THEORY OF COMPUTING, Volume 20 (7), 2024, pp. 1–62
www.theoryofcomputing.org

Lower Bound Techniques in the Comparison-Query Model and Inversion Minimization on Trees

Ivan Hu Dieter van Melkebeek Andrew Morgan

Received March 18, 2023; Revised July 1, 2024; Published December 7, 2024

Abstract. Given a rooted tree and a ranking of its leaves, what is the minimum number of inversions of the leaves that can be attained by ordering the tree? This variation of the well-known problem of counting inversions in arrays originated in mathematical psychology. It has the evaluation of the Mann–Whitney statistic for detecting differences between distributions as a special case.

We study the complexity of the problem in the comparison-query model, the standard model for problems like sorting, selection, and heap construction. The complexity depends heavily on the shape of the tree: for trees of unit depth, the problem is trivial; for many other shapes, we establish lower bounds close to the strongest known in the model, namely the lower bound of $\log_2(n!)$ for sorting n items. For trees with n leaves we show, in increasing order of closeness to the sorting lower bound:

A conference version of this paper appeared in the Proceedings of the 34th ACM-SIAM Symposium on Discrete Algorithms (SODA'23) [14].

ACM Classification: Theory of computation → Computational complexity and cryptography

AMS Classification: 68Q17, 68Q25

Key words and phrases: query complexity, comparison-query model, lower bounds, permutahedron, sensitivity, connectivity, inversions, trees, Mann–Whitney, Gaussian binomial coefficients, Gaussian polynomials

 DOI: 10.4086/toc.2024.v020a007

(a) $\log_2((\alpha(1-\alpha)n)!) - O(\log n)$ queries are needed whenever the tree has a subtree that contains a fraction α of the leaves. This implies a lower bound of $\log_2((\frac{k}{(k+1)^2}n)!) - O(\log n)$ for trees of degree k.

(b) $\log_2(n!) - O(\log n)$ queries are needed in case the tree is binary.

(c) $\log_2(n!) - O(k \log k)$ queries are needed for certain classes of trees of degree k, including perfect trees with even k.

The lower bounds are obtained by developing two new techniques for any problem Π in the comparison-query model and applying them to inversion minimization on trees. Both techniques can be described in terms of the Cayley graph of the symmetric group with adjacent-rank transpositions as the generating set, or equivalently, in terms of the edge graph of the permutahedron, the polytope spanned by all permutations of the vector $(1, 2, \ldots, n)$. Consider the subgraph consisting of the edges between vertices with the same value under Π. We show that the size of any decision tree for Π must be at least

(i) the number of connected components of this subgraph, and

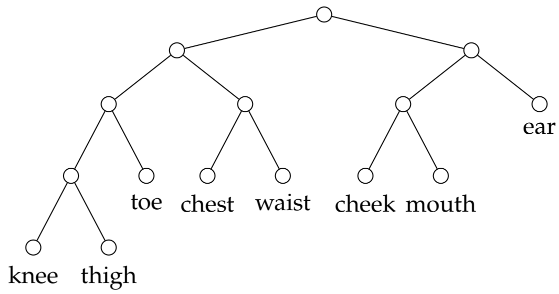
(ii) the factorial (or Γ function) of the average degree of the complementary subgraph, divided by n.

Lower bounds on query complexity then follow by taking the base-2 logarithm. Technique (i) represents a discrete analog of a classical technique in algebraic complexity and allows us to establish (c) and a tight lower bound for counting cross inversions, as well as unify several of the known lower bounds in the comparison-query model. Technique (ii) represents an analog of sensitivity arguments in Boolean complexity and allows us to establish (a) and (b).

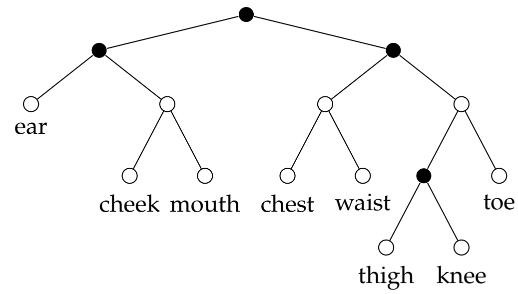
Along the way to proving (b), we derive a tight upper bound on the maximum probability of the distribution of cross inversions, which is the distribution of the Mann–Whitney statistic in the case of the null hypothesis. Up to normalization, the probabilities appear in the literature as the coefficients of polynomials formed by the Gaussian binomial coefficients, also known as Gaussian polynomials.

