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 [32]. The popular criteria to reconstruct an optimal tree – maximum parsimony and maximum likelihood

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

[†]Department of Computer Science, UC Davis. {balaji, apple, gusfield}@cs.ucdavis.edu

[‡]Department of Mathematics and Computer Science, Lehman College, and the Graduate Center, City University of New York. stjohn@lehman.cuny.edu

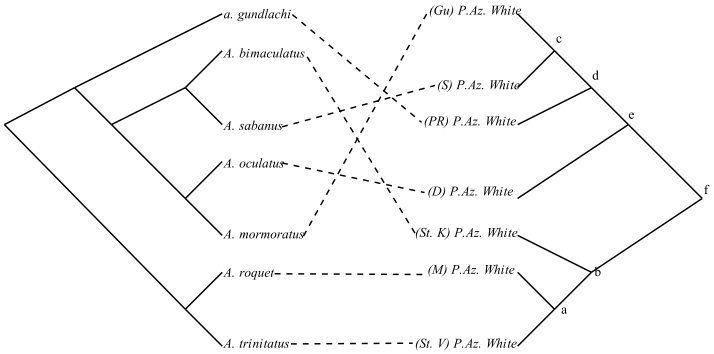

Figure 1: A tanglegram from Charleston and Perkins [9]: phylogenetic trees for lizards in the Caribbean tropics and strains of malaria found there ([9], p 86), joined by dashed lines that represent the parasite-host relationship. The crossing number is 7, and the footrule distance is 10. This is not optimal; an alternative layout which interchanges the children of nodes c and d improves these to 4 and 6, respectively. The optimal drawings have crossing number 1 and distance 2, respectively.

– are NP-hard [16, 29], so heuristic methods (i.e. [22, 31]) 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 [18].

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 Fig. 1 and Sect. 2). Tanglegrams are widely used in biology, including, to compare evolutionary histories of host and parasite species and to analyze genes of species in the same geographical area [28, 34].

Another potential application of the crossing distance in tanglegrams is to gauge the extent of *horizontal gene transfer*. This application follows from the discussion of the phylogenies of the transposable element ω and its hosts in [8, pg. 204–206]. They draw the two trees as shown in Fig. 2 shows a drawing of the phylogeny for ω and the phylogeny for its hosts with twelve crossings, and concludes that "The phylogeny of ω and its hosts are significantly different, indicating frequent horizontal transmission." This motivates the desire to have a method to minimize the number of crossings. There are twelve crossings in Fig. 2 (with fourteen leaves); there is an optimal layout with only three crossings as shown in Fig 3 Would the authors still consider the phylogenies to be "significantly different"?

Drawings with fewer crossings or with matching leaves close together are useful in biological analysis. A drawing imposes an order among the leaves of the tree. Therefore comparing the drawings of the trees is equivalent to comparing the permutations of the leaves. We focus on two natural measures of complexity that are used for comparing permutations: the crossing number (or Kendall-τ) and Spearman's footrule distance [10]. These measures are widely used, including, in ranking search results on the web and in voting systems [14, 12]. In comparing tanglegrams, the former measures the number of times edges between the leaves cross, and the latter, the proximity

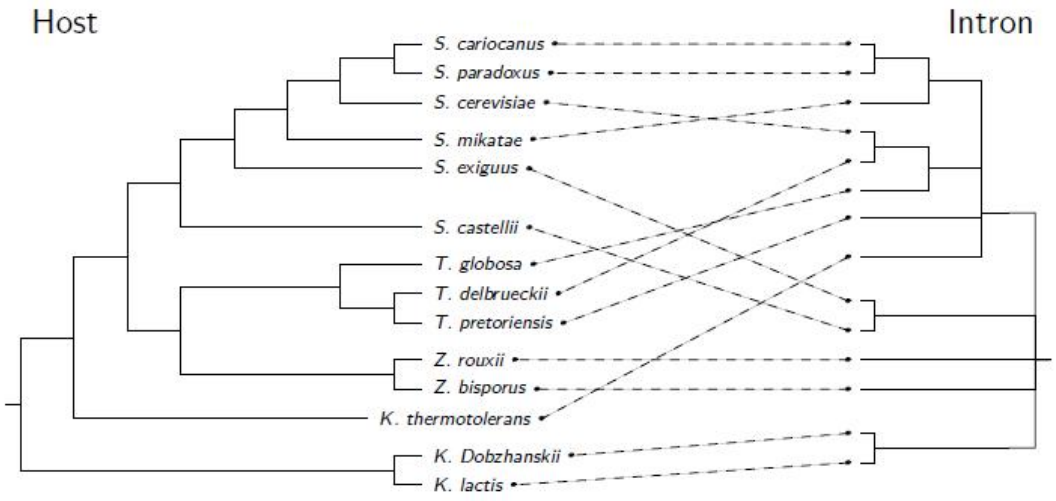

Figure 2: Fig. 6.6B on p. 206 in [8]. This drawing has twelve crossings.

