Diameters of groups generated by transposition trees

Benjamin Kraft

Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

Let $G = \langle S \rangle$ be a group, and let Γ be its Cayley graph. Computing the diameter of Γ is a computationally hard problem which comes up in several contexts. Thus, it is useful to be able to compute bounds on the diameter of Cayley graphs. In [1] Ganesan studied the case where S is a minimal set of transpositions which generate G, and provided an algorithm to find an upper bound on $\operatorname{diam}\Gamma$ without examining each permutation. Expanding on this work, we give several new algorithms to compute upper bounds on the diameter of Γ, without examining individual elements of G. In particular, we give one algorithm which is much faster to compute than Ganesan's, and one which produces better bounds than previous algorithms.

Keywords: group diameter, symmetric group, transposition tree

1. Introduction

Computing diameters of Cayley graphs is a well-known problem in finite group theory. Given a group G generated by some set S with $S = S^{-1}$, let Γ be the Cayley graph of G with respect to S, so that the vertices of Γ are the elements of G, and the edges of Γ are undirected edges between g and gs for any $s \in S$ and $g \in G$. The *diameter* of G (written $\operatorname{diam} G$ or $\operatorname{diam}\Gamma$) is the maximum distance between two vertices of Γ, or, equivalently, the smallest

Email address: benkraft@mit.edu (Benjamin Kraft)

number d such that any element of G can be written as a product of d elements of S. Certain cases of the problem of finding or bounding diameters of groups were studied by Babai and Hetyei [2], Babai and Seress [3, 4], Helfgott and Seress [5], and Bamberg et al. [6].

A well-publicized example of the diameter of a Cayley graph was the announcement in 2010 that any Rubik's Cube can be solved in 20 moves [7]. The result took sophisticated algorithms and over 35 years of CPU time; in general, computing the diameters of Cayley graphs is computationally difficult.

In this paper we consider symmetric groups as generated by minimal sets of transpositions. A *transposition graph* is a graph G on n labeled vertices that corresponds to a set S of transpositions in S_n, where $(i, j) \in S$ if and only if vertices i and j are adjacent in G. A *transposition tree* is a transposition graph that is a tree, which happens if and only if S is a minimal generating set.

The diameters of symmetric groups as generated by transpositions were studied by Akers and Krishnamurthy in a much-cited paper [8]; their paper includes specific results for a few types of transposition trees. However, their only general bound is the following.

Theorem 1 (Akers & Krishnamurthy)**.**

$$\operatorname{diam} \Gamma \leq \max_{\pi \in S_n} \left\{ c(\pi) - n + \sum_{i=1}^{n} \operatorname{dist}_T(i, \pi(i)) \right\}. \tag{1}$$

Unfortunately, this bound is computationally intractable, as it requires iterating over each permutation in S_n.

Ganesan [1] gave an algorithm β for computing bounds on the diameter of S_n as generated by transposition trees which looks only at the tree, rather than examining each individual permutation in S_n. This is a significant improvement over the previous algorithms – instead of computing a quantity for every element of S_n, as naïve approaches and Akers and Krishnamurthy's do, he gives a simple algorithm. However, Ganesan's algorithm is still somewhat slow, and the bound can be improved significantly.

We provide three new algorithms, α, ζ, and η. Our first algorithm, α, is harder to compute than β, but produces a better bound; its bound is always at least as good as β, can be quadratically better on certain trees, and does somewhat better empirically. On the other hand, η is computationally very fast – it can be computed easily even for trees on hundreds of nodes, and in practice it does only a little worse than β.

In Section 2 we discuss the previous work by Ganesan, in addition to some new properties of his algorithm. In Sections 3, 4, and 5, we discuss our new algorithms, α, η, and ζ, respectively. Section 6 compares our algorithms to each other and to previous algorithms. Section 7 discusses open questions for further research. Appendix Appendix A gives examples of how each algorithm works on specific families of transposition trees.

2. Previous work: Algorithm β

2.1. Transposition trees

To define Ganesan's algorithm, we must first explain the framework for understanding transposition trees. We can think of a permutation in S_n as acting on a set of moveable *markers* on the vertices of the transposition tree. We place a marker $\pi(i)$ at each vertex i of T. Then swapping the markers at vertices i and j corresponds to multiplying π by (i, j) (on the right if we multiply permutations right-to-left). Then expressing a permutation as a product of the generators in the transposition tree corresponds to giving a sequences of edges in the transposition tree, such that sequentially swapping the markers along each edge moves each marker to its corresponding vertex. We call moving a set of markers to their corresponding vertices *homing* those markers.

For example, consider $S_4 = \langle (1,2), (1,3), (1,4) \rangle$; we can represent it by the transposition tree shown in the first part of Figure 1. Suppose we want to show the permutation $(1, 2, 3)$ on this tree. We put marker 2 on vertex 1, marker 3 on vertex 2, marker 1 on vertex 3, and marker 4 on vertex 4, as shown in the second part of Figure 1. Now, if we swap the markers on vertices 1 and 2, we get the permutation $(1, 2, 3)(1, 2) = (1, 3)$, as shown. Subsequently swapping the markers on vertices 1 and 3 would give us the

