Metric Tree Weight Adjustment and Infinite Complete Binary Trees As Groups

A Thesis

Submitted to the Faculty

of

Drexel University

by

Craig Schroeder

in partial fulfillment of the

requirements for the degree

of

Master of Science in Computer Science

2006

Dedications

This work is dedicated to my family.

Acknowledgements

My parents, Dennis (late) and Irene Schroeder, for raising me and encouraging me to take education seriously.

Dr. Ali Shokoufandeh for being a great advisor. He has given me direction and assistance with many important decisions I have made, extending well beyond the boundaries of this thesis. He has also reviewed countless proofs, ideas, and full thesis drafts and contributed many ideas to the thesis. Without him, this thesis would not exist.

Dr. Eric Schmutz for serving on my thesis committee and providing ideas, insight, and suggestions.

Dr. Jeremy Johnson and Dr. Pawel Hitczenko for serving on my thesis committee.

Hal Finkel for providing a continuous supply of ideas and feedback, and for scrutinizing drafts. When new ideas came up, he has always been there to listen to them.

Quinn Thomas for insisting that I continue studying the things that interest me most even if there is resistance to it and even after this thesis is finished.

Dr. William Regli for teaching me how to do research, showing me the ropes, and giving me a lot of advice over the years.

Dr. Robert Boyer and Dr. Justin Smith for direction and insight when I found myself surrounded by a world of possibilities but little sense of which were worth following.

Fatih Demirci for getting me up to speed in the lab, leaving behind a lot of resources to refer to, and providing a great example of how to write a thesis.

Jeff Abrahamson for technical contributions and especially for causing me to realize that I was undertaking a proof with no clue what I was trying to prove.

Many others who have contributed to this thesis in the form of ideas, examples, a usable LaTeX thesis style, encouragement, and support.

This work was supported in part by ITR NSF [EIA-0205178].

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation or the other supporting government and corporate organizations.

Contents

List of Tables

List of Figures

Abstract

Metric Tree Weight Adjustment and Infinite Complete Binary Trees As Groups

Craig Schroeder

Advisor: Ali Shokoufandeh

Metric trees are an important type of metric because many problems are efficiently solvable on metric trees that are hard on more general metrics. For this reason, many algorithms use metric trees. At the current time, there is only one known efficient, approximate embedding into a metric tree, and the trees that this embedding produces are typically far from balanced. While an unbalanced tree is acceptable for many applications, a balanced metric tree is more suitable for others. This thesis considers part of the metric tree balancing problem. We find efficient algorithms for computing new weights for metric trees, to be applied after the tree is topologically balanced. One of these algorithms uses the L_2 norm, yielding an optimal solution. A second algorithm solves the same problem under the L_∞ norm, where the solution is also optimal. Both algorithms are considered in a local case, where they update weights locally after tree rotations.

By extending binary trees to an infinite, complete binary tree, tree operations like rotations and child flips become closed. By carefully defining the tree rotations and child flips, the tree operations form a group. To our knowledge, this group structure has not been studied before. We remedy this by proving many important theorems, including its finite generation and a complete list of its normal subgroups.

1. Introduction

1.1 Problem

Humans can easily and almost instantly recognize objects. This is called object recognition, and the task is quite difficult for computers. Object recognition has received much attention and remains a hot area of research in computational vision and pattern recognition. One variation of the object recognition problem may be formulated as the problem of, given some object, finding the best match from a database of models.

Several approaches used in object recognition break objects into features and a perform many-to-many matching on the features as points in a metric space [14, 17, 18]. However, this approach relies on the ability to take an arbitrary metric, embed it into a tree, and take a caterpillar decomposition. Unfortunately, only one provably good algorithm is known for performing this embedding, and the trees that it produces do not lead to good caterpillar decompositions, because the trees it produces are highly unbalanced. The metric embedding is described in [2], and the use of the caterpillar decomposition for embedding is described in [13].

The global update algorithms for updating the metric after balancing the tree are a useful tool for completing the metric tree balancing process. Algorithms for restructuring the tree are not considered in this thesis, as the difficulty of this task is beyond the scope of this thesis. It is not clear whether a polynomial runtime algorithm for this task exists. We treat the much simpler problem of assigning edge weights. The update algorithms return the tree to a metric by assigning edge weights to the balanced tree. In both cases the assignment is optimal.

Adding new entries to the metric using the global update algorithms is inefficient. Local update algorithms are derived for both metrics similar in nature to the global versions. The local updates are applied after individual rotation operations used in the insertion. While the local updates need not result in a globally optimal assignment of weights, it does yield an optimal update of the edges immediately around the rotation.

Attempting to treat the question of balancing metric trees leads to the study of binary trees and tree operations as a group. A group is a set with a binary operator. The operator must satisfy four properties:

closure, associativity, identity, and inverse. A good introductory and reference text on group theory is [7].

The link between group theory and graphs is far from new, and much of this is focused on automorphism groups of finite graphs [5, 4]. The connection surfaces in the classification theorem for finite simple groups, as many of the sporadic groups are conveniently realized as automorphism groups or subgroups of automorphism groups of graphs. Examples of this are the Janko group J_2, the Higman-Sims group HS, the McLaughlin group McL, and the Conway groups Co_1, Co_2, and Co_3 [3].

There are also links between trees and groups. For example, when binary trees of size n are used as vertices of a graph with a directed edge when the trees are related by a single left rotation, the resulting graph contains a Hamiltonian path, which is used to enumerate all such trees efficiently [11, 12]. Interesting relationships have been explored between trees, groups, and finite automata [8, 1]. Lavrenyuk considers techniques for developing finite state actions on homogeneous trees of finite size [10].

Integers may be broken down as factors of primes. In a similar way, groups may be broken down into simple groups. The groups may then be assembled using direct products, semidirect products, and other approaches. Groups are broken down into smaller groups by taking a quotient with respect to a subgroup. This quotient is well-defined precisely when the subgroup is normal. A more precise treatment of this can be found in [7]. The ability to decompose groups makes finding normal subgroups worthwhile.

1.2 Definitions

This section is intended to provide basic definitions and important background information useful to readers unfamiliar with the background assumed in the body of the thesis. Readers familiar with these areas may skip this section.

1.2.1 Tree Metrics and Embeddings

A **metric space** consists of a set X and a function $d : X \times X \to \mathbb{R}$ that satisfies four properties [21]:

- **Non-negativity:** $\forall x, y \in X, d(x,y) \geq 0$
- **Identity of Indiscernibles:** $\forall x, y \in X, d(x,y) = 0$ if and only if $x = y$
- **Symmetry:** $\forall x, y \in X, d(x,y) = d(y,x)$

- **Triangle Inequality:** $\forall x,y,z \in X, d(x,y)+d(y,z) \geq d(x,z)$.

Inparticular, X may be finite or infinite. The most common and familiar example of a metric is the Euclidean metric on $\mathbb{R}^n$ defined by:

$$d(x,y) = \|x-y\|_2^2 = \sum_i (x_i - y_i)^2.$$

A second example of a metric is the Manhattan distance metric, defined on $\mathbb{R}^n$ by:

$$d(x,y) = \|x-y\|_1 = \sum_i |x_i - y_i|.$$

A third example is defined on $\mathbb{R}^n$ by:

$$d(x,y) = \|x-y\|_\infty = \max_i |x_i - y_i|.$$

These three examples are normed vector spaces also known as the L_2, L_1, and L_∞ norms, respectively.

Any metric may be expressed as a possibly infinite, weighted, undirected, connected graph $G = (V,E)$. The vertices of the graph correspond to the elements in the set, so $X = V$. The distance $d(x,y)$ for $x,y \in V$ is defined to be the length of the shortest path from x to y, where the length of a path is the sum of the weights of the edges along that path.

These conditions guarantee that the graph corresponds to a metric. Positive edge weights ensure the first two properties. Symmetry is guaranteed because the graph is undirected, and a path from x to y corresponds to a path from y to x of the same length. The triangle inequality follows from the definition of the distance, since the path from x to y to z is a path from x to z of length $d(x,y)+d(y,z)$.

Any metric may be expressed as a graph. Let G be the complete graph on $V = X$ with edge weights $w(x,y) = d(x,y)$. The triangle inequality ensures that the shortest path from x to y will be the path consisting of only the edge (x,y).

A useful representation for a finite metric is a **distance matrix**. The metric is defined over a finite set $X = \{x_1, \ldots, x_n\}$ with $d(x_i, x_j) = D_{ij}$, where D is an $n \times n$ matrix. The properties of a metric force D to be symmetric with zeroes along the diagonal. This representation is used in Chapters 2 and 3.

Finally, a **tree metric** can be defined similar to the graph representation of a metric. In this case, the graph is a tree, and the set X of elements defining the metric is the set of *leaves* of the tree. Non-leaf nodes of the tree are not elements of the metric space. The distances, as with the graph representation, are defined as the length of the unique path bewteen two leaves. Metric trees satisfy a **four-point condition**:

$$d(w,x) + d(y,z) \leq \max\left(d(w,y) + d(x,z), d(w,z) + d(x,y)\right).$$

This condition is a necessary and sufficient condition for a metric to be a tree metric [9].

A **metric embedding** can be defined as a mapping $\phi : X \to X'$ from a metric space on X with distance function d into a metric space on X' with distance function d'. The **distortion** of an embedding is a measure of how well the distances of the metric are preserved by the embedding. Three examples of measures of distortion are variations of the examples for distance metrics. The L_2^2 norm may be used as a measure for the distortion between distance matrices D and E:

$$\|D - E\|_2^2 = \sum_{ij} (D_{ij} - E_{ij})^2.$$

This distortion measure is used in Chapter 2. The L_1 norm leads to the distortion measure:

$$\|D - E\|_1 = \sum_{ij} |D_{ij} - E_{ij}|.$$

The L_∞ norm leads to the distortion measure:

$$\|D - E\|_\infty = \max_{ij} |D_{ij} - E_{ij}|.$$

This distortion measure is used in Chapter 3. Note that the distortion measure and the distance metrics are in general unrelated.

It is known that embedding an arbitrary metric into a tree metric with minimum distortion is NP-hard under the L_1, L_2, and L_∞ distortion measures. A polynomial time 3-approximation, i.e., the distortion oftained is at most a factor of three of the optimal distortion, exists for the L_∞ distortion measure [2].

1.2.2 Group Theorey

This section is intended to be a quick introduction to the group theory used in this thesis. For a more complete and authoratative introduction, please see [7].

A **group** is a set of elements G and a binary operation $\cdot : G \times G \to G$. The binary operation must satisfy four properties.

- **Identity:** There exists an identity element e such that for any $x \in G$, $e \cdot x = x \cdot e = x$.
- **Closure:** For any elements $x, y \in G$, $x \cdot y \in G$. That is to say the group is closed under the binary operation.
- **Associativity:** For any elements $x, y, z \in G$, $x \cdot (y \cdot z) = (x \cdot y) \cdot z$.
- **Inverse:** For each element $x \in G$ there exists a unique element $w \in G$ such that $w \cdot x = x \cdot w = e$. The element w is called the inverse of x and is denoted $w = x^{-1}$.

A group is said be **Abelian** if $x \cdot y = y \cdot x$ for all $x, y \in G$. That is, the binary operation is commutative. When a group is Abelian, the group operation is often written as addition: $x + y$. If the group is not Abelian, the binary operation is generally written as multiplication: $x \cdot y$ or xy. Because the groups studied in this thesis are not Abelian, the binary operation is written using concatenation as xy.

A **subgroup** is a subset H with the same binary operation that itself satisfies the properties of a group. Subgroups are fundamental to the study of groups.

The elements of a group G may be thought of as bijective functions that act on the elements of a set X. Then, a (left) **group action** is a function $\cdot : G \times X \to X$ that satisfies two properties [20]:

- **Identity:** If $e \in G$ is the identity element in G and $x \in X$ then $e \cdot x = x$.

- **Associativity:** For all $g,h \in G$ and $x \in X$, $g \cdot (h \cdot x) = (gh) \cdot x$.

A right group action is similar, except that the binary operation is defined with the group applied to the other side as $\cdot : X \times G \to X$. Right action by g may be expressed as left action by g^{-1}, so the notions are not fundamentally different.

Note that any action leads to a group and is a way of creating a group structure from bijective functions. Used in this way, associativity may be treated as a definition of the binary operator as the composition of functions g and h. Closing set of functions under composition ensures the closure property. Combining the two properties gives us the element e that fulfills the identity property for the group:

$$\begin{aligned}
(eg) \cdot x &= e \cdot (g \cdot x) \\
&= g \cdot x \\
eg &= g \\
(ge) \cdot x &= g \cdot (e \cdot x) \\
&= g \cdot x \\
ge &= g.
\end{aligned}$$

The associativity property for actions implies associativity for the group:

$$\begin{aligned}
(f(gh)) \cdot x &= f \cdot ((gh) \cdot x) \\
&= f \cdot (g \cdot (h \cdot x)) \\
&= (fg) \cdot (h \cdot x)) \\
&= ((fg)h) \cdot x \\
f(gh) &= (fg)h.
\end{aligned}$$

Because the functions are bijections, they have inverses. The inverse g^{-1} of a function g corresponds to the group inverse:

$$\begin{aligned}
(g^{-1}g)\cdot x &= g^{-1}\cdot(g\cdot x)\\
&= x\\
&= e\cdot x\\
g^{-1}g &= e\\
(gg^{-1})\cdot x &= g\cdot(g^{-1}\cdot x)\\
&= x\\
&= e\cdot x\\
gg^{-1} &= e.
\end{aligned}$$

This construction is used to define the group of tree operations studied in Chapters 4 and 5.

A **group homomorphism** is a mapping $\phi : G \to H$ between groups G and H such that:

$$\phi(ab) = \phi(a)\phi(b).$$

An **group ismorphism** is a group homomorphism that is also a bijection. An **automorphism** is a group isomorphism from a group to itself. The automorphism group, $Aut(G)$, is the group constructed from the action of automorphisms on G using the construction above. In particular, the binary operation is composition.

Conjugation of a group element h by another element h is defined as ghg^{-1}. For each h, the set of all f for which $f = ghg^{-1}$ for some g is called the **conjugacy class** of h. A subgroup of G that is closed under conjugation by elements of G is called a **normal subgroup**. If $H \subseteq G$ is a normal subgroup, then for any $g \in G$ and $h \in H \subseteq G$, $ghg^{-1} \in H$. The subgroup formed by closing a subgroup $H \subseteq G$ under conjugation by elements of the group G is called the **conjugate closure** of H and is denoted $\langle H \rangle^G$ [19]. In particular, conjugate closures are always normal subgroups, and the conjugate closure of a normal subgroup is itself. Thus, there is a one-to-one correspondence between (distinct) conjugate closures and normal subgroups.

The **quotient** of a group G by a subgroup H, denoted G/H is a set of sets:

$$G/H \quad = \quad \{gH : g \in G\}.$$

Alternatively, the quotient may be defined in terms of an equivalence relation:

$$x \sim y \quad \iff \quad \exists h \in H, xh = y.$$

The binary operation is taken to be:

$$(fH)(gH) \quad = \quad (fg)H.$$

However, this operation is not in general well-defined. Let $aH = bH$ and $cH = dH$. The binary operation is well-defined if and only if: $(aH)(cH) = (bH)(dH)$. Because $b \in aH$, there exists $g \in H$ such that $b = ag$. Similarly, $d \in cH$, so there exists a $h \in H$ such that $d = ch$. Then, being well-defined implies:

$$\begin{aligned}
(aH)(cH) &= (bH)(dH) \\
(ac)H &= (bd)H \\
(ac)H &= (agch)H \\
a^{-1}(ac)H &= a^{-1}(agch)H \\
cH &= (gch)H \\
c^{-1}cH &= c^{-1}(gch)H \\
H &= (c^{-1}gch)H \\
H &\ni (c^{-1}gch)e \\
c^{-1}gch &\in H \\
c^{-1}gc &\in H.
\end{aligned}$$

Here, $c \in G$ is arbitrary. Further, I can choose $b = ag$ for any $a \in G$ and $g \in H$. It follows that H is closed under conjugation and is thus a normal subgroup. Being a normal subgroup is also a sufficient condition. This is shown by choosing a, b, c, d, arguing the existance of g and h as before, and reversing the order of the logic above. A quotient can be taken if and only if subgroup is normal.

Simple groups are groups that contain no normal subgroups other than the trivial group and the group itself, which are always normal subgroups. Simple groups are the group analog of a prime number. Quotients cannot be used to break simple groups into smaller groups.

If $K = G/H$, can G be obtained from K and H? The answer is "yes," and possibly in many ways. The simplest way is the **direct product**, denoted $H \oplus K$:

$$H \oplus K = \{(h,k) : h \in H, k \in K\}.$$

The binary operation is:

$$(a,b)(c,d) \quad = \quad (ac,bd).$$

The identity element is (e,e), and the inverse of (h,k) is (h^{-1},k^{-1}). Associativity and closure follow from the associativity and closure of groups H and K. It is left so show that $(H \oplus K)/H \cong K$. Note H is a subgroup of $H \oplus K$ because $H \cong \{(h,e), h \in H\} = (H,e)$. It is normal because $(a,b)(h,e)(a^{-1},b^{-1}) = (aha^{-1},bb^{-1}) = (aha^{-1},e) \in H$. The quotient is $(a,b)(H,e) = (aH,be) = (H,b)$. But $K \cong \{(H,k), k \in K\}$. Note that $H \oplus K \cong K \oplus H$, so that K is also a normal subgroup of $H \oplus K$, and $(H \oplus K)/K \cong H$.

A second way to construct G is by a **direct product**, denoted $H \rtimes K$, which differs in its definition for the binary operation:

$$\begin{aligned} (a,b)(c,d) \quad &= \quad (a\phi(b)(c),bd) \\ \phi \quad &: \quad H \rightarrow Aut(K), \end{aligned}$$

where ϕ is a group homomorphism from H to $Aut(K)$. The semidirect product is in general not uniquely defined, in that multiple choices of ϕ may exist. The choice $\phi(h) = id$ turns the semidirect product into a direct product. Nontrivial choices of ϕ lead to general direct products. In general, $H \rtimes K$ and $K \rtimes H$ are distinct, and K is not a normal subgroup of $H \rtimes K$. In fact, if either of these are false, then the semidirect product is in fact a direct product. As with a direct product, but H and K are subgroups, and H is a normal subgroup. Eg, $K \cong \{(e,k), k \in K\}$. Because $\phi(e) = id$, $(a,e)(c,e) = (ac,e)$, and $H \cong \{(h,e), h \in H\}$. The subgroup $H \cong (H,e)$ is normal because:

$$\begin{aligned}(a,b)(h,e)(a^{-1},b^{-1}) &= (a,b)(h\phi(e)(a^{-1}),eb^{-1}) \\ &= (a,b)(ha^{-1},eb^{-1}) \\ &= (a\phi(b)(ha^{-1}),beb^{-1}) \\ &= (a\phi(b)(ha^{-1}),e) \\ &\in (H,e).\end{aligned}$$

The quotient is $(a,b)(H,e) = (a\phi(b)(H),be) = (aH,b) = (H,b)$, as with the direct product. It is not hard to see that K being a normal subgroup makes the semidirect product a direct product:

$$\begin{aligned}(a,b)(e,k)(a^{-1},b^{-1}) &= (a\phi(b)(e),bk)(a^{-1},b^{-1}) \\ &= (ae,bk)(a^{-1},b^{-1}) \\ &= (a,bk)(a^{-1},b^{-1}) \\ &= (a\phi(bk)(a^{-1}),bkb^{-1}).\end{aligned}$$

Because the element is in (e,K) by assumption, $a\phi(bk)(a^{-1}) = e$ and $\phi(bk)(a^{-1}) = a^{-1}$. However, bk and a^{-1} are arbitrary, so that $\phi(k) = id$, making the semidirect product a direct product. If $H \rtimes K \cong K \rtimes H$, then K is a normal subgroup of $H \rtimes K$, and $H \oplus K \cong H \rtimes K \cong K \rtimes H$.

1.3 Outline of Thesis

This thesis is laid out as follows. Chapter 2 considers the problem of balancing metric trees under the L_2^2 norm. The primary focus of the chapter is on the problem of local updates, updating the weights immediately around the nodes involved in a tree rotation. The chapter solves the problem in closed form and is applicable to both left and right rotations. The chapter also solves the degenerate case of the local weight update problem, where one of the five edges is missing, and two of the remaining weights are not independent. The chapter also discusses an efficient way to maintain and update supplemental information

needed to perform the local updates. This chapter then generalizes the update algorithm to the case of a global update on the entire tree, expressing the result as the solution to a linear equation. The solution in each case is optimal.

Chapter 3 considers the same problem under the L_∞ norm. The solution for the local update case is considered first, and the solution is expressed in the form of a linear programming problem of constant size. The degenerate case is also considered, and the solutions to both are optimal. The chapter then considers the problem of updating and maintaining supplemental information needed to compute the local updates under the L_∞ norm. Next, the global update case is considered, and its solution is expressed as a linear programming problem whose size is linear in the number of edges in the metric tree. Simple distortion bounds are also presented in this chapter.

At this point, the emphasis of the thesis shifts from direct computation of edge weights after tree operations have been performed to a systematic analysis of the tree operations themselves. Though the results obtained do not yield an algorithm for balancing the metric trees, the group theoretical treatment of tree operations has, to the best of our knowledge, never been done previously.

Chapter 4 extends binary trees to an infinite, complete binary tree and formulates tree operations on the infinite tree. Using the infinite tree and the formulation of its operations, the operations are shown to satisfy the properties of a group. The chapter starts with exploratory and elementary results, but it quickly builds machinery for proving that the group is finitely generated. The chapter switches attention to subgroups of the group of operations, building up powerful machinery on the simpler, restricted subgroups. Machinery is built up in the group and subgroups to prove that the groups contain all finite groups and finite permutations. The chapter also proves that group of tree operations contains infinitely many copies of itself and gives a complete characterization of the group operations.

Chapter 5 focuses on identifying the normal subgroup structure of the group. The chapter begins by considering the effects on the group when operations at the root are removed to weaken the operations. This approach is used along with some additional machinery to construct the first normal subgroup of the group of tree operations. This, in turn, shows that the group is not simple. The chapter then turns to the conjugate closure operation and uses it to build up the machinery needed to solve the problem of finding normal subgroups completely, ending in a complete list of all normal subgroups. The chapter then switches focus to the much more common, but thus far unconsidered, subgroup generated by rotations alone. Finite

generation and a complete characterization are proven for this group.

Finally, in Chapter 6 we summarize the contributions of the thesis and present our conclusions.

2. Balancing Metric Trees under L_2

Given some distance metric, it is useful to be able to embed the metric into a tree structure. What we want to do is take this a step further by balancing the tree while minimizing the distortion of the resulting balanced tree. We are balancing using standard rotations, and these rotations can affect the distortion of the tree. In this chapter, we use the L_2 norm to calculate the distortion of the tree. This chapter is devoted to updating the weights of the tree to minimize this distortion, either locally or globally.

2.1 Local Edge Weight Updates

The case of a local update is considered first. In this case, it is assumed that we have a tree approximating the original metric, and we have just finished performing a rotation operation.

2.1.1 Notation

Let $D = [D_{ij}]_{n\times n}$ be the original metric; this metric is arbitrary. Let T be the tree obtained after a rotation, and let $E = [E_{ij}]_{n\times n}$ be the metric for this tree. Only the portion of the tree immediately around the site of the rotation needs to be considered in detail. The relevant nodes are shown before and after the rotation in Figure 2.1. The following text focuses on the tree T as it appears after the rotation.

Let T_a, where $a \in \{m,n,p,r\}$, denote the nodes of T reachable from α and β through a. Let L_a denote the leaves of T_a. We will use $\mu = |L_m|$, $\nu = |L_n|$, $\rho = |L_p|$, $\zeta = |L_r|$.

Define these edge lengths: $v = \alpha\beta$, $w = \alpha r$, $x = \alpha m$, $y = \beta n$, $z = \beta p$. These are the quantities to be determined, and they must be nonnegative.

