On the diameter of the rotation graph of binary coupling trees

V. Fack, S. Lievens and J. Van der Jeugt[1]
Department of Applied Mathematics and Computer Science,
University of Ghent, Krijgslaan 281-S9, B-9000 Gent, Belgium

Abstract

A binary coupling tree on $n+1$ leaves is a 0–2-tree in which each leaf has a distinct label. The rotation graph G_n is defined as the graph of all binary coupling trees on $n+1$ leaves, with edges connecting trees that can be transformed into each other by a single rotation. In this paper we study distance properties of the graph G_n. Exact results for the diameter $d(G_n)$ for values upto $n = 10$ are obtained. For larger values of n we prove upper and lower bounds for the diameter, which yield the result that the diameter $d(G_n)$ grows like $n \lg(n)$.

Corresponding author : V. Fack, Department of Applied Mathematics and Computer Science, University of Ghent, Krijgslaan 281-S9, B-9000 Gent, Belgium.
Tel. ++ 32 9 2644808; Fax ++ 32 9 2644995; E-mail Veerle.Fack@rug.ac.be.

[1]Research Associate of the Fund for Scientific Research – Flanders (Belgium)

1 Introduction

Binary coupling trees arise in the context of the quantum theory of angular momentum, where they represent coupling schemes in the bra/kets of a $3nj$-coefficient [4]. A summation formula for the evaluation of a given $3nj$-coefficient can be obtained from a sequence of basic operations, called flops, transforming one coupling tree into another. The complexity of the obtained summation formula is determined by the number of flops used in a particular sequence to transform one tree into the other. The objective is to construct an optimal summation formula, or equivalent a sequence of flops with minimum length, for any two binary coupling trees.

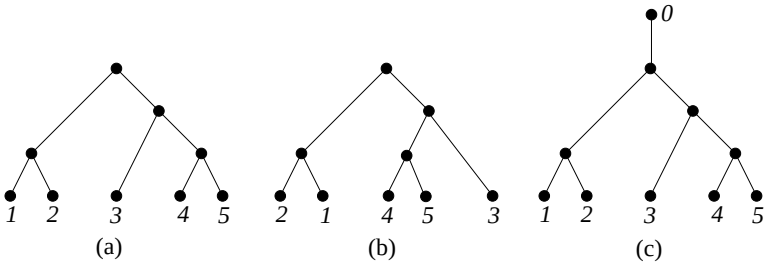
In the terminology of trees as defined by Knuth [9], a coupling tree in a $3nj$-coefficient can be seen as a 0–2-tree with $n+1$ leaves, and a flop is similar to a rotation operation. The set of all binary coupling trees for a given n-value can be seen as the vertices of a graph G_n, where an edge connects two binary coupling trees that are related by one rotation. In order to tackle the problem of finding an optimal sequence of flops for any two binary coupling trees, we study the structure of the associated rotation graph G_n of binary coupling trees, in particular its distance properties.

Section 2 defines binary coupling trees and the rotation graph G_n, and describes some basic properties of the graph G_n. In Section 3 we introduce the concept of distance between binary coupling trees, being the length of a shortest unweighted path between the corresponding vertices in the rotation graph G_n. Exact results for some distance properties (such as distance degree sequence and diameter) are given for small values of n ($n \leq 10$). The size of G_n is growing exponentially in n, so for large n-values we look for theoretical bounds for the diameter of G_n. In Section 4 we obtain an explicit upper bound by constructing a path between two arbitrary binary coupling trees and showing that its length is necessarily bounded by $n \lg(n) + O(n)$. Section 5 shows how an $\Omega(n \lg(n))$ lower bound for the diameter can be obtained from an upper bound for the number of trees within a certain distance of any given tree, for which the technique of short encodings introduced by Sleator *et al* in [13] can be used. We conclude that the diameter $d(G_n)$ grows like $n \lg(n)$.

2 Binary coupling trees and the graph G_n

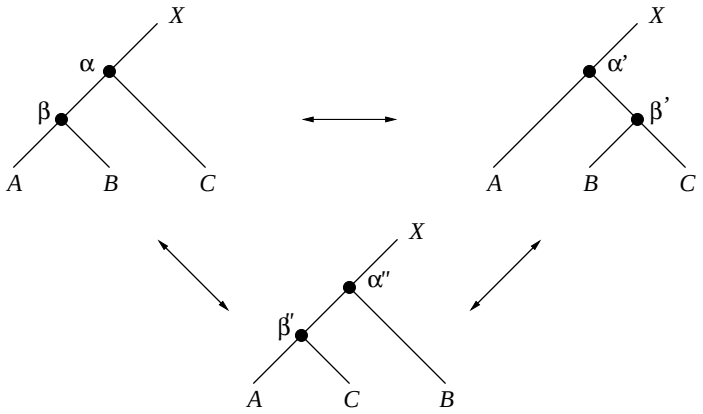
Following Knuth [9, Section 2.3] we say that a *tree* is a finite set of one or more nodes such that (a) there is one specially designated node called the root of the tree and (b) the remaining nodes are partitioned into $m \geq 0$ disjoint sets $T_1, T_2, \ldots, T_m$, and each of these sets in turn is a tree. A *0–2-tree* is a tree in which each node has exactly two or zero children. Finally, a *binary coupling tree* is a 0–2-tree in which each leaf (i.e. a node with no children) has a unique label. Without loss of generality, we can assume that these labels are the integer numbers between 1 and $n+1$ if the binary coupling tree has $n+1$ leaves. We denote, for fixed $n \geq 1$, the set of all binary coupling trees on $n+1$ leaves, or equivalently on n non-leaf nodes, as $\mathcal{T}_n$. Figure 1(a) and (b) give two representations of the same binary coupling tree. Note that binary coupling trees are *unordered*, or using the terminology of [9], *oriented*. Sometimes, it will be convenient to attach an extra leaf with label 0 to the root and regard the binary coupling tree as a *free tree*; this is shown in Figure 1(c). The set of such *extended binary coupling trees* on $n+2$ leaves will be denoted as $\tilde{\mathcal{T}}_n$.

Figure 1: Binary coupling trees



If (α, β) is an *internal edge*, i.e. α and β are both non-leaf nodes, then we can perform two *rotations* around this edge, as shown in Figure 2. Herein, A, B and C can be leaves or arbitrary subtrees and X is either empty or contains the root of the tree. Note that a rotation is invertible; if T_2 is obtained by performing a rotation on T_1, then T_1 can be obtained by performing a rotation on T_2. This is also indicated in Figure 2. In the literature, other names for rotations appear: *flops* [5], *nearest neighbour interchanges* [3] and *crossovers* [12].