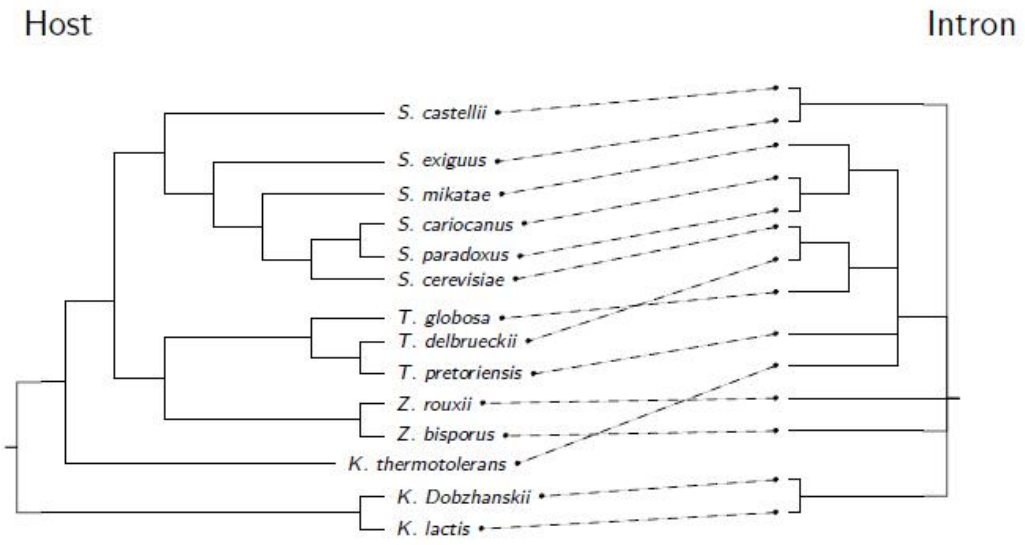

Figure 3: The optimal layout for the tanglegram show in Fig. 2 has three crossings.

of the leaves in the two trees. By Spearman's footrule, the distance between a pair of drawings is the sum of the distances between the positions of the leafs in the permutations. These are defined more formally in the next section. We focus on the complexity of these ranking problems and give efficient algorithms for drawing tanglegrams. The results in this paper were first presented at the ISBRA conference [33].

Algorithmic results for tanglegram drawing problems Crossing minimization in tanglegrams has parallels to crossing minimization in graphs. Computing the minimum number of crossings in a graph is NP-complete [17]. However, it can be verified in linear time that a graph has a planar drawing (with zero crossings) [21, 30]. Computing the minimum number of crossings is fixed-parameter tractable [24]. Analogously, crossing minimization in tanglegrams is NP-complete, while the special case of planarity can be decided in linear time [15]. Fernau et al. [15] also showed an FPT algorithm when the trees are binary.

Fernau et al. [15] showed the test for planarity by a reduction to the upward flow problem [2]. Independently, Lozano et al. [26] showed a simple dynamic programming based solution that gives a planar drawing in $O(n^2)$ time. Fernau et al. [15] also showed the NP-completeness by a reduction to MAX-CUT problem and a fixed-parameter algorithm for binary trees. In recent work, Buchin et al. [7] showed a fixed parameter tractable algorithm for complete tanglegrams (where every leaf has the same depth). The running time was further improved by Böcker et al. [5].

In this work we use the parallel between drawing graphs and tanglegrams described above. We show a new characterization to utilize results from graph drawing literature. For planarity testing our reduction adds an edge between the roots of the trees and runs the planar graph drawing algorithm. This method also leads to a fixed-parameter algorithm to minimize the number of crossings. Our algorithm improves the running time of Fernau et al. [15] and answer their conjecture for d-ary trees for $d > 2$ in the negative. Unlike the previous methods our algorithms for fixed-parameter tractability do not require restrictions and hold for arbitrary trees.

The problem of minimizing the number of crossings where one tree is fixed and the layout of the other tree is allowed to vary is called the *one-tree crossing minimization (OTCM)* problem and has been studied previously. Dwyer and Schreiber shows an $O(n \log n)$ algorithm for balanced trees. For binary trees of arbitrary topology, Fernau et al. [15] 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 $O(n \log n)$.

Previous work on tanglegrams is limited to crossing minimization. We introduce 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. The complexity of minimizing the distance when both trees can be modified is open.

For the praxis of tanglegram drawing, we show integer programming formulations to obtain tanglegram drawings that minimize the two optimization measures discussed. This work was independent of Nöllenburg et al. [27] who show an integer programming formulation like ours.

Dwyer and Schreiber [13] proposed a heuristic for crossing minimization in tan-

glegrams using the polynomial algorithm for OTCM. We show a lower bound on the worst case behavior of this heuristic.

In other related work, Buchin et al. [7] showed that under certain complexity theoretic assumptions there are no constant factor approximation algorithms for minimizing crossings. They show that crossing minimization is NP-hard even for complete tanglegrams. They also show an $O(n^3)$ time 2-approximation algorithm for complete tanglegrams. Bansal et al. [1] define a *generalized tanglegram* where the number of leaves in the two trees may be different and a leaf in one tree may match multiple leaves in the other tree. Dwyer and Schreiber [13], Zainon and Calder [34] and Holten and van Wijk [20] implement various heuristics for visualizing tanglegrams. Nöllenburg et al. [27] show experimental evaluation of some heuristics, an exact branch-and-bound algorithm and an ILP formulation.

The rest of the paper is organized as follows. After giving formal definitions in section 2, we first discuss one-tree optimization problems 3. We then discuss our reduction method for two-tree crossing minimization problems in section 4. Integer programming solutions for both crossing and distance minimization problems are discussed in section 5. Finally, in section 6 we show bounds on the worst case behavior of the Dwyer and Schreiber heuristic.