2.1.2 Distortion

The total distortion after the rotation under the L_2 norm is:

$$\|E - D\|_2 = \sum_{i,j} (E_{ij} - D_{ij})^2. \tag{2.1}$$

Figure 2.1: Local structure of the tree T (a) before and (b) after rotation about the node β

This quantity depends on the to-be-determined variables $v, w, x, y, z \geq 0$. We will start by splitting the sum into ten parts $\{P_{mm}, P_{mn}, P_{mp}, P_{mr}, P_{nn}, P_{np}, P_{nr}, P_{pp}, P_{pr}, P_{rr}\}$ based on which subtree the leaves i and j are in. Define:

$$P_{ab} = \sum_{\substack{i \in L_a,\, j \in L_b \\ a,b \in \{m,n,p,r\}}} (E_{ij} - D_{ij})^2. \tag{2.2}$$

Combining (2.1) and (2.2) we can express the distortion in terms of P:

$$\sum_{i,j} (E_{ij} - D_{ij})^2 = \sum_{a,b \in \{m,n,p,r\}} P_{ab}. \tag{2.3}$$

In this way, the contributions to the distortion for distances between leaves in L_a and leaves in L_b are contained in P_{ab}. Note that P_{mm}, P_{nn}, P_{pp}, and P_{rr} do not depend on v, w, x, y, or z. We have thus cut down the part to be optimized down to:

$$P_{mn} + P_{mp} + P_{mr} + P_{np} + P_{nr} + P_{pr}. \tag{2.4}$$

Consider the distance between two distinct leaves i and j. In the six parts we are interested in, the paths

between i and j will go through two of $\{m,n,p,r\}$. The path goes from i to a to b and finally to j, where $a,b \in \{m,n,p,r\}$. Define δ_{ia}, δ_{ab}, and δ_{bj} to be the distances between i and a, a and b, and b and j along the tree. Then, we can write:

$$E_{ij} = \delta_{ia} + \delta_{ab} + \delta_{bj}. \tag{2.5}$$

At this point, it is worthwhile to simplify things by defining two new quantities:

$$R_{ab} = \sum_{i \in L_a, j \in L_b} (\delta_{ia} + \delta_{bj} - D_{ij})^2, \tag{2.6}$$

$$K_{ab} = \sum_{i \in L_a, j \in L_b} (\delta_{ia} + \delta_{bj} - D_{ij}). \tag{2.7}$$

Next, rewrite P_{ab}, for $a \neq b$, as follows:

$$\begin{aligned} P_{ab} &= \sum_{i \in L_a, j \in L_b} (\delta_{ia} + \delta_{ab} + \delta_{bj} - D_{ij})^2 \\ &= \sum_{i \in L_a, j \in L_b} (\delta_{ia} + \delta_{bj} - D_{ij})^2 + 2 \sum_{i \in L_a, j \in L_b} \delta_{ab}(\delta_{ia} + \delta_{bj} - D_{ij}) + \sum_{i \in L_a, j \in L_b} \delta_{ab}^2 \\ &= R_{ab} + 2\delta_{ab} K_{ab} + |L_a||L_b|\delta_{ab}^2. \end{aligned}$$

Because R_{ab} do not depend on v, w, x, y, or z, we may drop it from the quantity being optimized. Call the quantity being optimized M:

$$M = \sum_{\substack{a,b \in \{m,n,p,r\} \\ a \neq b}} (2\delta_{ab} K_{ab} + |L_a||L_b|\delta_{ab}^2). \tag{2.8}$$

2.1.3 Edge Weight Updates after Rotation

Restating δ_{ab} in terms of unknown variables v, w, x, y, and z we have:

$$\begin{aligned}
\delta_{mn} &= v+x+y, \\
\delta_{mp} &= v+x+z, \\
\delta_{mr} &= w+x, \\
\delta_{np} &= y+z, \\
\delta_{nr} &= v+w+y, \\
\delta_{pr} &= v+w+z.
\end{aligned} \tag{2.9}$$

Next, we compute partial derivatives of M with respect to v, w, x, y, and z, and let them be zero.

$$\begin{aligned}
\frac{\partial}{\partial v}M &= 2 \sum_{(a,b)\in\{(m,n),(m,p),(n,r),(p,r)\}} (K_{ab}+|L_a||L_b|\delta_{ab}) = 0, \\
\frac{\partial}{\partial w}M &= 2 \sum_{(a,b)\in\{(m,r),(n,r),(p,r)\}} (K_{ab}+|L_a||L_b|\delta_{ab}) = 0, \\
\frac{\partial}{\partial x}M &= 2 \sum_{(a,b)\in\{(m,n),(m,p),(m,r)\}} (K_{ab}+|L_a||L_b|\delta_{ab}) = 0, \\
\frac{\partial}{\partial y}M &= 2 \sum_{(a,b)\in\{(m,n),(n,p),(n,r)\}} (K_{ab}+|L_a||L_b|\delta_{ab}) = 0, \\
\frac{\partial}{\partial z}M &= 2 \sum_{(a,b)\in\{(m,p),(n,p),(p,r)\}} (K_{ab}+|L_a||L_b|\delta_{ab}) = 0.
\end{aligned}$$

Collecting all this into matrix form as a linear equation, we get the beautiful equation:

$$\begin{pmatrix} \mu\nu+\mu\rho+\nu\zeta+\rho\zeta & \nu\zeta+\rho\zeta & \mu\nu+\mu\rho & \mu\nu+\nu\zeta & \mu\rho+\rho\zeta \\ \nu\zeta+\rho\zeta & \mu\zeta+\nu\zeta+\rho\zeta & \mu\zeta & \nu\zeta & \rho\zeta \\ \mu\nu+\mu\rho & \mu\zeta & \mu\rho+\mu\nu+\mu\zeta & \mu\nu & \mu\rho \\ \mu\nu+\nu\zeta & \nu\zeta & \mu\nu & \mu\nu+\nu\zeta+\nu\rho & \nu\rho \\ \mu\rho+\rho\zeta & \rho\zeta & \mu\rho & \nu\rho & \mu\rho+\nu\rho+\rho\zeta \end{pmatrix} \begin{pmatrix} v \\ w \\ x \\ y \\ z \end{pmatrix}$$
$$= -\begin{pmatrix} K_{mn}+K_{mp}+K_{nr}+K_{pr} \\ K_{mr}+K_{nr}+K_{pr} \\ K_{mp}+K_{mn}+K_{mr} \\ K_{mn}+K_{nr}+K_{np} \\ K_{mp}+K_{np}+K_{pr} \end{pmatrix}. \tag{2.10}$$

Noting the factors in rows and columns we can rewrite this a bit. Letting $s = \mu+\nu+\rho+\zeta$, $t = \mu+\zeta$, and $u = \nu+\rho$ we can rewrite (2.10) as:

$$\begin{pmatrix} tu & u & u & t & t \\ u & s/\zeta-1 & 1 & 1 & 1 \\ u & 1 & s/\mu-1 & 1 & 1 \\ t & 1 & 1 & s/\nu-1 & 1 \\ t & 1 & 1 & 1 & s/\rho-1 \end{pmatrix} \begin{pmatrix} v \\ \zeta w \\ \mu x \\ \nu y \\ \rho z \end{pmatrix} = -\begin{pmatrix} K_{mn}+K_{mp}+K_{nr}+K_{pr} \\ (K_{mr}+K_{nr}+K_{pr})/\zeta \\ (K_{mp}+K_{mn}+K_{mr})/\mu \\ (K_{mn}+K_{nr}+K_{np})/\nu \\ (K_{mp}+K_{np}+K_{pr})/\rho \end{pmatrix}. \tag{2.11}$$

The determinant is $16tu$. Let $\sigma = t-u$ and

$$\xi = \sum_{a,b\in\{\zeta,\mu,\nu,\rho\},a\neq b} \frac{a}{b}. \tag{2.12}$$

Solving yields:

$$\begin{pmatrix} v \\ \zeta w \\ \mu x \\ \nu y \\ \rho z \end{pmatrix} = -\frac{1}{4} \begin{pmatrix} \frac{\xi}{tu} & \frac{2\mu-s}{\mu u} & \frac{2\zeta-s}{\zeta u} & \frac{2\rho-s}{\rho t} & \frac{2\nu-s}{\nu t} \\ \frac{2\mu-s}{\mu u} & \frac{\zeta s}{\mu u} & -\frac{\sigma}{u} & 0 & 0 \\ \frac{2\zeta-s}{\zeta u} & -\frac{\sigma}{u} & \frac{\mu s}{\zeta u} & 0 & 0 \\ \frac{2\rho-s}{\rho u} & 0 & 0 & \frac{\nu s}{\rho t} & \frac{\sigma}{t} \\ \frac{2\nu-s}{\nu u} & 0 & 0 & \frac{\sigma}{t} & \frac{\rho s}{\nu t} \end{pmatrix} \begin{pmatrix} K_{mn}+K_{mp}+K_{nr}+K_{pr} \\ (K_{mr}+K_{nr}+K_{pr})/\zeta \\ (K_{mp}+K_{mn}+K_{mr})/\mu \\ (K_{mn}+K_{nr}+K_{np})/\nu \\ (K_{mp}+K_{np}+K_{pr})/\rho \end{pmatrix}. \tag{2.13}$$

Moving the factors back into the matrix gives us:

$$\begin{pmatrix} v \\ w \\ x \\ y \\ z \end{pmatrix} = -\frac{1}{4} \begin{pmatrix} \frac{\xi}{tu} & \frac{2\mu-s}{\mu\zeta u} & \frac{2\zeta-s}{\mu\zeta u} & \frac{2\rho-s}{\nu\rho t} & \frac{2\nu-s}{\nu\rho t} \\ \frac{2\mu-s}{\mu\zeta u} & \frac{s}{\mu\zeta u} & -\frac{\sigma}{\mu\zeta u} & 0 & 0 \\ \frac{2\zeta-s}{\mu\zeta u} & -\frac{\sigma}{\mu\zeta u} & \frac{s}{\mu\zeta u} & 0 & 0 \\ \frac{2\rho-s}{\nu\rho t} & 0 & 0 & \frac{s}{\nu\rho t} & \frac{\sigma}{\nu\rho t} \\ \frac{2\nu-s}{\nu\rho t} & 0 & 0 & \frac{\sigma}{\nu\rho t} & \frac{s}{\nu\rho t} \end{pmatrix} \begin{pmatrix} K_{mn}+K_{mp}+K_{nr}+K_{pr} \\ K_{mr}+K_{nr}+K_{pr} \\ K_{mp}+K_{mn}+K_{mr} \\ K_{mn}+K_{nr}+K_{np} \\ K_{mp}+K_{np}+K_{pr} \end{pmatrix}. \tag{2.14}$$

2.1.4 Degeneracies and Negative Values

At this point, there are two concerns. The first concern is that this solution will be undefined. This occurs when one or more of μ, ν, ρ, or ζ is zero. (A physical interpretation is that one of the edge lengths is arbitrary because the edge is irrelevant and does not contribute to the distortion.)

Let $r = 0$ (i.e., we are rotating about the root of the tree). Then $K_{mr} = K_{nr} = K_{pr} = 0$. We no longer need to solve for w. Thus, the entire equation can be reduced by one degree:

$$\begin{pmatrix} v \\ x \\ y \\ z \end{pmatrix} = -\frac{1}{4} \begin{pmatrix} \frac{\xi}{tu} & \frac{2\zeta-s}{\mu\zeta u} & \frac{2\rho-s}{\nu\rho t} & \frac{2\nu-s}{\nu\rho t} \\ \frac{2\zeta-s}{\mu\zeta u} & \frac{s}{\mu\zeta u} & 0 & 0 \\ \frac{2\rho-s}{\nu\rho t} & 0 & \frac{s}{\nu\rho t} & \frac{\sigma}{\nu\rho t} \\ \frac{2\nu-s}{\nu\rho t} & 0 & \frac{\sigma}{\nu\rho t} & \frac{s}{\nu\rho t} \end{pmatrix} \begin{pmatrix} K_{mn}+K_{mp} \\ K_{mp}+K_{mn} \\ K_{mn}+K_{np} \\ K_{mp}+K_{np} \end{pmatrix}. \tag{2.15}$$

This does not remove all occurrences of ζ from the denominators, so it is necessary to reconsider the

original matrix with nonzero combinations of μ, ν, ρ, and ζ:

$$\begin{pmatrix} \mu\nu+\mu\rho+\nu\zeta+\rho\zeta & \nu\zeta+\rho\zeta & \mu\nu+\mu\rho & \mu\nu+\nu\zeta & \mu\rho+\rho\zeta \\ \nu\zeta+\rho\zeta & \mu\zeta+\nu\zeta+\rho\zeta & \mu\zeta & \nu\zeta & \rho\zeta \\ \mu\nu+\mu\rho & \mu\zeta & \mu\rho+\mu\nu+\mu\zeta & \mu\nu & \mu\rho \\ \mu\nu+\nu\zeta & \nu\zeta & \mu\nu & \mu\nu+\nu\zeta+\nu\rho & \nu\rho \\ \mu\rho+\rho\zeta & \rho\zeta & \mu\rho & \nu\rho & \mu\rho+\nu\rho+\rho\zeta \end{pmatrix}. \quad (2.16)$$

By examining the characteristic polynomial of this matrix, we observe that the rank is five when $\mu,\nu,\rho,\zeta \neq 0$. If one of these is zero, the rank drops immediately to thre. If two are zero, the rank becomes one. After three are zero, the matrix is identically zero. By construction of our matrix, the rotating nodes always have two children. Thus, $\mu,\nu,\zeta \neq 0$ and only $\rho = 0$ is possible.

Consider the distances δ_{ab} again, ignoring those relying on ζ (Since there are no nodes using these distances). We can restate:

$$\begin{aligned} \delta_{mn} &= v+x+y, \\ \delta_{mp} &= v+x+z, \\ \delta_{np} &= y+z. \end{aligned}$$

There are dependencies on $v+x$, but no other dependence on either v or x. Letting $v = 0$, we have:

$$\begin{pmatrix} \mu\nu+\mu\rho & \mu\nu+\mu\rho & \mu\nu & \mu\rho \\ \mu\nu+\mu\rho & \mu\rho+\mu\nu & \mu\nu & \mu\rho \\ \mu\nu & \mu\nu & \mu\nu+\nu\rho & \nu\rho \\ \mu\rho & \mu\rho & \nu\rho & \mu\rho+\nu\rho \end{pmatrix} \begin{pmatrix} 0 \\ x \\ y \\ z \end{pmatrix} = - \begin{pmatrix} K_{mn}+K_{mp} \\ K_{mp}+K_{mn} \\ K_{mn}+K_{np} \\ K_{mp}+K_{np} \end{pmatrix}. \quad (2.17)$$

The equation can now be reduced to:

$$\begin{pmatrix} \mu\rho+\mu\nu & \mu\nu & \mu\rho \\ \mu\nu & \mu\nu+\nu\rho & \nu\rho \\ \mu\rho & \nu\rho & \mu\rho+\nu\rho \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = - \begin{pmatrix} K_{mp}+K_{mn} \\ K_{mn}+K_{np} \\ K_{mp}+K_{np} \end{pmatrix}. \tag{2.18}$$

The determinant of the coefficient matrix is $4\mu^2\nu^2\rho^2 > 0$, so the matrix is invertible. Note that $s = \mu+\nu+\rho$ here:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = -\frac{1}{4\mu\nu\rho} \begin{pmatrix} s & 2\rho-s & 2\nu-s \\ 2\rho-s & s & 2\mu-s \\ 2\nu-s & 2\mu-s & s \end{pmatrix} \begin{pmatrix} K_{mp}+K_{mn} \\ K_{mn}+K_{np} \\ K_{mp}+K_{np} \end{pmatrix}. \tag{2.19}$$

For practical implementation, it may be better to choose v and x equal, in which case, assign $x/2$ to each.

The second concern is harder to deal with; it may be possible for the values computed to be negative. Changing negative values to zero seems like a fair approach. A much better approach is to use a quadratic programming formulation, which we do in Section 2.2.

2.1.5 Efficient Computation

We make use of K_{ab} and $|L_a|$ in the computation of the edge weights. (The matrix depends only on $|L_a|$.) Thus, balancing the tree can be done efficiently if these can be stored and updated upon rotation. $|L_a|$ is trivial; it is the number of leaves beyond a and does not change. However, we must also update these for nodes α and β: $|L_{\beta|r}| = |L_n| + |L_p|$ and $|L_{\alpha|r}| = |L_m| + |L_n| + |L_p|$. Here, $L_{\alpha|r}$ and $L_{\beta|r}$ are taken to be the leaves reachable from node r through α and β. Consider K_{ab}:

$$K_{ab} = \sum_{i \in L_a, j \in L_b} (\delta_{ia} + \delta_{bj} - D_{ij}). \tag{2.20}$$

If we define $S_a = \sum_{i \in L_a} \delta_{ia}$, $a \in \{m, n, p, r, \alpha, \beta\}$ then K_{ab} simplifies to:

$$K_{ab} = |L_b|S_a + |L_a|S_b - \sum_{i \in L_a, j \in L_b} D_{ij}. \tag{2.21}$$

Note that:

$$\begin{aligned} S_\beta &= \sum_{i \in L_n \cup L_p} \delta_{i\beta} \\ &= \sum_{i \in L_n} (\delta_{in} + \delta_{n\beta}) + \sum_{i \in L_p} (\delta_{ip} + \delta_{p\beta}) \\ &= \sum_{i \in L_n} \delta_{in} + |L_n|\delta_{n\beta} + \sum_{i \in L_p} \delta_{ip} + |L_p|\delta_{p\beta} \\ &= S_n + S_p + |L_n|y + |L_p|z. \end{aligned}$$

Similarly, we get S_α from S_β:

$$\begin{aligned} S_\alpha &= \sum_{i \in L_m \cup L_{\beta|r}} \delta_{i\alpha} \\ &= \sum_{i \in L_m} (\delta_{im} + \delta_{m\alpha}) + \sum_{i \in L_{\beta|r}} (\delta_{i\beta} + \delta_{\beta\alpha}) \\ &= \sum_{i \in L_m} \delta_{im} + |L_m|\delta_{m\alpha} + \sum_{i \in L_{\beta|r}} \delta_{i\beta} + |L_{\beta|r}|\delta_{\beta\alpha} \\ &= S_m + S_\beta + |L_m|x + |L_{\beta|r}|v. \end{aligned}$$

Let:

$$F_{ab} = \sum_{i \in L_a, j \in L_b} D_{ij}. \tag{2.22}$$

Store F_{ab} for all node pairs a, b, noting that $F_{ab} = F_{ba}$. Consider the process of updating F_{ab}. Let a's neighbors be i, j, and k, and assume that k is its neighbor along the path ab, as shown in Figure 2.2. Then:

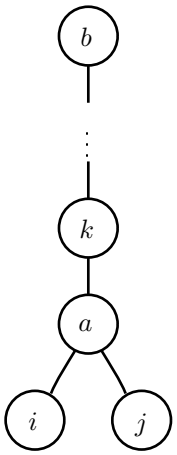


Figure 2.2: Let a's neighbors be i, j, and k. Assume that k is the neighbor along the path ab.

$$\begin{aligned} F_{ab} &= \sum_{u\in L_a, v\in L_b} D_{uv} \\ &= \sum_{u\in L_i\cup L_j, v\in L_b} D_{uv} \\ &= \sum_{u\in L_i, v\in L_b} D_{uv} + \sum_{u\in L_j, v\in L_b} D_{uv} \\ &= F_{ib} + F_{jb}. \end{aligned}$$

Thus, we can compute an entry by looking just beyond one of the ends (looking beyond a in this case).

Next we will consider the update rules. Consider first the ways in which F_{ab} can remain unchanged. It is unchanged if $a,b \in T_m \cup T_n \cup T_p \cup T_r$. This is because the value of F_{ab} depends only on which leaves lie beyond a and b. Since L_a and L_b are not changed, F_{ab} is unchanged. F_{ab} is also unchanged if $a = \alpha, b \in T_m$ or $a = \beta, b \in T_p$. This is because the nodes beyond α and β in these cases are the same before and after the rotation. The cases that need to be updated are $F_{\alpha\beta}$, $F_{\alpha x}$, $F_{\beta y}$ where $x \in T_n \cup T_p \cup T_r$ and $y \in T_m \cup T_n \cup T_r$.

Next we can update by noting that $F_{\alpha x} = F_{rx} + F_{mx}$ for $x \in T_n \cup T_p$. Similarly, $F_{\beta y} = F_{ny} + F_{py}$ for $x \in T_m \cup T_r$. Finally, we can compute the rest; $F_{\alpha w} = F_{mw} + F_{\beta w}$ for $w \in T_r$, $F_{\beta z} = F_{pz} + F_{\alpha z}$ for $z \in T_n$, and $F_{\alpha\beta} = F_{\alpha n} + F_{\alpha p}$. This completes the updates on F_{ab}.

2.1.6 Global Edge Weight Updates

The same ideas can be applied to the entire tree after an arbitrary change. In this case, we will be updating all weights in the tree. Reusing the notation from the local updates, the total distortion P of the tree is:

$$P = \sum_{i,j} (E_{ij} - D_{ij})^2. \tag{2.23}$$

Let w_k, $1 \leq k \leq 2n-3$, represent the weights to be assigned to the edges of the tree. Because the weights of the edges leaving the root of the tree are not independent, for the purposes of computing weights, we take one of the weights to be zero and do not assign to it a variable. There are n leaves, $2n-2$ edges, and $2n-3$ weights.

Let $A_{ijk} = 1$ if w_k is on the path from leaf i to leaf j; $A_{ijk} = 0$ otherwise. Note that $A_{ijk} = A_{jik}$, so that A is symmetric in its first two indices. Note that A_{ijk} is fixed, since it is determined by the tree's topology and the leaf and weight labels. We can use A_{ijk} to express E_{ij} in terms of the weights w_k:

$$E_{ij} = \sum_k A_{ijk} w_k. \tag{2.24}$$

Combining (2.23) and (2.24) we have:

$$P = \sum_{i,j} \left(\sum_k A_{ijk} w_k - D_{ij} \right)^2. \tag{2.25}$$

We must minimize P with respect to the variables w_k. We accomplish this by computing the partials of P with respect to those variables.

$$\begin{aligned}
\frac{\partial}{\partial w_k}P &= \frac{\partial}{\partial w_k}\sum_{i,j}\left(\sum_m A_{ijm}w_m - D_{ij}\right)^2 \\
&= \sum_{i,j}\frac{\partial}{\partial w_k}\left(\sum_m A_{ijm}w_m - D_{ij}\right)^2 \\
&= 2\sum_{i,j}\left(\sum_m A_{ijm}w_m - D_{ij}\right)\frac{\partial}{\partial w_k}\left(\sum_m A_{ijm}w_m - D_{ij}\right) \\
&= 2\sum_{i,j}\left(\sum_m A_{ijm}w_m - D_{ij}\right)\frac{\partial}{\partial w_k}\sum_m A_{ijm}w_m \\
&= 2\sum_{i,j}A_{ijk}\left(\sum_m A_{ijm}w_m - D_{ij}\right) \\
&= 2\sum_{i,j}A_{ijk}\sum_m A_{ijm}w_m - 2\sum_{i,j}A_{ijk}D_{ij}.
\end{aligned}$$

Setting these to zero gives us $2n-3$ equations in $2n-3$ unknowns, which we can write this as a system of equations:

$$Mw = b, \tag{2.26}$$

where $w = (w_k)$ is the unknown vector, and the kth entry of the right hand vector is:

$$b_k = \sum_{i,j}A_{ijk}D_{ij}. \tag{2.27}$$

Substituting these and comparing with the equations, we find that M_{km} must be given by:

$$M_{km} = \sum_{i,j}A_{ijk}A_{ijm}. \tag{2.28}$$

The weights are now $w = M^{-1}b$, provided M is invertible. As before, the weights obtained in this way may be negative. When this happens, these weights must be forced into the feasible region, such as by

making them zero.

Theorem 1. *Matrix M is invertible.*

A splitting tree T is a tree in which every non-leaf has degree three. Let T have n leaves and $2n-3$ edges. Call the weights of these edges w_k, $1 \leq k \leq 2n-3$, and label the leaves 1 through n. Define $\delta : [n] \times [n] \rightarrow \mathbb{R}$ so that $\delta(i,j)$ is the weight of the path from leaf i to leaf j, the sum of the weights of the edges along the path.

Claim 2. *Every splitting tree T has the property that there exists a list L of $2n-3$ pairs $(i,j) \in [n] \times [n]$, such that the $2n-3$ weights $\delta(L_i)$ uniquely determine the $2n-3$ weights w_k.*

Proof: The proof is inductive on the size of the tree. For the inductive step, $n \geq 3$. Let l be some leaf, w_{2n-3} be the weight of edge (l,m) for some node m, w_{2n-4} be the weight of edge (m,r) for some r, and w_{2n-5} be the weight of edge (m,s) for some s. Create from this tree T on n leaves a tree T' on $n-1$ leaves by removing leaf c and node m. Let $w'_k = w_k$ for $k < 2n-5$ and replace the edges (m,r) and (m,s) with an edge (r,s) whose weight is $w'_{2n-5} = w_{2n-5} + w_{2n-4}$. Compute path weights $\delta'(i,j)$ using edge weights w'. Tree T' is a splitting tree, since the remaining nodes and leaves have the same degree as in T. By the inductive hypothesis, this tree has a list L' of $2n-5$ pairs $(i,j) \in [n] \times [n]$, such that the $2n-5$ weights $\delta'(L'_i)$ uniquely determine the $2n-3$ weights w_k. We can write the path weights $\delta'(L'_i)$ as a linear combination of the edge weights w'_k:

$$\delta'(L'_i) = \sum_{k=1}^{2n-5} B'_{ik} w'_k, \tag{2.29}$$

where $B'_{ik} = 1$ if the edge with weight w'_k is along path L'_i, and $B'_{ik} = 0$ otherwise. Because w'_k are uniquely defined in this way, $|B'| \neq 0$. Let a be a leaf in T reachable from l through r, and let b be some leaf in T reachable from l through s. Let $L_i = L'_i$ for $1 \leq i \leq 2n-5$, $L_{2n-4} = (l,a)$, and $L_{2n-3} = (l,b)$. Letting single indices $B'^{[i]}$ indicate columns. The corresponding matrix B has the following form:

$$B = \begin{pmatrix} B'^{[1]} & \cdots & B'^{[2n-6]} & B'^{[2n-5]} & B'^{[2n-5]} & 0 \\ * & \cdots & * & 0 & 1 & 1 \\ * & \cdots & * & 1 & 0 & 1 \end{pmatrix}, \tag{2.30}$$

where $*$ indicate entries whose values do not affect later computations. The determinant $|B|$ is

$$\begin{aligned} |B| &= \begin{vmatrix} B'^{[1]} & \cdots & B'^{[2n-6]} & B'^{[2n-5]} & B'^{[2n-5]} & 0 \\ * & \cdots & * & 0 & 1 & 1 \\ * & \cdots & * & 1 & 0 & 1 \end{vmatrix} \\ &= -\begin{vmatrix} B'^{[1]} & \cdots & B'^{[2n-6]} & B'^{[2n-5]} & B'^{[2n-5]} \\ * & \cdots & * & 1 & 0 \end{vmatrix} + \begin{vmatrix} B'^{[1]} & \cdots & B'^{[2n-6]} & B'^{[2n-5]} & B'^{[2n-5]} \\ * & \cdots & * & 0 & 1 \end{vmatrix} \\ &= 2\begin{vmatrix} B'^{[1]} & \cdots & B'^{[2n-6]} & B'^{[2n-5]} & B'^{[2n-5]} \\ * & \cdots & * & 0 & 1 \end{vmatrix} \\ &= 2\begin{vmatrix} B'^{[1]} & \cdots & B'^{[2n-6]} & B'^{[2n-5]} \end{vmatrix} \\ &= 2|B'|. \end{aligned} \tag{2.31}$$

The first expansion by minors is along the last column, and the second is along the last row, noting that all other cofactors will have zero determinant due to the duplicate columns $B'^{[2n-5]}$. The important observation is that $|B| \neq 0$, implying that the path weights $\delta(L_i)$ uniquely define edge weights w_k.

