Exact Bipartite Crossing Minimization under Tree Constraints*

Frank Baumann[1], Christoph Buchheim[1], and Frauke Liers[2]

[1] Technische Universität Dortmund, Fakultät für Mathematik, Vogelpothsweg 87, 44227 Dortmund, Germany
[2] Universität zu Köln, Institut für Informatik, Pohligstraße 1, 50969 Köln, Germany

Abstract. A tanglegram consists of a pair of (not necessarily binary) trees T_1, T_2 with leaf sets L_1, L_2. Additional edges, called tangles, may connect nodes in L_1 with those in L_2. The task is to draw the tanglegram with a minimum number of tangle edge crossings while making sure that no crossing occurs between edges within each tree. This problem has relevant applications in computational biology, e.g., for the comparison of phylogenetic trees.
In this work, we show that the problem can be formulated as a quadratic linear ordering problem (QLO) with additional side constraints. In [2] it was shown that, appropriately reformulated, the QLO polytope is a face of some cut polytope. It turns out that the additional side constraints arising in our application do not destroy this property. Therefore, any polyhedral approach to max-cut can be used in our context. We present experimental results for drawing random and realistic tanglegrams on binary and on general trees. Using both linear and semidefinite programming techniques, we show that our approach is very efficient in practice.

Key words:tanglegram, graph drawing, computational biology, crossing minimization, quadratic programming, maximum cut problem

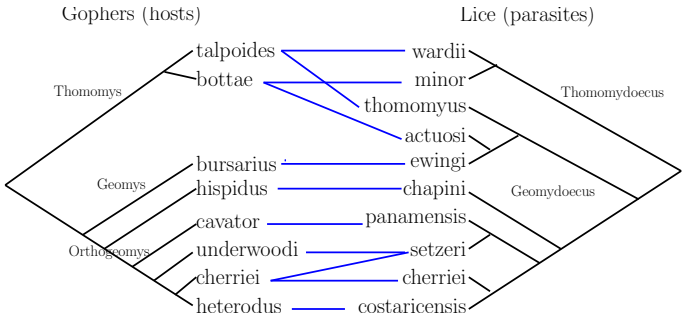
1 Introduction

A *tanglegram* [11] consists of a pair of trees T_1, T_2. Furthermore, there exist correspondences between the leaves L_1, L_2 of the trees that are represented by edges. The latter are called *tangles* or *tangle edges*. When visualizing a tanglegram, it is natural to ask for a drawing in which no edge crossings occur within either of the trees. Furthermore, the number of tangle edge crossings should be minimized. Here we require that the leaves in L_1 and L_2 are drawn on two parallel lines, while the trees are drawn in the two regions outside of these lines.

* Financial support from the German Science Foundation (DFG) is acknowledged under contracts Bu 2313/1–1 and Li 1675/1–1.

The task of drawing tanglegrams arises in several relevant applications, e.g., in computational biology for the comparison of phylogenetic trees [11]. A phylogenetic tree represents a hypothesis of the evolutionary history of a set of species. These species are drawn as the leaves of the tree, their ancestors as inner nodes. Different reconstruction methods may lead to a set of different candidate trees; a tanglegram layout then allows to compare a pair of such trees visually.

As another application, consider a phylogenetic tree of some set of species that serve as hosts for a certain set of parasites. The hypothesis that the evolution of hosts and their parasites is strongly correlated can be tested by analyzing a tanglegram layout. A tangle edge then specifies which host is affected by which parasite. Whereas in the first application the number of tangle edges incident to a leaf is always one, in the latter their degree can be higher, as shown below in a tanglegram from Hafner et al. [6] (here the hypothesis seems to be true).