2 Preliminaries

We define tanglegrams and their drawings following [28, 15]: 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 π_i be the permutation on the leaves induced by $<_i$, $i = 1, 2$, $\pi_i : L(T_i) \rightarrow [n]$. Then, Spearman's **footrule distance** [4, 10] 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 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 $(T_1, T_2; M)$.
RESULT: $<_1, <_2$ such that $cr(T_1, T_2, M, <_1, <_2)$ is minimal.

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 all the 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 [13, 15], we exploit the optimal substructure property of the problem and recursively work on the subtrees. Our results are due to the 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, $cr(T_1, T_2, M, <_1, <_N) < cr(T_1, T_2, M, <_1, <_2)$, contradicting the optimality of $<_2$. □

Our algorithm uses *2-3 finger trees* [6, 23] to maintain the list of leaves in each subtree. 2-3 finger trees are ordered search tree with fast split and append operations. At every internal node, the lists of leaves in the subtrees are compared to count the number of inversions and merged. The run-time analysis follows the analysis merging sorted lists in Brown and Tarjan [6].

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.

The base case 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 [23].

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.

Algorithm 1 Container merging for the OTCM problem. The inputs p, q, and the output $result$ are finger trees sorted according to $<_1$.

```
1:  count ← 0
2:  result ← ⟨⟩
3:  while |q| > 0 do
4:      (q_h, q) ← head/tail(q) // pops the first element of q
5:      (r, p) ← split(p, q_h) // splits p, removes elements less than q_h into r
6:      result ← result ++ r // appends elements smaller than q_h.
7:      result ← result ++ ⟨q_h⟩
8:      count ← count + |p| // the number of crossings for q_h
9:  end while
10: result ← result ++ p
11: return (count, result)
```

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, while maintaining a count of the number of crossings incurred. Switching the order of the arguments in the algorithm reveals the number of crossings if the layout of the two child nodes is switched, from which we can determine the optimal layout for these nodes.

Complexity Kaplan and Tarjan [23] describe split and append ($+\!\!+$) operations on 2-3 finger trees. The operation $(t_L, t_R) \leftarrow \mathrm{split}(t, v)$ takes $O(\log(1 + \min(|t_L|, |t_R|)))$

time, and $t_1 +\!\!+ t_2$ takes $O(\log(1 + \min(|t_1|, |t_2|)))$ time. Therefore, the head/tail split on line 4 and append on line 7 take only $O(1)$ time. The values $|p|$ and $|q|$ can be computed in $O(1)$ time as shown by Hinze and Patterson [19], where they maintain the trees with size information.

The call to split in line 5 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 ith element in q, the total time taken in line 5 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 6 and 10 as well; these lines append to $result$ all of p, in $|q| + 1$ pieces. W.l.o.g., we will assume $|p| > |q|$.

The total complexity is bounded by the shared complexity of lines 5, 6 and 10. Since the sum of the logarithms is maximized when all the d_is are equal [23], 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, and $l \leq r$ and $l + r = n$. Using induction, assume that $T(m) \in O(m \log m)$ for all $m < n$.

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

□

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. 4 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.

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

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.

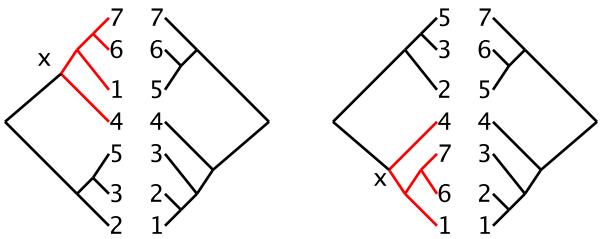

Figure 4: One-tree footrule distance minimization with respect to the identity permutation $1, \ldots, 7$. Consider the configuration of the subtree rooted at x. In the left figure, the configuration of this subtree is optimal and contributes 4 to the overall distance. If the same layout were to be used 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.

4 A Reduction Method for Two-tree Crossing Problems

When only one tree is allowed to vary, the crossing minimization problem (OTCM) can be solved in polynomial time. But the TTCM problem, where both the trees are mutable, is NP-hard [15]. However, the problem is fixed-parameter tractable and the special case of checking if a tanglegram has a drawing with zero crossings (planarity testing) can be solved in linear time [15]. In this section we show a new characterization that helps us to use the graph drawing algorithms to obtain planar drawings and a fixed-parameter algorithm.

4.1 Two-tree Tanglegram Planarity

A graph is *planar* if it can be drawn on a plane such that no two edges cross, i.e., apart from its endpoints an edge contains no point in common with any other edge. In a planar drawing every vertex is placed as a point on the plane and edges are drawn as curves between the vertices. The interior of an edge is the set of points on the curve not including its end points.

To avoid planar drawings of the tanglegrams that are not proper we will use a natural extension of tanglegrams.

Definition 1. *An* augmented *tanglegram is a tanglegram with the roots of the two trees joined by an edge. This edge is called the* augmented edge *and labelled A.*

To prove the next lemma we need some definitions and results on *plane duals* of a graph. The brief discussion below follows Diestel [11, Sec. 4.6]. For more details on plane graphs, please refer to chapter 4 in Diestel [11].