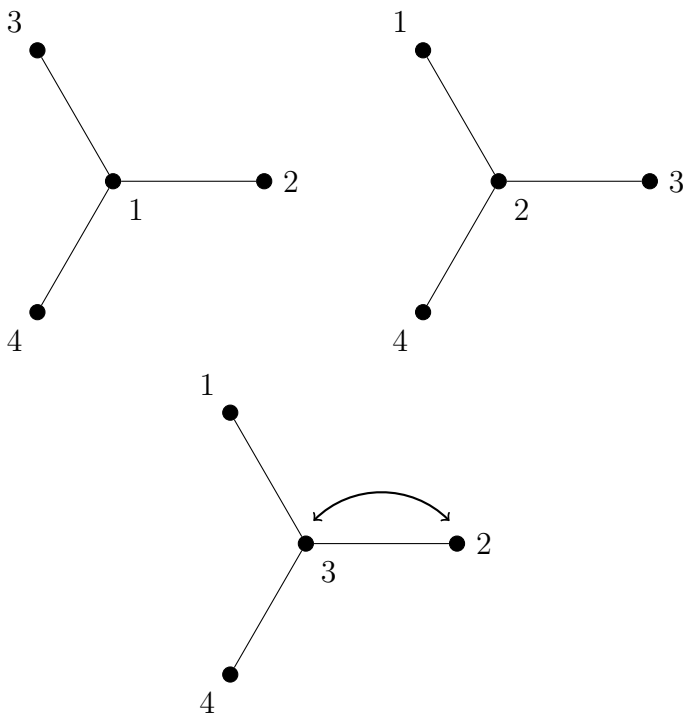

Figure 1: Transposition trees for S_4 as generated by $(1,2)$, $(1,3)$, and $(1,4)$. On the left, the original transposition tree, with the labelled nodes. In the center, the markers are in the permutation $(1,2,3)$. On the right, we have swapped the markers on vertices i and j, giving the permutation $(1,2,3)(1,2) = (1,3)$.

permutation $(1,2,3)(1,2)(1,3) = (1)$, so we have moved each marker to its corresponding vertex, or, equivalently, written $(1,2,3) = (1,3)(1,2)$ as a product of the given set of generators.

2.2. Algorithm $\boldsymbol{\beta}$

We next define Ganesan's algorithm β. Algorithm β iterates over the transposition tree, removing two vertices at each step, and recursively computing a bound for the diameter of the Cayley graph Γ of the group G.

Although Ganesan defines β in a non-recursive fashion, we find it easier to define it recursively. We begin with notation. Let T be a tree on n vertices with diameter d. Denote by $V(T)$ the set of vertices of T, and let $L(T)$ be the set of leaves of T; if $I \subset V(T)$, let $T - I$ be the induced subgraph obtained by removing the vertices in I from T.

Algorithm 1. *Nondeterministically let $\beta(T)$ be any of the values*

$$2d - 1 + \beta(T - \{i, j\})$$

where $i, j \in L(T)$ and $\operatorname{dist}_T(i, j) = d$. If T has order 0 or 1, let $\beta(T) = 0$. Then let $\beta_{\max}(T)$ be the maximum possible value produced. (Note that while β is not deterministic, the maximum value it produces is a well-defined quantity, which we can compute recursively by taking a maximum over i and j.)

Theorem 2 (Ganesan [1])**.** $\operatorname{diam}\Gamma \leq \beta_{\max}$.

We omit the details of Ganesan's proof, but the idea is that there is always a pair of maximally separated vertices whose markers can be homed in $2d - 1$ moves.

Algorithm β has many advantages over previous methods, such as the formula (1) of Akers and Krishnamurthy. Their formula requires iterating over every possible permutation in S_n to compute a bound; β involves only the properties of the transposition tree. However, while algorithm $\beta_{\max}$ is a significant improvement over the previous methods, it has two main defects. First, it is computationally slow, since one must compute all possible values of $\beta$and only the maximal one is a bound. Second, the bound it produces is often significantly larger than the actual diameter.

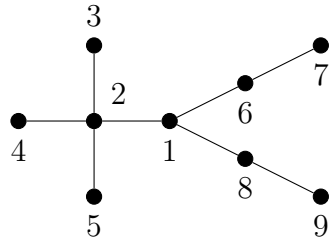

Figure 2: The transposition tree T.

We now discuss some properties of Ganesan's β, and why simply computing an arbitrary β does not necessarily produce a bound.

2.3. Non-maximal $\boldsymbol{\beta}$

Ganesan conjectures that $\beta_{\min}$ is also an upper bound on $\operatorname{diam}\Gamma$. While we are unable to find a counterexample to this claim (primarily because, in general, β is relatively far off the exact diameter, and computing the diameter of large transposition trees exactly is quite hard), there is reason to suspect that, at the least, such a proof cannot proceed recursively.

In particular, consider the transposition tree T on 9 vertices from Ganesan [1], shown in Figure 2, and suppose we start with the permutation $(3,7)(5,9)$. If we are to prove $\beta_{\min}$ is an upper bound for $\operatorname{diam}\Gamma$ recursively, we must show that we can home *any* two maximally separated markers in $2\operatorname{diam}T - 1 = 7$ moves. However, we will show that homing 7 and 9 in the above permutation requires 8 moves.

In particular, to home both markers, we must use each of the swaps $(2,3)$, $(2,5)$, $(1,2)$, $(1,6)$, $(6,7)$, $(1,8)$, and $(8,9)$ at least once, since each is on either all paths from 3 to 7 or all paths from 5 to 9. Then to home both markers in 7 moves, we must use each swap exactly once. However, after the swap $(1,2)$, we can end up with only one of 7 and 9 on the right side of that edge; thus the other must require another use of $(1,2)$, so we cannot home both in 7 moves.

2.4. Non-unique values of β

Ganesan asks whether algorithm β produces a unique value on almost all trees as $n \to \infty$. This problem remains open, but as a partial result, we exhibit a large infinite family of trees on which it always takes on multiple values, and give computational results for all trees on at most 17 vertices that suggest that unique β values become less common for large n, and that there exist trees with arbitrarily many values of β.