Tanglegrams also occur in hierarchical clusterings, which can be visualized by so-called *dendrograms*. Dendrograms consist of trees where the elements to be clustered are identified with the leaves. Internal nodes determine clusters that contain the elements or sub-clusters. A tanglegram layout helps comparing the results of different clustering methods. Moreover, tanglegrams occur when analyzing software projects in which a tree represents package, class, method hierarchies. Hierarchy changes are analyzed over time, or automatically generated decompositions are compared with human-made ones. This application yields tanglegrams on trees that are not binary in general [10].

In the next section, we review related work. In Section 3, we introduce an exact model for tanglegrams that can be applied to pairs of general (not necessarily binary) trees with arbitrary tangle-edge density. To this

end, we show that the task is to optimize a quadratic function over the linear ordering polytope intersected with further hyperplanes. We will show that the corresponding polytope is isomorphic to a face of a cut polytope. We present experimental results exploiting this structure in Section 4, showing results for both random and realistic instances. The results prove that our approach is very efficient in practice.

2 Related Work

Most of the literature is concerned with the case of binary trees and leaves that are in one-to-one correspondence. Whereas several of the presented methods could easily be generalized to arbitrary tangle layers, an extension to non-binary trees is usually not possible. When allowing general trees, one extreme case would be a star, where all leaves are adjacent to the root node. If both T_1 and T_2 are stars, there are no constraints on the orders of leaves on either shore, so that the problem specializes to the bipartite crossing minimization problem [9, 2].

We are not aware of any implementation of an exact method for drawing tanglegrams with non-binary trees. Fernau et al. [5] showed the NP-hardness of drawing tanglegrams. They also presented a fixed-parameter algorithm for binary tanglegrams. Recently, an improved fixed-parameter algorithm was presented by Böcker et al. [1] which can solve large binary instances quickly in practice, provided that the number of crossings is not too large. Finally, while in the recent paper by Venkatachalam et al. [13] the focus is on binary instances, a fixed-parameter algorithm for general tanglegram instances is presented. According to our knowledge, this is the only algorithm that could deal with non-binary trees. However, no implementation or running times are provided making it impossible to evaluate its practical performance.

Besides analyzing the performance and quality of several heuristics in a computational study for binary tanglegrams with 1–1 tangles, Nöllenburg et al. [10] also implemented a branch-and-bound algorithm and an exact integer-programming (IP) based approach for this case.

As we will compare our approach with the exact IP-approach of [10], we describe it in more detail in the following. A feasible but not necessarily optimal tanglegram layout is given as an input. For each inner node, a binary variable x_i is introduced. In the case of complete binary trees with n leaves each, this gives rise to $2n-2$ variables. If $x_i = 1$, the subtree rooted in node i is flipped with respect to the input drawing, otherwise it remains unchanged. As by definition there are no crossings within the

trees, the number of crossings can be determined by counting the number of tangle crossings. Let (a, c) and (b, d) be tangle edges with $a, b \in L_1$ and $c, d \in L_2$. Let i be the lowest common ancestor of a, b in T_1 and j that of c, d in T_2. If the tangle edges cross each other in the input drawing, then a crossing occurs in the output drawing if and only if either both subtrees below i and j are flipped or both remain unchanged. This can be expressed as $x_i x_j = 1$ or $(1 - x_i)(1 - x_j) = 1$. Similarly, if the edges do not cross each other in the input drawing, then there is a crossing in the output drawing if and only if either $(1 - x_i)x_j = 1$ or $x_i(1 - x_j) = 1$.

Thus minimizing the number of tangle edge crossings reduces to minimizing the sum of the given products. The latter is an instance of the unconstrained quadratic binary optimization problem, which is well-known to be equivalent to a maximum cut problem in some associated graph with an additional node [3]. In an undirected graph $G = (V, E)$, the cut $\delta(W)$ induced by a set $W \subseteq V$ is defined as the set of edges (u, v) such that $u \in W$ and $v \notin W$. If edge weights are given, the weight of a cut is the total weight of edges in the cut. Now the maximum cut problem asks for a cut of maximal weight or cardinality.