Let $G = (V, E)$ be a planar multigraph and let $F(G)$ be its faces. A graph (V^*, E^*) with faces $F(V^*, E^*)$ is called a *plane dual* of G, if there are bijections

$$\begin{array}{ccc} F \to V^* & E \to E^* & V \to F^* \\ f \mapsto v^*(f) & e \mapsto e^* & v \mapsto f^*(v) \end{array}$$

and satisfies the following conditions:

(i) $v^*(f) \in f$ for all $f \in F$;

(ii) $|e^* \cap G| = |\mathring{e} \cap \mathring{e}^*| = |e \cap G^*| = 1$ for all $e \in E$, where $\mathring{e}, \mathring{e}^*$ are the interiors of the edges e, e^*. That is, the edges e, e^* intersect at exactly one point in the interior of the edges;

(iii) $v \in f^*(v)$ for all $v \in V$.

We denote such a graph (V^*, E^*) as G^*.

We can construct a plane dual G^* as follows. We pick a point $v^*(f)$ in each face f of G. The vertices are connected by edges e^* satisfying (ii) above. It can then be shown that there is a bijection $V \to F^*$ satisfying (iii).

It is easy to show that if G^* is a plane dual of G, then G is plane dual of G^*.

Note that by the bijection on edges, for every edge e in G, there is a corresponding edge e^* in G that intersects e on the plane. For a set of edges C in G the corresponding set of edges in G^*, $\{e^*|e \in C\}$, is represented as C^*.

We need the following result for our proof.

Proposition 3. *([11, Prop. 4.6.1]) For any connected plane multigraph G, an edge set $C \subseteq E(G)$ is the edge set of a cycle of G iff $C^* := \{e^*|e \in C\}$ is a minimal cut in G^*.*

Since G and G^* are duals of each other, a minimal cut in in G corresponds to a cycle in G^*.

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

Proof. The "only if" direction of the lemma is straightforward. For the other direction, suppose that the augmented tanglegram G has a planar drawing. The set of matching edges between the leaves, M, together with the augmented edge A separate G into two trees. Therefore, $C = M \cup A$ is a cut. It is also a minimal cut because no proper subset of C is a cut in G. By proposition 3, there is a cycle in the plane dual G^* corresponding to a minimal cut in G. Consider the cycle C^* in G^* corresponding to this minimal cut. C^* has $n + 1$ vertices and edges. There is an edge A^* in the dual graph that intersects the augmented edge A. Let A^* be an edge between vertices u^*, v^* in the dual graph.

The intuition of the proof is the following. If there is a proper planar drawing of the augmented tanglegram, then one of the faces corresponding to the vertices u^* or v^* corresponds to the unbounded face, and the edges of the cycle intersect the matching edges in M in a linear order. We will deduce this linear order by the sequence in which the edges of the cycle C^* intersects the edges of M.

Let the vertices of cycle C^* be $u^*, u_1^*, \ldots, u_{n-1}^*, v^*, u^*$, in order. Consider the sequence of matching edges that are intersected by the edges $(u^*, u_1^*), (u_1^*, u_2^*), \ldots, (u_{n-1}^*, v^*)$. This sequence defines a linear order on the leaves.

Now, $T' = (V(T_1) \cup L(T_2), E(T_1) \cup \{M\})$ is a tree, where $L(T_2)$ are the leaves of T_2. In the planar drawing, the curve formed by the edges of $C^* = M^* \cup \{A^*\}$ intersect the edges M in some order, $<'$. Therefore there is a proper drawing of the tree T' such

that the leaves are ordered by $<'$. Similarly there is a drawing of the second tree with the same linear order. These two drawings of the tree can be pasted together to form a proper planar drawing of the tanglegram. □

Theorem 5. *Whether a tanglegram admits a planar drawing can be decided in linear time.*

Proof. Apply a planar graph drawing algorithm on the augmented tanglegram. Since the planar graph drawing algorithm runs in linear time [21, 30], from the previous lemma we can decide the planarity of the input tanglegram in linear time. □

We can build the dual of the planar drawing in linear time. Since the proof of lemma 4 is constructive, we have a linear time algorithm to obtain the planar drawing of the tanglegram, if one exists.

Please refer to the appendix for an alternate construction using edge contraction, instead of constructing the dual as above.

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. [15]. 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 reduce the crossing minimization of tanglegrams to the crossing minimization problem in graphs. We use the elegant work of Kawarabayashi and Reed [24] to provide an FPT algorithm for TTCM, thus answering the conjecture of Fernau et al. [15] in the negative.

Like in the planar graph drawing problem, we create an augmented tanglegram but in this case we also 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.

Let $T = (T_1, T_2; M)$ be a tanglegram. We build a graph $G = (V, E)$ based on the augmented tanglegram of T. As described earlier we want to augment the tanglegram with duplicate edges that would forbid drawings with crossings involving internal edges. This is pictorially represented in Fig. 5. Since we do not want multiple edges between a pair of vertices we add a new vertex in the middle of each duplicate edge to make the edges unique.