Proposition 1. *For any tree T on n vertices with multiple values of β, and any rooted tree R on m vertices, there is a tree T' on $n + 2m$ vertices, depending on R, which also has multiple values of β; each R produces a distinct T'.*

Proof. Consider any tree T on n vertices with multiple values of β (for example, the tree shown in Figure 2, for which β can produce 20 or 22), and let i and j be two maximally-separated vertices. Then for any rooted tree R on m vertices, attach a copy of R to each of i and j, such that i and j are the roots of the trees, resulting in a new tree T' on $n + 2m$ vertices. To compute β, we may always pick a vertex maximally distant from the root of R, and remove both of the corresponding vertices in T'. Eventually, we will get back to T, at which point we continue as if computing the value of β for T, which must result in multiple possible values. Then $\beta(T')$ must take on at least that many possible values as well. Note that each R produces a distinct T' as desired. □

Otter [9] showed that the number of unrooted trees grows as $Ca^n n^{-5/2}$, and the number of rooted trees grows as $Da^n n^{-3/2}$ as $n \to \infty$, where $C \approx 0.5349\ldots$, $D \approx 0.4399\ldots$, and $a \approx 2.9956\ldots$. Then the number of such examples with n vertices grows at least as the number of rooted trees on $\lfloor \frac{n-9}{2} \rfloor$ vertices, i.e., as $\Theta(a^{n/2} n^{-3/2})$. This grows more slowly than the number of unrooted trees, but still shows that many trees have multiple β values.

Proposition 2. *There exist trees with arbitrarily many values of β.*

Proof. Consider the tree T_n shown in Figure 3. We will show by induction that it has at least n values of β, namely

$$\{10{+}4j \mid 0 \le j \le n{-}2\} \cup \{14n{-}6\} = \{10n, 10n{+}4, \ldots, 14n{-}12, 14n{-}8, 14n{-}6\}.$$

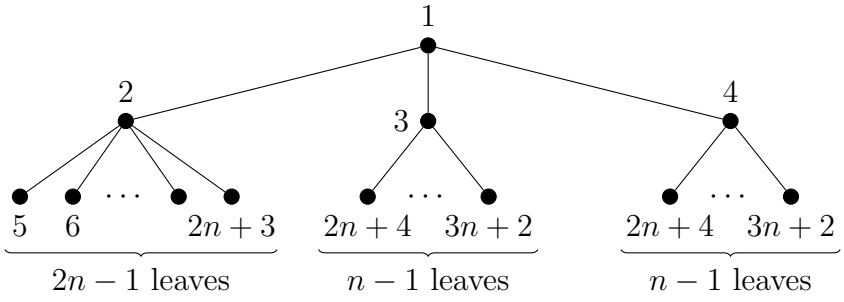

Figure 3: T_n, with n values of β.

First, Ganesan shows, and the computation is simple to check, that when $n = 2$, the tree $T_2 = T$ is the one shown in Figure 2, which has two values of β, 20 and 22.

Now consider some T_n, and suppose that T_{n-1} has the desired n values of β. Then we can remove from T_n the pairs of vertices $\{5, 2n+4\}$ and $\{6, 3n+3\}$, leaving a tree isomorphic to T_{n-1}, and giving us the values of β of $7 + 7 + \beta(T_{n-1})$ which can take on any of the values in $\{10n + 4, 10n + 8, \ldots, 14n - 12, 14n - 8, 14n - 6\}$. Finally, if we remove the pairs $\{2n + 4, 3n + 3\}, \{2n + 5, 3n + 4\}, \ldots, \{3n + 2, 4n + 1\}$, and then $\{4, 5\}$ and $\{3, 6\}$, we get a star graph on $2n+1$ vertices, so $\beta(T_n) = 7(n-1)+5+5+\beta(S_{2n-1}) = 10n$. Then T_n has all the claimed values of β, so by induction we are done. □

Table 1 gives the number of trees with non-unique β on at most 17 vertices.

3. Algorithm α

3.1. Motivation

In this section we define an algorithm α for computing bounds on the diameter of S_n as generated by transposition trees. Algorithms α and $\beta_{\max}$ have the same computational complexity, although in practice, α is slower than $\beta_{\max}$. However, α obtains a much tighter bound in certain cases, and a somewhat better bound in many cases; in all cases its bound is at least as

n	Unrooted trees	Trees with multiple values of β			Fraction with multiple values
		2 values	3 values	4 values	
1	1				0.0%
2	1				0.0%
3	1				0.0%
4	2				0.0%
5	3				0.0%
6	6				0.0%
7	11				0.0%
8	23				0.0%
9	47	1			2.1%
10	106	1			0.9%
11	235	9			3.8%
12	551	9			1.6%
13	1301	79	2		6.2%
14	3159	83	2		2.7%
15	7741	606	26		8.2%
16	19320	673	24		3.6%
17	48629	4676	298	6	10.2%

Table 1: Trees with multiple values of β.

good as that of $\beta_{\max}$. Details of the performance of each algorithm are given in Section 6.

Because computing exact diameters of groups is hard, it is difficult to tell exactly how much α differs from the actual diameter. However, we suspect that the bound given by α is as close as possible to the actual diameter as we can get without considering individual permutations or working non-recursively.

3.2. Definition

To define the algorithm α recursively; we first introduce some notation. Again, let T be a tree on n vertices with diameter d, let $V(T)$ be the set of vertices of T, and let $L(T)$ be the set of leaves of T. If $I \subset V(T)$, let $T - I$ be the induced subgraph obtained by removing the vertices in I from T. In addition, let $C(T) \subset V(T)$ be the center of T. (The center of a tree T is the set of vertices v such that $\max\{\text{dist}_T(u, v) \mid u \in V(T)\}$ is minimal.) Then $|C(T)|$ must be 1 or 2. If $|C(T)| = 1$ then we also use $C(T)$ to refer to the unique central vertex of T.