While Nöllenburg et al. used this model only for instances with 1–1 tangles, they briefly note that it could be extended to leaves of higher degree as well. However, their model cannot be generalized to instances with non-binary trees in a straightforward way. In many applications, the trees are not necessarily binary. In the next section we will present an exact model for tanglegrams that neither restricts the degree of inner nodes in the trees nor the number of tangles incident to a leaf.

3 An Exact Model for General Tanglegrams

The problem of drawing tanglegrams is closely related to bipartite crossing minimization. As argued above, the latter problem can be considered a special case of the former. Therefore, we first review approaches for drawing bipartite graphs.

3.1 Bipartite Crossing Minimization

Let $G = (V_1 \cup V_2, E)$ be a bipartite graph. The task is to draw G with straight line edges. The nodes in V_1 and V_2 have to be placed on parallel lines H_1 and H_2 such that the number of edge crossings is minimal. Both heuristic and exact methods [9] exist for this problem.

Assume for a moment that the nodes on the first layer H_1 are fixed, and only the nodes on layer H_2 are permuted. For each pair of nodes

on H_2, we introduce a variable x_{uv} such that $x_{uv} = 1$ if u is drawn to the left of v and $x_{uv} = 0$ otherwise. For edges (i,k) and (j,l) with $i,j \in H_1$ and $k,l \in H_2$, such that i is left of j, a crossing exists if and only if l is left of k. We thus have to punish x_{lk} in the objective function. The task of minimizing the number of crossings is now equivalent to determining a minimum linear ordering on the nodes of H_2. Exploiting $x_{uv} = 1 - x_{vu}$, we can eliminate half of the variables and only keep those with $u < v$. Note that bipartite crossing minimization with one fixed layer is already NP-hard [4].

If the nodes on both layers are allowed to permute, the number of crossings depends on the order of the nodes on each layer. Therefore, the problem can be modeled as a quadratic optimization problem over linear ordering variables. We write the quadratic linear ordering problem (QLO) in its general form as

$$\begin{array}{lll} & \min & \sum_{(i,j,k,l)\in I} c_{ijkl} x_{ij} x_{kl} \\ (QLO) & \text{s.t.} & x \in P_{LO} \\ & & x_{ij} \in \{0,1\} \ \text{ for all } (i,j) \in J \end{array}$$

where P_{LO} is the linear ordering polytope. The index set I consists of all quadruples (i,j,k,l) such that $x_{ij}x_{kl}$ occurs as a product in the objective function, while J is the set of all pairs (i,j) for which a linear ordering variable x_{ij} is needed. For the bipartite crossing minimization case, I and J are given as

$$\begin{aligned} I &= \{(i,j,k,l) \mid i,j \in H_1,\ i<j, \text{ and } k,l \in H_2,\ k<l\} \\ J &= \{(i,j) \mid i,j \in H_1 \text{ or } i,j \in H_2,\ i<j\} \end{aligned}$$

In order to linearize the objective function, we introduce a new binary variable y_{ijkl} for each $(i,j,k,l) \in I$, modeling the product $x_{ij}x_{kl}$. Applying the standard linearization, the corresponding linearized quadratic linear ordering problem (LQLO) can be written as

$$\begin{array}{lll} & \min & \sum_{(i,j,k,l)\in I} c_{ijkl} y_{ijkl} \\ (LQLO) & \text{s.t.} & x \in P_{LO} \\ & & x_{ij} \in \{0,1\} \qquad\qquad \text{for all } (i,j) \in J \\ & & y_{ijkl} \le x_{ij}, x_{kl} \qquad \text{for all } (i,j,k,l) \in I \\ & & y_{ijkl} \ge x_{ij} + x_{kl} - 1 \quad \text{for all } (i,j,k,l) \in I \\ & & y_{ijkl} \in \{0,1\} \qquad \text{for all } (i,j,k,l) \in I. \end{array}$$

In [2], the above model was introduced for bipartite crossing minimization. Additionally, a quadratic reformulation of the constraints defining P_{LO} was given: it was shown that a 0/1 vector (x,y) satisfying $y_{ijkl} =$