Let r_1 and r_2 denote the roots of T_1 and T_2, and $IE = E(T_1) \cup E(T_2) \cup \{(r_1, r_2)\}$. We will create a new graph G as follows. Let the vertices of G be

$$V(G) = V(T_1) \cup V(T_2) \cup \bigcup_{e_i \in IE} \{v_{i,1}, \ldots, v_{i,n}\}$$

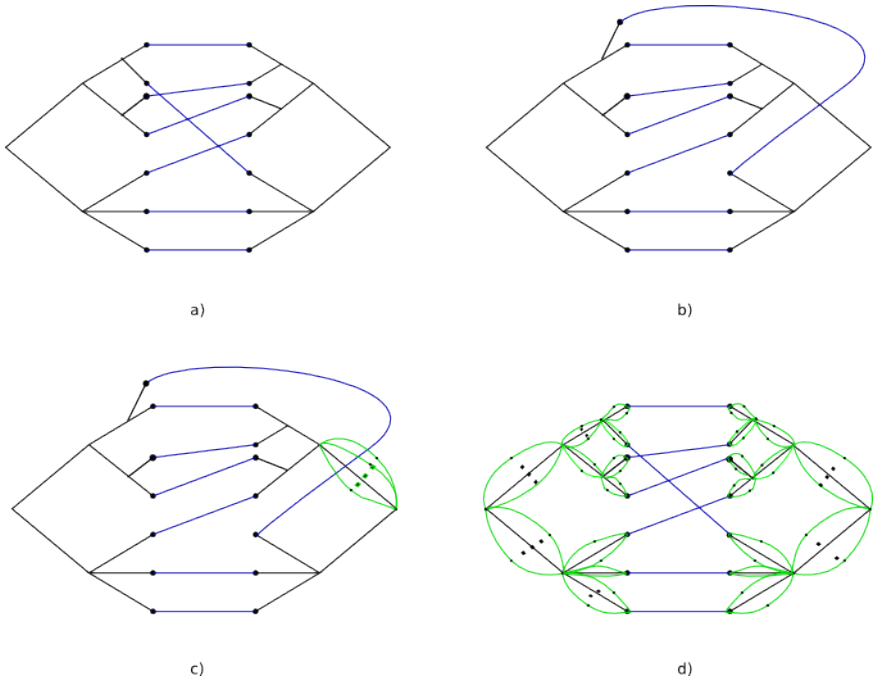

Figure 5: a) a tanglegram instance; b) an alternative drawing of the augmented tanglegram that has a lower crossing number; c) to avoid bad drawings like b), the internal edge is "thickened" by n edges; and d) the graph G encoding the n-augmented tanglegram from a).

and the edges of G be

$$E(G) = IE \cup \bigcup_{\substack{e_i=(x_i,y_i)\in IE,\\ j=1,\ldots,n}} \{(x_i v_{i,j}), (v_{i,j} y_i)\} \ .$$

We call G, the n-*augmented* tanglegram of T.

Lemma 6. *In any drawing of the n-augmented tanglegram with the minimal number of crossings, only the matching edges cross.*

Proof. Since a pair of edges can cross at most once, if an edge e crosses $(x_i, v_{i,j})$ or $(v_{i,j}, y_i)$ for some $(x_i, y_i) \in IE$ and $j \in [n]$, then it must cross n such edges in an optimal drawing. If any two edges in IE cross they will produce n^2 crossings which is more than any proper drawing will produce. Therefore every crossing must involve a matching edge.

Suppose that a matching edge e crosses $(x_i, v_{i,j})$ or $(v_{i,j}, y_i)$ for some $(x_i, y_i) \in IE$ and $j \in [n]$, then it must cross n such edges, as we argued before. Let $z_1, \ldots, z_l$ be the leaves descending from (x_i, y_i) in the respective tree. Then, e can be redrawn

to cross the matching edges from the leaves $z_1, \ldots, z_l$. This redrawing reduces the number of crossings to $l < n$.

Therefore, in any optimal solution only matching edges cross. □

Lemma 7. *If there is a drawing with k crossings in which only matching edges cross then there is a proper drawing with at most k crossings.*

Proof. The proof is similar to the proof of lemma 4. The "only if" direction is straight forward. To prove the other direction suppose that the augmented tanglegram has k crossings. Since this is not planar we create a new graph by adding vertices where the edges cross. Further, for every leaf l in tree T_i for $i \in \{1, 2\}$, add a vertex v_l on the matching edge out of l closer to the leaf than the crossing vertices added previously. Let $M_i = \{(l, v_l) : l \text{ is a leaf in } T_i\}$ for $i \in \{1, 2\}$. We call the new graph $\bar{G}$. $\bar{G}$ is planar by construction. If the drawing of the augmented tanglegram had k crossings then $|V(\bar{G})| = |V(G)| + 2n + k$.

The edges of M_1 together with the n augmented edges form a minimal cut in $\bar{G}$, therefore the corresponding dual edges form a cycle of size $2n$ in $\bar{G}^*$. Now the dual edges cross the edges in M_1 in some order starting with the one closest to the augmented edges. As we argued in lemma 4 this sequence of leaves form a linear order in a proper drawing. Similarly we get a proper drawing of T_2 using the edges of M_2 and the augmented edges.

Pasting these layouts together we get a proper drawing of the tanglegrams with k crossings. □

Please refer to the appendix for an alternate construction using edge contraction, instead of constructing the dual as above.

Theorem 8. *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.*

Proof. Given a tanglegram T of size n we create an n-augmented tanglegram G. The size of G is $O(n^2)$. We will use the linear-time algorithm by Reed and Kawarabayashi [24] on G to obtain a drawing with the optimal number of crossings. By lemma 6 only matching edges cross. And by lemma 7 we can redraw this graph to obtain a proper drawing for the tanglegram T. These steps take time linear in the size of G, i.e., quadratic in n.