The base case is the tree with $n = 2$ leaves i and j and an edge connecting them. In this case, B is just the 1×1 identity matrix, and $L_1 = (i, j)$. Note that in the general case, this construction produces a B for any splitting tree such that $|B| = 2^{n-2}$. ■

Proof of Theorem 1: Return to the situation where we have a rooted tree T on n leaves. Removing the root and setting the weight of one of the edges to the root to zero yields a splitting tree. From this, we have a nonsingular matrix B that expresses the path weight $\delta(L_i)$ in terms of the weights w_k. If $L_i = (a, b)$, then we let $L_{i1} = a$ and $L_{i2} = b$. The relationship between A_{ijk} and B_{mk} can be expressed as:

$$B_{mk} = A_{L_{m1}L_{m2}k}. \tag{2.32}$$

Regarding A as an $n^2 \times (2n-3)$ matrix by merging the first two indices, then $M = A^T A$. The relationship between A and B implies that A contains a $(2n-3) \times (2n-3)$ submatrix with nonzero determinant, so that A has rank $2n-3$. It follows that M has rank $2n-3$ and is invertible. ■

2.2 Quadratic Programming Formulations

The least squares formulation left one question unanswered. That is, how should the case of negative edge weights be handled? An alternate formulation using quadratic programming solves this problem, while retaining a polynomial runtime. There are again three situations that need to be treated: local update, degenerate local update, and global update.

2.2.1 Local Update

Much of the least squares framework is directly applicable to quadratic programming. In particular, if we expand (2.8) we get:

$$\begin{aligned} M &= 2\delta_{mn}K_{mn} + 2\delta_{mp}K_{mp} + 2\delta_{mr}K_{mr} + 2\delta_{np}K_{np} + 2\delta_{nr}K_{nr} + 2\delta_{pr}K_{pr} \\ &\quad + |L_m||L_n|\delta_{mn}^2 + |L_m||L_p|\delta_{mp}^2 + |L_m||L_r|\delta_{mr}^2 + |L_n||L_p|\delta_{np}^2 + |L_n||L_r|\delta_{nr}^2 + |L_p||L_r|\delta_{pr}^2. \\ &= 2\delta_{mn}K_{mn} + 2\delta_{mp}K_{mp} + 2\delta_{mr}K_{mr} + 2\delta_{np}K_{np} + 2\delta_{nr}K_{nr} + 2\delta_{pr}K_{pr} \\ &\quad + \mu\nu\delta_{mn}^2 + \mu\rho\delta_{mp}^2 + \mu\zeta\delta_{mr}^2 + \nu\rho\delta_{np}^2 + \nu\zeta\delta_{nr}^2 + \rho\zeta\delta_{pr}^2. \end{aligned}$$

This suggests the following definitions for a matrix Q and vectors $\mathbf{x}$ and c:

$$Q = \begin{pmatrix} \mu\nu & 0 & 0 & 0 & 0 & 0 \\ 0 & \mu\rho & 0 & 0 & 0 & 0 \\ 0 & 0 & \mu\zeta & 0 & 0 & 0 \\ 0 & 0 & 0 & \nu\rho & 0 & 0 \\ 0 & 0 & 0 & 0 & \nu\zeta & 0 \\ 0 & 0 & 0 & 0 & 0 & \rho\zeta \end{pmatrix},$$

$$\mathbf{x} = \begin{pmatrix} \delta_{mn} \\ \delta_{mp} \\ \delta_{mr} \\ \delta_{np} \\ \delta_{nr} \\ \delta_{pr} \end{pmatrix}, \qquad c = \begin{pmatrix} K_{mn} \\ K_{mp} \\ K_{mr} \\ K_{np} \\ K_{nr} \\ K_{pr} \end{pmatrix}.$$

This allows us to write M as a quadratic programming objective function:

$$\frac{1}{2}M = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + c^T\mathbf{x}. \tag{2.33}$$

This objective function has five inequality constraints. In particular, the six equations from (2.9) and the nonnegativity conditions $v, w, x, y, z \geq 0$ yield constraints on $\mathbf{x}$. We only need five of those equations to express the variables v, w, x, y, and z. In matrix form, we have:

$$\mathbf{x} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ w \\ x \\ y \\ z \end{pmatrix}. \tag{2.34}$$

We need to invert this equation, but the matrix is not square. We may overcome this by ignoring the last row of the matrix, invert as a 5×5 matrix, and insert zeros as the last column.

$$\begin{pmatrix} v \\ w \\ x \\ y \\ z \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 0 & 1 & -1 & -1 & 1 & 0 \\ -1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & -1 & 0 \\ 1 & -1 & 0 & 1 & 0 & 0 \\ -1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} \mathbf{x} \geq 0.$$

This takes care of the inequality constraints, but there is still a problem with this formulation. In particular, there are six independent variables where before there were just five. We must eliminate one degree of freedom by introducing a constraint. This constraint may be found by examining the nullspace of the transpose of the 6×5 matrix in (2.34). The nullspace is $(1,-1,0,0,-1,1)$. This leads to the constraint $\delta_{mn} - \delta_{mp} - \delta_{nr} + \delta_{pr} = 0$, where the equality to zero is obtained by substitution of (2.9). At this point, it is worth noting that the form of the nullspace implies that we could not have ignored the third or fourth rows before taking the inverse, since the nullspace is identically zero on those coordinates. Taking advantage of the fact that $a \geq 0$ and $-a \geq 0$ imply $a = 0$, we can express all of the constraints as one matrix inequality:

$$\begin{pmatrix} 0 & 1 & -1 & -1 & 1 & 0 \\ -1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & -1 & 0 \\ 1 & -1 & 0 & 1 & 0 & 0 \\ -1 & 1 & 0 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 \\ -1 & 1 & 0 & 0 & 1 & -1 \end{pmatrix} \mathbf{x} \geq 0 \tag{2.35}$$

It only remains to show that we can solve this problem in polynomial time. In particular, the problem can be solved in polynomial time to any prescribed error with the ellipsoid method if Q is positive-definite [15, 16, 6]. However, we have $|L_a| \geq 1$ for $a \in \{m,n,p,r\}$ or $\mu,\nu,\rho,\zeta \geq 1$. Then, Q is a diagonal matrix with positive elements along the diagonal. It follows that Q is positive-definite, and the problem of local updates can be solved in polynomial time. This statement has actually only been shown to be true in the general case, since it does not consider the degenerate case of rotations at the root, which is the topic of the next section.

2.2.2 Degenerate Local Update

The degenerate case is where the rotation was performed at the root. In this case, $|L_r| = \zeta = 0$, so the analysis above does not suffice. Further, δ_{mr}, δ_{nr}, and δ_{pr} do not make sense. Substituting $\zeta = 0$ in Q, we see that three of the six diagonal entries become zero, and those portions will contribute nothing to the sum. Further $K_{mr} = K_{nr} = K_{pr} = 0$ as before. Removing the terms that do not contribute from the definitions of Q, $\mathbf{x}$, and c, define:

$$Q' = \begin{pmatrix} \mu\nu & 0 & 0 \\ 0 & \mu\rho & 0 \\ 0 & 0 & \nu\rho \end{pmatrix},$$

$$\mathbf{x}' = \begin{pmatrix} \delta_{mn} \\ \delta_{mp} \\ \delta_{np} \end{pmatrix}, \qquad c' = \begin{pmatrix} K_{mn} \\ K_{mp} \\ K_{np} \end{pmatrix}.$$

Once again we have a diagonal matrix Q' with positive entries along the diagonal. If we can update the constraints appropriately, we will have a solution in polynomial time using the ellipsoid method. We return to (2.34) and remove the third, fifth, and sixth rows to account for the rows eliminated from $\mathbf{x}$:

$$\mathbf{x}' = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} v \\ w \\ x \\ y \\ z \end{pmatrix}.$$

This 3×5 matrix has a nullspace with two basis vectors $(0,1,0,0,0)$ and $(1,0,-1,0,0)$. The first states that $w = 0$, and the second states that $v + x = 0$, which are precisely what we determined in the least squares formulation. We let $v = x$ and remove v and w from the equation, yielding:

$$\mathbf{x}' = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

The constraints $x, y, z \geq 0$ are obtained by inverting this equation:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = 2 \begin{pmatrix} 1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \end{pmatrix} \mathbf{x}' \geq 0.$$

Unlike the general case, we do not have any extra degrees of freedom, so there are no equality constraints needed. The degenerate case of the local update problem is solvable in linear time by this formulation.

2.2.3 Global Update

We now turn our attention to the global update problem for formulation as a quadratic programming problem. This formulation, like the local formulation, begins by reusing some of the results from the least squares formulation. We begin by rewriting (2.25) by expanding the square:

$$\begin{aligned}
P &= \sum_{i,j}\left(\sum_k A_{ijk}w_k - D_{ij}\right)^2 \\
&= \sum_{i,j}\left(\left(\sum_k A_{ijk}w_k\right)^2 - 2D_{ij}\sum_k A_{ijk}w_k + D_{ij}^2\right) \\
&= \sum_{i,j}\left(\sum_k A_{ijk}w_k\right)^2 - 2\sum_{i,j} D_{ij}\sum_k A_{ijk}w_k + \sum_{i,j} D_{ij}^2 \\
&= \sum_{i,j,k,m} A_{ijk}w_k A_{ijm}w_m - 2\sum_k w_k \sum_{i,j} D_{ij}A_{ijk} + \sum_{i,j} D_{ij}^2 \\
&= \sum_{k,m} w_k M_{km} w_m - 2w_k \sum_{i,j} D_{ij}A_{ijk} + \sum_{i,j} D_{ij}^2.
\end{aligned}$$

We simplify this by defining

$$\begin{aligned}
c_k &= -\sum_{i,j} D_{ij}A_{ijk}, \\
P' &= \frac{1}{2}P - \frac{1}{2}\sum_{i,j} D_{ij}^2.
\end{aligned}$$

In particular, P' differs from P by a positive constant factor and a constant shift, so minimizing P' and P are equivalent. Writing out P' gives:

$$\begin{aligned} P' &= \frac{1}{2}\sum_{k,m}\sum_{k} w_k M_{km} w_m + \sum_k w_k \sum_{i,j} c_k \\ &= \frac{1}{2} w^T M w + c^T w \end{aligned}$$

The global update problem is solved in polynomial time if we can show two things. The first is that M is positive-definite. From (2.28) we know that M is positive-semidefinite. Theorem 1 strengthens this and shows that M is positive-definite. This leaves only the final piece, expressing the constraints in linear form. This is trivial, because the constraints are expressed as $w \geq 0$.

Note that the choice of weights as variables made the constraints trivial but the matrix M less so. The same choice is possible with the local update case by substituting (2.34) into (2.33) and taking the column of weights to be the new variable vector.

2.3 Conclusion

The problem of exact computation of local edge weights for a metric tree under the L_2 norm can be formulated as a linear system of equations with a fixed number of constraints, thus allowing efficient re-computation of edge weights after a rotation operation. Updating all edges in the tree at once under the same norm can be formulated as a system of linear equations that is linear in the size and admits a polynomial time solution. Both solutions, however, may result in negative weights being chosen for edges, so the solutions obtained may need to be revised. An alternate formulation in terms of quadratic programming solves the negative weight problem by including the constraints in the actual solution. The result is an optimal polynomial algorithm for both local and global updates.

3. Balancing Metric Trees under L_∞

In this section, we shift our focus from the L_2 to the L_∞ norm. There is an efficient 3-approximate embedding of an arbitrary distance metric into a tree [2]. This makes the L_∞ norm particularly attractive for trees. As with the L_2 norm, the focus is on rotations. Both local and global edge weight updates are considered.

3.1 Local Edge Weight Updates

The case of a local update is considered first. In this case, it is assumed that we have a tree approximating the original metric, and we have just finished performing a rotation operation. This section begins with an introduction to the notation used throughout the chapter. The notation is the same as the notation from the previous chapter and is included for purposes of being self-contained.

3.1.1 Notation

Let $D = [D_{ij}]_{n\times n}$ be the original metric, and $E = [E_{ij}]_{n\times n}$ denote the metric for the tree T after a single rotation has been performed. Let T_a, where $a \in \{m,n,p,r\}$, denote the nodes of T reachable from α and β through a. Let L_a denote the leaves of T_a. We will use $\mu = |L_m|$, $\nu = |L_n|$, $\rho = |L_p|$, and $\zeta = |L_r|$. Define these edge lengths: $v = \overline{\alpha\beta}$, $w = \overline{\alpha r}$, $x = \overline{\alpha m}$, $y = \overline{\beta n}$, and $z = \overline{\beta p}$. These are the quantities to be determined, and they must be nonnegative.

3.1.2 Distortion

The total distortion after the rotation under the L_∞ norm is:

$$P = \|E - D\|_\infty = \max_{i,j} |E_{ij} - D_{ij}|. \tag{3.1}$$

Consider the pairwise distances split into ten categories $P_{ab} \in \{P_{mm}, P_{mn}, P_{mp}, P_{mr}, P_{nn}, P_{np}, P_{nr}, P_{pp}, P_{pr}, P_{rr}\}$, where one leaf is in subtree L_a, the other leaf is in L_b, and $a,b \in \{m,n,p,r\}$:

Figure 3.1: Local structure of the tree T (a) before and (b) after rotation about the node β

$$P_{ab} = \max_{i \in L_a, j \in L_b} |E_{ij} - D_{ij}|. \tag{3.2}$$

Combining (3.1) and (3.2) we can express P in terms of P_{ab}:

$$P = \max_{a,b} P_{ab}. \tag{3.3}$$

Changing the weights of edges around the rotation does not affect distances between nodes lying completely within a subtree L_a, so P_{aa} is unaffected by the choice of weights.

Fix a choice of a, b, where $a \neq b$. We now deal with distances between leaves in L_a and L_b. Let γ_{ab} denote contribution of edge weights v, w, x, y, and z to the weight of a path from a leaf in L_a to a leaf in L_b. Fix $v = w = x = y = z = 0$ so that $\gamma_{ab} = 0$, and compute the quantities:

$$P^{*+}_{ab} = \max_{i \in L_a, j \in L_b} (E^*_{ij} - D_{ij}), \tag{3.4}$$

$$P^{*-}_{ab} = \max_{i \in L_a, j \in L_b} -(E^*_{ij} - D_{ij}). \tag{3.5}$$

The star superscript indicates that the value was computed with $\gamma_{ab} = 0$. Now, unfix v, w, x, y, and z and consider that $\gamma_{ab} \geq 0$, as it will be in the general case, and let:

$$P^{+}_{ab} = \max_{i\in L_a, j\in L_b}(E_{ij} - D_{ij}), \tag{3.6}$$

$$P^{-}_{ab} = \max_{i\in L_a, j\in L_b} -(E_{ij} - D_{ij}). \tag{3.7}$$

These two sets of quantities are related:

$$\begin{aligned} P^{+}_{ab} &= \max_{i\in L_a, j\in L_b}(E_{ij} - D_{ij}) \\ &= \max_{i\in L_a, j\in L_b}((E^{*}_{ij} + \gamma_{ab}) - D_{ij}) \\ &= \max_{i\in L_a, j\in L_b}(\gamma_{ab} + (E^{*}_{ij} - D_{ij})) \\ &= \gamma_{ab} + \max_{i\in L_a, j\in L_b}(E^{*}_{ij} - D_{ij}) \\ &= \gamma_{ab} + P^{*+}_{ab}. \end{aligned}$$

Similarly, we have:

$$\begin{aligned} P^{-}_{ab} &= \max_{i\in L_a, j\in L_b} -(E_{ij} - D_{ij}) \\ &= \max_{i\in L_a, j\in L_b} -((E^{*}_{ij} + \gamma_{ab}) - D_{ij}) \\ &= \max_{i\in L_a, j\in L_b}(-(E^{*}_{ij} - D_{ij}) - \gamma_{ab}) \\ &= \max_{i\in L_a, j\in L_b} -(E^{*}_{ij} - D_{ij}) - \gamma_{ab} \\ &= P^{*+}_{ab} - \gamma_{ab}. \end{aligned}$$

Using (3.8) and (3.8) we can express P_{ab} in terms of P^{*+} and P^{*-}:

$$\begin{aligned} P_{ab} &= \max(P^{+}_{ab}, P^{-}_{ab}) \\ &= \max(P^{*+}_{ab} + \gamma_{ab}, P^{*-}_{ab} - \gamma_{ab}). \end{aligned}$$

The quantities P_{ab}^{*+} and P_{ab}^{*-} are not affected by the choice of weights (and thus the choice of γ_{ab}). Unfix a and b, and let $P^{eq} = \max_a P_{aa}$. Our distortion can be written:

$$P = \max\left(P^{eq}, \max_{ab}(P_{ab}^{*+} + \gamma_{ab}), \max_{ab}(P_{ab}^{*-} - \gamma_{ab})\right). \tag{3.8}$$

The inner maxima are over six quantities. Thus, the outer maximum is to be computed over 13 quantities.

3.1.3 Edge Weight Updates after Rotation

Using the variables v, w, x, y, and z, we have: $\gamma_{mr} = w+x$, $\gamma_{mn} = x+v+y$, $\gamma_{mp} = x+v+z$, $\gamma_{nr} = w+v+y$, $\gamma_{np} = y+z$, and $\gamma_{pr} = w+v+z$.

$$\begin{aligned}
\text{MINIMIZE:} \quad & M \\
\text{SUBJECT TO:} \quad & P_{mr}^{*+} + w + x \le M, && P_{mr}^{*-} - w - x \le M, \\
& P_{mn}^{*+} + x + v + y \le M, && P_{mn}^{*-} - x - v - y \le M, \\
& P_{mp}^{*+} + x + v + z \le M, && P_{mp}^{*-} - x - v - z \le M, \\
& P_{nr}^{*+} + w + v + y \le M, && P_{nr}^{*-} - w - v - y \le M, \\
& P_{np}^{*+} + y + z \le M, && P_{np}^{*-} - y - z \le M, \\
& P_{pr}^{*+} + w + v + z \le M, && P_{pr}^{*-} - w - v - z \le M, \\
& P^{eq} \le M, && v, w, x, y, z, M \ge 0.
\end{aligned}$$

In canonical form:

$$\begin{aligned}
\text{MAXIMIZE:} \quad & -M \\
\text{SUBJECT TO:} \quad & M - w - x \ge P_{mr}^{*+}, && M + w + x \ge P_{mr}^{*-}, \\
& M - x - v - y \ge P_{mn}^{*+}, && M + x + v + y \ge P_{mn}^{*-}, \\
& M - x - v - z \ge P_{mp}^{*+}, && M + x + v + z \ge P_{mp}^{*-}, \\
& M - w - v - y \ge P_{nr}^{*+}, && M + w + v + y \ge P_{nr}^{*-}, \\
& M - y - z \ge P_{np}^{*+}, && M + y + z \ge P_{np}^{*-}, \\
& M - w - v - z \ge P_{pr}^{*+}, && M + w + v + z \ge P_{pr}^{*-}, \\
& M \ge P^{eq}, && v, w, x, y, z, M \ge 0.
\end{aligned}$$

This is a linear programming problem with 13 constraints and six variables. Observe that $v = w = x = y = z = 0$ and M sufficiently large is a feasible solution and provides an upper bound on M. Further, $M \geq 0$, showing that M is bounded and thus has an optimal solution.

Look back to the first statement of the constraints. Each constraint consists of a quantity that is less than M, and the equations collectively state that M is (at least) the maximum of all values to be maximized. Because M is to be minimized, it will be no larger than the maximum. Thus, we are minimizing the maximum distortion, which is what we want to do. (Note that M, being the maximum, will always be nonnegative, so the explicit requirement that it be so is redundant.)

Special Case: Rotating the Root

It still remains to consider the special case, where α becomes the root. This eliminates all equations involving w, since those deal with paths that no longer exist. This leaves:

$$\begin{aligned}
\text{MINIMIZE:} \quad & M \\
\text{SUBJECT TO:} \quad & P^{*+}_{mn} + x + v + y \leq M, \quad P^{*+}_{mp} + x + v + z \leq M, \\
& P^{*+}_{np} + y + z \leq M, \quad P^{*-}_{mn} - x - v - y \leq M, \\
& P^{*-}_{mp} - x - v - z \leq M, \quad P^{*-}_{np} - y - z \leq M, \\
& P^{eq} \leq M, \quad v, x, y, z, M \geq 0.
\end{aligned}$$

By letting $x = v$, we are able to reduce this to:

$$\begin{aligned}
\text{MINIMIZE:} \quad & M \\
\text{SUBJECT TO:} \quad & P^{*+}_{mn} + 2x + y \leq M, \quad P^{*+}_{mp} + 2x + z \leq M, \\
& P^{*+}_{np} + y + z \leq M, \quad P^{*-}_{mn} - 2x - y \leq M, \\
& P^{*-}_{mp} - 2x - z \leq M, \quad P^{*-}_{np} - y - z \leq M, \\
& P^{eq} \leq M, \quad x, y, z, M \geq 0.
\end{aligned}$$

This is just a linear programming problem with seven constraints and four variables. Again, it is always feasible by letting $x = y = z = 0$ and letting M be arbitrarily large. Each constraint consists of a quantity that is less than M, and the equations collectively state that M is (at least) the maximum of all values to be maximized. Because M is to be minimized, it will be no larger than the maximum. Thus, we are minimizing the maximum distortion, which is what we want to do.

3.2 Global Edge Weight Updates

Note that the local edge weight update formulation extends to allow the optimal choice of edge lengths for the entire tree. In this section, we consider the problem of choosing edge weights for each edge of the tree such that the L_∞ norm will be minimized.

3.2.1 Formulation

In this case, there will be N leaves, $N-1$ internal nodes, $2N-2$ edges, of which two should be chosen equal (those connected to the root). We also have the M variable, leaving $2N-2$ variables in the problem. Each pairwise distance yields two constraints. The quantity P^{eq} is zero since it consists of distances between leaves and themselves. This yields a total of N^2-N constraints. In particular, given a tree structure and an ideal distance metric, the weights of the tree can be chosen optimally (under the L_∞ norm) in polynomial time.

This can be expressed in the form $Ax \geq b$. The variables x_i for $1 \leq i < 2N-1$ correspond to weights for each edge, with x_1 corresponding to the weight of both edges connected to the root, and $x_{2N-2} = M$ and plays the part of the maximum. Let $S = \frac{1}{2}(N^2-N)$ denote the number of distinct paths, and let p_j, $j \leq S$, denote the paths between distinct leaves listed in some order. Observe that b_j is the distance between the leaves at the endpoints of path p_j according to the original metric. The objective to be maximized is just $-x_{2N-2}$.

Finally, we must describe the coefficients in A. The case corresponding to the variable M is $A_{i,(2N-2)} = 1$. The rest of this paragraph assumes $j < 2N-2$. If $i \leq S$ then $A_{ij} = 1$ if weight x_i occurs along the path p_j, and $A_{ij} = 0$ otherwise. The remaining elements are determined by $A_{(i+M),j} = -A_{ij}$.

3.2.2 Efficient Computation

Let nodes r and s lie just beyond b, as shown in Figure 3.2. Then, $L_b = L_r \cup L_s$, $L_r \cap L_s = \emptyset$. Let δ_{ab} denote the distance between a and b. Then, $x = \delta_{ab}$. Note that the starred quantities depend on x and thus depend on the δ_{ab}. However, the unstarred quantities do not:

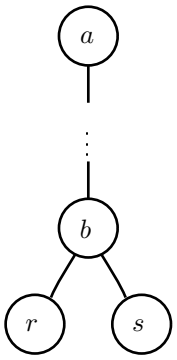


Figure 3.2: Nodes r and s are adjacent to b but not along the path from b to a.

$$
\begin{aligned}
P_{ab}^{*+} &= -\delta_{ab} + P_{ab}^{+} \\
&= -\delta_{ab} + \max_{i \in L_a, j \in L_b} (E_{ij} - D_{ij}) \\
&= -\delta_{ab} + \max_{i \in L_a, j \in L_r \cup L_s} (E_{ij} - D_{ij}) \\
&= -\delta_{ab} + \max(\max_{i \in L_a, j \in L_r} (E_{ij} - D_{ij}), \max_{i \in L_a, j \in L_s} (E_{ij} - D_{ij})) \\
&= \max(-\delta_{ab} + \max_{i \in L_a, j \in L_r} (E_{ij} - D_{ij}), -\delta_{ab} + \max_{i \in L_a, j \in L_s} (E_{ij} - D_{ij})) \\
&= \max(-\delta_{ab} + P_{ar}^{+}, -\delta_{ab} + P_{as}^{+}) \\
&= \max(\delta_{ar} + P_{ar}^{*+}, \delta_{as} + P_{as}^{*+}).
\end{aligned}
$$

A similar approach holds for the other sign:

$$
\begin{aligned}
P^{*-}_{ab} &= \delta_{ab} + P^{-}_{ab} \\
&= \delta_{ab} + \max_{i \in L_a, j \in L_b} (E_{ij} - D_{ij}) \\
&= \delta_{ab} + \max_{i \in L_a, j \in L_r \cup L_s} (E_{ij} - D_{ij}) \\
&= \delta_{ab} + \max\left(\max_{i \in L_a, j \in L_r} (E_{ij} - D_{ij}), \max_{i \in L_a, j \in L_s} (E_{ij} - D_{ij}) \right) \\
&= \max\left(\delta_{ab} + \max_{i \in L_a, j \in L_r} (E_{ij} - D_{ij}) \delta_{ab} + \max_{i \in L_a, j \in L_s} (E_{ij} - D_{ij}) \right) \\
&= \max(\delta_{ab} + P^{-}_{ar}, \delta_{ab} + P^{-}_{as}) \\
&= \max(-\delta_{ar} + P^{*-}_{ar}, -\delta_{as} + P^{*-}_{as}).
\end{aligned}
$$

This allows us to build up a table of these values and obtain each additional entry in constant time. However, any modification to the tree can invalidate these entries. If a and b are leaves, then $x = E_{ij}$ and $E^{*}_{ij} = E_{ij} - x = 0$. Further, being leaves implies $\{a\} = L_a$ and $\{b\} = L_b$. Thus, $P^{*+}_{ab} = (E^{*}_{ab} - D_{ab}) = -D_{ab}$ and $P^{*-}_{ab} = -(E^{*}_{ab} - D_{ab}) = D_{ab}$. This gives us base cases from which the values can be recursively computed.