$x_{ij}x_{kl}$ is contained in (LQLO) if and only if

$$x_{ik} - y_{ijik} - y_{ikjk} + y_{ijjk} = 0 \text{ for all } (i,j,k,l) \in I. \tag{1}$$

Furthermore, the constraints (1) yield a minimum equation system for (LQLO). Note that (LQLO) is a quadratic binary optimization problem where the feasible solutions need to satisfy further side constraints, namely those restricting the set of feasible solutions to linear orderings. As unconstrained binary quadratic optimization is equivalent to the maximum cut problem [3], the task is to intersect a cut polytope with a set of hyperplanes.

In general, the convex hull of the corresponding feasible incidence vectors has a structure that is very different from that of a cut polytope. In the above context, however, it was shown in [2] that the hyperplanes (1) cut out faces of the cut polytope. Exploiting this result, both IP- and SDP-based methods originally designed for maximum cut problems were used to solve the quadratic linear ordering problem. It turned out that the SDP-based approach outperformed the IP-based techniques.

3.2 Modeling Tanglegrams

Crossing minimization in tanglegrams can be seen as a generalization of bipartite crossing minimization. The set of feasible orderings is implicitly restricted by the given tree structures. Starting from the model discussed above, we formalize these restrictions as follows: let us consider a triple of leaves a, b, c in one of the trees, say T_1. In case all pairwise lowest common ancestors coincide, all relative orderings between a, b, and c are feasible. However, if the lowest common ancestor of, say, a and b is on a lower level than that of, say, a and c (in this case, the former is a descendant of the latter), then c must not be placed between a and b, as an intra-tree crossing would be induced; see Figure 1.

Therefore, we derive a betweenness restriction for every triple of leaves such that two of the leaf pairs have different lowest common ancestors. Each such betweenness restriction of the form 'c cannot be placed between a and b' can be written in linear ordering variables as $x_{ac}x_{cb} = 0$ and $x_{ca}x_{bc} = 0$. In the linearized model (LQLO), the latter amounts to requiring

$$y_{accb} = 0 \text{ and } y_{cabc} = 0\,. \tag{2}$$

For binary trees with n leaves each, all triples of leaves have two different lowest common ancestors, so in this case the number of additional equations is $2\binom{n}{3}$.

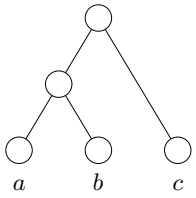


Fig. 1. Leaf c is not allowed to lie between a and b.

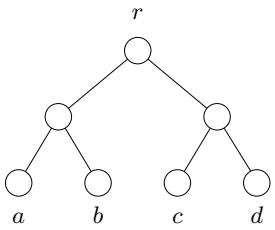


Fig. 2. Variables x_{ac} and x_{bd} can be identified.

In summary, we now obtain a quadratic linear ordering problem (QLO) on a smaller number of variables, with additional constraints of the form (2), where

$$J = \{(i,j) \mid i,j \text{ are leaves of the same tree, } i < j\}$$
$$I = \{(i,j,k,l) \mid (i,j),(k,l) \in J \text{ belong to different trees}\} .$$

For complete binary trees with n leaves each, the total number of linear ordering variables is $2\binom{n}{2}$. The same number of variables is necessary in the corresponding bipartite crossing minimization model [2].

As mentioned above, the polytope corresponding to the linearized problem (LQLO) is isomorphic to a face of a cut polytope [2]. Since all y-variables are binary, constraints of the form (2) are always face-inducing for (LQLO). In summary, we derive the following result:

Theorem 1. *The problem of drawing tanglegrams with a minimum number of edge crossings can be solved by optimizing over a face of a suitable cut polytope.*

3.3 Binary Case

In the binary case, the model introduced in the last sections is closely related to the model presented in [10]. To see this, first observe that the two equations (2) can be written as

$$x_{ab} = x_{bc} . \tag{3}$$

This replacement does not affect the set of feasible solutions, even the corresponding LP-relaxations of (LQLO) are equivalent. Note however that introducing the y-variables allows to strengthen the model, see Theorem 1.