The factor involving k in the running time depends on which of the subroutines is used in the algorithm of [24], and the choice of the subroutine depends on various factors, viz., the tree-width, face-width, embeddability in a surface of small genus.

□

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 running time 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.

To formulate an integer program for a given instance, we introduce variables for the nodes of the trees. The relative position of the leaves is determined by the order of the children at the internal nodes. These constraints are represented by linear equations on the variables. The objective, to minimize the number of crossings or the footrule distance, is also represented by a linear expression on the variables. We use the linear programming solver, CPLEX, to solve the linear program. The values for the variables determine an optimal drawing for the tanglegram. The details follow.

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, for the first tree, 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. In a layout y_k is set to 1 if c_1, c_2 are placed to the left and right, respectively, otherwise $y_k = 0$. For all leaves i in the subtree below c_1 and j in the subtree below c_2, if $i < j$ then $x_{i,j} = 1 \iff y_k = 1$, i.e., $x_{i,j} = y_k$. 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 the following linear inequalities. $z_{i,j} - x_{i,j} + x'_{i,j} \geq 0; z_{i,j} + x_{i,j} - x'_{i,j} \geq 0; z_{i,j} - x_{i,j} - x'_{i,j} \leq 0; z_{i,j} + x_{i,j} + x'_{i,j} \leq 2$.

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 (DP) formulation In the dynamic programming algorithm for one-tree distance minimization (section 3.2), every subtree is rooted at every possible position so that its leaves are located starting at position i for all $i \in [n]$. Here we will generate equations to allow placing each subtree of either tree at every position. The constraints will eliminate mutually incompatible configurations. The sum of the distances of the matching edges can be calculated for each legal layout of the trees. The objective is to determine the optimal solution among all possible layouts.

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 its children with l and r leaves in the subtrees below them. Either i is placed as the left child or the right child. If i is placed as the left child then its leaves take positions p through $p + l - 1$ and the leaves below j take positions $p + l$ through $p + l + r - 1$.

$y_{k,p} = 1$ implies that the 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.

To calculate the distance contributed by each leaf we introduce the variable $z_{l,r,r'}$. 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 $\min \sum_{\text{leaf } l} \sum_{r\in[n]} \sum_{r'\in[n]} |r - r'| z_{l,r,r'}$. The absolute values can be converted to simple linear terms by standard techniques [3].

Related Distance (RD) formulation In this formulation we will use the relative distance between the pair of leaves on the same tree. This distance is determined by the the order of the children at the least common ancestor.

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 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 $y_i = 0$. Now the order of the children c_1, c_2 determine the distance between the leaves in its subtrees.

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

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

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 [25]. 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 $\min \sum_i |x_i - x'_i|$. As before, we will convert the absolute values to linear forms using standard techniques [3].

5.3 Timing

The performance of these formulations were tested on a random set of tanglegrams. A random tanglegram is given by a pair of random trees. To generate a random tree we

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**	**RD**		**DP**	
	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

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 ILP formulation for the crossing minimization problem is very fast. Both the ILP formulations for the distance version are slower in comparison.

For the distance minimization problem, the relative 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 Buchin et al. [7] shows that, under certain complexity theoretic assumptions, there is no constant-factor approximation algorithm for TTCM, Dwyer and Schreiber [13] present a heuristic 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 converged to a good solution after ten or fewer iterations. We call this approach "seesawing".

For a tanglegram $(T_1, T_2; M)$, we say a drawing $(<_1, <_2)$ is **seesaw-optimal** if it cannot be improved by seesawing, that is, if

$$cr(T_1, T_2, M, <_1, <_2) = cr(T_1, T_2, M, \cdot, <_2) = cr(T_1, T_2, M, <_1, \cdot) \ .$$

We prove the lower bound on the worst case performance of this heuristic 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. In an extreme case, there are planar tanglegrams of size $\Omega(n)$ with drawings with $\Omega(n^2)$ crossings that cannot be improved by seesawing.

Theorem 9. *For any N, there is an $n > N$, a tanglegram of size n, and a seesaw-optimal drawing of that tanglegram with $\Omega(n^2)$ more crossings than an optimal drawing.*

Lemma 10. *Let A be a tanglegram with k leaves per tree, p crossings in an optimal layout, and q crossings in a seesaw-optimal layout, and B be a tanglegram with m leaves per tree, s crossings in an optimal layout, and r crossings in a seesaw-optimal layout. Then there is a tanglegram $\texttt{over}(A, B)$ with km leaves per tree, $ks + pm^2$ or fewer crossings in an optimal layout, and $kr + qm^2$ crossings in a seesaw-optimal layout.*

Proof. Let A be the tanglegram $(A_1, A_2; M_A)$, and let B be the tanglegram $(B_1, B_2; M_B)$. Assume, w.l.o.g., that the leaves in A are in $[k]$ and that M_A is the identity matching. Assume that the leaves in B are in $[m]$ and that M_B is the identity matching.

We form a new tanglegram $\texttt{over}(A, B) = (C_1, C_2; M_C)$ from A and B by replacing each leaf i in A_j with a tree $B'_{j,i}$, for $j \in \{1, 2\}$. $B'_{j,i}$ is formed by replacing each leaf l in B_j with a leaf labeled $(i-1)m + l$. $\texttt{over}(A, B)$ has km leaves in each tree. We use the identity matching for M_C.