Unfortunately, these quantities are all invalidated when L_a or L_b changes, when any edge between a and b changes, or when any edge inside of L_a or L_b changes. This leads immediately to an $O(n^2)$ algorithm for recomputing the twelve quantities required to do the update, but it would be nice to have a more efficient updating algorithm. Unfortunately, maxima are not linear in the way sums are, so we cannot split these expressions up further. Assuming an upper bound of $O(n)$ rotations, this yields a cubic algorithm.

Next we will compute:

$$
P^{eq} = \max_{i,j \in L_a} |E_{ij} - D_{ij}|. \tag{3.9}
$$

If a is a leaf, then $P^{eq} = P_{aa} = |E_{aa} - D_{aa}| = |0 - 0| = 0$. Otherwise, assume that r and s are the children of a, with $L_a = L_r \cup L_s$ and $L_r \cap L_s = \emptyset$. We have:

$$\begin{aligned} P^{eq} &= P_{aa} \\ &= \max_{i,j\in L_a} |E_{ij} - D_{ij}| \\ &= \max\left(\max_{i,j\in L_r} |E_{ij} - D_{ij}|, \max_{i,j\in L_s} |E_{ij} - D_{ij}|, \max_{i\in L_r, j\in L_s} |E_{ij} - D_{ij}|, \max_{i\in L_s, j\in L_r} |E_{ij} - D_{ij}|\right) \\ &= \max(P_{rr}, P_{ss}, P_{rs}, P_{sr}) \\ &= \max(P_{rr}, P_{ss}, P_{rs}). \end{aligned}$$

This requires the additional computation:

$$\begin{aligned} P_{rs} &= \max(P_{rs}^{+}, P_{rs}^{-}) \\ &= \max(P_{rs}^{*+} + \delta_{ar} + \delta_{as}, P_{rs}^{*-} - \delta_{ar} - \delta_{as}). \end{aligned}$$

3.3 Distortion Lower Bounds

In this section, we seek inequalities that remain unchanged by local updates and inequalities that limit how much distortion may be caused by the local update.

3.3.1 Invariant Inequalities

Consider again the canonical form of the solution.

$$\begin{aligned} \text{MAXIMIZE:} \quad & -M \\ \text{SUBJECT TO:} \quad & M - w - x \geq P_{mr}^{*+}, \qquad M + w + x \geq P_{mr}^{*-}, \\ & M - x - v - y \geq P_{mn}^{*+}, \qquad M + x + v + y \geq P_{mn}^{*-}, \\ & M - x - v - z \geq P_{mp}^{*+}, \qquad M + x + v + z \geq P_{mp}^{*-}, \\ & M - w - v - y \geq P_{nr}^{*+}, \qquad M + w + v + y \geq P_{nr}^{*-}, \\ & M - y - z \geq P_{np}^{*+}, \qquad M + y + z \geq P_{np}^{*-}, \\ & M - w - v - z \geq P_{pr}^{*+}, \qquad M + w + v + z \geq P_{pr}^{*-}, \\ & M \geq P^{eq}, \qquad v, w, x, y, z, M \geq 0. \end{aligned}$$

One can create new constraints by adding the second six constraints to the first six constraints and divide by two. With these six new constraints and six weaker constraints obtained by applying the observation that $v, w, x, y, z \geq 0$ to the first six constraints, we have 13 constraints that do not depend on v, w, x, y, or z:

$$\begin{aligned}
M \geq P^{*+}_{mr}, \quad & M \geq \tfrac{1}{2}(P^{*+}_{mr} + P^{*-}_{mr}), \\
M \geq P^{*+}_{mn}, \quad & M \geq \tfrac{1}{2}(P^{*+}_{mn} + P^{*-}_{mn}), \\
M \geq P^{*+}_{mp}, \quad & M \geq \tfrac{1}{2}(P^{*+}_{mp} + P^{*-}_{mp}), \\
M \geq P^{*+}_{nr}, \quad & M \geq \tfrac{1}{2}(P^{*+}_{nr} + P^{*-}_{nr}), \\
M \geq P^{*+}_{np}, \quad & M \geq \tfrac{1}{2}(P^{*+}_{np} + P^{*-}_{np}), \\
M \geq P^{*+}_{pr}, \quad & M \geq \tfrac{1}{2}(P^{*+}_{pr} + P^{*-}_{pr}), \\
M \geq P^{eq}.
\end{aligned}$$

It is also very important to note that these constraints do not depend on the combinations of variables v, w, w, y, and z that were used to construct the constraints, and this combination is the only thing separating the optimal choice of weights before and after the rotation. Thus, these 13 lower bounds hold before and after the rotation.

3.3.2 Lower Bound on Distortion Change

It will be helpful to write out the optimization problem prior to the rotation. Here, the variable naming is similar in that w is the weight connected to r, x is the weight next to m, ..., and v is the weight between α and β. Then, the original problem is:

$$\begin{aligned}
\text{MAXIMIZE:} \quad & -M \\
\text{SUBJECT TO:} \quad & M - v - w - x \geq P^{*+}_{mr}, \quad M + w + v + x \geq P^{*-}_{mr}, \\
& M - x - y \geq P^{*+}_{mn}, \quad M + x + y \geq P^{*-}_{mn}, \\
& M - x - v - z \geq P^{*+}_{mp}, \quad M + x + v + z \geq P^{*-}_{mp}, \\
& M - w - v - y \geq P^{*+}_{nr}, \quad M + w + v + y \geq P^{*-}_{nr}, \\
& M - v - y - z \geq P^{*+}_{np}, \quad M + v + y + z \geq P^{*-}_{np}, \\
& M - w - z \geq P^{*+}_{pr}, \quad M + w + z \geq P^{*-}_{pr}, \\
& M \geq P^{eq}, \quad v, w, x, y, z, M \geq 0.
\end{aligned}$$

With the optimal solution before the rotation, we get an assignment of v, w, x, y, and z. Applying this to our new problem, with an extra variable ε to account for changes in M gives:

$$
\begin{aligned}
\text{MAXIMIZE:} \quad & -M \\
\text{SUBJECT TO:} \quad & M+\varepsilon-w-x \geq P^{*+}_{mr}, && M+\varepsilon+w+x \geq P^{*-}_{mr}, \\
& M+\varepsilon-x-v-y \geq P^{*+}_{mn}, && M+\varepsilon+x+v+y \geq P^{*-}_{mn}, \\
& M+\varepsilon-x-v-z \geq P^{*+}_{mp}, && M+\varepsilon+x+v+z \geq P^{*-}_{mp}, \\
& M+\varepsilon-w-v-y \geq P^{*+}_{nr}, && M+\varepsilon+w+v+y \geq P^{*-}_{nr}, \\
& M+\varepsilon-y-z \geq P^{*+}_{np}, && M+\varepsilon+y+z \geq P^{*-}_{np}, \\
& M+\varepsilon-w-v-z \geq P^{*+}_{pr}, && M+\varepsilon+w+v+z \geq P^{*-}_{pr}, \\
& M+\varepsilon \geq P^{eq}, && v,w,x,y,z,M \geq 0.
\end{aligned}
$$

If we agree to assume $\varepsilon \geq 0$, some of these constraints follow from the definition of the remaining variables and the observation that $v \geq 0$. Note that ε can be increased without losing the feasibility of a solution. While forcing it to be nonnegative may omit the optimal solution, it cannot cause the system to be infeasible. Eliminate these equations:

$$
\begin{aligned}
& M+\varepsilon-x-v-y \geq P^{*+}_{mn}, && M+\varepsilon+w+x \geq P^{*-}_{mr}, \\
& M+\varepsilon-w-v-z \geq P^{*+}_{pr}, && M+\varepsilon+y+z \geq P^{*-}_{np}.
\end{aligned}
$$

Letting $\varepsilon = v$ causes these to follow as well, and a feasible solution results. Thus, v is a bound on the distortion, which is also trivially observed by examining the way the rotation affects the tree. Unfortunately, there is no limit to how large v may be.

To see this, consider a tree with all edge weights equal to one, except the weight where v would be. Let this weight be arbitrarily large. To see how bad it is, consider the distance from any point beyond r to any point beyond p. This path has lost the edge of length s, so let the other edges change by Δw and Δz. For the distances between m and n, let the changes be Δx and Δy. This puts bounds of $M \geq s-\Delta w-\Delta z$ and $M \geq s-\Delta x-\Delta y$. The distances from r to m give a distortion bound of $M \geq s+\Delta w+\Delta x$. Similarly, the distance from n to p gives a distortion bound of $M \geq s+\Delta x+\Delta y$. Adding up all four inequalities gives: $4M \geq 4s$, so that the distortion is bounded below by s, which was chosen arbitrarily large.

3.4 Conclusion

The problem of exact computation of local edge weights for a metric tree under the L_∞ norm can be formulated as a linear programming problem with a fixed number of constraints, thus allowing efficient

recomputation of edge weights after a rotation operation. Simultaneously updating all weights in the tree can also be formulated as a linear programming problem. In this case, the linear programming problem has a linear number of variables and constraints and thus admits a polynomial time solution.

4. Tree Operations As Groups

There are a large number of ways to balance a single tree. Understanding how the balancing operations work and interact can provide insight when the way balancing is performed matters. We consider and analyze tree operations using the tools of abstract algebra. Flips can always be performed on a metric tree without affecting the distortion of the tree, as is shown in 5.4. We concentrate on rotations and flips. It is possible to consider tree operations in such a way that they form a group structure. Although the group structure is complicated and does not yield a simple answer to the balancing question, it is worth studying in its own right.

4.1 Infinite Complete Binary Trees

Consider a binary tree t rooted at a node n_0. All nodes have two children. Label the left child of n_i as n_{2i+1} and the right child n_{2i+2}. Letting $L(i) = Li = 2i+1$ and $R(i) = Ri = 2i+2$, we have the notation n_{Ri} and n_{Li} for the children of n_i. Similarly, let $\pi i = \lfloor \frac{i-1}{2} \rfloor$, so that the parent of n_i is $n_{\pi i}$. The first three levels, including the root, are shown in Figure 4.1.

Next, consider operations that we can perform on the binary tree t. We can perform a left rotation on node n_i. Denote this operation r_i, and denote its affect on the tree t as $r_i t$, using concatenation to indicate operator composition and operator action. Similarly, we can perform a left rotation at node n_i, which we denote l_i. We can also perform a child flip (or just "flip") at node n_i, which we denote f_i. Analogously, $l_i t$ and $f_i t$ are trees resulting from the left rotation and flip. In particular, r_0 and l_0 are rotations around the root. The operations f_0, r_0, and l_0 are shown in Figure 4.2. The convention used in these illustrations, and the convention we shall use for the remainder of the figures in this chapter, is that "leaves" of these diagrams are really subtrees, and those subtrees are not changed.

Let t_0 be a fixed, infinite, complete binary tree. We will now define a set T of infinite, complete binary trees.

- $t_0 \in T$.
- $\forall t \in T, \forall i, r_i t \in T$.

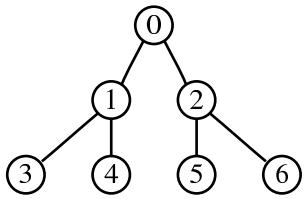
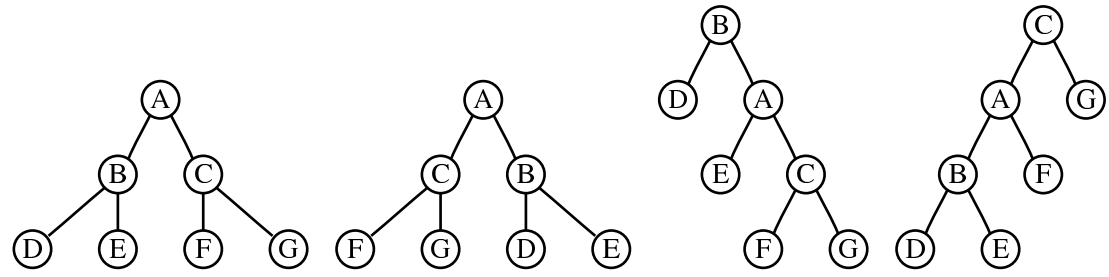


Figure 4.1: First three levels of the infinite tree. The numbers indicate the position with respect to the root.



(a) t_0 (b) f_0t_0 (c) r_0t_0 (d) l_0t_0

Figure 4.2: The three generating operations applied to the root of a tree.

- $\forall t \in T, \forall i, l_i t \in T$.
- $\forall t \in T, \forall i, f_i t \in T$.

Similarly, we define a set S of actions. We denote the composition of two operations by concatenation and refer to this operation accordingly as multiplication. For example, flipping about the root followed by right rotation about the root is $r_0 f_0$. Note that the operations are performed right to left, as is more evident when written as $r_0 f_0 t = r_0(f_0 t)$. Let e be the identity action.

- $\forall i, r_i \in S$
- $\forall i, l_i \in S$
- $\forall i, f_i \in S$
- $\forall a, b \in S, ab \in S$

Additionally, we let T_i be the subtree rooted at n_i and S_i the subgroup of S generated by rotations and flips in subtree T_i. From this definition and two observations about the generators of S, we can make a very important claim about S.

Proposition 3. *The set S is a group under composition.*

Proof: Inverses arise from relations in S. The first such relation follows from the observation that flips undo themselves:

$$f_i^2 = e, \tag{4.1}$$

and the second relation comes from the observation that left and right rotations are inverse operations:

$$l_i r_i = r_i l_i = e. \tag{4.2}$$

These two observations show that the generators have inverses: $f_i^{-1} = f_i$, $r_i^{-1} = l_i$, and $l_i^{-1} = r_i$. Inverses for composite elements are obtained from the formula $(\alpha\beta)^{-1} = \beta^{-1}\alpha^{-1}$. Observe that S has an identity element, since $f_i^2 = e \in S$. Closure under composition is explicitly provided by the definition of S. Associativity follows from the associativity of function composition, and it follows that S is a group. ■

Note that l_i and r_i possess inverses because they have been explicitly included. It was not actually necessary to do this. This statement is made explicit by the following proposition.

Proposition 4. *The operation r_i can be obtained from the composition of flips and left rotations, and the operation l_i can be obtained from the composition of flips and right rotations.*

Proof: The proof is constructive, and follows from the observation of the following two identities:

$$r_i = f_i f_{Li} l_i f_i f_{Li}, \tag{4.3}$$

$$l_i = f_i f_{Ri} r_i f_i f_{Ri}. \tag{4.4}$$

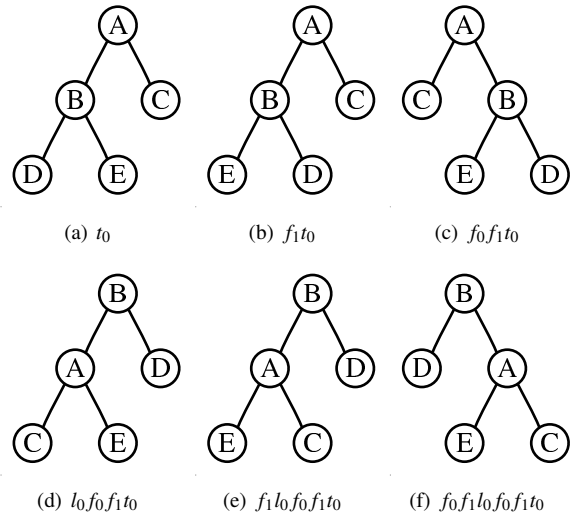


Figure 4.3: The three generating operations applied to the root of a tree.

■

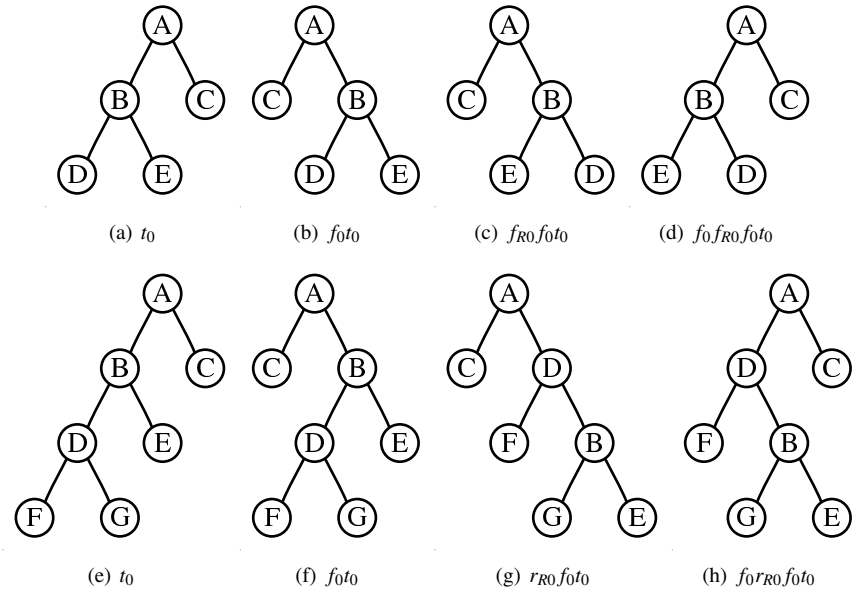
Note that f_{Li} is a flip about the left child of n_i. The first identity is illustrated in Figure 4.3. The second identity is just a mirror image. It also follows algebraically from the first identity and the flip relations below. Multiplying the first equation by l_i and the second by r_i gives us the relations:

$$(l_i f_i f_{Li})^2 = (r_i f_i f_{Ri})^2 = e. \tag{4.5}$$

Comparing t_0 to $l_0f_0f_1t_0$ in Figure 4.3 shows that $l_0f_0f_1$ performs two operations on t_0. It swaps n_0, n_1 and swaps T_2, T_3. Described in this way, it is clear why it is its own inverse.

Proposition 5. *An operation applied to the subtree T_{Li} of a node n_i is related to the same operation applied to the subtree T_{Ri} by the flip f_i.*

Proof: This follows immediately from the following two relations, illustrated in Figure 4.4:



(a) t_0 (b) f_0t_0 (c) $f_{R0}f_0t_0$ (d) $f_0f_{R0}f_0t_0$

(e) t_0 (f) f_0t_0 (g) $r_{R0}f_0t_0$ (h) $f_0r_{R0}f_0t_0$

Figure 4.4: Illustration of $\alpha_{Li} = f_i\alpha_{Ri}f_i$ at the root with $\alpha_i = f_i$ and $\alpha_i = r_i$.

$$\alpha_{Ri} = f_i\alpha_{Li}f_i, \tag{4.6}$$

$$\alpha_{Li} = f_i\alpha_{Ri}f_i. \tag{4.7}$$

■

Theorem 6. *The group S is finitely generated; in particular* $S = \langle r_0, f_0, r_1, f_1\rangle$.

Proof: It is possible to write down relations that reduce an operation on node i in terms of operations on its ancestors. Let α_i denote one of l_i, r_i, f_i. Then, we have the relations:

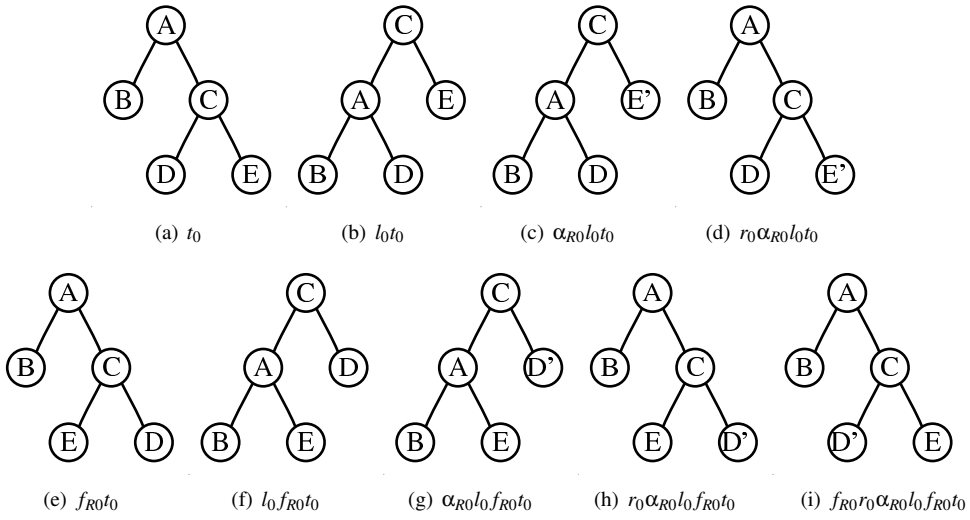


Figure 4.5: Level-reduction illustrations at the root for the fourth and third cases, respectively.

$$\alpha_{LLi} = l_i\alpha_{Li}r_i, \tag{4.8}$$

$$\alpha_{LRi} = f_{Ri}r_i\alpha_{Ri}l_if_{Ri}, \tag{4.9}$$

$$\alpha_{RLi} = f_{Li}l_i\alpha_{Li}r_if_{Li}, \tag{4.10}$$

$$\alpha_{RRi} = r_i\alpha_{Ri}l_i. \tag{4.11}$$

The last two relations are shown in Figure 4.5. In the figure, nodes with a primed label indicate that the subtree rooted at that node has changed in some arbitrary way described by α_i. Note that α_{LLi}, α_{LRi}, α_{LLi}, and α_{LRi} work by rotating an element closer to the root so that the operation α_i can be applied instead as $\alpha_{\pi i}$. Using these reduction formulas, we can write any of l_i, r_i, or f_i in terms of l_0, r_0, f_0, l_1, r_1, f_1, l_2, r_2, and f_2, so that:

$$S = \langle l_0, r_0, f_0, l_1, r_1, f_1, l_2, r_2, f_2 \rangle . \tag{4.12}$$

It immediately follows that S is finitely generated. In fact, we note that the only rotations required for the reductions (not including α, if it is a rotation) are two levels up. Thus, we are able to express flips two levels below the root in terms of flips at the first two levels and rotations at the top level. Combining (4.3) and (4.4) with (4.6) and (4.7), we can conclude:

$$S = \langle r_0, f_0, r_1, f_1, r_2, f_2 \rangle . \tag{4.13}$$

Using (4.6) and (4.7), we are able to express r_2 and f_2 in terms of the other four generators. This reduces the number of generators further, giving us the desired result:

$$S = \langle r_0, f_0, r_1, f_1 \rangle . \tag{4.14}$$

■

Some of the relations above hint at the fact that the elements in S might not commute. This observation is made explicit in the following proposition.

Proposition 7. *The group S is not Abelian.*

Proof: Assume to the contrary that S is Abelian. Expand (4.5):

$$e = (r_i f_i f_{Ri})^2 = r_i^2 f_i^2 f_{Ri}^2 = r_i^2 ee = r_i^2 . \tag{4.15}$$

This implies that rotations are their own inverses, which is false. It follows that S is not Abelian. ■

There are, however, some commutativity relations. In particular, consider that we have two actions α_i, β_j. These actions change subtrees A and B respectively. If $A \cap B = \emptyset$ then $\alpha_i \beta_j = \beta_j \alpha_i$; in particular $\alpha_{Li} \beta_{Ri} = \beta_{Ri} \alpha_{Li}$.

4.2 Subgroup of Invariant Operations

The operations that leave a metric unchanged are f_i, r_0, and l_0. Call this subgroup $F = \langle r_0, f_0, f_i \rangle$. Note that (4.8), (4.9), (4.10), and (4.11) are valid for any α_i that acts only on the subtree rooted at i. Thus, we can use the relations, applied with $i = 0$, to move r_i, l_i, and f_i to the level below the root. This allows us to conclude that f_i are included in a definition of F with only three generators: $F = \langle r_0, f_0, f_1 \rangle$. Unfortunately, F is not a normal subgroup of S, as established by the complete classification of normal subgroups of S in Theorem 52.

4.3 Subgroups of F

An important part of understanding a group's structure is understanding what subgroups it contains. Before getting into the machinery for dealing with this question carefully, we will point out a couple groups that exist as subgroups of this group.