When using the linear equations (3) instead of the quadratic equations (2), we end up with a set of equivalence classes of linear ordering variables, such that all pairwise orderings corresponding to variables

in the same class can only be flipped simultaneously. Two variables x_{ac} and x_{bd} belong to the same class if and only if there is a node r such that a, b and c, d are descendants of different children of r; see Figure 2. In the binary case, a class of linear ordering variables thus corresponds to the decision of flipping the children of node r or not, which is modeled explicitly by a single variable in the model of Nöllenburg et al. [10].

However, in the general case where node r has k children, there are $k!$ different orderings. As these cannot be modeled by a single binary variable, the model of Nöllenburg et al. [10] cannot be applied here.

4 Computational Results

We implemented the model explained in Section 3.2. Instead of adding equations (3) explicitly, we used one variable for each equivalence class of linear ordering variables, thereby significantly reducing the number of variables. For evaluating the IP-based methods, we used CPLEX 11.2 [8]. The naive approach is to solve the linearized model (LQLO) using a standard integer programming solver. More advanced approaches solve the quadratic reformulation (1), using separation of cutting planes for max-cut, both in the context of integer and semidefinite programming. For the SDP approaches, we used the bundle method by Rendl et al. [12]. For comparison, we also implemented the IP approach [10] that only works for binary tanglegrams. For the tested binary instances, the running times for solving the latter are very comparable to the model proposed here and are omitted in the following. This behavior can be expected since our model generalizes [10].

We generated random instances on general binary, ternary and quad trees. I.e., the degree of each internal node is at most 2, 3 or 4, respectively. Each tree has n leaves, either having 1–1 tangles or a certain tangle-edge density d%. Instances are generated following the description in [10], with obvious extensions to the more general cases considered here. Finally, we solved realistic binary tanglegram instances from [10] arising in applications in biology and general realistic instances from visualizing software hierarchies [7].

Average results are always computed over 5 randomly generated instances. For each instance, we imposed an upper limit of 10h of cpu time. Instances that could not be solved within this limit count with 10h in the averages. Runs were performed on Intel Xeon machines with 2.33GHz.

In Table 1, we present the average cpu time in seconds for realistic binary instances. Table 2 shows results for random ternary and quad trees,

n	SDP	std	ref	std+cyc	ref+cyc
0–49	<1	<1	<1	<1	14
50–99	3	<1	2	<1	428
100–149	31	<1	7	<1	1100
150–199	125	1	32	1	1282
200–249	437	1	66	2	2704
250–299	4786	2	2529	7	10602
300–349	6483	2	80	9	8862
400–449	20508	12	12067	69	14931

Table 1. Average cpu time in seconds for realistic 1–1 binary trees having n leaves each [10]. Instances are grouped by their number of leaves. The nine largest instances with up to 540 leaves could not all be computed within the time and memory constraints and are omitted.

respectively. Figure 4 visualizes the results from Table 2 for $n = 128$. Running times for realistic general tanglegram instances are presented in Table 3. In case an entry is missing in the tables, we could not solve the instances by the corresponding methods due to memory allocation constraints.

n	d	SDP	std	ref	std+cyc	ref+cyc	SDP	std	ref	std+cyc	ref+cyc
64	1	3	1	<1	<1	1	<1	<1	<1	<1	<1
	5	15	12	4	67	6	2	3	1	1	<1
	10	18	27	19	1301	17	3	8	2	5	1
	15	78	54	23	3893	37	4	16	6	57	3
	20	41	54	48	12508	66	4	19	7	65	3
128	1	165	16	32	78	63	32	19	3	2	4
	5	362	448	280	22253	448	80	59	35	1045	51
	10	436	2179	791	36000	2746	73	1406	119	10147	495
	15	4049	3247	1551	36000	5583	77	844	352	31582	1184
	20	8293	2326	1408	36000	15426	1420	1560	1908	36000	10111

Table 2. Average cpu time in seconds for random general ternary (left) and quad trees (right) having n leaves each, density d%.