Before considering seesaw-optimality, we show that an optimal layout of $\texttt{over}(A, B)$ has $ks + pm^2$ or fewer crossings. We can see this by picking an optimal layout for the embedded A and each embedded B in $\texttt{over}(A, B)$. Each B incurs s crossings. Since there are k of them, they contribute ks crossings. The embedded A incurs p crossings, but each one of these is, instead of a crossing of leaves, a crossing of B_1 and B_2. Each of these crossings creates m^2 leaf crossings, leaving a total of $ks + pm^2$ crossings in this layout of $\texttt{over}(A, B)$. Any optimal layout must have at most this many crossings.

We will now pick a drawing for $\texttt{over}(A, B)$ and show that it has $kr + qm^2$ crossings and is seesaw-optimal. The drawing that we pick is one where the copy of A is seesaw-optimal and each copy of B is seesaw-optimal.

The argument for the number of crossings is the same as that for the upper bound on the optimal crossings. To show that this layout is seesaw-optimal, we need to show that fixing one leaf order and solving OTCM on the other tree can not decrease the number of crossings. We begin seesawing by finding the layout that minimizes $cr(C_1, C_2, M_C, \cdot, <_2)$. The analysis below would apply just as well if we were to fix $<_1$.

By Lemma 1, after fixing $<_2$, any optimal layout for a copy of B_1 in C_1 is an optimal drawing for B_1 in isolation. Above the copies of B_1 we have a copy of A_1. Each pair of leaves c, c' of A has been replaced by a pair of copies of B_1 such that if, say, $c < c'$, then $d < d'$ for all leaves $d \in B'_{1,c}$ and $d' \in B'_{1,c'}$. Using Algorithm 1, any optimal drawing for the copy of A_1 in C_1 is reached by the same procedure as a drawing for the original A_1 was reached. It therefore can not reduce the number of crossings. □

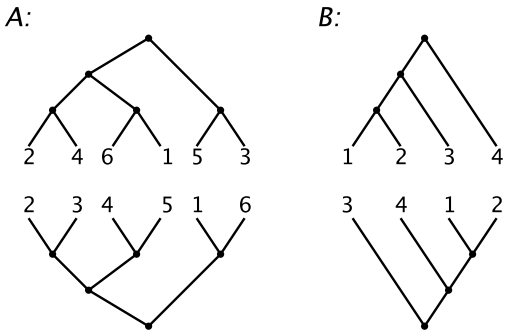

Figure 6: Tanglegrams for use in Theorem 9. A has a drawing with only 1 crossing, but seesawing can only find a drawing with 5 crossings. B has a planar drawing, but seesawing can only find a drawing with 1 crossing.

Given a tanglegram A with a leaves, b crossings in a seesaw-optimal layout, and c crossings in an optimal layout. Iterating $\texttt{over}(A, A)$ $m-1$ times produces a tree with a^m leaves, $ba^{m-1}\left(\frac{a^m-1}{a-1}\right)$ crossings in a seesaw-optimal layout, and $ca^{m-1}\left(\frac{a^m-1}{a-1}\right)$ crossings in an optimal layout. The difference between the optimal and seesaw-optimal layouts, in this tree of size a^m, is greater than $(b-c)(a^{2m-2})$, which is $\Omega((a^m)^2)$.

To complete the proof, we need only an example of a tanglegram drawing with a seesaw-optimal layout that is not optimal. Two examples are provided in Fig. 6. The figure on the right shows that we can even find planar tanglegrams where seesawing only finds layouts 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 measure for tanglegrams and shown an efficient one-tree drawing algorithm. We conjecture that the two-tree footrule distance minimization problem is NP-complete.

Future work includes improving drawing heuristics for tanglegrams with the distance measure. Our ILP solution for the crossing measure is efficient, but the ILP solution for the footrule 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 permutations, footrule distance can be computed in linear time while counting crossings takes $\Omega(n \log n)$ time. However, for tanglegrams crossing minimization takes $\Omega(n \log n)$ time while footrule distance seems harder to optimize.

Acknowledgement We would like to thank an anonymous reviewer for suggesting an alternate proof for Lemma 4 and 6 and a number of other suggestions to improve the manuscript.

References

[1] M. S. Bansal, W.-C. Chang, O. Eulenstein, and D. Fernández-Baca. Generalized binary tanglegrams: Algorithms and applications. In *Bioinformatics and Computational Biology, First International Conference, (BICoB)*, LNCS 5462, pages 114–125, 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] D. Bertsimas and R. Weismantel. *Optimization Over Integers*. Dynamic Ideas, Belmont, Massachusetts, May 2005.

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

[5] S. Böcker, F. Hffner, A. Truss, and M. Wahlström. A faster fixed-parameter approach to drawing binary tanglegrams. In *The Fourth International Workshop on Parameterized and Exact Computation (IWPEC 2009)*, LNCS 5917, 2009.

[6] M. R. Brown and R. E. Tarjan. Design and analysis of a data structure for representing sorted lists. *SIAM J. Comput.*, 9(3):594–614, 1980.

[7] 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*, pages 324 – 335. Springer-Verlag, 2008.

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

[9] M. Charleston and S. Perkins. Lizards, malaria, and jungles in the Caribbean. In R. Page, editor, *Tangled Trees: Phylogeny, Cospeciation, and Coevolution*, pages 65–92. University Of Chicago Press, 2003.