Proposition 8. *The group $\mathbb{Z}$ is a subgroup of F.*

Proof: Consider the subgroup $\langle r_0 \rangle$. The elements are just r_0^n for all integers n. Note that $r_0^{-n} = l_0^n$. This forms a subgroup isomorphic to $\mathbb{Z}$. ■

Proposition 9. *The group D_4 is a subgroup of F.*

Proof: Consider the subgroup $\langle f_0, f_1 \rangle$. Since $f_2 = f_0 f_1 f_0$ and $f_1 f_2 = f_2 f_1$, we have $f_1 f_0 f_1 f_0 = f_0 f_1 f_0 f_1$. Note that $f_0 f_0 = f_1 f_1 = e$, so that the f_0 and f_1 terms must alternate. The relation above tells us that $(f_0 f_1)^4 = e$. Thus, we obtain the eight elements: $\{e, f_0, f_1, f_0 f_1, f_1 f_0, f_0 f_1 f_0, f_1 f_0 f_1, f_0 f_1 f_0 f_1\}$. Letting $f = f_0$ and $r = f_0 f_1$ we see that we have D_4 the dihedral group of order eight. ■

Example: One of the subgroups contained in S is $\langle f_0, r_0 \rangle$. A little inspection reveals the additional relation $(f_0 r_0)^6 = e$, indicating the presence of a group not yet observed in S: $\mathbb{Z}_6$. The main point of this example, however, is to point out the large number of relations that even this highly restricted subset contains. First, however, let $r = r_0$ and $s = f_0 r_0$. This subgroup contains these relations:

- $s^6 = e$,
- $(s^3 r^n s^3 r^{-n})^2 = e, \qquad n \geq 2,$

- $(s^2r)^6 = e$,
- $(s^3rsr^{-2})^4 = e$,
- $(s^3r^{-2}sr)^4 = e$.

The fact that there is a countable sequence of relations is an indication that the group is not finitely presented. This subgroup is very interesting, and we call it G. It will be considered in more detail in Section 4.4. For now, it is just an example. The operation s is also very important. It is a permutation consisting of two cycles, one of order two and one of order three, which explains why it has order six. This also sheds light on where the sequence of relations comes from. The operation s^3 swaps the root with one of its children. The second relation swaps two nodes, moves to a different spot in the tree, swaps two nodes, then moves back to the original spot. These observations will prove useful in building up machinery in later sections.

Theorem 10. *The group S contains an infinite number of copies of itself, which are disjoint except for the identity element.*

Proof: We begin the proof by establishing the weaker result that S actually contains a subgroup isomorphic to itself. More precisely:

$$S' = \langle r_1, f_1, r_3, f_3 \rangle \cong S. \tag{4.16}$$

This follows from the observation that these operations are just the original operations, but applied to a subtree of the root. In particular, their interaction is identical to the interaction of the generators of S. Next, we strengthen this to the observation that every subtree is an isomorphic copy of S:

$$S_i'' = \langle r_i, f_i, r_{Li}, f_{Li} \rangle \cong S \tag{4.17}$$

This is just a generalization of the previous observation. Building on this, we note that we can find a subset of these that are disjoint, except for the identity element:

$$S_i''' = \langle r_{RL^n0}, f_{RL^n0}, r_{LRL^n0}, f_{LRL^n0} \rangle \cong S, \tag{4.18}$$

so that for $i \neq j$:

$$S_i''' \cap S_j''' = \{e\}. \tag{4.19}$$

■

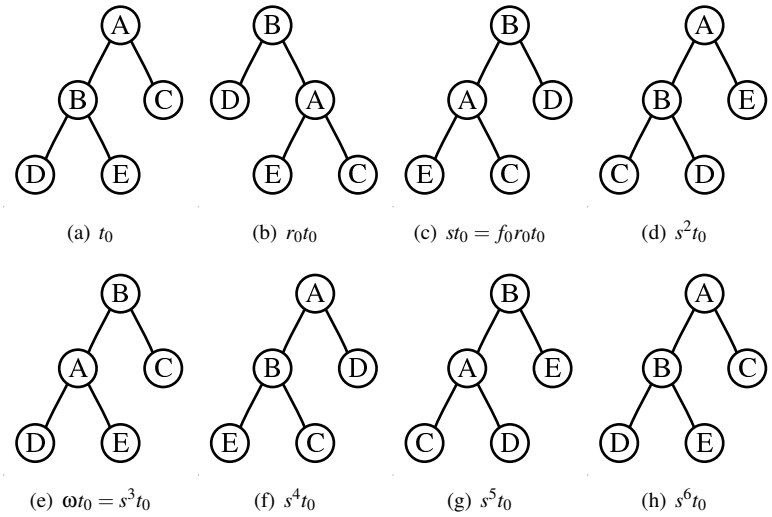
4.4 Understanding $\langle r_0, f_0 \rangle$

We begin with the definition $G = \langle r_0, f_0 \rangle$, as this group will be studied in more depth. This analysis begins by identifying important elements that will become the building blocks for a construction that will ultimately construct copies of all finite groups in G.

This construction begins with the simple operator that is contained in G. It is simply $s = f_0 r_0$. It has two effects on the tree. First, it swaps n_0 and n_1. Second, it cycles the three nodes n_2, n_3, and n_4 and the subtrees rooted at those nodes. (It moves n_2 to the location n_4 occupied originally.) This simple operation is worth a quick note. First, it can be treated as a simple permutation on five elements. Second, it has order six. Thus, $s^6 = e$. A critical observation is that the two cycles can be isolated. In particular, s^4 is just the three cycle and s^3 is just the two cycle. In our case, we care about the latter, and we shall give it a special name: $\omega = s^3$. This operation is the primary building block of constructing the contents of G. It simply swaps n_0 and n_1. Figure 4.6 shows the effects s and a few of its powers have on an initial tree.

4.5 Outer Ribbon

Consider the nodes $\ldots, n_{R^20}, n_{R0}, n_0, n_{L0}, n_{L^20}, \ldots$. These nodes lie along the left and right sides of the tree, which we shall call the outer ribbon. This set of nodes is preserved by r_0 and its inverse. We will refer to these nodes as $\ldots, m_{-2}, m_{-1}, m_0, m_1, m_2, \ldots$. In particular, m_0 is the root and m_1 is its left child. ω swaps these two nodes and leaves all other nodes of the tree untouched. r_0 has the effect of shifting these nodes



(a) t_0 (b) $r_0 t_0$ (c) $st_0 = f_0 r_0 t_0$ (d) $s^2 t_0$

(e) $\omega t_0 = s^3 t_0$ (f) $s^4 t_0$ (g) $s^5 t_0$ (h) $s^6 t_0$

Figure 4.6: Illustration of s and a few of its powers.

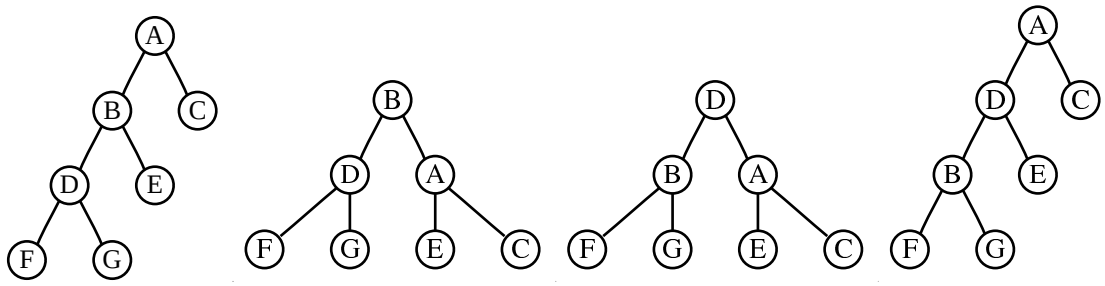
in the following way: $w_i \to w_{i-1}$. We will be dealing with only r_0 and ω for a while, so the effect of the rotations on the rest of the tree is inconsequential; as long as the number of applications of r_0 equals the number of applications of its inverse, the rest of the tree will be returned to its original position. Note that since ω does not affect the rest of the tree, and the rotations do not cause the elements m_i to mix with the rest of the tree, removing them from the sequence will not affect the rest of the tree.

Proposition 11. *The group G contains the operator ω_{ij} that swaps elements m_i and m_j along the outer ribbon.*

Proof: This proof is constructive and uses the operator ω to construct ω_{ij}. First, we begin by constructing ω_{ij} for the case where $j = i+1$:

$$\omega_{i,i+1} = r_0^{-i} \omega r_0^i. \tag{4.20}$$

Note that the number of right rotations is precisely balanced by the number of left rotations. Thus, the rest of the tree is unchanged by these operations. Note also that the above definition is valid for all integers i, including those that are negative. In particular, $\omega_{01} = \omega$, and the comma is omitted when its absence does



(a) t_0 (b) $r_0 t_0$ (c) $\omega r_0 t_0$ (d) $\omega_{12} = r_0^{-1} \omega r_0 t_0$

Figure 4.7: Illustration of (4.20) with $i = 1$.

not cause ambiguity. The equation (4.20) is shown in Figure 4.7 for the case where $i = 1$.

Using this we construct the operator ω_{ij} in its full generality using the recursive definition:

$$\omega_{ij} = \omega_{i,j-1} \omega_{j-1,j} \omega_{i,j-1} \tag{4.21}$$

where $i < j-1$ and (4.20) is the base case $i = j-1$. Note that (4.20) preserves the rest of the tree, so (4.21) will as well. The case $i > j$ is covered by $\omega_{ij} = \omega_{ji}$. Finally, the case $i = j$ follows from $\omega_{ii} = e$. ■

Theorem 12. *The group G contains a copy of S_n, the group of permutations on n elements, for all $n \geq 1$.*

Proof: Consider a permutation on the numbers $\{1, \ldots, n\}$. Decompose this permutation into a product of two-cycles. Replace each $(a\,b)$ with the element ω_{ab}. The result is an operator that acts on $\{m_1, \ldots, m_n\}$ in precisely the same way as the original permutation acted on the integers. Repeat this for all $\sigma \in S_n$ to construct elements σ_m. Let the set of these elements be S'_n. It is clear that this construction creates an isomorphism from S_n to S'_n, so that $S'_n \cong S_n$. ■

Theorem 13. *The group G contains a subgroup isomorphic to every finite group.*

Proof: Because all finite groups are a subgroup of some permutation group S_n, it follows that there is a copy of all finite groups in G. In fact, the same construction can be repeated on the half of the ribbon with negative indices to construct a disjoint second copy of S_n. Because $G \subset F \subset S$, it follows that these

groups contain copies of all finite groups. Repeating the construction starting at different indices, we have a countable number of copies of any finite group. ■

4.6 Constructing Permutations on Trees

Using what we have learned, we are ready to construct permutation groups in S, F, and G. This construction will made independently for S, F, and G. Each uses its own machinery to reach that conclusion. It should be noted that the construction on G implies the other two. Nevertheless, we have chosen this presentation, as it illustrates the amount of freedom we have in each group to work with and the powerful machinery that the groups possess.

We will use a different notation for swaps in the full group S to avoid confusion with the swaps in G, as the two will be indexed differently. We will use γ_{ij} to indicate the operation swapping n_i and n_j but leaving the rest of the nodes untouched. We call two subgroups disjoint when their intersection is the subgroup consisting of only the identity element.

We now begin the process of showing that γ_{ij}'s are all in S, a result stated and proved as Proposition 15. We already know that two of them are, and we shall use them as base cases for constructing the rest. The two of interest that we have seen are:

$$\gamma_{01} = \omega_{01}, \quad \gamma_{02} = \omega_{-1,0}. \tag{4.22}$$

Proposition 14. *The operator $\gamma_{0i} \in S$, for all $i \geq 0$.*

Proof: The case $i = 0$ is just the identity. The cases $i = 1$ and $i = 2$ are from (4.22). The rest we construct recursively using the two relations:

$$\gamma_{0Li} = r_i^{-1} \gamma_{0i} r_i, \tag{4.23}$$

$$\gamma_{0Ri} = r_i \gamma_{0i} r_i^{-1}. \tag{4.24}$$

To see that these are correct, consider the first one (the second one is similar). The elements of the set

$T_i - \{n_i, n_{Li}\}$ are moved by the rotation, untouched by the swap, and restored by the inverse of the rotation. The elements of the set $S - T_i - \{n_0\}$ are left untouched by all three operations and are thus untouched by the swap. This leaves just three nodes to be considered: n_0, n_i, and n_{Li}. n_0 is untouched by the first rotation. It is swapped to n_i, then rotated to n_{Li}. n_{Li} is rotated to n_i by the first rotation, swapped to n_0, and left untouched by the left rotation. n_i is rotated to n_{Ri}, left untouched by the swap, and rotated back to n_i by the left rotation. Thus, the only nodes that are not restored to their original positions are n_0 and n_{Li}, and these nodes are swapped. This completes the construction. ■

We immediately use Proposition 14 to form the basis for the following much stronger theorem.

Proposition 15. *The operator $\gamma_{ij} \in S$, for all $i, j \geq 0$. That is, it is possible to swap arbitrary nodes in S.*

Proof: This theorem extends Proposition 14 using the following construction:

$$\gamma_{ij} = \gamma_{0i}\gamma_{0j}\gamma_{0i}. \tag{4.25}$$

■

At this point, we have the machinery to state and prove one of the more important theorems regarding *S*.

Theorem 16. *Every operator representing a finite permutation of nodes is in S.*

Proof: Choose an arbitrary finite permutation and express it as the product of swaps in the same way a permutation would be written as the product of two-cycles. Because the permutation is finite, only a finite number of swaps are required. By Proposition 15, each of these swap operators is in *S*, so their product, the arbitrary permutation, is in *S*. ■

This theorem is important in understanding *S*. It plays a key role in characterizing *S* in Theorem 24 and in the classification of its normal subgroups in Theorem 52.

Corollary 17. *There are an infinite number of disjoint copies of S_n in S, for any $n \geq 1$.*

Proof: Choose an infinite number of disjoint subtrees, such as T_{RL^k0}, for $k \geq 0$. Choose n nodes in each subtree. Permutations on these n nodes are a finite permutation in *S*, so that each element in S_n exists in

each subtree. Because the subtrees are disjoint, the permutations acting on them must be disjoint, with the exception of the identity element. ■

At this point, we switch our attention from S to its subgroup F.

Proposition 18. *The operator $\gamma_{ij} \in F$, for all $i, j \geq 0$.*

Proof: Choose some element n_i that is not the root. Consider the path from n_i to n_0: $(n_i, n_{p_1}, n_{p_2}, \ldots, n_{p_k}, n_0)$. Let $(q_1, \ldots, q_m)$ be the subsequence of $(p_1, p_2, \ldots, p_k)$ such that n_i lies in the right subtree of n_{p_j}. Let $\kappa = f_{q_m} \cdots f_{q_2} f_{q_1}$, so κn_i lies on the outer ribbon, i.e., working from n_i to the root, we flipped children whenever the node was in the right subtree. Since flips affect ordering of children only for the node being flipped, we end up with a path from the root to the node that follows left children. This location is on the outer ribbon. In fact, if n_i were s levels below the root, then $\kappa n_i = m_s$. Note that $\kappa n_0 = n_0$, since no flips move the root. We claim that:

$$\gamma_{0i} = \kappa^{-1} \omega_{0s} \kappa. \tag{4.26}$$

First, note that all elements that are not n_0 or n_i will be moved by κ to some place that is neither n_0 nor m_s, because n_0 and n_i are moved to those locations. Thus, these elements will be moved somewhere (possibly moved nowhere), left unchanged by ω_{0s}, then moved back by κ^{-1}. Node n_0 is left unchanged by κ, moved to m_s by the swap, then moved to n_i by κ^{-1}. Node n_i is moved to m_s by κ, to n_0 by ω_{0s}, and is left there by κ^{-1}. Thus, this is a swap with the root as claimed. The operators γ_{ij} are constructed as in Proposition 15. ■

As with S, we can make the jump to arbitrary permutations.

Theorem 19. *Every operator representing a finite permutation of nodes is in F.*

Proof: The proof is similar to the proof of Theorem 16. ■

With the finite permutation theorem established for F, we turn our attention to establishing the same theorem for G.

Proposition 20. *The operator $\gamma_{ij} \in G$, for all $i, j \geq 0$.*

Proof: The tricks used to construct swaps in F and S will not work here. Thus, we will have to develop a little more machinery to construct them here. First, though, note that $\gamma_{01} = \omega_{01}$ and $\gamma_{02} = \omega_{-1,0}$ are in G as was the case for F.

Recall the operator $s = f_0 r_0$. It swapped n_0 and n_1 and cycled the subtrees $T_2 \to T_3 \to T_4 \to T_2$. Then, s^2 loses its two-cycle and cycles the subtrees $T_2 \to T_4 \to T_3 \to T_2$. What we want is a way to move any subtree rooted two levels below the root to T_2 without moving the root node. This will have the effect of moving every node in the subtree up one level in the tree, so that we can use a swap with an element closer to the root to construct a swap with the original node. We can construct such an operator ν_i for each of the of the four subtrees rooted two levels below the root:

- $n_i \in T_3$: $\nu_i = s^2$,
- $n_i \in T_4$: $\nu_i = s^4$,
- $n_i \in T_5$: $\nu_i = s^2 f_0$,
- $n_i \in T_6$: $\nu_i = s^4 f_0$.

Now, we can construct γ_{0i} directly. Using $n_k = \nu_i n_i$ as a definition of k, we have:

$$\gamma_{0i} = \nu_i^{-1} \gamma_{0k} \nu_i. \tag{4.27}$$

Since k is strictly closer to the root than i, this constructs the remaining swaps recursively by distance from the root. The correctness argument is similar to the one used with κ and F. All elements other than n_0 and n_i are preserved because ν_i moves them where γ_{0k} will not touch them, and ν_i^{-1} moves them back. The other two nodes move as follows: $n_0 \to n_0 \to n_k \to n_i$ and $n_i \to n_k \to n_0 \to n_0$. The operators γ_{ij} are constructed as in Proposition 15.

As with S and F, we can make the jump to arbitrary permutations.

Theorem 21. *Every operator representing a finite permutation of nodes is in G.*

Proof: The proof is similar to the proof of Theorem 16. ■

The theorems up to this point have been about finding operators that exist in S. The next theorem is about establishing that there are things that can be done to trees for which S possesses no operator. The theorem regards the tree as a linked data structure, where each node has a pointer to its left child and a pointer to its right child.

Proposition 22. *Any operation that requires changing an infinite number of child pointers cannot be in S.*

Proof: Let $\rho(s)$ be the number of "child pointers" that are changed by the action $s \in S$. $\rho(f_i) = 2$, since the only two children of n_i are changed. $\rho(r_0) = \rho(r_0^{-1}) = 2$. $\rho(r_i) = \rho(r_i^{-1}) = 3$ for $i > 0$. Further, we have, for $a, b \in S$, $\rho(ab) \leq \rho(a) + \rho b$. Since actions $s \in S$ consist of a finite number of such operations, we find that a necessary condition for s being in S is that $\rho(s)$ be finite. This leads immediately to a way of constructing operations not in S. For example, the action that swaps n_{4^i} and n_{4^i+1} for all $i \geq 1$. Each swap changes six child pointers, and there is no interaction between swaps due to the two-layer separation between the levels of the swaps. ■

We now begin building the machinery to prove the converse of Proposition 22.

Theorem 23. *There exists an operator in S that swaps disjoint subtrees T_i and T_j.*

Proof: We will do this by showing that we can move subtrees beyond a certain level closer to the root and swap every pair above that level. Let $h(x)$ be the depth of node x, if x is a node, or the depth of the root of x, if x is a subtree. We take $h(n_0) = 0$, that is to say the root has depth zero.

We begin with the subtrees rooted at the great-grandchildren of the root subtrees rooted below them. There eight nodes at this depth: $n_7, \ldots, n_14$. Either the subtrees x, y are at or below different nodes or they are both below a single node. Either way, the approach is to find for each possibility an operation ν such that $\nu^{-1}\alpha\nu$ has the effect of swapping x and y, such that α is an operation that swaps subtrees w, z, and $h(w) + h(z) < h(x) + h(y)$. Thus, we are performing induction on $h(x) + h(y)$. Table 4.1 summarizes suitable choices for ν for each possible combination of grandchildren.

The next possibility is that one subtree is at the third level or beyond, but the other subtree is not. In this case, we are looking for a ν that will bring the subtree furthest from the root closer to the root while preserving the level of the other subtree. The choices for ν for these cases are summarized in Table 4.2.

At this point, all we have to deal with is the pairs of subtrees at the first and second levels below the root. These comprise the base cases and are shown in Table 4.3. These are operations α that swap the subtrees

	7	8	9	10	11	12	13	14
7	r_0	r_0	r_0	r_0	r_1	r_1	r_1	r_1
8	r_0	r_0	r_0	r_0	r_2	$r_1 f_3$	$r_1 f_3$	l_2
9	r_0	r_0	$l_1 f_4$	l_1	r_2	$l_1 f_4$	$l_1 f_4$	l_2
10	r_0	r_0	l_1	l_1	l_1	l_1	l_1	l_1
11	r_1	r_2	r_2	l_1	r_2	r_2	l_0	l_0
12	r_1	$r_1 f_3$	$l_1 f_4$	l_1	r_2	$r_2 f_5$	l_0	l_0
13	r_1	$r_1 f_3$	$l_1 f_4$	l_1	l_0	l_0	l_0	l_0
14	r_1	l_2	l_2	l_1	l_0	l_0	l_0	l_0

Table 4.1: Choosing ν in the case where both nodes are three levels below the root.

	1	2	3	4	5	6
7	-	r_1	-	r_0	r_1	r_1
8	-	$r_1 f_3$	-	r_0	$r_1 f_3$	$r_1 f_3$
9	-	$l_1 f_4$	r_0	-	$l_1 f_4$	$l_1 f_4$
10	-	l_1	r_0	-	l_1	l_1
11	r_2	-	r_2	r_2	-	l_0
12	$r_2 f_5$	-	$r_2 f_5$	$r_2 f_5$	-	l_0
13	$l_2 f_6$	-	$l_2 f_6$	$l_2 f_6$	l_0	-
14	l_2	-	l_2	l_2	l_0	-

Table 4.2: Choosing ν in the case where exactly one node is three levels below the root.

rooted at the specified nodes.

This completes the construction of subtree swaps. ■

Theorem 16 and Theorem 23 provide the basis for the following theorem, which provides a necessary and sufficient condition for an operation on an infinite tree being found in S.

Theorem 24. *An operator exists in S if and only if it requires changing only a finite number of child pointers and preserves the infinite complete binary tree structure.*

Proof: We now have the machinery that we will need to construct an arbitrary finite change α to a tree, i.e., any change to a tree that requires changing only a finite number of child pointers and preserves the infinite complete binary tree structure.

We claim that we can identify a set of subtrees $U = \{T_{a_1}, \ldots, T_{a_k}\}$ that satisfies the following properties.

	1	2	3	4	5	6
1	-	f_0	-	-	$r_0f_1l_0$	$f_0l_0f_2r_0f_0$
2	f_0	-	$f_0r_0f_1l_0f_0$	$l_0f_2r_0$	-	-
3	-	$f_0r_0f_1l_0f_0$	-	f_1	f_1yf_1	$f_1f_2yf_2f_1$
4	-	$l_0f_2r_0$	f_1	-	y	f_2yf_2
5	$r_0f_1l_0$	-	f_1yf_1	y	-	f_2
6	$f_0l_0f_2r_0f_0$	-	$f_1f_2yf_2f_1$	f_2yf_2	f_2	-

Table 4.3: Choosing α when both nodes are children or grandchildren of the root. Here, $y = l_0r_2f_5l_2r_0$. The operation y effectively flips subtrees T_4, t_5.

Let N be the set of nodes that are not contained in any of the subtrees in U.

- U is finite.
- The subtrees in U are disjoint.
- N is finite.
- No child pointer in any subtree in U is changed.

To show that we can always do this, we will construct such a set U. Let M be the set of nodes with a child pointer changed by α, $n = \max_{m \in M} h(m)$, and $U = \{T_a : h(a) = n+1\}$. Observe that M and U are finite, and the subtrees in U are disjoint. Let $N = \{a : h(a) \leq n\}$, so N is also finite. By the way n was chosen, no child pointers in any subtree in U will be changed. It follows that we can always choose U and N with these properties.

Note that the subtrees of U will be preserved by α. Thus, changes of this type consist of moving the subtrees in U and the nodes in N around. From this observation we can construct the change α.

The subtrees in U move to new locations in the tree, but they are not modified. Using the subtree swap operation constructed above, we can move these subtrees around in a finite number of steps. Note that the subtrees are not necessarily permuted, i.e., subtrees may be moved to locations where none of the subtrees were rooted before. This construction is inductive.

Let t be a subtree as it is currently situated due to the operations that have been performed so far, γ. Let β be the effects that have yet to be performed, i.e., $\beta = \alpha\gamma^{-1}$. Then, βt will be the subtree as it would be

situated after applying α to the corresponding subtree in U. At each step of the construction, at least one of these two conditions is true for t:

- $h(t) \geq n$ and $h(\beta t) \geq n$,
- $\beta t = t$.

While there is a t such that $h(\beta t) = n$ and $\beta t \neq t$, then we will swap subtrees t and βt. Now, t is correctly placed. (After updating β to β', we will have $t = \beta' t$.)

While there is a t such that $h(\beta t) > n$ and $h(t) = n$, we must move this subtree down a level. Consider now the locations of the nodes of N before and after β is applied to the tree. $|\{\beta x : x \in N, h(\beta x) < n\}| + |\{\beta x : x \in N, h(\beta x) = n\}| + |\{\beta x : x \in N, h(\beta x) > n\}| = |\{x : x \in N, h(x) < n\}| + |\{x : x \in N, h(x) = n\}| + |\{x : x \in N, h(x) > n\}|$.

Because all $t \in U$, $h(t) < n$ are correctly placed, and the number of nodes $x, h(x) < n$ is constant, it follows that $|\{\beta x : x \in N, h(\beta x) < n\}| = |\{x : x \in N, h(x) < n\}|$.