Figure 2: Rotations on binary coupling trees



For fixed $n \geq 1$, we build the *rotation graph* G_n as follows: each vertex of G_n represents an element from $\mathcal{T}_n$. Two vertices are adjacent if and only if the two binary coupling trees they represent are related through a single rotation. Some simple properties of G_n were proved in [12]; see also [4]. We summarize them here:

- $|G_n| = |\mathcal{T}_n| = (2n-1)!! = 1 \cdot 3 \cdot \ldots \cdot (2n-1)$,
- G_n is regular of degree $2(n-1)$,
- G_n is connected.

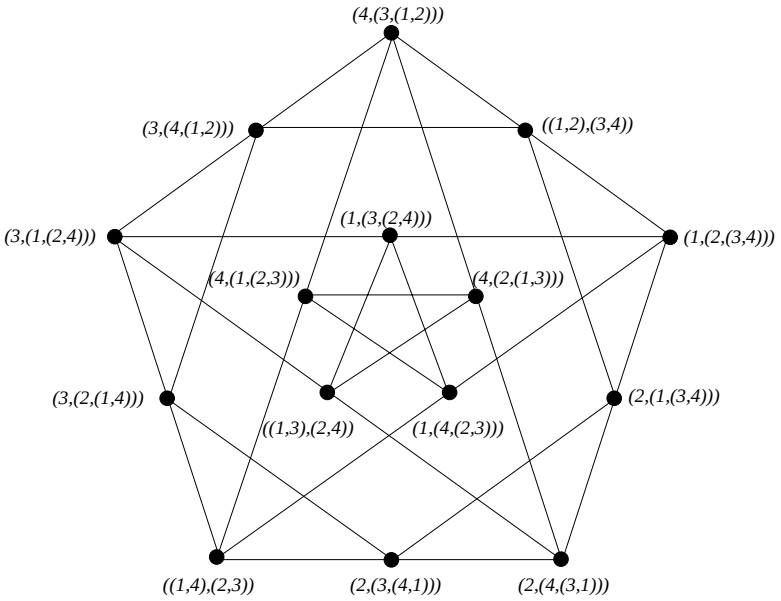
Example 1 As can be seen in Figure 3, the graph G_3 has $1 \cdot 3 \cdot 5 = 15$ vertices, while every vertex has $2 \cdot 2 = 4$ neighbours. In Figure 3, every vertex is labelled with a *bracket notation* of the binary coupling tree it represents. A bracket notation of a binary coupling tree gives the way in which the leaves are coupled to form the binary coupling tree. Possible bracket notations of the binary coupling tree in Figure 1(a) are:

$$((1,2),(3,(4,5))) \qquad \text{or} \qquad ((2,1),((4,5),3)).$$

It is worth mentioning that G_3 is the line graph of the Petersen graph. Using Theorem 9 of [12] one can prove that for $n > 3$ the graph G_n is not the line graph of another graph. ∎

Let σ be an element of S_{n+2}, the group of all permutations on $n+2$ elements; σ acts on $T \in \tilde{\mathcal{T}}_n$ by permutation of the $n+2$ leaf labels. It is clear that if T_1 and $T_2 \in \tilde{\mathcal{T}}_n$

Figure 3: The rotation graph G_3



are related through a single rotation, $\sigma(T_1)$ and $\sigma(T_2)$ are also related through a single rotation. Thus $\sigma(G_n)$ is isomorphic with G_n. Furthermore, for $n \geq 3$ no element of S_{n+2}, the identity permutation excepted, fixes G_n completely. Indeed, if σ has a cycle of length 2,

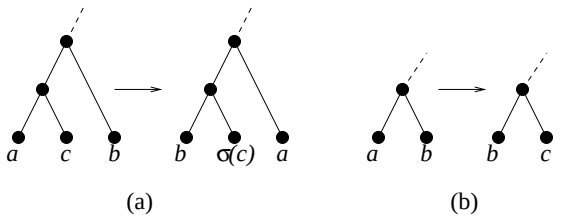
$$\sigma = (a\ b)\ldots,$$

then any tree of the form indicated in Figure 4(a) is not fixed under σ. If σ has no cycle of length 2 and $\sigma \neq \mathrm{id}$, then it must have a cycle of length > 2,

$$\sigma = (a\ b\ c\ldots)\ldots.$$

In this case, all trees of the form indicated in Figure 4(b) are not fixed under σ. Thus, we can conclude that for $n \geq 3$, $S_{n+2} \subseteq \mathrm{Aut}(G_n)$, the automorphism group of G_n. For $n = 3, 4, 5, 6$, equality holds; we have verified this using the `nauty` program [11]. For larger values of n, the question whether equality is achieved or not remains open.

Figure 4: Trees that are not fixed under σ



3 Distance in G_n

Given two binary coupling trees T_1 and T_2 from $\mathcal{T}_n$, the *distance* $d(T_1, T_2)$ between T_1 and T_2 is the minimum number of rotations needed to transform T_1 into T_2. Equivalently, we can say that the distance between T_1 and T_2 equals the length of a shortest path between their corresponding vertices in G_n. It is easy to see that $(\mathcal{T}_n, d)$ forms a metric space.

In this paper, we are primarily concerned with computing or estimating the *diameter* $d(G_n)$ of G_n, i.e. the maximal length of any shortest path in G_n:

$$d(G_n) = \max \{d(T_1, T_2) | \, T_1, T_2 \in \mathcal{T}_n\} .$$

The diameter and many other distance related properties (eccentricity, radius, center, periphery, ... [1]) follow easily if we know the *distance degree sequence* for every vertex of G_n. The distance degree sequence for a vertex v of G_n is the sequence

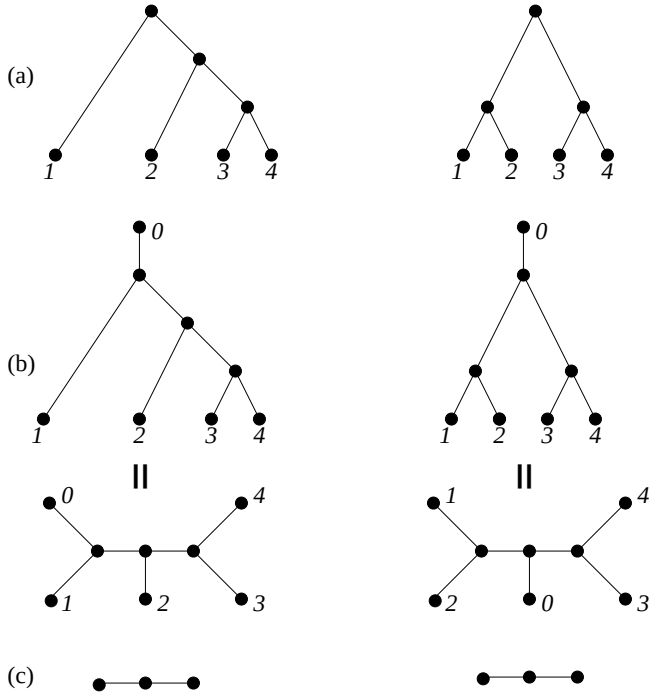
$$dds(v) = (d_0(v), d_1(v), d_2(v), \ldots),$$

where $d_i(v)$ is the number of vertices at distance i from v.