1 Overview

The result of a hierarchical cluster analysis on a set X of items can be thought of as an unordered rooted tree T with leaf set X. To visualize the tree, or to spell out the classification in text, one needs to decide for every internal node of T in which order to visit its children. Figure 1a represents an example of a classification of eight body parts from the psychology literature [6]. It is obtained by repeatedly clustering nearest neighbors where the distance between two items is given by the number of people in a survey who put the items into different classes [22]. The ordering of the resulting binary tree in Figure 1a is the output produced by a particular implementation of the clustering algorithm.



(a) Initial tree ordering



(b) Optimal tree ordering

Figure 1: Classification of body parts

Another ordering is given in Figure 1b; black marks the nodes whose children have been swapped from the ordering in Figure 1a. Figure 1b has the advantage over Figure 1a that the leaves now appear in an interesting global order, namely head-to-toe: ear, cheek, mouth, chest, waist, thigh, knee, toe. Indeed, Figure 1b makes apparent that the anatomical order correlates perfectly with the clustering. In general, given a tree T and a ranking ρ of its leaves, one might ask "how correlated" is T with ρ? Degerman [6] suggests evaluating the orderings of T in terms of the number of inversions of the left-to-right ranking σ of the leaves with respect to the given ranking ρ, and use the minimum number over all orderings as a measure of (non)correlation. For further reference, we collect the formal definitions of the underlying notions.

Definition 1.1 (tree ordering, ranking, inversion, $\mathrm{Inv}_{\cdot}(\cdot)$)**.** An *ordering* of a rooted tree T consists of an ordering, for every non-leaf v of T, of the children of v. A *ranking* ρ of a set X of n items is a bijection from X to $[n]$. Given two rankings σ and ρ, an *inversion* of σ with respect to ρ is a pair of items $x_1, x_2 \in X$ such that $\rho(x_1) < \rho(x_2)$ but $\sigma(x_1) > \sigma(x_2)$. The number of inversions is denoted by $\mathrm{Inv}_\rho(\sigma)$. An inversion in an array A of distinct values is an inversion of σ with respect to ρ where σ denotes the ranking by array index and ρ the ranking by value; in this setting we write $\mathrm{Inv}(A)$ for $\mathrm{Inv}_\rho(\sigma)$.

The minimum number of inversions over all tree orderings can be used to compare the quality of different trees T for a given ranking ρ, or of different rankings ρ for a given tree T. This mimics the use of the number of inversions in applications like collaborative filtering in recommender systems, rank aggregation for meta searching the web, and Kendall's test for dependencies between two random variables. In particular, the Mann–Whitney test for differences between random variables can be viewed as a special case of our optimization problem. The test is widely used because of its nonparametric nature, meaning that no assumptions need to be made about the distribution of the two variables; the distribution of the statistic in the case of the null hypothesis (both variables have the same distribution) is always the same. The test achieves this property by only considering the relative order of the samples. It takes a sequence A of a samples from a random variable Y, a sequence B of b samples from another random variable Z, and computes the statistic $U \doteq \min(\mathrm{XInv}(A, B), \mathrm{XInv}(B, A))$ that is the minimum of the number $\mathrm{XInv}(A, B)$ of cross inversions from A to B, and vice versa, defined as follows.

Definition 1.2 (cross inversions, $\mathrm{XInv}_{\cdot}(\cdot,\cdot)$)**.** Let ρ be a ranking of X, and $A, B \subseteq X$. A *cross inversion* from A to B with respect to ρ is a pair $(x_1, x_2) \in A \times B$ that is out of order with respect to ρ, i. e., such that $\rho(x_1) > \rho(x_2)$. The number of cross inversions is denoted by $\mathrm{XInv}_\rho(A, B)$. For two arrays A and B of distinct values, a cross inversion from A to B is a cross inversion from the set of entries in A to the set of entries in B where ρ denotes the ranking by value; in this setting we write $\mathrm{XInv}(A, B)$ for $\mathrm{XInv}_\rho(A, B)$.

The statistic U coincides with the optimum value of our optimization problem with the tree T in Figure 2 as input. The leftmost a leaves correspond to the samples A, the rightmost b leaves to the samples B, and the ranking ρ to the value order of the combined $a + b$ samples.