Next, note that the nodes $x, h(x) = n$ can be split into two groups: those in N and those in a subtree in U. The only difference at this point is that there are $h(t) = n$ where $h(\beta t) > n$. However, $\beta t = t$ whenever $h(\beta t) \leq n$. Thus, the number of nodes at depth n in a subtree is strictly greater before β. Thus, $|\{\beta x : x \in N, h(\beta x) = n\}| > |\{x : x \in N, h(x) = n\}|$. Combining the these equations we have:

$$|\{x : x \in N, h(x) > n\}| > |\{\beta x : x \in N, h(\beta x) > n\}| \geq 0. \tag{4.28}$$

This implies that there is some element $x \in N, h(x) > n$; swap T_x and t. At this point, it is true that one of these is true for each t:

- $h(t) > n$ and $h(\beta t) > n$,
- $\beta t = t$,

so the conditions now hold for $n+1$. Initially, the conditions trivially hold for $n = 1$. Let $H = \max_{t \in U} h(\alpha t)$. When $n > H$, the conditions imply that all subtrees have been correctly placed. It follows that the algorithm will terminate after a finite number of steps and is correct. Note that each step requires a finite number of

swaps, since each swap either correctly places a subtree at a level or moves a subtree out of the level, and neither operation undoes the other.

At this point in the algorithm, all of the subtrees in U have been correctly placed. The positions held by elements in N are precisely the positions in which elements of N must end up. This is because the other positions are taken up by subtrees, which are correctly placed and cannot be modified. Thus, moving elements of N to the right place is simply a matter of performing a permutation on the nodes in N. Since N is finite, we already have an algorithm for doing this. It follows that we have a construction for performing the arbitrarily chosen valid operation α. This completes the proof. ■

5. Generating Generators

The focus of this chapter is understanding the structure and normal subgroups of the group S. It begins with an analysis of the effects of removing generators at the root. It continues by introducing an invariant that leads to the isolation of the first nontrivial normal subgroup and the consequent conclusion that S is not simple. Finally, the concept of a conjugate closure is introduced, and it is used to build the machinery that ultimately reveals all of the normal subgroups of S.

5.1 Generator Removal

The topic for this section is the effect of removing generators from the root of S. Let $S - \kappa = \langle \{f_i, r_i\} - \kappa \} \rangle$. That is, all of the original generators for S are used, and the generators listed in κ are removed. Note that the original set of generators contains an infinite number of flips and rotations. It is not true, for example, that $S - \{f_0\} = \langle r_0, f_1, r_1 \rangle$.

Proposition 25. *Removing f_0 from the original set of generators of S does not change the group generated, so that $S - \{f_0\} = S$.*

Proof: The element $f_0 \in S - \{f_0\}$ because:

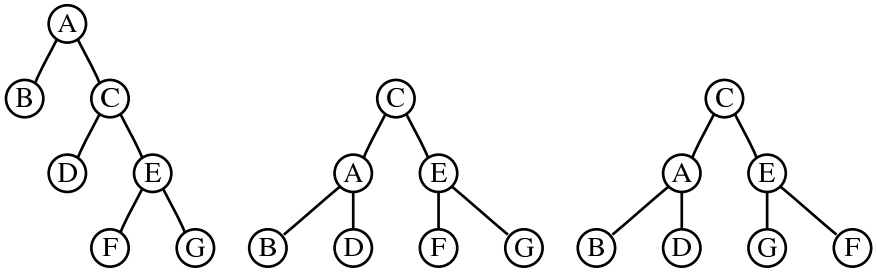
$$f_0 = l_0 l_2 f_2 r_0 r_1 f_1 r_1 l_0 f_2 l_0. \tag{5.1}$$

This identity is illustrated in Figure 5.1. ■

Proposition 26. *The group $S - \{f_0\}$ is finitely generated by $\langle r_0, f_1, r_1, f_2 \rangle$.*

Proof: Requiring the elements from (5.1) to be present along with the rest of the usual generators of S, we have $S = \langle r_0, f_1, r_1, f_2, r_2 \rangle$. We can trim this down by generating f_0 differently:

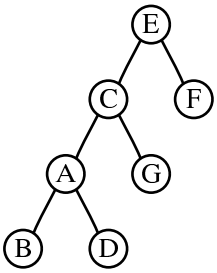
$$f_0 \quad = \quad r_0 f_2 r_0 l_1 f_1 l_1 l_0 f_2 r_0 r_0 l_1. \tag{5.2}$$



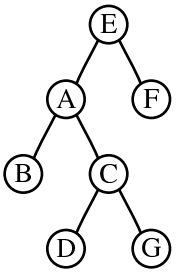
(a) t_0

(b) l_0t_0

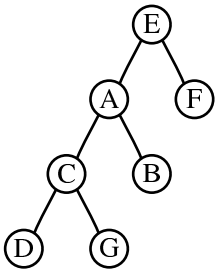
(c) $f_2l_0t_0$



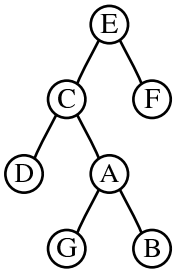
(d) $l_0f_2l_0t_0$



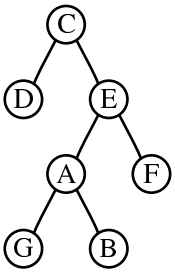
(e) $r_1l_0f_2l_0t_0$



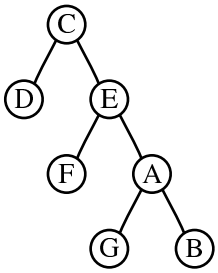
(f) ξ



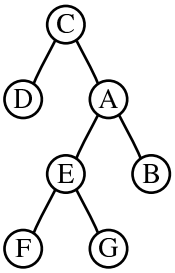
(g) $r_1\xi$



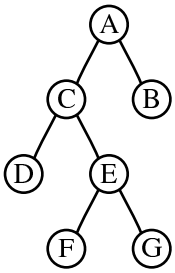
(h) $r_0r_1\xi$



(i) $f_2r_0r_1\xi$

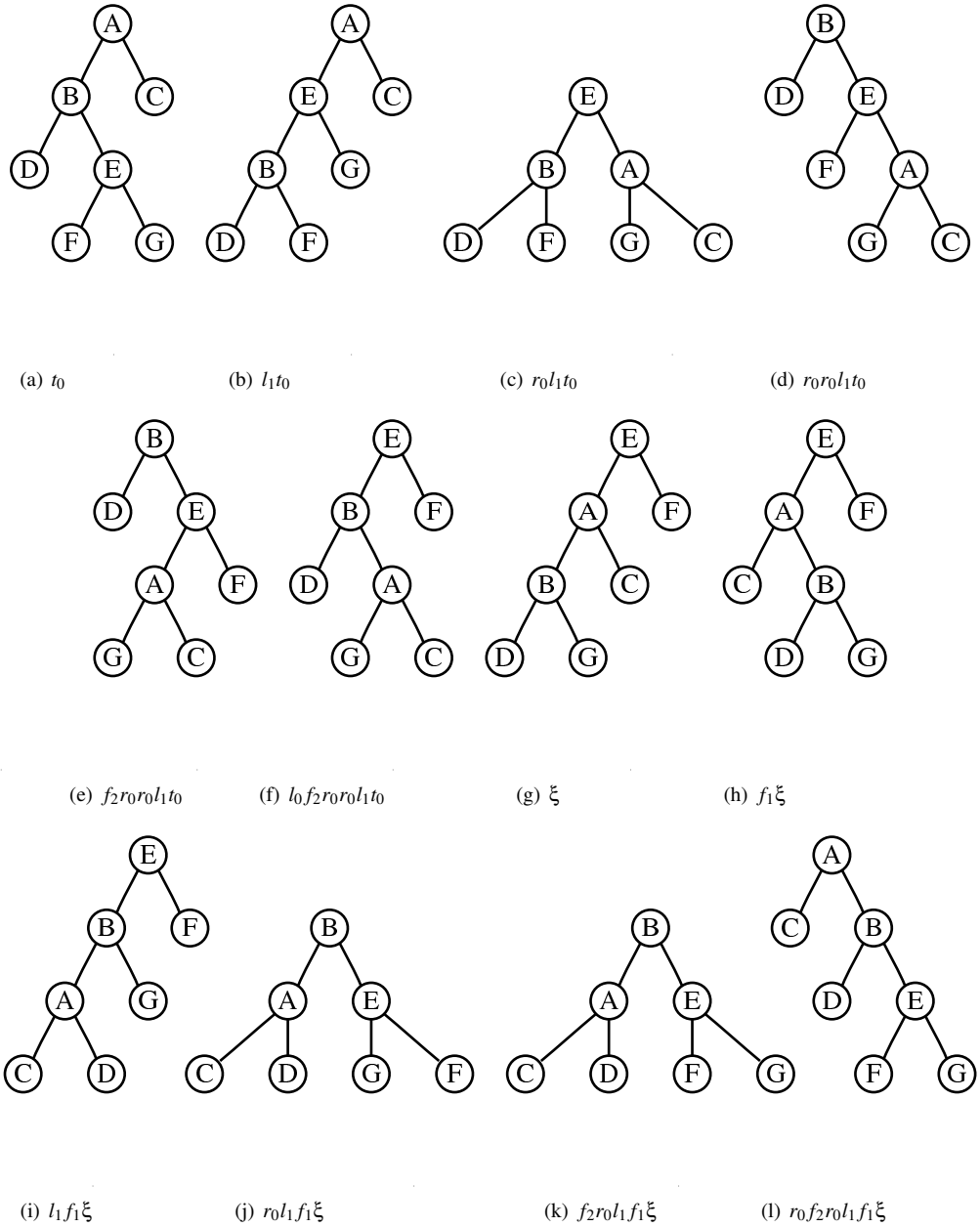


(j) $l_2f_2r_0r_1\xi$



(k) $l_0l_2f_2r_0r_1\xi$

Figure 5.1: Illustration of (5.1) with $\xi = f_1r_1l_0f_2l_0t_0$.



(a) t_0 (b) $l_1 t_0$ (c) $r_0 l_1 t_0$ (d) $r_0 r_0 l_1 t_0$

(e) $f_2 r_0 r_0 l_1 t_0$ (f) $l_0 f_2 r_0 r_0 l_1 t_0$ (g) ξ (h) $f_1 \xi$

(i) $l_1 f_1 \xi$ (j) $r_0 l_1 f_1 \xi$ (k) $f_2 r_0 l_1 f_1 \xi$ (l) $r_0 f_2 r_0 l_1 f_1 \xi$

Figure 5.2: Illustration of (5.2) with $\xi = l_1 l_0 f_2 r_0 r_0 l_1 t_0$.

This identity is illustrated in Figure 5.2. It allows us to immediately remove two generators:

$$S = \langle r_0, f_1, r_1, f_2 \rangle . \tag{5.3}$$

This is the desired result. ■

Consider group $V = S - \{f_0, r_0\}$. Because no operations in this group can affect the root, the element $r_0 \notin S$, so that $V \neq S$. Let S_1 and S_2 be all of the operations in subtrees T_1 and T_2, respectively.

Proposition 27. *The group V is isomorphic to the direct product of S and itself, or $V \cong S \times S$.*

Proof: Note that V can be written in terms of generators in S_1 and generators in S_2. Further, the generators in S_1 commute with the generators in S_2. Using this, we can commute the operators in S_1 to the left of all operators in S_2. Thus, for any $\alpha \in V$ we can find $\beta_1 \in S_1$ and $\beta_2 \in S_2$ such that $\alpha = \beta_1 \beta_2$. Noting that $S \cong S_1 \cong S_2$ lets us write:

$$V = S_1 \times S_2 \cong S \times S. \tag{5.4}$$

■

Next, we take a look at the group $W = S - \{r_0\}$. This group is slightly more interesting than V, which turns out to be a normal subgroup of W.

Proposition 28. *The group V is a normal subgroup of W, and $W \cong (S \times S) \rtimes Z_2$.*

Proof: To prove that V is a normal subgroup, we need to show that if we pick any $x \in W$ and $y \in V$, then $xyx^{-1} \in V$. The following observations make this process a lot easier.

1. The requirements are trivial if $x \in V$; in particular, this case follows from V being a group.
2. Let $x \in W$ and $y, z, xyx^{-1}, xzx^{-1} \in V$. Then $x(yz)x^{-1} \in V$. This is a simple consequence of $x(yz)x^{-1} = (xyx^{-1})(xzx^{-1})$ and the assumptions.
3. Let $w, x \in W$ and assume $xyx^{-1}, wyw^{-1} \in V$ for any $y \in V$. Then $(wx)y(wx)^{-1} \in V$. To see this, note that $(wx)y(wx)^{-1} = w(xyx^{-1})w^{-1}$, where $xyx^{-1} \in V$.

Because of these three observations, it is sufficient to consider x as any generator of W that is not in V and y as any generator of V. The second observation extends this to all y. The first and third observations extend this to all x and y and complete the proof.

By the definitions of W and V, we have $x = f_0$. Every generator of V is of the form α_{Li} or α_{Ri} for some generating operation α_i. Because $\alpha_{Li} = f_0\alpha_{Ri}f_0$ and $\alpha_{Ri} = f_0\alpha_{Li}f_0$, we have shown everything that needs to be shown.

We have a normal subgroup, so we can take quotients. Let $Q = W/V$ and let $\phi(x) = xV$, so that $\phi : W \to Q$. We claim that $Q = \{V, f_0V\}$. We do this by showing that for any $xV \in Q$, where $x \neq \{e, f_0\}$ and x is written as a concatenation of generators, there is a shorter concatenation of generators y such that $xV = yV$. We proceed in cases. In each case, let w be an arbitrary element in W and α be a generator of V. Let β be the generator such that $\alpha = f_0\beta f_0$.

- If $x = w\alpha$ then choose $y = w$.
- If $x = w\alpha f_0 = wf_0\beta$ then choose $y = wf_0$.
- If $x = wf_0f_0$ then choose $y = w$.

If x contains at least two generators, look at the last generator. If it is not f_0, the first case applies. Otherwise, look at the generator before it. If it is not f_0, the second case applies. Otherwise, the last two generators are f_0, and $f_0f_0 = e$, so that the two generators cancel each other and yield a shorter description. This leaves x being a generator or the identity. If it is a generator other than f_0, the first case applies. This leaves e and f_0, which we claim to be Q. We need only show that they are distinct. This is easy, since all generators of V lack the ability to moving an element across the root, which f_0 can do.

In the same vein as the simplification of Q, we can take an element in W and use $\alpha f_0 = f_0\beta$ to shift all of the occurrences of f_0 to the rightmost positions in the string, and the f_0's will cancel in pairs. Thus, we can write every element in W in the form vq, where $q \in \{e, f_0\}$ and $v \in V$. Let $(v, q) = vq$ and $(w, p) = wp$. We want to define a multiplication of these ordered pairs so that the product is just $vqwp$.

$$\begin{aligned}(v,q)(w,p) &= (vq)(wp)\\ &= vqwp\\ &= vqw(q^{-1}q)p\\ &= v(qwq^{-1})qp\\ &= (v(qwq^{-1}),qp).\end{aligned}$$

Thus, W is just the semidirect product of V and Q, i.e.,

$$W = V \rtimes Q, \tag{5.5}$$

where the homomorphism $\phi(e)(x) = x$ and $\phi(f_0)(x) = f_0 x f_0$. What is more, V itself can be broken down. In particular, let S_1 and S_2 be all of the operations in subtrees T_1 and T_2, respectively. Then, $V = S_1 \times S_2$. However, $S \cong S_1 \cong S_2$, so that:

$$W \cong (S \times S) \rtimes Z_2. \tag{5.6}$$

■

5.2 Flip Parity

There are many representations of the identity in terms of generators $\{r_i, l_i, f_i\}$, e.g., $r_0 l_0 = f_0^2 = (r_0 f_0)^6 = e$. The number of flips in each representation of the identity has always been even. This observation is not a coincidence. In fact, it is possible to identify whether an even or odd number of flips were used to describe an operator $\alpha \in S$ given only the trees t_0 and αt_0 and without actually finding such an α.

One interesting place to look is the relative order of invariant subtrees, as rotations do not affect this. However, this ordering does not contain enough information to distinguish the parity of the number of flips,

e.g., $(f_0 r_0)^3$ has the same effect on the position or ordering of any set of invariant subtrees as the identity. If we keep track of more information, we can remedy this situation. This forms the basis of the main theorem of this section. However, some simpler results are needed to build up to that theorem.

Choose a set of invariant subtrees such that the number of nodes not in such a subtree is finite. Replace each subtree with a leaf. Each node of the tree is either an internal node with two children or a leaf. Give each node and leaf a unique symbol.

Proposition 29. *Each subtree contains an odd number of symbols.*

Proof: We show this inductively on the size of the subtree. If the subtree is rooted at a leaf, the number of symbols is one. Otherwise, the number of symbols in a subtree T_i is the sum of the number of symbols in T_{Li} and T_{Ri} plus the symbol for n_i. Because n_i is not a leaf, both T_{Li} and T_{Ri} exist, and the number of symbols contained in them is odd by the inductive hypothesis. Thus, the number of symbols in T_i is the sum of three odd numbers and is odd itself. ■

Let $\sigma(t)$ be the list of symbols of the tree t as obtained by an inorder traversal of the tree. Then, σ_{t_0} is the list obtained from the original tree. Let α be an operator expressed as a string of generators from $\{r_i, l_i, f_i\}$.

Theorem 30. *The permutation $\sigma(\alpha t_0)$ is an even permutation if and only if α contains an even number of flips.*

Proof: Assume that the invariant subtrees were chosen so that they are never changed at any point in the application of α. Look at the effect of a single generating operation on the parity of the permutation. Because rotations do not affect the relative ordering of elements in a tree, $\sigma(t) = \sigma(r_i t) = \sigma(l_i t)$. All that remains is $\sigma(f_i t)$. The permutation $\sigma(t)$ has the form:

$$[a_1 \ldots a_u\, b_1 \ldots b_v\, c\, d_1 \ldots d_w\, e_1 \ldots e_x], \tag{5.7}$$

where $a_1 \ldots a_u$ are symbols occurring before symbols from the subtree T_i, $b_1 \ldots b_v$ are the symbols from T_{Li}, c is the symbol for n_i, $d_1 \ldots d_w$ are the symbols from T_{Ri}, and $e_1 \ldots e_x$ are the symbols occurring after the symbols from T_i. Note that v and w are odd. After applying f_i we have $\sigma(f_i t)$, which has the form:

$$[a_1 \ldots a_u\, d_1 \ldots d_w\, c\, b_1 \ldots b_v\, e_1 \ldots e_x]. \tag{5.8}$$

Move b_1 in front of d_1 by repeatedly swapping b_1 with the symbol before it. There are $w+1$ swaps required to do this, and the result is:

$$[a_1 \ldots a_u\, b_1\, d_1 \ldots d_w\, c\, b_2 \ldots b_v\, e_1 \ldots e_x]. \tag{5.9}$$

Repeat with $b_2, \ldots, b_v$:

$$[a_1 \ldots a_u\, b_1 \ldots b_v\, d_1 \ldots d_w\, c\, e_1 \ldots e_x], \tag{5.10}$$

Do the same thing with c:

$$[a_1 \ldots a_u\, b_1 \ldots b_v\, c\, d_1 \ldots d_w\, e_1 \ldots e_x], \tag{5.11}$$

This is just $\sigma(t)$. The process required shifting v elements b_k, requiring $(w+1)$ swaps each. Shifting c requires w swaps. Thus, $v(w+1)+w$ swaps were required. Because v and w are both odd, the number of swaps is odd, and the parity the permutation is inverted. Because each flip inverts the parity and each non-flip preserves the parity, the parity is zero if and only if the number of flips is even. ■

Note that this was proven with the restriction that the invariant subtrees remain invariant through the action of α. This restriction is not necessary, as is shown by the following proposition.

Proposition 31. *The parity of the number of flips in the description of* α *can be determined from any set of invariant subtrees.*

Proof: The restriction on the invariant subtrees can be lifted so that the choice need not depend on the representation of the operator α. Compare $\sigma(t)$ and $\sigma(t')$, where t' is formed by replacing an invariant

subtree T_i with two invariant subtrees T_{Li} and T_{Ri}. Let t contain $n+1$ symbols and t' contain $n+3$ symbols, and let all of the symbols in t and t' be the same, with a new symbols for n_{Li} and n_{Ri}. The permutation $\sigma(t)$ looks like:

$$[a_1 \ldots a_u\, c\, e_1 \ldots e_{n-u}], \tag{5.12}$$

and the permutation $\sigma(t')$ looks like:

$$[a_1 \ldots a_u\, b\, c\, d\, e_1 \ldots e_{n-u}]. \tag{5.13}$$

The permutation $\sigma(\alpha t)$ looks like:

$$[g_1 \ldots g_v\, c\, h_1 \ldots h_{n-v}], \tag{5.14}$$

and the permutation $\sigma(\alpha t')$ looks like:

$$[g_1 \ldots g_v\, b\, c\, d\, h_1 \ldots h_{n-v}]. \tag{5.15}$$

Let $\tau\sigma(t) = \sigma(\alpha t)$ and $\tau'\sigma(t') = \sigma(\alpha t')$. We claim that τ and τ' have the same parity. Let $p = (a_1 \ldots a_u\, b)(d\, e_x \ldots e_1)$ be a permutation expressed as the product of two disjoint cycles. Then:

$$[b\, a_1 \ldots a_u\, c\, e_1 \ldots e_{n-u}\, d] = p\sigma(t'). \tag{5.16}$$

Similarly, let $q = (g_1 \ldots g_v\, b)(d\, h_{n-v} \ldots h_1)$. Then,

$$[b\, g_1 \ldots g_v\, c\, h_1 \ldots h_{n-v}\, d] = q\sigma(\alpha t'). \tag{5.17}$$

Let $\kappa p\sigma(t') = q\sigma(\alpha t')$. If τ is expressed in cycle notation as $\tau = C_1 \ldots C_n$, then κ is expressed in cycle notation as $\kappa = C_1 \ldots C_n(b)(d)$. Let τ be expressed as the product of j two cycles. Then, κ is also expressed as the product of j two cycles. Because a cycle on m symbols can be expressed as the product of $m-1$ two cycles, p and q can both be written as the product of n two cycles. Because $\tau' = q^{-1}\kappa p$, τ' can be expressed as the product of $2n+j$ two cycles, and $(2n+j)-j$ is even, τ and τ' have the same parity, as desired.

The ability to replace an invariant subtree with subtrees further from the root is important, as it allows us to remove the assumption that the subtrees remain invariant through the application of α. Instead, we can replace subtrees that would be disturbed during the application of α with subtrees that will not be affected. After α has been applied, the process can be reversed. The fact that the original subtrees were invariant implies that the process can always be reversed. Further, the change in invariant subtrees will not affect the parity of the permutation produced by α. It follows that the parity can be computed using any set of invariant subtrees, without knowing α's representation. ■

It is worth mentioning a very simple corollary of Theorem 30.

Corollary 32. *If $\alpha = \beta$ are two descriptions of the same operator of the same operator, then both α and β contain an even number of flips or an odd number of flips.*

Proof: This follows from Theorem 30. ■

5.2.1 Group S Is Not Simple

At this point, we are in a position to close a problem that has shadowed progress in describing S. Can S be decomposed into smaller subgroups? We showed earlier that in the absence of r_0, the answer is "yes." Now, we show that S has a normal subgroup and use it to write S as a semidirect product of smaller subgroups.

Theorem 33. *Let the group of operations in S that can be expressed with an even number of flips be U. Then, U is a normal subgroup of S.*

Proof: We claim that U and f_0U are disjoint. This follows from a previous corollary, because $\alpha \in U$ has an even number of flips and $\beta \in f_0U$ has an odd number of flips. We also claim that these two sets contain all elements of S. If $\alpha \in S$ contains an even number of flips, then $\alpha \in U$ by definition. Otherwise, $\alpha = f_0 f_0 \alpha = f_0(f_0\alpha)$, and $f_0\alpha$ has an even number of flips. Then, $\alpha \in f_0U$.

Next, we show that U is a normal subgroup of S. Let $\alpha \in U$ and $\beta \in S$, where α has $2n$ flips and β has m flips. The operator $\beta\alpha\beta^{-1}$ can be expressed with $2n+2m$ flips, and $\beta\alpha\beta^{-1} \in U$. ■

From this follows an important conclusion.

Corollary 34. *The group S is not simple.*

Proof: The group S contains the proper normal subgroup U. ■

Proposition 35. *The group S can be written as a semidirect product. More specifically,*

$$S = U \rtimes \langle f_0 \rangle .$$

Proof: We express $\alpha \in S$ as the product $\beta\gamma$, where $\beta \in U$ and $\gamma \in \{e, f_0\}$. Identify this element α with the ordered pair (β, γ). Similarly, express α' with the ordered pair (β', γ'). Consider the product $\alpha\alpha'$:

$$\begin{aligned}
(\beta,\gamma)(\beta',\gamma') &= (\beta\gamma)(\beta'\gamma') \\
&= \beta\gamma\beta'(\gamma^{-1}\gamma)\gamma' \\
&= \beta(\gamma\beta'\gamma^{-1})\gamma\gamma' \\
&= (\beta\gamma\beta'\gamma^{-1}, \gamma\gamma'),
\end{aligned}$$

because $\gamma\gamma' \in \{e, f_0\}$ and $\beta\gamma\beta'\gamma^{-1} \in U$. This leads to:

$$S = U \rtimes \langle f_0 \rangle . \tag{5.18}$$

■

It is worth a note that f_0 is not special; any flip would have sufficed. In particular, it is true for any fixed n that:

$$S = U \rtimes \langle f_n \rangle . \tag{5.19}$$

5.2.2 Conjugate Closures and Normal Subgroups in S

To get a better understanding of what the normal subgroups of S are, it helps to look at conjugate closures. A conjugate closure $C \subset S$ is a subgroup such that $x \in C, s \in S \implies xsx^{-1} \in C$. Conjugate closures correspond exactly to normal subgroups. Closures under conjugation ensures that it is a normal subgroup. Conversely, the definition of normal is that it is closed under conjugation. Let $\langle C \rangle^S$ denote the conjugate closure of $C \subset S$ with respect to the group S.