It is obvious that many vertices in G_n give rise to the same distance degree sequence. Indeed, if two binary coupling trees differ only by a permutation of their labels, then they have the same distance degree sequence. Two such binary coupling trees are said to be of the same *type*. As indicated in [4] and in Figure 5(a), there are two different types of binary coupling trees on 4 leaves, yet the distance degree sequence of these two types is identical. This can be understood by considering the corresponding elements from $\tilde{\mathcal{T}}_n$; indeed, these elements are equal upto a permutation of the labels, see Figure 5(b). The *skeleton* of an extended binary coupling tree is the free tree obtained by deleting all leaves

and corresponding edges from the extended binary coupling tree, see Figure 5(c). Note that two extended binary coupling trees are equal upto a permutation of the leaf labels if and only if their skeletons are isomorphic.

Figure 5: (a) The two types in $\mathcal{T}_3$, (b) their corresponding extended binary coupling trees and (c) the corresponding skeletons



We have written a program to calculate the distance degree sequences for values of $n \leq 10$. This program is inspired by some techniques of [10]. The key ideas to this program are:

- assign to each element of $\mathcal{T}_n$ a unique number between 0 and $|\mathcal{T}_n| - 1$; these numbers can then be used as index in an array,
- use efficient data structures to fastly calculate the neighbours of any given tree, since calculating the neighbours must be done for (almost) all trees,
- use the minimal number of bits per tree, since there are $(2n-1)!!$ trees and these must all reside in the main memory of the computer.

Table 1: Number of types and skeletons for $n \leq 10$

n	2	3	4	5	6	7	8	9	10
types	1	2	3	6	11	23	46	98	207
skeletons	1	1	2	2	4	6	11	18	37

To determine the diameter of G_n for some small fixed n, it is sufficient to calculate the distance degree sequence for all skeletons with n nodes. Indeed, since rotations are performed around internal edges, two extended binary coupling trees which have the same skeleton will have the same distance degree sequence. Table 1 lists the number of types and skeletons for values of n upto 10. The sequence giving the number of types is sequence A001190 of [14]; it is also known as the Wedderburn-Etherington sequence. The number of skeletons is sequence A000672 of [14]. Note that the number of skeletons is (much) smaller than the number of types, yielding a substantial decrease of the required computation time. This technique was first used by Jarvis *et al* [7]. Distance degree sequences upto $n = 7$ are given in [4]; the complete results upto $n = 10$ can be found at URL `http://allserv.rug.ac.be/~jvdjeugt/BCT`. The diameter of G_n for $n \leq 10$ is shown in Table 2.

Table 2: Diameter of G_n for $n \leq 10$

n	2	3	4	5	6	7	8	9	10
$d(G_n)$	1	3	5	7	10	12	15	18	21

4 An upper bound for the diameter of G_n

For all elements T_1, T_2 of $\mathcal{T}_n$, we will construct a path between their corresponding vertices in G_n. Robinson [12], Culik II and Wood [2] and Li *et al* [10] used the same technique to obtain an $O(n^2)$, $4n\lg(n) + O(n)$, $n\lg(n) + O(n)$ upper bound for the diameter of G_n respectively. Here, and in the rest of this paper, lg denotes the logarithm in base 2. We will follow the lines indicated in [10] to obtain an explicit upper bound of the order $n\lg(n) + O(n)$ for the diameter of G_n.

The *level of a node in a tree* is defined recursively as follows [9, Section 2.3]: the level

of the root is zero and the level of any other node is one more than the level of its parent. The *level of a tree* T, denoted as $l(T)$, is the maximum level of any of its nodes. If T is an element of $\mathcal{T}_n$, the number of nodes with level i, denoted as l_i, satisfies:

$$l_0 = 1 \text{ and } 2 \leq l_i \leq 2^i, \text{ for } 0 < i \leq l(T).$$

Since the total number of nodes of T is $2n+1$, we have

$$1 + \sum_{i=1}^{l(T)} 2 \leq 1 + \sum_{i=1}^{l(T)} l_i = 2n+1 \leq 1 + \sum_{i=1}^{l(T)} 2^i,$$

and hence

$$\lceil \lg(n+1) \rceil \leq l(T) \leq n. \tag{1}$$

An element $S \in \mathcal{T}_n$ is a *spine* if and only if $l(S)$ equals n. Note that spines exist for every $n \geq 1$; indeed, there are $\frac{(n+1)!}{2}$ spines in $\mathcal{T}_n$.

The path between T_1 and T_2 is constructed in three steps:

1. transform T_1 into a spine S_1,
2. transform T_2 into a spine S_2 and,
3. transform the spine S_1 into S_2 (or vice versa).

In this section, we will determine an explicit upper bound for the number of rotations needed in each step, yielding an explicit upper bound for the diameter of G_n.

Suppose T is a binary coupling tree, that is not a spine. Choose a leaf x of T that has maximum level. Since T is not a spine, there is an internal edge of T, that is not on the path from the root node of T to x, but that has a node in common with an edge on this path. Performing the appropriate rotation around this internal edge will increase the level of T by one. Hence, one can transform an arbitrary element T of $\mathcal{T}_n$ into a spine using $n - l(T)$ rotations.

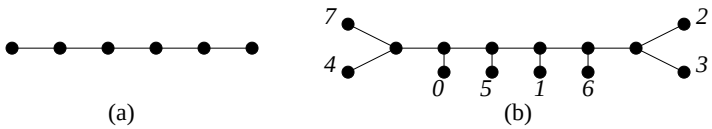
Thus, given the bound in (1), one can transform any binary coupling tree on $n+1$ leaves into a spine using at most

$$n - \lceil \lg(n+1) \rceil \tag{2}$$

rotations.