[10] 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.

[11] R. Diestel. *Graph Theory*. Graduate Texts in Mathematics, Volume 173. Springer-Verlag, third edition edition, 2005.

[12] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW*, pages 613–622, 2001.

[13] 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.

[14] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists. In *SODA*, pages 28–36, 2003.

[15] H. Fernau, M. Kaufmann, and M. Poths. Comparing trees via crossing minimization. In *FSTTCS*, pages 457–469, 2005.

[16] L. R. Foulds and R. L. Graham. The Steiner problem in phylogeny is NP-complete. *Adv. in Appl. Math.*, 3(1):43–49, 1982.

[17] M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 4(3):312–316, 1983.

[18] D. M. Hillis, T. A. Heath, and K. St. John. Analysis and visualization of tree space. *Systematic Biology*, 54(3):471–482, 2005.

[19] R. Hinze and R. Paterson. Finger trees: A simple general-purpose data structure. *Journal of Functional Programming*, 16(2):197–217, 2006.

[20] D. Holten and J. J. van Wijk. Visual comparison of hierarchically organized data. *Comput. Graph. Forum*, 27(3):759–766, 2008.

[21] J. E. Hopcroft and R. E. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974.

[22] J. P. Huelsenbeck and F. Ronquist. Mrbayes: Bayesian inference of phylogeny, 2001.

[23] H. Kaplan and R. E. Tarjan. Purely functional representations of catenable sorted lists. In *STOC*, pages 202–211. ACM, 1996.

[24] K. Kawarabayashi and B. Reed. Computing crossing number in linear time. In *STOC*, pages 382–390, 2007.

[25] J. Lee. All-different polytopes. *Journal of Combin. Optim.*, 6(3):335–352, 2002.

[26] 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.

[27] M. Nöllenburg, D. Holten, M. Völker, and A. Wolff. Drawing binary tanglegrams: An experimental evaluation. In *ALENEX*, pages 106–119. SIAM, 2009.

[28] R. D. E. Page. *Tangled Trees: Phylogeny, Cospeciation, and Coevolution*. University Of Chicago Press, 2002.

[29] 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.

[30] W. K. Shih and W.-L. Hsu. A new planarity test. *Theor. Comput. Sci.*, 223(1-2):179–191, 1999.

[31] D. L. Swofford. *PAUP*. Phylogenetic Analysis Using Parsimony (*and Other Methods). Version 4*. Sinauer Associates, Sunderland, Massachusetts, 2002.

[32] 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.

[33] B. Venkatachalam, J. Apple, K. St. John, and D. Gusfield. Untangling tanglegrams: Comparing trees by their drawings. In *Proceedings of the Fifth International Symposium on Bioinformatics Research and Applications (ISBRA 09)*, LCNS 5542, pages 88–99, 2009.

[34] 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.

Appendix

A Alternate Proofs

Here we will provide alternate proofs for Lemma 4 and Lemma 7.

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

Proof. The "only 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. 7, 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. 7). 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.

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. 7) 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. 7). Rearranging the

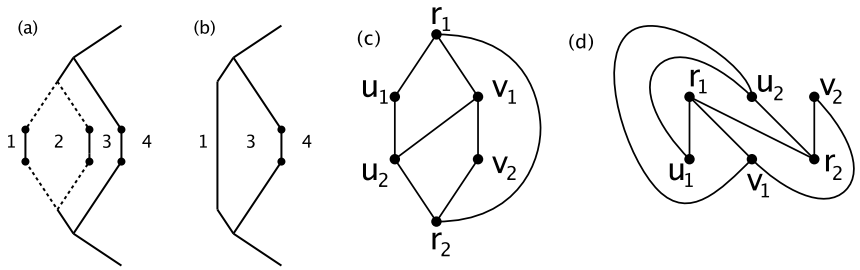

Figure 7: (a) & (b): Contraction process: After contracting the dashed internal edges, planar region 2 vanishes. The new edge can be thought of as containing the region 2 within it, and is called a super-edge. (c) & (d): Avoiding $K_{3,3}$ minor: There are at most 3 edges between pairs (u_1, v_1) and (u_2, v_2). (d) is not proper. The edges can be rearranged to form (c).

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 for the tanglegram. For the curves of the planar graph we set a convention for left and right children and remember the order of 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.

The edge contraction and expansion takes time linear in the size of the input graph. Since planar drawing of the graph takes linear time, together with the redrawing algorithm described above gives a linear time algorithm for planar drawing of a tanglegram, if one exists.

Alternate proof to lemma 7

Lemma (7)**.** *If there is a drawing with k crossings in which only matching edges cross then there is a proper drawing with at most k crossings.*

Proof. To prove this lemma we use the idea from the proof above retrace its steps.

Contract all the internal edges of the trees except the two edges out of the root of the two trees. The contraction of edges does not introduce any new crossings. We now have the reduced graph on six vertices $r_1, u_1, v_1, r_2, u_2, v_2$, the root and two internal vertices in the two trees respectively. (Unlike in lemma 4 the super-edges represent the merger of planar regions and also crossings between edges, and we cannot limit the number of super edges to three.) We can rearrange the reduced graph to form a proper drawing of the tanglegram. Finally, we expand the edges in the reverse order of contraction.

This gives a proper drawing with at most k crossings. The algorithm to redraw takes time linear in the size of the input graph. □