Proposition 36. *The conjugate closures generated by r_0 and r_1 are the same, so that $\langle r_0 \rangle^S = \langle r_1 \rangle^S$.*

Proof: We shall prove by demonstrating containment in each direction:

$$\begin{aligned}
r_0 \in \langle r_0 \rangle^S \quad &\implies \quad r_2^{-1} r_0 r_2 \in \langle r_0 \rangle^S \\
&\implies \quad r_2^{-1} r_0 r_2 r_0^{-1} = r_1 r_0^{-1} r_2 \in \langle r_0 \rangle^S \\
&\implies \quad (r_0^{-1} r_1 r_0^{-2} r_2) r_1 r_0^{-1} r_2 (r_0^{-1} r_1 r_0^{-2} r_2)^{-1} = r_2^{-1} \in \langle r_0 \rangle^S \\
&\implies \quad r_2 \in \langle r_0 \rangle^S \\
&\implies \quad f_0 r_2 f_0 = r_1 \in \langle r_0 \rangle^S \\
&\implies \quad \langle r_1 \rangle^S \subseteq \langle r_0 \rangle^S .
\end{aligned}$$

The other direction is similar:

$$\begin{aligned} r_1 \in \langle r_1 \rangle^S &\implies (f_0 r_0)^{-1} r_1 (f_0 r_0) \in \langle r_1 \rangle^S \\ &\implies (f_0 r_0)^{-1} r_1 (f_0 r_0) r_1 = r_0 \in \langle r_1 \rangle^S \\ &\implies \langle r_0 \rangle^S \subseteq \langle r_1 \rangle^S . \end{aligned}$$

■

From this we are able to make a simple but important generalization.

Proposition 37. *The conjugate closures generated by r_0 and r_i are the same, for $i \geq 0$, so that $\langle r_0 \rangle^S = \langle r_i \rangle^S$.*

Proof: By choosing an element ν to move a subtree T_i to T_1, for $i > 0$, we have $r_i = \nu^{-1} r_1 \nu \in \langle r_1 \rangle^S$. Combined with r_0 itself, we have $r_i \in \langle r_1 \rangle^S$, for all $i \geq 0$. Because $r_1 = \nu r_i \nu^{-1} \in \langle r_i \rangle^S$, it follows that $\langle r_i \rangle^S = \langle r_1 \rangle^S = \langle r_0 \rangle^S$. ■

Proposition 38. *The operator $f_i f_j \in \langle r_0 \rangle^S$, for $i, j \geq 0$.*

Proof: We begin by showing that $f_1 f_2 \in \langle r_0 \rangle^S$:

$$\begin{aligned} e \in \langle r_0 \rangle^S &\implies e = (f_1 f_0 r_0)^2 \in \langle r_0 \rangle^S \\ &\implies f_1 f_0 r_0 f_1 f_0 r_0 \in \langle r_0 \rangle^S \\ &\implies f_1 f_0 r_0 f_1 f_0 \in \langle r_0 \rangle^S \\ (f_1 f_0)^{-1} r_0^{-1} (f_1 f_0) \in \langle r_0 \rangle^S &\implies f_0 f_1 r_0^{-1} f_1 f_0 \in \langle r_0 \rangle^S \\ &\implies (f_1 f_0 r_0 f_1 f_0)(f_0 f_1 r_0^{-1} f_1 f_0) \in \langle r_0 \rangle^S \\ &\implies f_1 f_0 f_1 f_0 = f_1 f_2 \in \langle r_0 \rangle^S . \end{aligned}$$

Using this result, a similar argument shows that $f_3 f_6 \in \langle r_0 \rangle^S$:

$$
\begin{aligned}
e \in \langle r_0 \rangle^S &\implies e = (f_3 f_1 f_0 r_0^2)^2 \in \langle r_0 \rangle^S \\
&\implies f_3 f_1 f_0 r_0^2 f_3 f_1 f_0 r_0^2 \in \langle r_0 \rangle^S \\
&\implies e = (f_3 f_1 f_0 r_0^2)^2 \in \langle r_0 \rangle^S && (5.20) \\
&\implies f_3 f_1 f_0 r_0^2 f_3 f_1 f_0 r_0^2 \in \langle r_0 \rangle^S && (5.21) \\
&\implies f_3 f_1 f_0 r_0^2 f_3 f_1 f_0 \in \langle r_0 \rangle^S \\
r_0^{-2} \in \langle r_0 \rangle^S &\implies (f_3 f_1 f_0)^{-1} r_0^{-2} (f_3 f_1 f_0) \in \langle r_0 \rangle^S \\
&\implies (f_3 f_1 f_0 r_0^2 f_3 f_1 f_0)(f_3 f_1 f_0)^{-1} r_0^{-2} (f_3 f_1 f_0) \in \langle r_0 \rangle^S \\
&\implies f_3 f_1 f_0 f_3 f_1 f_0 = f_3 f_6 f_1 f_2 \in \langle r_0 \rangle^S \\
&\implies (f_3 f_6 f_1 f_2)(f_1 f_2) = f_3 f_6 \in \langle r_0 \rangle^S . && (5.22)
\end{aligned}
$$

Using rotations and flips, it is possible to move any subtree $T_i \subset T_1$, $i \neq 1$ to T_3 using an operation $\nu \in S_1$. The construction is similar to other constructions in this thesis and the details are omitted. This leads to the conclusion that:

$$f_i f_6 \in \langle r_0 \rangle^S , \tag{5.23}$$

where $i \in T_1 - \{n_1\}$. The same is possible with f_6, leading to:

$$f_3 f_j \in \langle r_0 \rangle^S , \tag{5.24}$$

where $j \in T_2 - \{n_2\}$. Combining (5.22), (5.23), and (5.24), we get $f_i f_j \in \langle r_0 \rangle^S$, where $i, j \geq 3$. Next, we show that $f_1 f_0 \in \langle r_0 \rangle^S$:

$$
\begin{aligned}
r_1 r_0^{-1}, f_2 f_1 \in \langle r_0 \rangle^S &\implies f_2 f_1 r_1 r_0^{-1} \in \langle r_0 \rangle^S \\
&\implies (r_1 r_0^{-1} f_2 f_0 f_2) f_2 f_1 r_1 r_0^{-1} (r_1 r_0^{-1} f_2 f_0 f_2)^{-1} = f_2 r_1 r_0^{-1} f_2 r_0 \in \langle r_0 \rangle^S \\
&\implies (f_1 f_2)(f_2 r_1 r_0^{-1} f_2 r_0) \in \langle r_0 \rangle^S \\
&\implies f_1 r_1 r_0^{-1} f_2 r_0 \in \langle r_0 \rangle^S \\
&\implies (r_0^{-1} f_0 r_1 r_0^{-1} f_2 r_0^{-1} f_0 f_1) f_1 r_1 r_0^{-1} f_2 r_0 (r_0^{-1} f_0 r_1 r_0^{-1} f_2 r_0^{-1} f_0 f_1)^{-1} = f_1 f_0 \in \langle r_0 \rangle^S .
\end{aligned}
$$

The last important element we need is $f_3 f_0 \in \langle r_0 \rangle^S$:

$$
\begin{aligned}
f_0 r_0^{-1} f_0 \in \langle r_0 \rangle^S &\implies (f_1 f_0)(f_0 r_0^{-1} f_0) r_0^{-1} = f_1 r_0^{-1} f_0 r_0^{-1} \in \langle r_0 \rangle^S \\
&\implies f_0 (f_1 r_0^{-1} f_0 r_0^{-1}) f_0 = f_3 f_0 \in \langle r_0 \rangle^S .
\end{aligned}
$$

Combining these we have $f_i f_j \in \langle r_0 \rangle^S$, where $i, j \geq 0$. ■

This is all of the machinery that we need to completely describe $\langle r_0 \rangle^S$.

Theorem 39. *The element r_0 generates the group U, and $\langle r_0 \rangle^S = U$.*

Proof: We prove this by induction on the size of the description of an $\alpha \in U$. In particular, assume that if β is shorter than α and has an even number of flips, then $\beta \in U$. If $\alpha = r_i \beta$, then β has the same number of flips and is shorter by one generator, so that $\beta \in U$ and $\alpha = r_i \beta \in U$. The same argument applies for the cases $\alpha = r_i^{-1} \beta$, $\alpha = \beta r_i$, and $\alpha = \beta r_i^{-1}$.

Assume α has the form $\alpha = f_i \beta f_j$. Then, β has an even number of flips and is shorter than α by two generators so that $\beta \in U$, $f_j \beta f_j \in U$, and $(f_i f_j)(f_j \beta f_j) = f_i \beta f_j = \alpha \in U$. The base case and only unhandled case is the identity element, which is trivially in U. ■

The next theorem requires three more simple propositions. The theorem will make the equivalent statement about $\langle f_0 \rangle^S$.

Proposition 40. *For any $n \geq 1$, $\langle r_i^n \rangle^S = \langle r_0 \rangle^S$.*

Proof: The direction $\langle r_i^n \rangle^S \subseteq \langle r_i \rangle^S$ is trivial. For the other direction:

$$
\begin{aligned}
r_{Li} r_i^n r_{Li}^{-1} \in \langle r_i^n \rangle^S \quad &\Longrightarrow \quad r_i^{-n} r_{Li} r_i^n r_{Li}^{-1} \in \langle r_i^n \rangle^S \\
&\Longrightarrow \quad (r_i f_{Li} r_{Li}^n) r_i^{-n} r_{Li} r_i^n r_{Li}^{-1} (r_i f_{Li} r_{Li}^n)^{-1} = r_{Li}^{-1} \in \langle r_i^n \rangle^S \\
&\Longrightarrow \quad \langle r_{Li} \rangle^S \subseteq \langle r_i^n \rangle^S \\
&\Longrightarrow \quad r_i \in \langle r_i^n \rangle^S .
\end{aligned}
$$

The proof is completed with Proposition 37. ■

We now shift our attention to the subgroup $\langle f_0 \rangle^S$ to make state the other proposition.

Proposition 41. *For any $n \geq 0$, $r_i \in \langle f_0 \rangle^S$.*

Proof: This is because:

$$
\begin{aligned}
f_1 f_0 f_1 = f_2 f_1 f_0 \in \langle f_0 \rangle^S \quad &\Longrightarrow \quad (f_2 f_1 f_0) f_0 = f_2 f_1 \in \langle f_0 \rangle^S \\
&\Longrightarrow \quad r_0 f_2 f_1 r_0^{-1} \in \langle f_0 \rangle^S \\
&\Longrightarrow \quad f_0 r_0 f_2 f_1 r_0^{-1} \in \langle f_0 \rangle^S \\
&\Longrightarrow \quad f_0 (f_1 f_0)(f_0 r_0 f_2 f_1 r_0^{-1})(f_1 f_0)^{-1} = f_2 f_0 r_0 f_2 f_0 r_0 \in \langle f_0 \rangle^S \\
&\Longrightarrow \quad (f_1 r_0 f_1)(f_2 f_0 r_0 f_2 f_0 r_0)(f_1 r_0 f_1)^{-1} = f_0 r_0^{-1} f_0 r_0^{-1} \in \langle f_0 \rangle^S \\
&\Longrightarrow \quad r_0^{-1} f_0 r_0^{-1} \in \langle f_0 \rangle^S \\
&\Longrightarrow \quad r_0 (r_0^{-1} f_0 r_0^{-1}) r_0^{-1} = f_0 r_0^{-2} \in \langle f_0 \rangle^S \\
&\Longrightarrow \quad r_0^{-2} \in \langle f_0 \rangle^S \\
&\Longrightarrow \quad r_i \in \langle f_0 \rangle^S .
\end{aligned}
$$

■

The third proposition is now easily within reach. While the proposition actually holds for all flips, the much weaker proposition puts the full characterization of $\langle f_0 \rangle^S$ easily within reach, and the stronger form of the proposition follows immediate from that.

Proposition 42. *The operator* $f_1 \in \langle f_0 \rangle^S$.

Proof: We can construct this operator:

$$\begin{aligned} r_0 f_0 r_0 r_1 \in \langle f_0 \rangle^S \quad &\Longrightarrow \quad (f_0 r_0 f_1) r_0 f_0 r_0 r_1 (f_0 r_0 f_1)^{-1} = r_1 f_1 r_1 \in \langle f_0 \rangle^S \\ &\Longrightarrow \quad f_1 \in \langle f_0 \rangle^S . \end{aligned}$$

■

Finally, we have what is necessary to quickly dispose of the theorem.

Theorem 43. *The conjugate closure of* f_0 *contains everything, so that* $\langle f_0 \rangle^S = S$.

Proof: Because $f_0, f_1, r_0, r_1 \in \langle f_0 \rangle^S$, it follows that $S = \langle f_0 \rangle^S$. ■

This theorem is quickly extended to cover any flip. It is important to establish the that f_0 generates everything, but it is not especially important to extend this to additional classes of operators, since these will follow immediately from the complete listing of normal subgroups of S. Nevertheless, the extension of the theorem to cover all flips is relatively straightforward and contains some elegant steps. It also shows the general approach of using the propositions to find more and more operators until eventually a generator is constructed, and it contains many of the same logical steps used in the listing of the normal subgroups.

Using the argument of moving T_i to T_1 using an operator ν and noting that $f_i = \nu^{-1} f_1 \nu$, we have $\langle f_1 \rangle^S = \langle f_i \rangle^S$ where $i \geq 1$. Repeating the proof that $r_0 \in \langle f_0 \rangle^S$ for the subtree T_1 we have $r_1 \in \langle f_1 \rangle^S$ so that $r_i \in \langle f_j \rangle^S$, where $i, j \geq 1$. Locating $f_0 \in \langle f_1 \rangle^S$ can literally be accomplished by reversing the proof that $f_1 \in \langle f_0 \rangle^S$:

$$\begin{aligned} r_1 f_1 r_1 \in \langle f_1 \rangle^S \quad &\Longrightarrow \quad (f_0 r_0 f_1)^{-1} r_1 f_1 r_1 (f_0 r_0 f_1) = r_0 f_0 r_0 r_1 \in \langle f_1 \rangle^S \\ &\Longrightarrow \quad f_0 \in \langle f_1 \rangle^S . \end{aligned}$$

It follows that $\langle f_i \rangle^S = \langle f_0 \rangle^S = S$, where $i \geq 0$.

At this point, we begin building up machinery for the listing of normal subgroups.

Proposition 44. *If $f_3f_6 \in K$, where K is any conjugacy closure, then $f_if_j \in K$, for all $i, j \geq 1$.*

Using arguments above, we have already determined that $f_if_j \in K$, where $i, j \geq 3$. We also have $f_7f_2 = r_0^{-1}f_3f_6r_0 \in K$, $f_1f_{14} = r_0f_3f_6r_0^{-1} \in K$, and finally for $i, j \geq 3$:

$$\begin{aligned}(f_1f_{14})(f_{14}f_i) = f_1f_i \in K \quad &\Longrightarrow \quad (f_1f_i)^{-1} = f_if_1 \in K \\ (f_if_7)(f_7f_2) = f_if_2 \in K \quad &\Longrightarrow \quad (f_if_2)^{-1} = f_2f_i \in K \\ (f_1f_3)(f_3f_2) \in K \quad &\Longrightarrow \quad f_1f_2 = f_2f_1 \in K.\end{aligned}$$

■

Next, we show that f_3f_6 is effectively a generator of U.

Proposition 45. *If $f_3f_6 \in K$, then $U = \langle r_0 \rangle^S \subseteq K$.*

Proof: This is sufficient to show that rotations are contained in K. In the following, let $i \geq 0$:

$$\begin{aligned}&r_1(f_4f_3)r_1^{-1} \in K \\ &\qquad \Longrightarrow \quad (f_1f_3)(r_1f_4f_3r_1^{-1})(f_3f_1) = f_4f_1r_1f_4f_1r_1 \in \langle f_1 \rangle^S \\ &\qquad \Longrightarrow \quad (f_1f_4)f_4f_1r_1f_4f_1r_1 \in \langle f_1 \rangle^S \\ &\qquad \Longrightarrow \quad r_1^{-1}(r_1f_4f_1r_1)r_1 \in \langle f_1 \rangle^S \\ &\qquad \Longrightarrow \quad f_4f_1r_1^2 \in \langle f_1 \rangle^S \\ &\qquad \Longrightarrow \quad (f_1f_4)f_4f_1r_1^2 \in \langle f_1 \rangle^S \\ &\qquad \Longrightarrow \quad r_1^2 \in \langle f_1 \rangle^S \\ &\qquad \Longrightarrow \quad r_i^2 \in \langle f_1 \rangle^S\end{aligned}$$

In particular, $U = \langle r_0 \rangle^S \subseteq K$. ■

The next observation is the crucial observation that almost by itself lists off the normal subgroups.

Theorem 46. *If an element $\alpha \in S$ moves an invariant subtree T_i to $\alpha T_i \neq T_i$, then $U \subseteq \langle \alpha \rangle^S$.*

Proof: Consider such an element $\alpha \in S$ that moves an invariant subtree T_i to $\alpha T_i \neq T_i$. If it is not the case that $T_i \cap \alpha T_i = \emptyset$, choose a $n_k \in T_i - \alpha T_i$ and use T_k instead. If no such n_k exists, find a $n_k \in \alpha T_i - T_i$ and use T_j, where $n_j = \alpha^{-1} n_k$. We may now assume $T_i \cap \alpha T_i = \emptyset$ and let $T_j = \alpha T_i$. If it is the case that $i < 3$ or $j < 3$, choose T_{LLi} and $\alpha T_{LLi} = T_{LLj}$. We may now assume $i, j \geq 3$.

Consider the operation $f_i \alpha f_i \alpha^{-1} \in \langle \alpha \rangle^S$. By the above argument:

$$\begin{aligned} \alpha f_i \alpha^{-1} T_j &= \alpha f_i T_i \\ &= f_j T_j \end{aligned}$$

Because f_i leaves the portions outside the tree T_j unmodified, it follows that $\alpha f_i \alpha^{-1} = f_j$ and $f_i \alpha f_i \alpha^{-1} = f_i f_j \in \langle \alpha \rangle^S$. The disjointness of T_i and T_j allows me to move them to T_3 and T_6 using conjugation, from which follows that $f_3 f_6 \in \langle \alpha \rangle^S$ and then by Proposition 45, $U \subseteq \langle \alpha \rangle^S$. ■

This theorem makes it possible to classify all conjugate closures with this property.

Proposition 47. *Let $\alpha \in S$ be an element that moves an invariant subtree T_i to $\alpha T_i \neq T_i$. If α contains an even number of flips, then $\langle \alpha \rangle^S = U$, and $\langle \alpha \rangle^S = S$ otherwise.*

Proof: If α contains an even number of flips, then $\alpha \in U$, $\langle \alpha \rangle^S \subseteq U$, and $\langle \alpha \rangle^S = U$. Otherwise, $\alpha \notin U$ and thus contains an odd number of flips. Then, $f_0 \alpha \in U \subseteq \langle \alpha \rangle^S$, since $f_0 \alpha$ contained an even number of flips. Finally, $f_0 \alpha \alpha^{-1} = f_0 \in \langle \alpha \rangle^S$ so that $\langle \alpha \rangle^S = S$. ■

This leaves the case where α does not move any invariant subtrees. In this case, only a finite number of nodes are moved around, and α is a finite permutation. First, we show that the distinction between even and odd permutations is the same as the distinction between an even and odd number of flips.

Proposition 48. *A permutation on n nodes contains an even number of flips if and only if the permutation is an even permutation.*

Proof: Looking back at the development of flips on the outer ribbon, we find that $s = f_0 r_0$, $\omega = s^3$, $\omega_{01} = \omega$, $\omega_{i,i+1} = r_0^{-i} \omega r_0^i$, and $\omega_{ij} = \omega_{i,j-1} \omega_{j-1,j} \omega_{i,j-1}$ all contain an odd number of flips, provided $i \neq j$. Thus, any two-cycle on the outer ribbon has an odd number of flips. Consider the swap γ_{kp} of two arbitrary but distinct nodes n_k and n_p in S. Let τ be the permutation that swaps m_i with n_k and m_j with n_p. Then, $\gamma_{kp} = \tau^{-1} \omega_{ij} \tau$,

so that γ_{kp} has an odd number of flips. If α corresponds to an even permutation, then it can be written as a product of an even number of two-cycles, each of which has an odd number of flips. In this case, α has an even number of flips. On the other hand, if α corresponds to an odd permutation, then it can be written as product of an odd number of two-cycles, each of which has an odd number of flips. In this case, α has an odd number of flips. ■

The next proposition is the permutation equivalent of Theorem 46. Because it is more straightforward and less powerful, we consider it only as a proposition.

Proposition 49. *Let $\alpha \in S$, $\alpha \neq e$ be an element that does not move any invariant subtrees, and let $\sigma_U \in S$ be the set of even permutation elements. Then, $\sigma_U \subseteq \langle\alpha\rangle^S$.*

Proof: Assume α contains an even number of flips and is not the identity, and let β be another finite permutation with an even number of flips. Set up a one-to-one correspondence ϕ between the nodes that α or β acts on and a symmetric group S_k. If $k < 5$, we include a few nodes that neither α nor β acts on so that $k \geq 5$. Note that conjugation in S_k corresponds to multiplication by a permutation on one side and its inverse on the other, which in turn corresponds to multiplication on one side by a permutation operation and on the other side by its inverse, which is just conjugation in S. Note also that $\phi(\alpha), \phi(\beta) \in A_k$, where A_k is the alternating group, since both are even permutations. Because A_k for $k \geq 5$ is a simple group (see, for instance, [7]), it follows that the operation β, as well as any other element of A_k, can be obtained from α through multiplication, inverses, and conjugation. Mapping this construction back into S using ϕ^{-1} yields a construction of $\beta \in \langle\alpha\rangle^S$.

Assume instead that α contains an odd number of flips. Then, the operation $r_0 \alpha r_0^{-1} \in \langle\alpha\rangle^S$ is also a permutation with an odd number of flips, but it involves at least one element not involved in α. Thus, $r_0 \alpha r_0^{-1} \alpha^{-1} \in \langle\alpha\rangle^S$ is a finite permutation that contains an even number of flips and is not the identity element. This element can then be used to construct $\beta \in \langle\alpha\rangle^S$ using the same argument. ■

Next, we show that permutations do not pull in other operations when closed under conjugate closure. This also firmly establishes a second proper normal subgroup for S.

Proposition 50. *Finite permutations σ_S form a conjugate closure of S.*

Proof: If α is a finite permutation, then $\nu\alpha\nu^{-1}$ is a finite permutation, for any $\nu \in S$. To see why this is the case, examine its behavior on each node. Begin with a node n_i that is modified by α. Then, $\nu\alpha\nu^{-1}$

modifies the node νn_i, since $(\nu\alpha\nu^{-1})(\nu n_i) = \nu(\alpha n_i) \neq \nu n_i$. On the other hand, if n_i is not modified by α then $(\nu\alpha\nu^{-1})(\nu n_i) = \nu(\alpha n_i) = \nu n_i$. Because $\nu\alpha\nu^{-1}$ and α change the same number of elements, and α changes a finite number of elements, it follows that $\nu\alpha\nu^{-1}$ changes only a finite number of elements and is a permutation. Finite permutations are closed under composition and inverses, so that the conjugate closure of a finite permutation must be a finite permutation. ■

The following the an analog of Proposition 47 for permutations.

Proposition 51. *Let $\alpha \in \sigma_S$, $\alpha \neq e$ be a finite permutation. If α contains an even number of flips, then $\langle\alpha\rangle^S = \sigma_U$, and $\langle\alpha\rangle^S = \sigma_S$ otherwise.*

Proof: If $\alpha \in \sigma_U$ is not the identity, then $\sigma_U = \langle\alpha\rangle^S$. $\sigma_U \subseteq \langle\alpha\rangle^S$ follows from the construction of each element of σ_U in $\langle\alpha\rangle^S$. $\sigma_U \supseteq \langle\alpha\rangle^S$ follows from the fact that every element in $\langle\alpha\rangle^S$ must be a finite permutation with an even number of flips. If $\alpha \in \sigma_S - \sigma_U$, then $\sigma_S = \langle\alpha\rangle^S$. $\sigma_S \supseteq \langle\alpha\rangle^S$ follows from the closure of finite permutations under conjugate closure. To get the other direction, let β be any odd permutation. Then, $\alpha\beta$ is an even permutation, and $\alpha\beta \in \langle\alpha\rangle^S$. It follows that $\alpha^{-1}\alpha\beta = \beta \in \langle\alpha\rangle^S$ and $\sigma_S \subseteq \langle\alpha\rangle^S$. ■

Theorem 52. *The following is an exhaustive list of the normal subgroups of S:*

- *$\{e\}$, the trivial subgroup,*
- *S, the entire group,*
- *U, the subgroup whose elements have an even number of flips,*
- *σ_S, the subgroup whose elements are finite permutations,*
- *$\sigma_U = U \cap \sigma_S$, the subgroup whose elements are finite permutations and have an even number of flips.*

Proof: Any element $\alpha \in S$ fits into exactly one of these cases: α satisfies the conditions of Theorem 46, α satisfies the conditions of Proposition 49, or $\alpha = e$. The first case is completely characterized by Proposition 47. The second case is completely characterized by Proposition 51. Finally, the identity element is trivially closed under conjugate closure, so that it, too, is a normal subgroup. In particular, this shows that $\langle\alpha\rangle^S$ is a normal subgroup in the list, for any $\alpha \in S$.