Furthermore, we define a set of rooted trees, the *central subtrees* of T, as follows. If $C(T) = \{a, b\}$, then define the central subtrees to be the connected components of T with the edge (a, b) removed, and with roots a and b. If $C(T) = a$, then define the central subtrees of T to be the connected components of $T - \{a\}$, with the vertices adjacent to a as the roots. Each subtree has depth at most $\lfloor \frac{d-1}{2} \rfloor$, and so we say subtrees with that depth have *maximum depth*.

Algorithm 2. *Let $\alpha(T) = \alpha_{\min}(T)$ be the minimum of the following values.*

(a) If T has at most 2 central subtrees with maximum depth, then

$$2d-1+\min\{\alpha(T-\{i,j\}) \mid i,j \text{ leaves of distinct central subtrees of } T\}.$$

(If not, skip this case.)

(b) $d + \min\{\alpha(T - \{i\}) \mid i \in L(T)\}$.

(c) $2d - 1 + \min\limits_{i \in L(T)} \left(\max\limits_{\substack{j \in L(T) \\ \text{dist}_T(i,j)=d}} \Big(\alpha(T - \{i, j\}) \Big) \right)$. *(If the maximum is empty for some i, we skip that i.)*

Finally, let $\alpha(T)$ be 0 if T has 0 or 1 vertices.

Theorem 3. $\operatorname{diam}\Gamma \leq \alpha(T) \leq \beta_{\max}(T)$.

Proof. We prove the first inequality by induction on n. In the base case, T has 0 or 1 vertices, and $\operatorname{diam}\Gamma = \alpha(T) = \beta(T) = 0$. Then suppose we have some tree T with n vertices, and that $\operatorname{diam}\Gamma \leq \alpha(T)$ for all trees on fewer than n vertices. Consider some permutation π of the markers on its vertices, where the marker i begins on vertex $\pi^{-1}(i)$.

First, suppose α takes branch (a) of the minimum, removing i and j. If $\operatorname{dist}_T(\pi^{-1}(i), i) < d$, we can move i's marker home in at most $d-1$ moves, then j's home in at most d moves, so we can home any i and j, and thus remove them from T, in $2d-1$ moves, as desired. Similarly, if $\operatorname{dist}_T(\pi^{-1}(j), j) < d$, we can do the same. If not, then $\operatorname{dist}_T(\pi^{-1}(i), i) = \operatorname{dist}_T(\pi^{-1}(j), j) = d$, so i and $\pi^{-1}(i)$ must be in distinct maximum depth central subtrees of T, as must j and $\pi^{-1}(j)$. But i and j are also in distinct central subtrees, so i and $\pi^{-1}(j)$ must be in the same subtree, as must j and $\pi^{-1}(i)$. So in $\lfloor \frac{d-1}{2} \rfloor$ moves each, we can move markers i and j to the roots of their respective trees. Then, in 1 move if the diameter is odd, and 3 if it is even, we can swap i and j, since they lie at the ends of a path of length 1 or 2. Then we can move them each another $\lfloor \frac{d-1}{2} \rfloor$ moves to get home. In either case, this takes $2d-1$ moves. Then we can home the rest of the markers in $\alpha(T - \{i, j\})$ moves, so we are done.

Now suppose α takes branch (b), removing i. Then we can move i's marker home in d moves, since $\operatorname{dist}_T(\pi^{-1}(i), i) \leq \operatorname{diam} T = d$. Again, we can home the remaining markers in $\alpha(T - \{i\})$ moves.

Finally, if α takes branch (c), we use a method similar to Ganesan's [1]. Suppose α attains the minimum with some vertex i. If $\operatorname{dist}_T(i, \pi^{-1}(i)) < d$, we home marker i in at most $d-1$ steps, then home marker j in at most d steps, for any j maximally separated from i. This takes at most $2d-1$ steps. Otherwise, $\operatorname{dist}_T(i, \pi^{-1}(i)) = d$. In this case, we first home marker $j = \pi^{-1}(i)$ in at most d steps. The last of these must exchange marker i (sitting at $\pi^{-1}(i)$) so that it is now on the (unique) vertex adjacent to $\pi^{-1}(i)$. Then it takes at most $d-1$ more steps to move back to i, for a total of $2d-1$ steps. Then for any i and some j, we can home i and j in $2d-1$ steps and home everything else in $\alpha(T - \{i, j\})$ moves, as desired.

Since we have proven that each step of α produces an upper bound, we are done, and $\operatorname{diam}\Gamma \leq \alpha(T)$.

For the second inequality, observe that the third step of α is a slightly stronger version of Ganesan's β, and α always takes the step which will produce the minimal possible value, $\alpha(T)$ is necessarily less than or equal to $\beta(T)$; in the worst case, α just does exactly what β does. Then $\alpha(T) \leq \beta_{\max}(T)$. □

4. Algorithm η

4.1. Motivation

Algorithm η incorporates a new step which removes several leaves at once, to compute a bound very quickly. It gives an acceptable bound, and does so very quickly, without backtracking on different orders of removing vertices; thus, it can be used even on very large trees.

4.2. Definition

Let $S(T)$ be the set of vertices maximally distant from the center. (If $|C(T)| = 2$, measure the distance to the nearer center.) Again, let $\eta(T) = 0$ if T has 0 or 1 vertices, and define η recursively as follows:

(a) If there exists a vertex i of T such that some other vertex j of T is the unique vertex at maximum distance from i, then let $\eta(T) = d + \eta(T - \{j\})$.

(b) Otherwise, let $\eta(T) = d|S(T)| - \left\lceil \frac{|S(T)|}{2} \right\rceil + \eta(T - S(T))$ as in Algorithm η.