The construction of a path between two arbitrary spines from $\mathcal{T}_n$ is easier to understand when working with extended binary coupling trees, i.e. elements of $\tilde{\mathcal{T}}_n$. We say that an element S from $\tilde{\mathcal{T}}_n$ is an *extended spine* if and only if its skeleton is a *degenerate tree*, i.e. a free tree in which each node has degree one or two. Figure 6(a) gives a representation of a degenerate tree on six nodes, while Figure 6(b) gives a representation of an extended spine of $\tilde{\mathcal{T}}_6$. Note that there are $\frac{(n+2)!}{8}$ extended spines in $\tilde{\mathcal{T}}_n$, for $n > 1$, so for $n > 2$, not every extended spine corresponds to a spine in $\mathcal{T}_n$. In the rest of this section, we will assume that $n > 1$.

Figure 6: (a) A degenerate tree on six nodes and (b) an extended spine of $\tilde{\mathcal{T}}_6$



The maximum number of rotations needed to transform two arbitrary extended spines of $\tilde{\mathcal{T}}_n$ into each other, is an upper bound for the number of rotations needed to transform two arbitrary spines of $\mathcal{T}_n$ into each other. The problem of transforming two arbitrary extended spines into each other is equivalent to transforming an arbitrary extended spine into some fixed extended spine since this only requires a permutation of the labels.

An extended spine of $\tilde{\mathcal{T}}_n$ has four *end leaves*, i.e. leaves for which there exists another leaf at distance two; in Figure 6(b), these are the leaves with label 7, 4, 2 and 3. Let S be an extended spine and let x be an end leaf. We say that S is *increasing* (resp. *decreasing*) *with respect to* x if and only if for all other leaves x_1 and x_2 the following property holds:

$$d(x_1, x) < d(x_2, x) \Rightarrow x_1 < x_2 \text{ (resp. } x_1 > x_2).$$

Herein, $d(x_i, x)$ denotes the distance of the leaf x_i to the leaf x, while the notation x_i itself is also used to denote the label of the leaf x_i. If the leaf label of x is known, then there is exactly one extended spine in $\tilde{\mathcal{T}}_n$ that is increasing with respect to x; we will use this extended spine as the fixed spine mentioned before.

Let S be an extended spine of $\tilde{\mathcal{T}}_n$ and let x be an end leaf. We say that $S \in \tilde{\mathcal{T}}_n$ is

concave with respect to x (resp. *convex with respect to* x) if and only if for all other leaves x_1 and x_2 the following property holds:

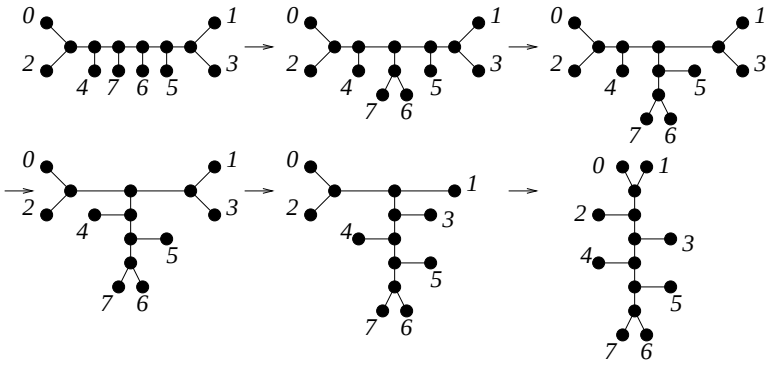
$$d(x_1, x) < d(x_2, x) \leq \left\lceil \frac{n}{2} \right\rceil + 1 \Rightarrow x_1 < x_2 \text{ (resp. } x_1 > x_2)$$

and

$$\left\lceil \frac{n}{2} \right\rceil + 2 \leq d(x_1, x) < d(x_2, x) \Rightarrow x_1 > x_2 \text{ (resp. } x_1 < x_2).$$

If $S \in \tilde{\mathcal{T}}_n$ is an extended spine which is concave (resp. convex) with respect to x, then we can transform S into an increasing (resp. decreasing) extended spine, again with respect to x, using at most $n-1$ rotations, see Figure 7. This is quite analogous to the *merge* step in the mergesort algorithm, where two sorted sequences are combined to form a single sorted sequence [8, Section 5.2.4].

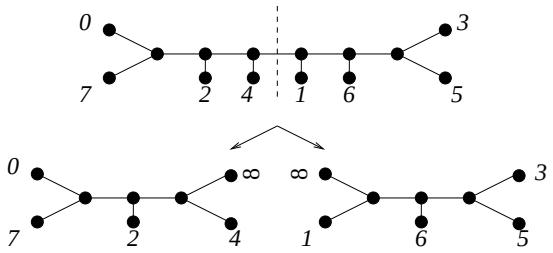
Figure 7: Merging an extended spine



Also the other ideas of the mergesort algorithm apply to our problem. Let $S \in \tilde{\mathcal{T}}_n$ be an extended spine that has to be transformed into an increasing or a decreasing one with respect to some leaf x. We cut S in two and attach to each of the two halves an extra leaf. These two extra leaves get the same label and the value of this label depends on whether we are transforming S into a increasing respectively a decreasing extended spine. In the former case the value of this label has to be greater then all other labels used ($+\infty$), while in the latter it has to be smaller ($-\infty$). In this way, we get two extended spines: one in $\tilde{\mathcal{T}}_{\lceil \frac{n}{2} \rceil}$, containing the leaf x, and one in $\tilde{\mathcal{T}}_{\lfloor \frac{n}{2} \rfloor}$. (To really match the definition, we would have to do an order preserving re-labelling.) Figure 8 gives an illustration of how

an extended spine is cut in two when this extended spine has to be transformed into an increasing one with respect to the leaf 0.

Figure 8: Cutting an extended spine in two



If we are transforming S into an increasing (resp. decreasing) extended spine with respect to the leaf x, we *recursively* transform the extended spine containing x into an increasing (resp. decreasing) one with respect to the original leaf x, while we recursively transform the other extended spine into a decreasing (resp. increasing) one with respect to the new leaf. As a consequence of the labelling of the extra leaf, this leaf does not change position when transforming the two extended spines; this means that we can indeed merge the two extended spines together.