Next, consider $A = \langle \alpha_1, \ldots, \alpha_n \rangle^S$ generated by multiple group elements, but not necessarily finitely many. We show that A is also in the list. If $n = 0$ or α_i for all i, then $A = \{e\}$. Otherwise, A is completely determined by the answer to two questions. Does α_i contain an even number of flips for any i? Does α_j move an invariant subtree? If the answer to both questions is "yes" and $i = j$ then $\langle \alpha_i \rangle^S = S$ and $A = S$. If the answer to both questions is "yes" and $i \neq j$ then $\langle \alpha_i \alpha_j \rangle^S = S$. This is because $\alpha_i \alpha_j$ has an odd number of flips, because α_i has an even number of flips and α_j has an odd number of flips. Further, $\alpha_i \alpha_j$ modifies an invariant subtree because α_i does and α_j does not. Again this leads to $A = S$. Otherwise, the answer to both questions is not "yes."

If the answer to the first question is "no" and the answer to the second question is "yes," then $\langle \alpha_i \rangle^S = U$ and $\alpha_k \in U$ for all k. Thus, $A = U$. If the answer to the first question is "yes" and the answer to the second question is "no," then $\langle \alpha_i \rangle^S = \sigma_S$ and $\alpha_k \in \sigma_S$ for all k. Thus, $A = \sigma_S$. The only remaining possibility is that both questions are answered "no," in which case $\langle \alpha_i \rangle^S = \sigma_U$ or $\langle \alpha_i \rangle^S = \{e\}$. The case where all elements generate the identity was already considered, so we may assume $\langle \alpha_i \rangle^S = \sigma_U$. $\alpha_k \in \sigma_U$ for all k. Thus, $A = \sigma_U$. ■

Proposition 53. *These quotient isomorphisms hold: $S/\sigma_S \cong U/\sigma_U$ and $S/U \cong \sigma_S/\sigma_U \cong \mathbb{Z}_2$.*

Proof: The second set is straightforward, as $S/U \cong \{e, f_0\} \cong \mathbb{Z}_2$ and $\sigma_S/\sigma_U \cong \{e, \gamma_{01}\} \cong \mathbb{Z}_2$. This leaves only $S/\sigma_S \cong U/\sigma_U$ to be shown.

Let $\iota : U \to S$ be the inclusion map. We claim that $\phi : U/\sigma_U \to S/\sigma_S$ defined by $\phi(\alpha\sigma_U) = \iota(\alpha)\sigma_S$ is a group isomorphism, from which the claim is immediate.

$$\begin{aligned}
\phi((\alpha\sigma_U)(\beta\sigma_U)) &= \phi((\alpha\beta)\sigma_U) \\
&= \iota(\alpha\beta)\sigma_S \\
&= (\iota(\alpha)\iota(\beta))\sigma_S \\
&= (\iota(\alpha)\sigma_S)(\iota(\beta)\sigma_S) \\
&= \phi(\alpha\sigma_U)\phi(\beta\sigma_U)
\end{aligned}$$

At this point, we define a function $\kappa : S \to U$. If $\alpha \in U$ then $\kappa(\alpha) = \alpha$. Otherwise, $\alpha\gamma_{01} \in U$, since γ_{01} is

an odd permutation operation. In this case, let $\kappa(\alpha) = \alpha\gamma_{01}$. Note that κ is not a homomorphism. However, it is true that $\kappa(\iota(\alpha)) = \kappa(\alpha) = \alpha$, so that $\kappa \circ \iota = id_U$.

Let $\psi : S/\sigma_S \rightarrow U/\sigma_U$ be defined by $\psi(\alpha\sigma_S) = \kappa(\alpha)\sigma_U$. Then, we have:

$$\begin{aligned}
\psi(\phi(\alpha\sigma_U)) &= \psi(\iota(\alpha)\sigma_S) \\
&= \kappa(\iota(\alpha))\sigma_U \\
&= \alpha\sigma_U \\
\phi(\psi(\alpha\sigma_S)) &= \phi(\kappa(\alpha)\sigma_U) \\
&= \iota(\kappa(\alpha))\sigma_S
\end{aligned}$$

If $\alpha \in U$, then $\iota(\kappa(\alpha))\sigma_S = \iota(\alpha)\sigma_S = \alpha\sigma_S$. Otherwise, $\iota(\kappa(\alpha))\sigma_S = \iota(\alpha\gamma_{01})\sigma_S = \alpha\gamma_{01}\sigma_S = \alpha\sigma_S$. It follows that $\phi^{-1} = \psi$, and ϕ is a group isomorphism as desired. ■

5.3 Tree Operations As Groups: Rotations

In this section, we will consider the group of rotations, $R = \langle r_i \rangle$. This is what we will call the rotation group; it is the group formed by the closure of tree rotations. Remember that we must add inverses when forming the closure. Unlike G, F, and S, left rotations are not generated by multiplying other elements.

Theorem 54. *The group R is finitely generated as* $R = \langle r_0, r_1, r_2, r_4 \rangle$.

Proof: We will consider the problem of finite generation by moving a subtree closer to the root without disturbing the subtree, performing the rotation, and then moving it back. We have done a construction like this before. This time we need to work a level down, so there will be more cases. In each case, we need an operation ν_i that will bring the subtree T_i closer to the root without disturbing it.

- $n_i \in T_7$: $\nu_i = r_0$.
- $n_i \in T_8$: $\nu_i = r_0$.
- $n_i \in T_9$: $\nu_i = r_0 r_1^{-1}$.

- $n_i \in T_{10}$: $\nu_i = r_1^{-1}$.
- $n_i \in T_{11}$: $\nu_i = r_2$.
- $n_i \in T_{12}$: $\nu_i = r_0^{-1} r_2$.
- $n_i \in T_{13}$: $\nu_i = r_0^{-1}$.
- $n_i \in T_{13}$: $\nu_i = r_0^{-1}$.

Then, we can define k by $n_k = \nu_i n_i$ and construct r_i by:

$$r_i = \nu_i^{-1} r_k \nu_i. \tag{5.25}$$

It follows that:

$$R = \langle r_0, r_1, r_2, r_3, r_4, r_5, r_6 \rangle. \tag{5.26}$$

Noting the the first two and last two cases can be combined at a higher level we can do a little better:

- $n_i \in T_3$: $\nu_i = r_0$
- $n_i \in T_9$: $\nu_i = r_0 r_1^{-1}$
- $n_i \in T_{10}$: $\nu_i = r_1^{-1}$
- $n_i \in T_{11}$: $\nu_i = r_2$
- $n_i \in T_{12}$: $\nu_i = r_0^{-1} r_2$
- $n_i \in T_6$: $\nu_i = r_0^{-1}$.

This removes two generators:

$$R = \langle r_0, r_1, r_2, r_4, r_5 \rangle . \tag{5.27}$$

We also have this relation: $r_5 = r_0 r_4 r_0^{-1}$. This eliminates another generator:

$$R = \langle r_0, r_1, r_2, r_4 \rangle . \tag{5.28}$$

■

Finite presentation seems unlikely. Using the above construction, we can generate relations like this:

$$r_0^{-n} r_1^{-p} r_0^n r_1^{-m} r_0^{-n+m} r_1^p r_0^{n-m} r_1^m = e. \tag{5.29}$$

This is valid for integers $n > m \geq 0$ and an arbitrary integer p, and is constructed by reducing $r_{L^n 1}$ in terms of the generators in two ways.

These are not all interesting generators, e.g., if $p = 0$ or $m = 0$, then the terms cancel out trivially. In fact, this can be generated for all p algebraically from the expression for $p = 1$:

$$r_0^{-n} r_1^{-1} r_0^n r_1^{-m} r_0^{-n+m} r_1 r_0^{n-m} r_1^m = e. \tag{5.30}$$

Here, $n > m > 0$. ($m = 0$ is trivial.) The simplest of these is:

$$r_0^{-2} r_1^{-1} r_0^2 r_1^{-1} r_0^{-1} r_1 r_0 r_1 = e. \tag{5.31}$$

5.3.1 Subgroups of the Rotation Group

Recall that the rotation group is $R = \langle r_0, r_1, r_2, r_4 \rangle$. Then, there are 15 subgroups of R formed by using fewer generators. Many of these are trivial and very easily described. One is the trivial group (remove all generators). $\langle r_0 \rangle \cong \langle r_1 \rangle \cong \langle r_2 \rangle \cong \langle r_4 \rangle \cong \mathbb{Z}$. Because r_1 and r_2 affect disjoint subtrees, they will commute. Similarly, r_2 and r_4 commute. Thus, $\langle r_1, r_2 \rangle \cong \langle r_2, r_4 \rangle \cong \mathbb{Z} \oplus \mathbb{Z}$.

It is an important question to us whether $\langle r_0 \rangle$ is a normal subgroup of R. In particular, it must be true that $r_1^{-1} r_0 r_1 \in \langle r_0 \rangle$, so that, $r_1^{-1} r_0 r_1 = r_0^k$. Then, $r_1^{-1} r_0^h r_1 = r_0^{hk}$ and $r_1^{-j} r_0^h r_1^j = r_0^{hk^j}$. Using an identity we have seen:

$$\begin{aligned}
e &= r_0^{-2} r_1^{-1} r_0^2 r_1^{-1} r_0^{-1} r_1 r_0 r_1 \\
&= r_0^{-2} r_1^{-1} r_0^2 (r_1^{-1} r_0^{-1} r_1) r_0 r_1 \\
&= r_0^{-2} r_1^{-1} r_0^2 r_0^{-k} r_0 r_1 \\
&= r_0^{-2} r_1^{-1} r_0^{3-k} r_1 \\
&= r_0^{-2} (r_1^{-1} r_0^{3-k} r_1) \\
&= r_0^{-2} r_0^{(3-k)k} \\
&= r_0^{(3-k)k-2} \\
&= r_0^{(3-k)k-2}.
\end{aligned}$$

Because r_0 has characteristic zero, it must be that $(3-k)k-2 = 0$, i.e., $k = 1$ or $k = 2$. If $k = 1$, then $r_1^{-1} r_0 r_1 = r_0$ and $r_0 r_1 = r_1 r_0$. This is a contradiction, since these rotations do not commute. Thus, it must be that $k = 2$. This leads to a contradiction, since:

$$\begin{aligned}
e &= r_0^{-n} r_1^{-1} r_0^n r_1^{-m} r_0^{-n+m} r_1 r_0^{n-m} r_1^m \\
&= r_0^{-n} r_1^{-1} r_0^n r_1^{-m+1} (r_1^{-1} r_0^{-n+m} r_1) r_0^{n-m} r_1^m \\
&= r_0^{-n} r_1^{-1} r_0^n r_1^{-m+1} r_0^{k(-n+m)} r_0^{n-m} r_1^m \\
&= r_0^{-n} r_1^{-1} r_0^n r_1^{-m+1} r_0^{(1-k)(n-m)} r_1^m \\
&= r_0^{-n} r_1^{-1} r_0^n r_1 r_0^{(1-k)(n-m)k^m} \\
&= r_0^{-n} r_0^{nk} r_0^{(1-k)(n-m)k^m} \\
&= r_0^{(1-k)(n-m)k^m + nk - n}
\end{aligned}$$

Again, the power must be zero:

$$\begin{aligned}
0 &= (1-k)(n-m)k^m + nk - n \\
0 &= (1-k)((n-m)k^m - n) \\
0 &= ((n-m)2^m - n).
\end{aligned}$$

This is not true for all $n > m > 0$. Thus, no such k may exist. It follows that $\langle r_0 \rangle$ is not a normal subgroup of R. Note that $\langle r_0 \rangle$ is the subgroup of free operations if flips are ignored.

Proposition 55. *The subgroup $\langle r_0 \rangle$ of R is not a normal subgroup.*

5.3.2 Eliminating a generator

We claim that $r_2 \in \langle r_0, r_1 \rangle$. In fact, $r_2 = r_0^2 r_1^{-1} r_0^{-1}$. Thus, we can rewrite R:

$$R = \langle r_0, r_1, r_4 \rangle . \tag{5.32}$$

In fact, we can get a rather interesting and elegant expression for r_2^n:

$$r_2^n = r_0^2 (r_1^{-1} r_0)^n r_0^{-2}. \tag{5.33}$$

Noting that r_2 and r_1 commute, we have $r_2^n r_1^m = r_1^m r_2^n$, i.e.,

$$r_0^2 (r_1^{-1} r_0)^{-n} r_0^{-2} r_1^{-m} r_0^2 (r_1^{-1} r_0)^n r_0^{-2} r_1^m = e. \tag{5.34}$$

We get a similar relationship by noting that r_2 commutes with r_4:

$$r_0^2 (r_1^{-1} r_0)^{-n} r_0^{-2} r_4^{-m} r_0^2 (r_1^{-1} r_0)^n r_0^{-2} r_4^m = e. \tag{5.35}$$

In particular, we can use the commutativity of disjoint operations to create more identities. E.g., we can construct $r_5 = r_0 r_4 r_0^{-1}$ and require that it commute with r_4:

$$r_0 r_4^{-m} r_0^{-1} r_4^{-n} r_0 r_4^m r_0^{-1} r_4^n = e. \tag{5.36}$$

It also commutes with r_1:

$$r_0 r_4^{-m} r_0^{-1} r_1^{-n} r_0 r_4^m r_0^{-1} r_1^n. \tag{5.37}$$

5.3.3 Characterizing R

In much the same way as it was possible to describe S in a very simple way, it is also very easy to describe R. Namely, R is the set of all operations that requires changing only a finite number of child pointers, preserves the infinite complete binary tree structure, and does not change the relative ordering of elements in the tree.

The observation that elements of R satisfy those properties is straightforward. Finiteness follows from the finiteness of S. The retention of relative ordering follows from the well-known fact that tree rotations do not affect the relative ordering of nodes. Thus, these restrictions are necessary. What isn't immediately obvious is that these restrictions precisely describe R's contents. As with the characterization of S, create a set U of subtrees that are not changed by the operation α being considered. The relative ordering of these subtrees is not changed.

Consider that k of the n subtrees in U are on the left side of the root. We wish to move one of these subtrees to the right side of the root, leaving $k-1$ on the left side. Because the relative ordering of the trees is fixed, there is only one subtree t that can be moved over. Before we can move it over, we need to draw it as close to the root as possible. We can do this using the operation l_1. Because the nodes n_1 and n_4 are ancestors of t (it is obtained by following left children from n_1), it follows that T_1 and T_4 are not in U, so that none of the subtrees in U are disturbed by the operation. It is also easy to see that as long as t is further from the root than n_4, it will be drawn one level closer to the root. This process is halted when $t = T_4$. Then, r_0 moves it to T_5, leaving $k-1$ subtrees on the left side. Thus, $r_0 l_1^{h(t)-2}$ is the operation that moves t from the left side of the root to T_5, which is to the right of the root. Further, no other elements of U are moved over. This can be seen by noting that l_1 cannot move elements across the root, and r_0 moves only T_4 across the root, leaving T_3 sill on the left side of the root. All subtrees in U on the left side of the root must be inside T_3, since T_1 cannot be in U (because T_4 is inside T_1).

The reverse operation is derived and justified using the same techniques. This operator is described by $l_0 r_2^{h(t)-2}$. We can now define two operations:

$$\sigma_{kt} = r_k l_{Lk}^{h(t)-h(n_k)-2}, \tag{5.38}$$

$$\tau_{kt} = l_k l_{Rk}^{h(t)-h(n_k)-2}. \tag{5.39}$$

These operations are analogous to the ones at the root, except that they apply to n_k instead. Note that k is a node index and t is a node.

The construction for generating α is recursive and simple. Let unprimed nodes and subtrees indicate the current tree and primed nodes and subtrees indicate the tree obtained by applying α to the original tree.

For a node k, count how many subtrees in U occur in T_{Lk} and T'_{Lk}. If there are more in T_{lK}, let t be the rightmost such subtree and apply σ_{kt}. If there are more in T'_{Lk}, let t be the leftmost subtree from U in T_{Rk}, and apply τ_{kt}. This is repeated until the number of subtrees on each side of n_k is right. Recurse on Lk and Rk.

The construction starts with $k = 0$ and ends each time $T_k \in U$. To see that the construction will terminate in a finite number of steps, it is important to make a few simple observations. First, each path from the root will eventually reach a node n_k where $T_k \in U$. This follows from the fact that the number of nodes not in a subtree in U is finite. The next thing to note is that either $T_k \in U$ or there is at least one subtree in U in T_{Rk} and T_{Lk}. Thus, the maximum depth is limited by the number of subtrees in U, which is finite.

To see that the construction actually works, consider the number of subtrees in U that are in T_{Lk} and T'_{Lk}. These are always equal. If k is not in any subtree of U, these will be positive and equal. Consider the parent of the root of a subtree of U, n_k. Assume without loss of generality that the root of the subtree was n_{Lk}. The number of subtrees from U in T_{Lk} is precisely one, since it is the subtree, and the subtrees in U are disjoint. But then the number of subtrees in T'_{Lk} is one. If this subtree weren't rooted at T'_{Lk}, then it would be to one side of the node n_{Lk}. There are an infinite number of nodes on the other side of the node that are not in any subtree U, which is a contradiction. It follows that $T'_{Lk} \in U$. Thus, subtrees are located at the right places in the tree. Because the ordering is fixed, it follows that these subtrees are actually in the correct places.

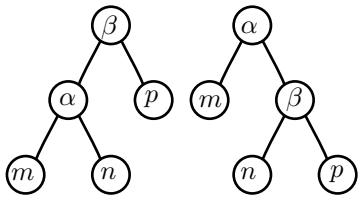
Now that the subtrees are in the right places, the rest of the nodes in the tree are uniquely determined by the preserved ordering. It follows that this construction has produced α.

Theorem 56. *The group R is the set of all operations that requires changing only a finite number of child pointers, preserves the infinite complete binary tree structure, and does not change the relative ordering of elements in the tree.*

5.4 Metric-Preserving Operations

There are to metric-preserving operations being considered; the first is the rotation at the root, and the second is a flip at an arbitrary node in the tree.

Originally, β is the root of the tree; after the (right) rotation, α is the root of the tree. Let the original weights be $d_{\alpha m}$, $d_{\alpha n}$, $d_{\alpha\beta}$, and $d_{\beta p}$. Let the desired weights after the rotation be $d'_{\alpha m}$, $d'_{\alpha\beta}$, $d'_{\beta n}$, and



(a) Before (b) After

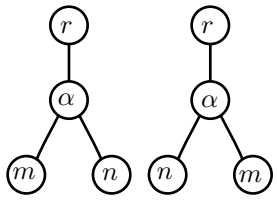
Figure 5.3: Right rotation about the root, β.

$d'_{\beta p}$. Note that the subscripts indicate endpoints for the edge, and the primed distances correspond to edge weights after rotation. Let T_a represent the subtree rooted at node a. We can assign these primed distances as follows:

$$\begin{aligned} d'_{\alpha m} &= d'_{\alpha\beta} = \frac{1}{2} d_{\alpha m}, \\ d'_{\beta n} &= d_{\alpha n}, \\ d'_{\beta p} &= d_{\alpha\beta} + d_{\beta p}. \end{aligned}$$

We claim that the metric represented by the tree (or rather the leaves of the tree) is invariant under this rotation with the new weights chosen in this way.

Consider a distance d_{ij}, where $i \in T_a$ and $j \in T_b$ are leaves of the subtrees. If $a = b$, then the path does not involve the portion of the tree being changed, so the distance is unchanged. Otherwise, $a \neq b$. Then, $d_{ij} = d_{ia} + d_{ab} + d_{bj}$, where d_{ia} is the distance from leaf i to node a. Note that d_{ia} and d_{bj} do not contain edges involved in the rotation, so they are unchanged by the rotation. It follows that d_{ij} is unchanged if and only if d_{ab} is unchanged. It in turn follows that the metric described by the tree is unchanged if d_{mn}, d_{mp}, and d_{np} are all unchanged by the rotation. This follows from the following three chains of equalities:



(a) Before (b) After

Figure 5.4: Child flip about the node α.

$$
\begin{aligned}
d_{mn} &= d_{\alpha m} + d_{\alpha n} = \frac{1}{2} d_{\alpha m} + \frac{1}{2} d_{\alpha m} + d_{\alpha n} = d'_{\alpha m} + d'_{\alpha\beta} + d'_{\beta n} = d'_{mn}, \\
d_{mp} &= d_{\alpha m} + d_{\alpha\beta} + d_{\beta p} = d'_{\alpha m} + d'_{\alpha\beta} + d'_{\beta p} = d'_{mp}, \\
d_{np} &= d_{\alpha n} + d_{\alpha\beta} + d_{\beta p} = d'_{\beta n} + d'_{\beta p} = d'_{np}.
\end{aligned}
$$

Thus, distances are preserved under right rotation about the root. Since left rotation about the root is just the inverse of this operation, it must also preserve distances.

In the case of child flips, the edge weights are not changed. All paths contain the same edges, and these edges' weights are not changed. It follows that the path lengths and the metric distances are also unchanged. The metric is invariant under child flips.

6. Conclusions

6.1 Summary

Metric trees have found important uses in computer vision and other active areas. Metric trees present efficient algorithms or efficient approximation algorithms for problems that are intractible on general metrics. This has made metric trees particularly attractive. At this time, only one efficient approximation algorithm is known for embedding arbitrary metrics into trees, and this algorithms produces trees that are typically far from balanced. While unbalanced trees are quite suitable for many domains, it can lead to suboptimal performance in others. This thesis introduces efficient algorithms for computing weights on a metric tree under L_2 and L_∞ norms. Both algorithms are optimal. Similarly, we present local versions of both algorithms that compute edge weights locally with the same optimality guarantees.

From this point, the thesis shifts its focus by extending the tree and considering its operations as a group. The group that results has, to the best of our knowledge, never been studied before. This thesis therefore initiates the study of this group. It explores the underlying structure of the group and constructs important operations within the group, such as flips and permutations. It gives a simple characterization of the group and proves its finite generation. It also considers two smaller groups, including one that is a group consisting of operations that do not affect the distortion of metric trees.

Normal subgroups are especially important both for practical applications and the general theory of the group itself. Normal subgroups make it possible to take quotients. While it is not true that the group of free operations or any useful subset of that group forms a normal subgroup, the group does in fact have normal subgroups. The thesis finds and proves a complete list of the normal subgroups. Finally, the thesis switches to the group consisting only of rotations and proves some of the same results for that group, such as finite generation and a complete characterization.

6.2 Future Work

There are a number of important questions that this thesis raises but does not solve. These are left for future work.

- Does an efficient algorithm exist to balance a metric tree with distortion at most a constant factor from optimal under L_2 or L_∞?
- Is the group S isomorphic to any other known group?
- Is the group U simple?
- The application for which S was originally studied failed because the group of free operations did not form a normal subgroup. Can the group properties of S be applied in other ways to the problem of balancing metric trees?
- Which properties of S can be restricted to finite subsets of S so that results of the study of S can be applied to finite trees? What is a good methodology for performing this restriction? Does S yield useful applications under this restriction? Does calling invariant subtrees "leaves" provide a suitable restriction?
- The group of only tree rotations should be studied in greater depth.
- Does the group generated by root rotations and flips contain all operations that do not cause distortion on metric trees?
- What is the quotient group S/σ_S? Does it lie in S, so that S may be expressed as a semidirect product using it?

Bibliography

[1] S. Sidki A. Brunner. Wreath operations in the group of automorphisms of the binary tree. *Journal of Algebra*, 257(1):51–64, 2002.

[2] R. Agarwala, V. Bafna, M. Farach, M. Paterson, and M. Thorup. On the approximability of numerical taxonomy (fitting distances by tree metrics). *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, 28(3):365–72, 1996.

[3] M. Aschbacher. *Sporadic Groups*. Cambridge Tracts in Mathematics. Cambridge University Press, 1994.

[4] L. Beinekey, R. Wilson, M. Keynes, and P. Cameron, editors. *Topics in Algebraic Graph Theory*. Encyclopedia of Mathematics and its Applications. Cambridge University Press.

[5] N. Biggs. *Algebraic graph theory*. Cambridge Mathematical Library (Cambridge University Press), second edition, 1993.

[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2006.

[7] D. Dummit and R. Foote. *Abstract Algebra*. Wiley, 1999.

[8] R. Grigorchuk. On the system of defining relations and the schur multiplier of periodic groups generated by finite automata. In *Groups St Andrews 1997 in Bath*, pages 290–317, 1999.

[9] M. Hazewinkel. Tree-tree matrices and other combinatorial problems from taxonomy, 1996.

[10] Y. Lavrenyuk, V. Mazorchuk, A. Oliynyk, and V. Sushchansky. Faithful group actions on rooted trees induced by actions of quotients. *Comm. Algebra*, 2005.

[11] J. Lucas. The rotation graph of binary trees is hamiltonian. *Journal of Algorithms*, 9:503–35, 1988.

[12] J. Lucas, D. Baronaigien, and F. Ruskey. On rotations and the generation of binary trees. *Journal of Algorithms*, 15:343–66, 1993.

[13] J. Matousek. On embedding trees into uniformly convex banach spaces. *Israel Journal of Mathematics*, 1999.

[14] R. Myers, R. Wilson, and E. Hancock. Bayesian graph edit distance. *IEEE PAMI*, 22(6):628–635, 2000.

[15] Y. Nesterov and A. Nemirovskii. Interior-point polynomial methods in convex programming. *Society for Industrial and Applied Mathematics*, 1994.

[16] Y. Nesterov and M. Todd. Primal-dual interior-point methods for selfscaled cones. *SIAM Journal on Optimization*, 8(2):324–64, 1998.

[17] A. Sanfeliu and K. S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:353–362, May 1983.

[18] T. Sebastian, P. Klein, and B. Kimia. Recognition of shapes by editing shock graphs. In *IEEE International Conference on Computer Vision*, pages 755–762, 2001.

[19] Wikipedia. Conjugate closure — wikipedia, the free encyclopedia, 2006. [Online; accessed 2-July-2006].

[20] Wikipedia. Group action — wikipedia, the free encyclopedia, 2006. [Online; accessed 2-July-2006].

[21] Wikipedia. Metric space — wikipedia, the free encyclopedia, 2006. [Online; accessed 2-July-2006].