Theorem 4. $\operatorname{diam}\Gamma \leq \eta(T)$.

Proof. We proceed by induction on n; with 0 or 1 vertices, the theorem is trivially satisfied. Suppose the markers are permuted by π, so that marker i is sitting on $\pi^{-1}(i)$, and let $|S(T)| = m$. First, home all markers in $S(T)$

which are not currently sitting on vertices in $S(T)$. Suppose k markers remain unhomed after this is complete; each must be sitting on a vertex in $S(T)$. Since each of the first $m-k$ markers was not on a vertex in $S(T)$, it must have taken at most $d-1$ moves, so we have used at most $(m-k)(d-1)$ moves so far. Now pick one remaining marker at a distance d from home, and call it i; it may be moved home in at most d moves. Then as in branch (c) of Algorithm α, we have already moved the marker that was on i once, so it may be homed in at most $d-1$ moves, and so on for all markers in the same cycle as i. We then repeat by picking a new marker, and so on. This takes $k(d-1)+c(\pi)$ moves, where $c(\pi)$ is the number of cycles of π. But $c(\pi) \leq \left\lfloor \frac{k}{2} \right\rfloor$, so in total, we have made at most

$$(m-k)(d-1)+k(d-1)+\left\lfloor \frac{k}{2} \right\rfloor = (d-1)m+\left\lfloor \frac{k}{2} \right\rfloor$$

moves. Then since $k \leq m$, we have

$$(d-1)m+\left\lfloor \frac{k}{2} \right\rfloor \leq (d-1)m+\left\lfloor \frac{m}{2} \right\rfloor = dm-\left\lceil \frac{m}{2} \right\rceil,$$

so all markers corresponding to vertices in $S(T)$ can be homed in $dm-\left\lceil \frac{m}{2} \right\rceil$ moves. The remaining vertices can be homed in $\eta(T-S(T))$ moves by induction, so all markers can be homed in $\eta(T)$ moves.

If instead we used step (b), we proceed as in the proof of Algorithm α. We can home any single marker in d moves, and then home the rest in $\eta(T-\{i\})$ moves, so by induction we are done, and η is a bound for $\operatorname{diam}\Gamma$. □

5. Algorithm ζ

5.1. Motivation

Algorithm ζ represents a simple "naïve approach" against which the speed and bounds of other algorithms can be compared. Since some of the steps of Algorithm α take a minimum, instead of a maximum, each individual value computed by those steps, before taking a minimum, is also an upper bound for $\operatorname{diam}\Gamma$. Thus, if we do not require as good a bound, we may simply arbitrarily pick vertices to remove, and obtain a weaker bound much more quickly.

5.2. Definition

As with α, we define ζ recursively. Again let $L(T)$ be the leaves of T.

Algorithm 3. *Let $\zeta(T)$ be any of the values $d + \zeta(T - \{i\})$, where $i \in L(T)$. Again, $\zeta(T)$ is zero if T has order zero or one. As with β, ζ is nondeterministic, but we can compute an arbitrary value of ζ by choosing a leaf randomly, or compute the minimum or maximum value of ζ by taking a minimum or maximum over i at each step.*

Theorem 5. $\operatorname{diam}\Gamma \leq \zeta_{\min}(T)$.

Proof. We prove this exactly as we proved branch (b) of Algorithm α. We induct on n. Suppose that in computing some particular value of $\zeta(T)$, we remove vertex i. Then we can move marker i home to vertex i in d steps. We then recurse, and see that we can home all remaining markers in $\zeta(T - \{i\})$ steps, and $\zeta(T)$ is an upper bound for $\operatorname{diam}\Gamma$. Then any $\zeta(T)$, and thus $\zeta_{\min}(T)$ is an upper bound for $\operatorname{diam}\Gamma$. □

6. Relative performance

6.1. Tightness of bound

In general, computing the exact diameters of Cayley graphs is computationally very hard. Thus, we do not know the absolute error of any of the bounds in general; they are certainly not always exact. However, we can compare the bounds computed by the different algorithms to each other.

In Table 2, we show the average bounds computed by each algorithm. For each n tested, 100 random (labelled) trees were generated, then each algorithm was run on each, and the results averaged. Algorithms too slow to run were omitted. "Any" means that a value of β or ζ was chosen arbitrarily; note that the values computed for "Any β" are not known to be upper bounds on $\operatorname{diam}\Gamma$.

Now, we move on to general relations between the algorithm. First, we prove that α always produces the best bound of the algorithms discussed.

Proposition 3. *$\alpha(T) \leq \beta_{\max}(T), \zeta(T), \eta(T)$, that is, α produces a better bound than $\beta_{\max}$, ζ, or η.*

Proof. We induct on n. Clearly when n is 0 or 1 we are done, since each bound is zero. Now suppose that $\alpha(T') \leq \beta_{\max}(T'), \zeta(T'), \eta(T')$ for trees T' with fewer than n vertices. Then let T have n vertices, and observe:

- $\alpha(T - \{i, j\}) \leq \beta_{\max}(T - \{i, j\})$ for all i and j, so option (c) of α is always at most $\beta_{\max}(T)$, so $\alpha(T) \leq \beta_{\max}(T)$.
- Similarly, $\alpha(T - \{i\}) \leq \zeta(T - \{i\})$ for all i, so option (b) of α is always at most $\zeta(T)$, so $\alpha(T) \leq \zeta(T)$.
- Finally, any vertex removed by option (c) of α is in $S(T)$, so an application of option (b) of η is equivalent to several applications of option (c) of α. And again, option (a) of η is equivalent to option (b) of α. Then by the same inductive method, $\alpha(T - S(T)) \leq \eta(T - S(T))$, and $\alpha(T - \{i\}) \leq \eta(T - \{i\})$, so $\alpha(T) \leq \eta(T)$.