Let $f(n)$ (resp. $g(n)$) denote the maximum number of rotations needed to transform an arbitrary extended spine $S \in \tilde{\mathcal{T}}_n$ into an increasing (resp. decreasing) one with respect to some fixed leaf x. It is clear that $f(1) = g(1) = 0$, since $|\tilde{\mathcal{T}}_1| = 1$. Furthermore, f and g are bounded by f_u and g_u satisfying the following recurrences:

$$\begin{cases} f_u(n) &= f_u(\lceil \frac{n}{2} \rceil) + g_u(\lfloor \frac{n}{2} \rfloor) + n - 1, \quad \text{for } n > 1 \\ g_u(n) &= g_u(\lceil \frac{n}{2} \rceil) + f_u(\lfloor \frac{n}{2} \rfloor) + n - 1, \quad \text{for } n > 1 \\ f_u(1) &= 0 \\ g_u(1) &= 0. \end{cases}$$

It is easy to see that $f_u(n)$ equals $g_u(n)$ for all values of n, so the recurrence for f_u simplifies to:

$$\begin{cases} f_u(n) &= f_u(\lceil \frac{n}{2} \rceil) + f_u(\lfloor \frac{n}{2} \rfloor) + n - 1, \quad \text{for } n > 1 \\ f_u(1) &= 0. \end{cases}$$

This is a well known recurrence [6, Section 3.3] and its solution is given by:

$$f_u(n) = n\lceil \lg n \rceil - 2^{\lceil \lg n \rceil} + 1. \tag{3}$$

As already noted, $f(n)$ is an upper bound for the number of rotations needed to transform an arbitrary spine $S_1 \in \mathcal{T}_n$ into another arbitrary spine $S_2 \in \mathcal{T}_n$. We thus have the following theorem:

Theorem 2 *The diameter of G_n satisfies*

$$d(G_n) \leq n\lceil \lg(n) \rceil - 2^{\lceil \lg(n) \rceil} + 1 + 2\left(n - \lceil \lg(n+1) \rceil\right). \tag{4}$$

This follows immediately by combining formulas (3) and (2). □

Remark 3 As already noted, Robinson [12] obtained a $O(n^2)$ upper bound for the diameter of G_n. One can also obtain an $O(n^2)$ upper bound by following the above method and by simulating a bubble sort [8, Section 5.2.2] on the spines instead of a merge sort. ∎

5 A lower bound for the diameter of G_n

If one can find an upper bound for the number of vertices within distance m of any given vertex v of a graph G, then one can obtain a lower bound for the diameter of that graph.

A lower bound is easily obtained by considering the following inequalities that hold for any vertex v of a graph G with maximal degree Δ:

$$d_0(v) = 1, \quad d_1(v) = \Delta, \quad d_i(v) \leq \Delta(\Delta - 1)^{i-1}, \text{ for } i > 1.$$

This gives the following inequality:

$$\frac{\Delta(\Delta - 1)^{\delta} - 2}{\Delta - 2} \geq \sum_{i=0}^{\delta} d_i(v) = |G|, \tag{5}$$

where δ denotes the diameter of G. This bound arises in the context of the so-called Moore graphs [1]. If we apply inequality (5) to the rotation graph G_n, we get a linear lower bound for the diameter of G_n.

Since Theorem 2 gives an $n\lg(n) + O(n)$ upper bound for the diameter of G_n, one might wonder whether a better lower bound, i.e. of the order $n\lg(n)$, can be established. This is indeed the case, as was already indicated in [10]. In the rest of this section we will show how to obtain an $\Omega(n\lg(n))$ lower bound for the diameter of G_n.

In [13] Sleator *et al* provide a tool for deriving an upper bound for the number of combinatorial objects within m transformations from a given object. They take advantage of the fact that, for many series of transformations, one can interchange the order of the transformations, without affecting the final outcome.

We will apply their technique of *short encodings* to binary coupling trees. The transformations we will be using are the rotations on binary coupling trees. As already noted, a binary coupling tree does not change if one exchanges the "left" and "right" child of any non-leaf node. This transformation is called an *exchange* [5] or a *twist* [13]. Sleator *et al* [13] give an example where the twist transformation is also counted. We do *not* want to take these twists into account. As we shall see, the twist transformation can be avoided by suitably doubling the transformations associated with rotations.

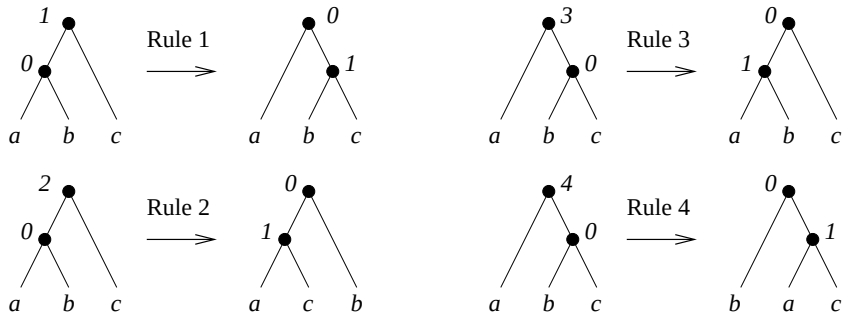
Suppose T and T' are elements of $\mathcal{T}_n$ with $d(T, T') = m$. This means that there exists a sequence of m rotations that carries T into T'; this sequence is called a *derivation*. Note that there may be many derivations that carry T into T', since it may be possible to interchange the order in which the rotations are applied.

At any time in a derivation, we will consider the elements T of $\mathcal{T}_n$ as *ordered* trees, i.e. twists are not allowed. When regarding T as an ordered tree, we will denote it as $\hat{T}$. We can *apply* one of the four *rules* (transformations) indicated in Figure 9 to $\hat{T}$ if and only if $\hat{T}$ contains a subtree identical to the tree on the left side of that rule. The result is $\hat{T}$ in which the left side of the rule is replaced by the right side. The numbers of the nodes in the left sides of the rules are called *position numbers*, while the numbers of the nodes in the right sides of the rules are called *right position numbers*. Their use will soon become apparent.

It will be convenient to think of applying a rule as destroying nodes and creating new ones. We will make sure that every non-leaf node in the derivation receives a unique *name* which can be used to identify that node. An *action* is a particular application of a rule, i.e. a derivation is a sequence of actions. The *required nodes* of an action are the nodes that are destroyed by that action. An action is *ready* if and only if the required nodes of that action exist.

In order to name each non-leaf node that appears in a derivation, we first number

Figure 9: The four rules that can be applied



the actions of that derivation. Each internal node of the initial tree $\hat{T}$ is named v_i, with $1 \leq i \leq n$, in pre-order. (The exact order is immaterial as long as we know what it is.) Next, each new node gets a name of the form $v_{j,0}$ or $v_{j,1}$, where j is the number of the action that created these nodes and where 0 and 1 refer to the right position numbers of the applied rule.