The first column *SDP* shows results obtained by semidefinite optimization, whereas the remaining columns refer to IP-based approaches for solving the model from Section 3. *std* refers to solving the standard linearization using CPLEX default, *ref* its quadratic reformulation (1). In the options *std+cyc* and *ref+cyc*, cycle inequalities for max-cut are additionally separated, all CPLEX cuts are switched off.

Clearly, for realistic binary trees with 1–1 tangles, the SDP approach usually needs considerably more time than the IP-based methods. Fur-

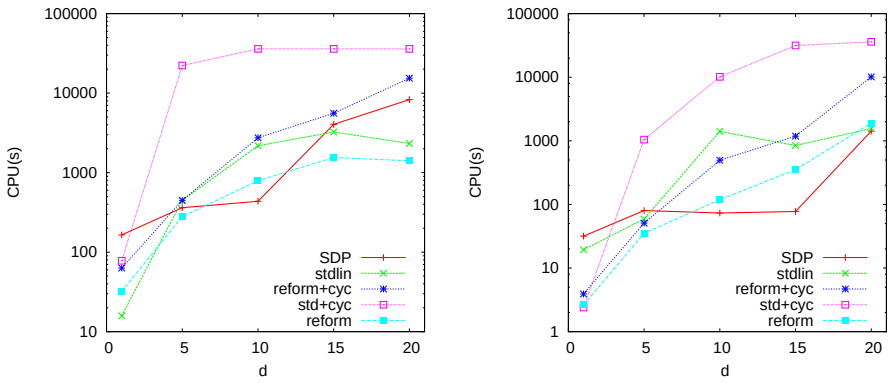


Fig. 3. Plot showing results for $n = 128$ of Table 2 for ternary (left) and quad trees (right).

instance	SDP	std	ref	std+cyc	ref+cyc
philips_orig_4a	27618	4725	73	4321	32692
philips_orig_4b	26093	4205	74	2756	23443
philips_orig_4c	36000	2666	122	4030	36000
philips_orig_4d	27891	3208	314	4178	28902
philips_orig_4e	36000	2789	90	5665	36000
philips_4a_4b	5238	1395	4	420	5
philips_4a_4c	4769	1858	3	637	4
philips_4a_4d	2924	1467	4	494	3
philips_4a_4e	2575	827	2	338	3
philips_4b_4c	6526	1965	8	510	7
philips_4b_4d	4872	2070	5	577	5
philips_4b_4e	2127	649	4	124	36
philips_4c_4d	4611	804	3	217	5
philips_4c_4e	6403	1074	3	396	5
philips_4d_4e	3738	891	4	811	11

Table 3. Cpu time in seconds for realistic general tanglegram instances [7].

thermore, memory requirements strongly increase with system size and so the largest instances could not be solved. On average, the fastest approaches for solving the largest instances are the pure standard linearization *std* and the quadratic reformulation *ref*.

In fact, we can optimize tanglegrams with more than 500 leaves in each tree. This is the range of sizes arising in realistic applications. The realistic instances can be solved particularly fast. Interestingly, cycle separation for max-cut usually does not pay off for binary 1–1 tanglegrams: the running time increases, even if the dual bounds are usually considerably

better when cycle separation is included. Often, an optimum solution can be determined in the root node. However, although the bound is weak in the standard linearization, after few branching steps the optimum LP solution is often feasible and the program can stop. We found similar characteristics for random binary tanglegram instances.