Then by induction the proposition holds for all n. □

However, $\zeta_{\min}$ cannot do much worse than $\alpha(T)$.

Proposition 4. $\alpha(T) \leq \zeta_{\min}(T) \leq \alpha(T) + \frac{n}{2}$.

Proof. We have already shown the first half of the inequality (in Proposition 3). We induct on n to show the second half. If n is 0 or 1, $\alpha(T) = \zeta_{\min}(T) = 0$, so we are done. Finally, we always have that $\alpha(T) = 2d + \alpha(T - \{i, j\})$ or $\alpha(T) = 2d - 1 + \alpha(T - \{i, j\})$ for some i and j, and similarly for $\zeta_{\min}$, so $\alpha(T) - \zeta_{\min}(T)$ can differ by at most 1 from $\alpha(T - \{i, j\}) - \zeta_{\min}(T - \{i, j\})$ for some i and j. Now $\zeta_{\min}(T - \{i, j\}) \leq \alpha(T - \{i, j\}) + \frac{n-2}{2}$ by induction, so $\zeta_{\min}(T) \leq \alpha(T) + \frac{n}{2}$. □

Finally, we note that the three bounds can be quite far apart in some cases.

Proposition 5. *There exist transposition trees T for which $\alpha(T), \zeta_{\min}(T) \leq \beta_{\min}(T), \beta_{\max}(T) \leq \zeta_{\max}(T)$ by $\Omega(n^2)$ in each case.*

n	α	$\beta_{\max}$	Any β	$\zeta_{\min}$	Any ζ	η
10	26.11	27.48	27.48	27.51	34.67	28.25
15	50.68	53.6	53.6	52.65	71.6	55.06
20	77.83	81.02	81.02	80.28	116.2	84.14
30		144.2	144.2		219.55	149.73
40		209.12	209.04		333.81	218.3
50			288.36		470.71	299.4
75			492.58		844.26	515.9
100			710.3		1257.45	741.7

Table 2: In this table we show the average strength of the bounds provided by each algorithm; recall that arbitrary β are not known to give a bound.

Proof. For $T = B_{n,n}$, defined in Appendix Appendix A, the values each algorithm produces are described there. The tree T has $O(n)$ vertices, but the given bounds differ by $\Omega(n^2)$. (In this case, the value of β is unique.) □

In addition, Algorithm η can also do $\Omega(n^2)$ worse than α, on the graph $B_{m,m}$ with two extra leaves adjacent to vertex $2m$ (the "handle" of the broom); we omit the computation of its values under the various algorithms. In practice, it seems to do better than $\zeta_{\max}$ or β, though.

The minimum value produced by Algorithm ζ can do at most linearly worse than $\alpha(T)$. In particular, whenever α takes branch (a) or (c) of its minimum, ζ can remove the same vertices, adding at most $2d$ instead of $2d - 1$. So $\zeta_{\min}(T) - \alpha(T) \leq \frac{n}{2}$.

6.2. Computational complexity

Computing a single value produced by algorithms β, ζ, and η is fairly fast. Each step of the algorithm takes polynomial time in the number of vertices, and the recursion takes at most n steps, so each is polynomial. However, computing all values of β (in order to obtain $\beta_{\max}$, which is known to be a bound) or ζ requires recursing on all possible orders of removing vertices. This can take superexponential (namely $\Omega(n!n^2)$) time in the worst case, although in practice it is generally computable in a reasonable amount of

n	α	$\beta_{\max}$	Any β	$\zeta_{\min}$	Any ζ	η
10	0.204	0.008	0.004	0.065	0.008	0.004
15	4.589	0.036	0.012	1.168	0.024	0.012
20	66.807	0.135	0.023	14.293	0.045	0.017
30		1.369	0.068		0.129	0.035
40		11.912	0.162		0.306	0.069
50			0.209		0.413	0.082
75			0.784		1.500	0.237
100			1.747		3.447	0.418
500						14.013

Table 3: Running times (in seconds) for each algorithm on trees of different sizes; recall that arbitrary values of β are not known to be bounds on $\operatorname{diam}\Gamma$ at all.

time. Similarly, Algorithm α can take superexponential time; in practice it is worse than β but still tractable for small trees. Thus, for computationally easy bounds, η is the best choice, but each of the algorithms is tractable for small trees.

Table 3 gives a practical idea of the time taken by each algorithm. For each n tested, 100 random (labelled) trees were generated, then each algorithm was timed on each independently, and the results averaged. (All computations were performed in *Mathematica*.) If the algorithm was too slow to run, its result is omitted. "Any" means that a value of β or ζ was chosen arbitrarily; note that the values computed for "Any β" are not known to be upper bounds on $\operatorname{diam}\Gamma$.

7. Conclusion

In summary, we found algorithms that compute upper bounds for the diameter of groups generated by transposition trees. Algorithm α is somewhat more difficult to compute than the previously known algorithm β, but produces stronger bounds in most cases, while algorithm η produces weaker bounds than β, but does so much more quickly.

Of course, even Algorithm α rarely returns the exact diameter of the group.

It appears that it may be difficult to go further without iterating over specific permutations, but it could be productive, for example, to look at how to do so efficiently, and thus produce a better bound which is still computationally easier than constructing the entire Cayley graph to compute the diameter. Alternately, it would be useful to have heuristics for use with Algorithm α – it might be possible to compute a bound which is marginally worse in most cases, and much easier to compute, by making an educated guess rather than recursing over every possible subtree. In addition, the properties of any of the algorithms could be interesting to explore. How often do α and η use each of their respective options? On which trees does each algorithm do worst? Finally, it may be possible to extend these results to transposition graphs which are not trees, or even to generating sets with elements of order greater than 2, and ideally even to find bounds on the diameter of any Cayley graph of S_n.