In order to build an encoding for the derivation D, with initial tree $\hat{T}$, we first (a) number the actions of D, (b) give each internal node a name and (c) compute the required nodes of each action. Furthermore, we associate a position number with each name, which is the position number of the node in the rule that is applied to the node with that name. If no such rule is applied, i.e. if the node with the matching name exists in $\hat{T}'$, the number 0 is associated with that node.

The construction of the encoding of a derivation D is done simultaneously with the construction of a canonical derivation D', which consists of a reordering of the actions of D, but for which the final outcome, i.e. $\hat{T}'$, is the same. In order to decide which action to do first, we determine which actions are ready, i.e. all actions that can be performed on $\hat{T}$. We then *choose* to do the action that destroys the node with the smallest name, when the names are lexicographically ordered. If we have already done i actions, we can decide in the same way what the $(i+1)$-th action must be. We repeat this process until all actions are applied.

The encoding now exists of the position numbers associated with the internal nodes, *in the order introduced by the canonical derivation* D'.

Figure 10: A derivation of length 4

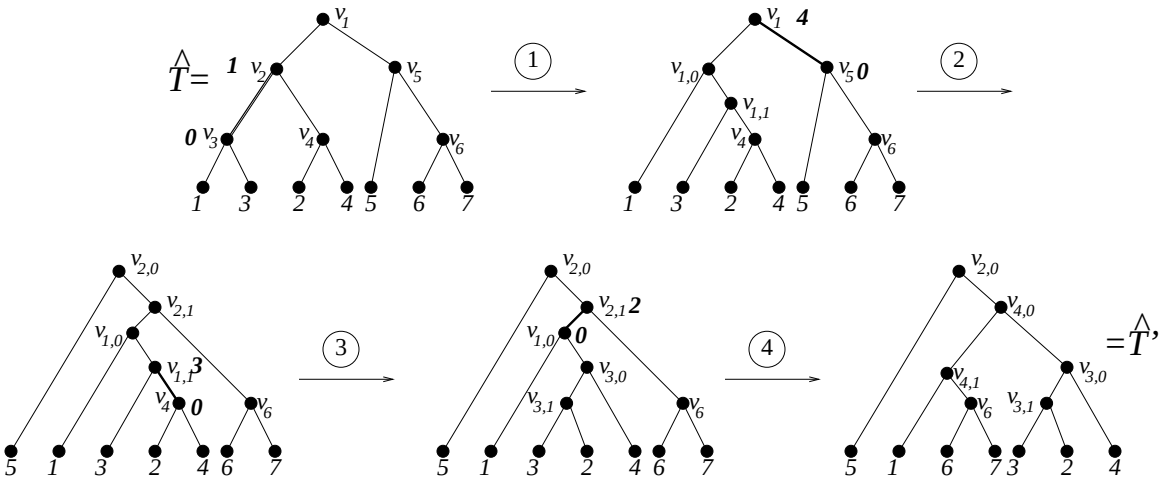


Table 3: Required nodes for each action of the derivation in Figure 10

action	required nodes
1	$v_2,\ v_3$
2	$v_1,\ v_5$
3	$v_4,\ v_{1,1}$
4	$v_{1,0},\ v_{2,1}$

Example 4 For the derivation in Figure 10, Tables 3 and 4 give the required nodes of each action and the association of the names with position numbers. Looking at the initial tree, or equivalently at Table 3, we see that actions 1 and 2 are ready. We choose to do action 2 first, because action 2 destroys the node with the smallest name. Thus, nodes v_1 and v_5 are destroyed and nodes $v_{2,0}$ and $v_{2,1}$ are created. Next, only action 1 is ready so we do action 1, hereby destroying the nodes v_2 and v_3 and creating the nodes $v_{1,0}$ and $v_{1,1}$. Now, both actions 3 and 4 are ready, but we choose to do action 3 and finally, action 4 is done. This results in the encoding given in the third row of Table 5. The canonical derivation of the derivation in Figure 10 is given in Figure 11. ∎

An internal node is required by at most one action, since it is destroyed by that action. Thus, choosing the ready action which destroys the internal node with the smallest name

Figure 11: The canonical derivation of the derivation in Figure 10

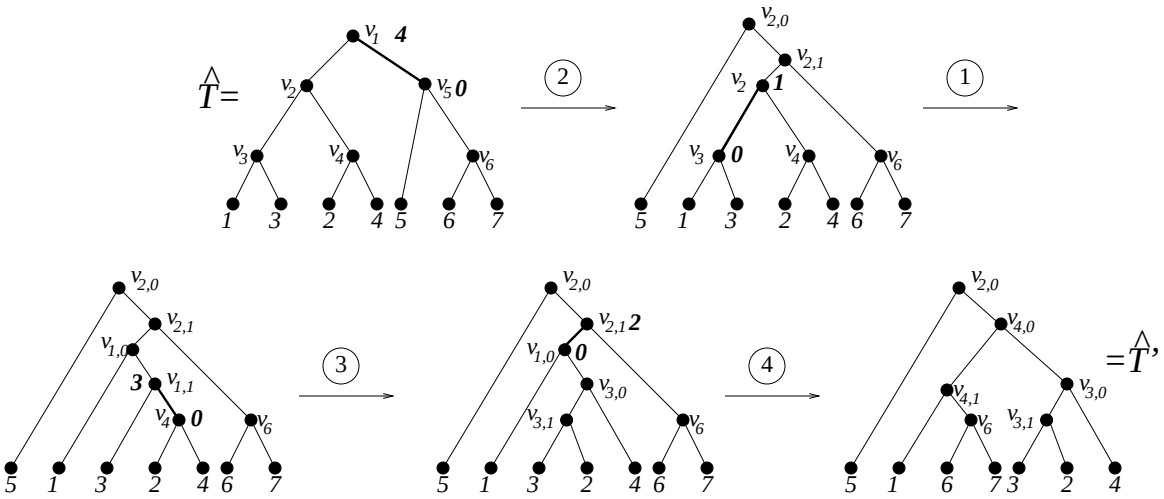


Table 4: Association of names with position numbers

v_1	v_2	v_3	v_4	v_5	v_6	$v_{1,0}$	$v_{1,1}$	$v_{2,0}$	$v_{2,1}$	$v_{3,0}$	$v_{3,1}$	$v_{4,0}$	$v_{4,1}$
4	1	0	0	0	0	0	3	0	2	0	0	0	0

Table 5: Encoding for the derivation in Figure 10

v_1	v_2	v_3	v_4	v_5	v_6	$v_{2,0}$	$v_{2,1}$	$v_{1,0}$	$v_{1,1}$	$v_{3,0}$	$v_{3,1}$	$v_{4,0}$	$v_{4,1}$
v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}
4	1	0	0	0	0	0	2	0	3	0	0	0	0

is well defined. Furthermore, at each stage in the encoding process, at least one action is ready. Indeed, if the first action of the derivation D is not yet applied, then this action is ready. If the first action of the derivation D is already applied, then there exists a number $i > 1$, such that the actions $1, 2, \ldots, i-1$ of the derivation D are already applied and such that action i is not yet applied. In this case, action i is ready. This proves that we can reorder the actions of derivation D to form the canonical derivation D'.