The picture changes when varying the density of the tangle edges: for big enough tangle-edge density the SDP approach usually outperforms the IP ones. However, memory requirements usually prohibit the solution of instances with more than 500 leaf nodes and tangle-edge density of 1%. On the IP side, reformulation is often preferable. Indeed, the best performance is found when the problems are quadratically reformulated. For $n = 512$ and 1% tangle-edge density, the average solution time is 2120.42 seconds. These instances cannot be solved within the given time limits when using only the standard linearization, with or without separation of cycle inequalities.

The instances for ternary and quad trees are computationally slightly more difficult. This is mainly due to the fact that the number of betweenness restrictions decreases when compared to binary trees. Here again, the SDP approach performs well for denser instances however memory requirements strongly increase with system size. For larger instances, best performance is often found for the quadratic reformulation.

Comparing *std* with *std+cyc* and *ref* with *ref+cyc* for not necessarily binary trees, it turns out that the performance of separating cycle inequalities improves for ternary and quad trees. The special case of a star, where the degree is maximal, is equivalent to the quadratic linear ordering problem, for which we know that separation of cycle inequalities improves over *std* [2].

The realistic general instances we tested had between 371 and 414 nodes and 131 tangles. The maximum degree of an internal node was 15. We show the results in Table 3. Here, solving the reformulation usually yields best performance. Note that these non-binary instances could not be solved before by any other exact method.

5 Acknowledgments

We are grateful to Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele for the permission to use their bundle program for the SDP-based approach, as well as to Martin Nöllenburg and Danny Holten for sending us realistic tanglegram instances. We thank Henning Fernau for drawing our attention to tanglegrams.

References

[1] S. Böcker, F. Hüffner, A. Truss, and M. Wahlström. A faster fixed-parameter approach to drawing binary tanglegrams. In *Proc. of International Workshop on Parameterized and Exact Computation (IWPEC 2009)*, 2009. To appear.

[2] C. Buchheim, A. Wiegele, and L. Zheng. Exact algorithms for the quadratic linear ordering problem. *INFORMS Journal on Computing*. To appear.

[3] C. De Simone. The cut polytope and the boolean quadric polytope. *Discrete Mathematics*, 79:71–75, 1989.

[4] P. Eades and N. C. Wormald. Edge crossings in drawing bipartite graphs. *Algorithmica*, 11:379–403, 1994.

[5] H. Fernau, M. Kaufmann, and M. Poths. Comparing trees via crossing minimization. *Journal of Computer and System Sciences*, In Press, 2009.

[6] M. S. Hafner, P. D. Sudman, F. X. Villablanca, T. A. Spradling, J. W. Demastes, and S. A. Nadler. Disparate rates of molecular evolution in cospeciating hosts and parasites. *Science*, 265:1087–1090, 1994.

[7] D. Holten. personal communication, 2009.

[8] ILOG, Inc. ILOG CPLEX 11.2, 2007. www.ilog.com/products/cplex.

[9] M. Jünger and P. Mutzel. 2-layer straightline crossing minimization: performance of exact and heuristic algorithms. *J. Graph Algorithms Appl*, 1:1–25, 1997.

[10] M. Nöllenburg, M. Völker, A. Wolff, and D. Holten. Drawing binary tanglegrams: An experimental evaluation. In *Proc. of the Workshop on Algorithm Engineering and Experiments, ALENEX 2009*, pages 106–119. SIAM, 2009.

[11] R. D. M. Page. *Tangled Trees: Phylogeny, Cospeciation, and Coevolution*. University of Chicago Press, 2002.

[12] F. Rendl, G. Rinaldi, and A. Wiegele. A branch and bound algorithm for max-cut based on combining semidefinite and polyhedral relaxations. In M. Fischetti and D. P. Williamson, editors, *IPCO 2007*, volume 4513 of *Lecture Notes in Computer Science*, pages 295–309. Springer, 2007.

[13] B. Venkatachalam, J. Apple, K. St. John, and D. Gusfield. Untangling tanglegrams: Comparing trees by their drawings. *Bioinformatics Research and Applications*, pages 88–99, 2009.