8. Acknowledgments

This research was done at the University of Minnesota Duluth REU, funded by NSF/DMS grant 1062709 and NSA grant H98230-11-1-0224. It would have been impossible without Joe Gallian, the director of the REU, and Adam Hesterberg, Davie Rolnick, and Eric Riedl, the program advisers. I would also like to thank Mike Develin and Gaku Liu for reading drafts of the paper, and David Moulton for suggesting some of the example graphs.

Appendix A. Examples of algorithms

Appendix A.1. Summary of examples

$\boldsymbol{P_n}$
$\alpha = \binom{n}{2}$ $\beta = \binom{n}{2}$ $\zeta = \binom{n}{2}$ $\eta = \binom{n}{2}$
$\boldsymbol{S_{n,k}}$ **(n even)**
$\alpha = \frac{n}{2}k(2k+1)$ $\eta = \frac{n}{2}k(2k+1)$
$\boldsymbol{S_{n,k}}$ **(n odd)**
$\alpha = \frac{n}{2}k(2k+1) - \frac{k}{2}$ $\eta = \frac{n}{2}k(2k+1) - \frac{k}{2}$
$\boldsymbol{B_{n,k}}$ **($n \geq k \geq 2$)**

$\alpha = \binom{n+1}{2} + \lfloor \frac{3}{2}(k-1) \rfloor \qquad \beta = n^2 - \binom{n-k+2}{2}$
$\zeta \in \{\binom{n+1}{2} + 2(k-1), \ldots, \binom{n+1}{2} + n(k-1)\}$
$\boldsymbol{B_{n,k}}$ $\boldsymbol{(k \geq n \geq 2)}$
$\alpha = \binom{n+1}{2} + \lfloor \frac{3}{2}(k-1) \rfloor \qquad \beta = n^2 - 1 + \lceil \frac{3}{2}(k-n+1) \rceil$
$\zeta \in \{\binom{n+1}{2} + 2(k-1), \ldots, \binom{n+1}{2} + n(k-1)\}$

Appendix A.2. Path graph $\boldsymbol{P_n}$

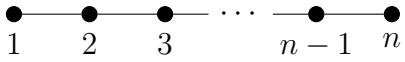

Figure A.4: The path graph on n vertices, P_n.

The diameter of the group generated by a path graph (shown in Figure A.4) is well-known to be $\binom{n}{2}$. $\alpha(P_n) = \beta(P_n) = \zeta(P_n) = \eta(P_n) = \binom{n}{2}$ as well; on this simple graph all of the algorithms do well. We can see this by induction: when n is 0 or 1, each algorithm returns $0 = \binom{0}{2} = \binom{1}{2}$, and then for some arbitrary n, we have, since removing either end of the path yields the same graph,

$$\begin{aligned}
\alpha(P_n) &= \min\left(2n-1+\alpha(P_{n-2}), n+\alpha(P_{n-1}), 2n-1+\alpha(P_{n-2})\right) \\
&= \min\left(2n-1+\binom{n-2}{2}, n+\binom{n-1}{2}, 2n-1+\binom{n-2}{2}\right) \\
\alpha(P_n) &= \binom{n}{2}.
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
\beta(P_n) &= 2n-1+\beta(P_{n-2}) = 2n-1+\binom{n-2}{2} = \binom{n}{2} \\
\zeta(P_n) &= n+\zeta(P_{n-1}) = n+\binom{n-1}{2} = \binom{n}{2} \\
\eta(P_n) &= n+\eta(P_{n-1}) = n+\binom{n-1}{2} = \binom{n}{2}
\end{aligned}$$

where η follows option (a).

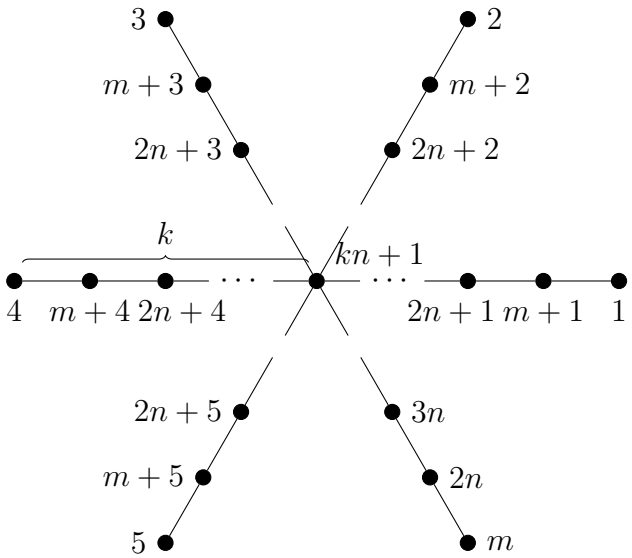

Figure A.5: The generalized star graph $S_{m,k}$.

Appendix A.3. Generalized star graph $S_{m,k}$

In the generalized star graph shown in Figure A.5, the diameter of the transposition tree is $2k$.