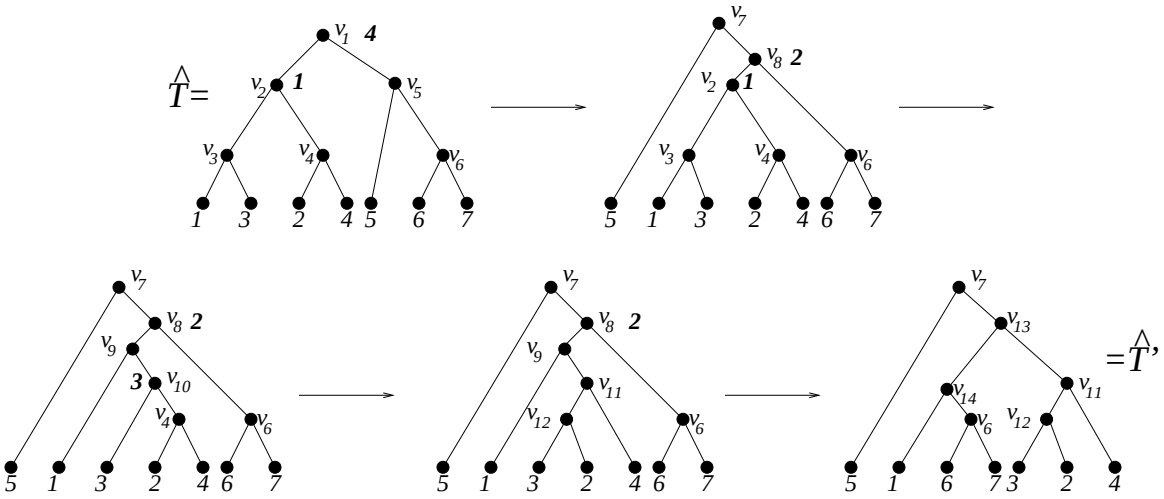
The only question that remains is whether the outcome of D' is identical to the outcome of D. Suppose actions i and j of the original derivation D are ready at the same time while constructing D'. Then these actions do not require a common node, since any node is required by at most one action. Robinson [12] already proved that two rotations around two edges that do not share a common node can be performed in either order without affecting the outcome. This proves that the outcome of D' equals the outcome of D.

Next, we explain how the canonical derivation D' can be reconstructed (decoded) when $\hat{T}$ and the encoding are given. The decoding procedure mimics the behaviour of the encoding procedure. We start from the initial tree $\hat{T}$ and name the internal nodes as in the encoding procedure. With each name, we associate the corresponding entries from the encoding. By inspecting all parent-child couples in the initial tree, we can build a list of actions that are ready. Note that the position numbers in the left sides of the rules are chosen in such a way that it is impossible that two actions sharing a node are ready simultaneously. We then apply the rule that destroys the node with the smallest name. This will obviously be the first action of D'. Application of this rule will create internal nodes v_{n+1} and v_{n+2}, corresponding with the $(n+1)$-th and $(n+2)$-th entry of the code Continuing in this manner, we can reconstruct D'.

Example 5 Next to the nodes of Figure 12 we have written the corresponding entries from the encoding. In order not to overload the figure the entries that equal zero are not shown. As can be seen, rules 1 and 4 can be applied to the initial tree. Because rule 4 destroys the node with the smallest name i.e. v_1, we apply this rule thus creating the nodes v_7 and v_8. Now we can only apply rule 1, yielding the third tree. Finally, we arrive at $\hat{T}'$. ∎

Using this technique, every tree $\hat{T}'$, with $d(\hat{T}, \hat{T}') = m$, can be encoded by an array

Figure 12: The decoding procedure



(code) of length $n+2m$. Since a code of length $n+2m$ has exactly m non-zero entries, and each non-zero entry can take values 1 upto 4, the number of codes $|C(n,m)|$ of length $n+2m$ is bounded by

$$|C(n,m)| \leq \binom{n+2m}{m} 4^m.$$

The number of trees at distance m from any given tree is thus bounded by the same number. We can even say more: the number of trees *within* distance m from any given tree is bounded by the same number. Indeed, if $d(T,T') = m-2l$, then there is a derivation of length m carrying T into T'; one only needs to go back and forth between T' and his predecessor on the path of length $m-2l$. If $d(T,T') = m-2l-1$, then one can construct a derivation of length $m-2l$ by making a detour along the common neighbour of T' and his predecessor on the path of length $m-2l-1$. This derivation of length $m-2l$ can be extended to a derivation of length m by going back and forth between T' and his predecessor in this derivation. Note that although the last argument does not hold when m equals 1, the bound still holds.

We have thus proved the following:

Lemma 6 *The number of trees within distance m from any binary coupling tree $T \in \mathcal{T}_n$ is less than or equal to $\binom{n+2m}{m} 4^m$.*

Theorem 7 *The diameter of G_n satisfies, for $n > 1$,*

$$d(G_n) > \frac{1}{4}\lg(n!) > \frac{1}{4}n\lg(\frac{n}{e}).$$

¿From Lemma 6 we know, when δ denotes the diameter of G_n, that:

$$\binom{n+2\delta}{\delta}4^{\delta} \geq (2n-1)!! = \frac{n!}{2^n}\binom{2n}{n}. \tag{6}$$

We can enlarge the lhs of (6), for $n > 0$ and $\delta > 1$, by

$$\frac{2^{n+2\delta}}{2n} > \binom{n+2\delta}{\delta}.$$

This inequality is proved in Appendix A. The rhs of (6) can be bounded using

$$\binom{2n}{n} \geq \frac{2^{2n}}{2n}. \tag{7}$$

Together, this yields

$$2^{4\delta} > n!,$$

from which the theorem follows. □

Remark 8 One slightly improves the lower bound from Theorem 7 when bounding the rhs of (6) by

$$\frac{(2n)!}{2^n n!} \geq \sqrt{2}\left(\frac{2n}{e}\right)^n\left(1 - \frac{1}{24n}\right). \tag{8}$$

instead of by (7). Inequality (8) is proved using Stirling's formula. ∎

Remark 9 Li *et al* ([10]) also gave an $O(n\lg(n))$ lower bound for the diameter of G_n, using the results of [13]. They sketched a way, using "flips" in plane triangulations and short encodings, to derive that the number of trees within distance m from any given tree is bounded by $3^n 2^{4m}$. When enlarging the lhs of (6) we find the number of trees within distance m ($m > 1$) to be bounded by

$$\frac{2^{n+4m}}{2n}.$$

Since this is smaller than $3^n 2^{4m}$, our lower bound will be better than the one found by Li *et al.* ∎