Proposition 6. *If m is even, $\alpha(S_{m,k}) = \eta(S_{m,k}) = \frac{m}{2}k(2k+1)$, and if m is odd, $\alpha(S_{m,k}) = \eta(S_{m,k}) = \frac{m}{2}k(2k+1) - \frac{k}{2}$.*

Proof. The value of α can be computed by induction, but it is ugly and not particularly instructive to do so, so we omit the proof. In fact, by induction on k we can see that η returns the same result. Clearly the results hold when $k = 0$. If m is even, all leaves are in the set $S(S_{m,k})$ defined in Section 4, so

$$\eta(S_{m,k}) = 2kn - \frac{m}{2} + \eta(S_{m,k-1}) = 2kn - \frac{m}{2} + \frac{m}{2}(k-1)(2k-1) = \frac{m}{2}k(2k+1)$$

as desired, and similarly in the odd case. This is, in some sense, the best-case graph for η – it does just as well as the other algorithms, and is extremely easy to compute. (We omit the computation of ζ and β as both are long but uninteresting.) □

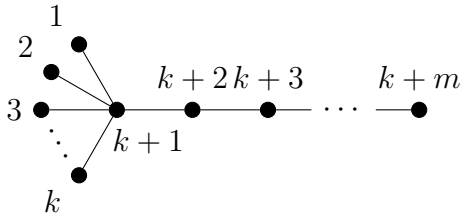

Figure A.6: The broom graph $B_{m,k}$

Appendix A.4. Broom graph $\boldsymbol{B_{m,k}}$

We define the broom graph on m vertices as shown in Figure A.6. It is an intermediate case between common trees, and thus some of the algorithms do poorly on it.

First, consider $m = 1$, i.e., a star graph on $k + 1$ vertices. Here $\beta_{\max} = \operatorname{diam}\Gamma = \lfloor \frac{3}{2}k \rfloor$, so since $\operatorname{diam}\Gamma \leq \alpha(T) \leq \beta_{\max}(T)$, $\alpha = \lfloor \frac{3}{2}k \rfloor$ too.

Otherwise, $m \geq 2$ and we have the following proposition.

Proposition 7. *If $m \geq 2$, then $\alpha(B_{m,k}) = \binom{m+1}{2} + \lfloor \frac{3}{2}(k-1) \rfloor$.*

Proof. We prove this by induction on $m + k$.

For the base case, let $m = 2$, and we have a star graph on $k + 2$ vertices. Then $\alpha(B_{2,k}) = \lfloor \frac{3}{2}(k+1) \rfloor = \lfloor \frac{3}{2}(k-1) \rfloor + 3$ as desired, since $B_{2,k} = B_{1,k+1}$. Furthermore, if $k = 1$, we have a path graph $B_{m,1} = P_{m+1}$ which has $\alpha(P_{m+1}) = \binom{m+1}{2}$ as desired.

Now, for the inductive step, let $m > 2$, $k > 1$, and suppose we have already shown the claim for all smaller values of $m + k$. Then

$$\begin{aligned}\alpha(B_{m,k}) &= \min(2n - 1 + \alpha(B_{m-1,k-1}), m + \alpha(B_{m-1,k}), m + \alpha(B_{m,k-1}), 2n - 1 + \alpha(B_{m-1,k-1})) \\ &= \min\left(2n - 1 + \binom{m}{2} + \left\lfloor \frac{3}{2}(k-2) \right\rfloor, \right. \\ &\qquad \left. m + \binom{m}{2} + \left\lfloor \frac{3}{2}(k-1) \right\rfloor, m + \binom{m+1}{2} + \left\lfloor \frac{3}{2}(k-2) \right\rfloor \right)\end{aligned}$$

$$= m + \binom{m}{2} + \left\lfloor \frac{3}{2}(k-1) \right\rfloor \qquad \text{(for } m > 2\text{)}$$
$$= \binom{m+1}{2} + \left\lfloor \frac{3}{2}(k-1) \right\rfloor$$

as desired. □

It is easy to compute β and ζ by a similar method; we omit the computation but state the results.

Proposition 8. *If $m \geq 2$, the value of $\beta(B_{m,k})$ is uniquely $m^2 - \binom{m-k+1}{2}$ when $m \geq k$ and $m^2 - 1 + \lfloor \frac{3}{2}(k-m+1) \rfloor$ when $m \leq k$.*

On the other hand, $\zeta(B_{m,k})$ can take on any integral value from $\binom{m+1}{2} + 2(k-1)$ to $\binom{m+1}{2} + m(k-1)$, for $k \geq 2$.

[1] A. Ganesan, Diameter of Cayley graphs of permutation groups generated by transposition trees, ArXiv e-printsarXiv:1111.3114.

[2] L. Babai, G. L. Hetyei, On the diameter of random cayley graphs of the symmetric group, Combinatorics, Probability and Computing 1 (03) (1992) 201–208. doi:10.1017/S0963548300000237.

[3] L. Babai, A. Seress, On the diameter of cayley graphs of the symmetric group, J. Comb. Theory Ser. A 49 (1) (1988) 175–179. doi:10.1016/0097-3165(88)90033-7.

[4] L. Babai, A. Seress, On the diameter of permutation groups, European Journal of Combinatorics 13 (4) (1992) 231 – 243. doi:10.1016/S0195-6698(05)80029-0.

[5] H. A. Helfgott, A. Seress, On the diameter of permutation groups, ArXiv e-printsarXiv:1109.3550.

[6] J. Bamberg, N. Gill, T. Hayes, H. Helfgott, Á. Seress, P. Spiga, Bounds on the diameter of Cayley graphs of the symmetric group, ArXiv e-printsarXiv:1205.1596.

[7] T. Rokicki, H. Kociemba, M. Davidson, J. Dethridge, God's number is 20 (2010).
URL http://www.cube20.org/

[8] S. B. Akers, B. Krishnamurthy, A group-theoretic model for symmetric interconnection networks, Computers, IEEE Transactions on 38 (4) (1989) 555–566. doi:10.1109/12.21148.

[9] R. Otter, The number of trees, The Annals of Mathematics 49 (3) (1948) pp. 583–599.
URL http://www.jstor.org/stable/1969046