6 Conclusion

In this paper we studied some distance properties of the rotation graph G_n of binary coupling trees. Upto $n = 10$ the distance degree sequences for G_n have been calculated explicitly, yielding also the exact value of the diameter $d(G_n)$. For larger values of n we have restricted ourselves to determining theoretical bounds for the diameter of G_n. Both an upper bound and a lower bound for $d(G_n)$ have been obtained in Theorems 2 and 7. By slightly enlarging the upper bound from Theorem 2, in order to yield a more aesthetic formula, we find the following theorem:

Theorem 10 *The diameter $d(G_n)$ of G_n satisfies, for $n > 1$:*

$$\frac{1}{4}n \lg(\frac{n}{e}) < d(G_n) < n \lceil \lg n \rceil + n - 2 \lceil \lg n \rceil + 1.$$

This implies that the diameter of G_n grows like $n \lg(n)$, i.e.

$$d(G_n) = \Theta(n \lg n).$$

Appendix A

We have to show that

$$\binom{n+2m}{m} < \frac{2^{n+2m}}{2n}$$

for all integers $n > 0$ and $m > 1$.

The cases $n = 1$ and $n = 2$ can easily be treated separately. For $n = 1$, the inequality follows from $2^{2m} = \sum_{i=0}^{m} \binom{2m+1}{i}$. For $n = 2$, it is easy to show that $r(m) = \binom{2m+2}{m} 2^{-2m}$ satisfies $r(m+1) < r(m)$. Since $r(1) = 1$, the result follows.

Generally, let $n(> 2)$ be fixed, and consider

$$f_n(m) = \frac{2^{n+2m}}{\binom{n+2m}{m}}.$$

The inequality to prove is $f_n(m) > 2n$. It is easy to verify that

$$f_n(m) - f_n(m-1) = f_n(m) \frac{2m + n - n^2}{4m(n+m)}.$$

Hence it follows that $f_n : \mathbb{Z}_+ \to \mathbb{Q}$ is decreasing in the integer interval $[0, m_0 - 1]$ and increasing in the integer interval $[m_0, +\infty]$, reaching its minimal value for $m = m_0 = (n^2 - n)/2$ and for $m = m_0 - 1 : \; f_n(m_0) = f_n(m_0 - 1)$. So it is sufficient to prove that $f_n(m_0) > 2n$, i.e.

$$\frac{2^{n^2}}{\binom{n^2}{(n^2-n)/2}} > 2n,$$

or in other words $g(n) > 2$, with

$$g(x) = \frac{2^{x^2}\Gamma(1 + \frac{x^2-x}{2})\Gamma(1 + \frac{x^2+x}{2})}{x\ \Gamma(1 + x^2)},$$

in terms of the classical Gamma-function. The desired inequality can be obtained from an asymptotic expansion of $g(x)$. The function $g(x)$ is a C_∞-function in the open interval $(0, +\infty)$, and so is the function $G(x) = g(1/x)$. Moreover, $G(x)$, $G'(x)$ and $G''(x)$ have a continuous extension in $x = 0$, so we can write

$$G(x) = G(0) + x\, G'(0) + \frac{x^2}{2} G''(\theta_x), \qquad \text{with } 0 < \theta_x < x.$$

Explicit calculation yields $G(0) = \sqrt{2\pi e}/2$, $G'(0) = 0$, and $|G''(\theta)| < 1$ for $0 < \theta < 1/2$. Thus,

$$|G(x) - \frac{\sqrt{2\pi e}}{2}| < \frac{x^2}{2}, \qquad \text{for } 0 < x < \frac{1}{2}.$$

This implies

$$|g(n) - \frac{\sqrt{2\pi e}}{2}| < \frac{1}{2n^2}, \qquad \text{for } n > 2.$$

Since $\frac{\sqrt{2\pi e}}{2} \sim 2.066$ and $\frac{1}{2n^2} < 0.06$ for $n \geq 3$, it follows that $g(n) > 2$ for $n \geq 3$.

References

[1] F. Buckley and F. Harary, Distance in Graphs (Addison–Wesley, 1990).

[2] K. Culik II and D. Wood, A note on some tree similarity measures, Inform. Process. Lett. 15 (1982) 39–42.

[3] W. H. Day, Properties of the nearest neighbor interchange metric for trees of small size, J. Theor. Biol. 101 (1983) 275–288.

[4] V. Fack, S. Lievens and J. Van der Jeugt, On rotation distance between binary coupling trees and applications for $3nj$-coefficients, Comput. Phys. Commun., in press, 1999.

[5] V. Fack, S. N. Pitre and J. Van der Jeugt, New efficient programs to calculate general recoupling coefficients. Part I: Generation of a summation formula, Comput. Phys. Commun. 83 (1994) 275–292.

[6] R. L. Graham, D. E. Knuth and O. Patashnik, Concrete Mathematics, A Foundation for Computer Science (Addison–Wesley, 1995).

[7] J. Jarvis, J. Luedeman and D. Shier, Comments on computing the similarity of binary trees, J. Theor. Biol. 100 (1983) 427–433.

[8] D. E. Knuth, Sorting and Searching, Volume 3 of The Art of Computer Programming (Addison–Wesley, 1973).

[9] D. E. Knuth, Fundamental Algorithms, Volume 1 of The Art of Computer Programming (Addison–Wesley, 1997).

[10] M. Li, J. Tromp and L. Zhang, On the nearest neighbour interchange distance between evolutionary trees, J. Theor. Biol. 182 (1996) 463–467.

[11] B. McKay. nauty. `http://cs.anu.edu.au/people/bdm/nauty/`.

[12] D. Robinson, Comparison of labeled trees with valency three, J. Comb. Theory 11 (1971) 105–119.

[13] D. D. Sleator, R. E. Tarjan and W. P. Thurston, Short encodings of evolving structures, SIAM J. Disc. Math. 5 (1982) 428–450.

[14] N. J. Sloane, On-line encyclopedia of integer sequences, `http://www.research.att.com/~njas/sequences/index.html`.