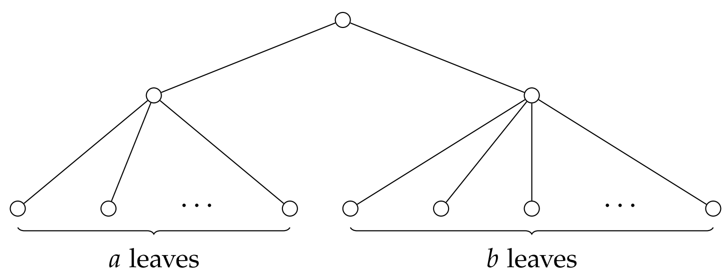


Figure 2: Mann–Whitney instance

We mainly study the value version of our optimization problem, which we denote by MinInv.

Definition 1.3 (inversion minimization on trees, $\mathrm{MinInv}(\cdot,\cdot)$, $\Pi_{\cdot}$)**.** Inversion minimization on trees is the computational problem with the following specification:

Input: A rooted tree T with leaf set X of size n, and a ranking ρ of X.

Output: $\mathrm{MinInv}(T, \rho)$, the minimum of $\mathrm{Inv}_\rho(\sigma)$ over all possible orderings of T, where σ denotes the left-to-right ranking of X induced by the ordering of T.

For any fixed tree T with leaf set X, we use the short-hand Π_T to denote the computational problem that takes as input a ranking ρ of X and outputs $\mathrm{MinInv}(T, \rho)$.

Degerman [6] observes that the ordering at each internal node can be optimized independently in a greedy fashion. In the setting of binary trees, for each node v, we can count the cross inversions from the leaves in the left subtree of v to the leaves in the right subtree of v. Between the two possible orderings of the children of a node v, we choose the one that yields the smaller number of cross inversions. Based on his observation, Degerman presents a polynomial-time algorithm for the case of binary trees T. A more refined implementation and analysis yields a running time of $O(d_{\mathrm{avg}}(T) \cdot n)$, where $d_{\mathrm{avg}}(T)$ denotes the average depth of a leaf in T. For balanced binary trees the running time becomes $O(n \log n)$. All of this can be viewed as variants of the well-known $O(n \log n)$ divide-and-conquer algorithm for counting inversions in arrays of length n.

For trees of degree $\deg(T) > 2$, the local greedy optimization at each internal node becomes more complicated, as there are many ways to order the children of each internal node. Exhaustive search results in a running time of $O((\deg(T)! + \deg(T) \cdot d_{\text{avg}}(T)) \cdot n)$, which can be improved to $O((\deg(T)^2 2^{\deg(T)} + \deg(T) \cdot d_{\text{avg}}(T)) \cdot n)$ using dynamic programming. The problem is closely related to the classical problem of minimum arc feedback set, and becomes NP-hard without any constraints on the degree. We refer to Section 10 for more details.

Query complexity. Rather than running time in the Turing machine model, our focus lies on query complexity in the comparison-query model. There we can only access the ranking $\rho : X \to [n]$ via queries of the form: Is $\rho(x_1) < \rho(x_2)$? For any fixed tree T, we want to determine the minimum number of queries needed to solve the problem.

The comparison-query model represents the standard model for analyzing problems like sorting, selection, and heap construction. Sorting represents the hardest problem in the comparison-query model as it is tantamount to knowing the entire ranking ρ. Its query complexity has a well-known information-theoretic lower bound of $\log_2(n!) = n \log_2(n/\mathrm{e}) + \frac{1}{2}\log_2(n) + O(1)$. Standard algorithms such as mergesort and heapsort yield an upper bound of $\log_2(n!) + O(n)$, which has been improved to $\log_2(n!) + o(n)$ recently [26]. We refer to Section 2 for an overview of results and techniques for lower bounds in the model.

Information theory only yields a very weak lower bound on the query complexity of inversion minimization on trees: $\log_2 \binom{n}{2} = 2\log_2(n) - O(1)$. The complexity of the problem critically depends on the shape of the tree T and can be significantly lower than the one for sorting. For starters, the problem becomes trivial for trees of depth one as their leaves can be arranged freely in any order. More precisely, the trees T for which the answer is identically zero, irrespective of the ranking ρ, are exactly those such that all root-to-leaf paths have only the root in common.

Arguably, the simplest nontrivial instances of inversion minimization are for trees T of the Mann–Whitney type in Figure 2 with $a = 1$ and $b = n - 1$. Depending on the rank r of the isolated leaf, an optimal ordering of T is either the left or the right part in Figure 3, where the label of each leaf is its rank under ρ.

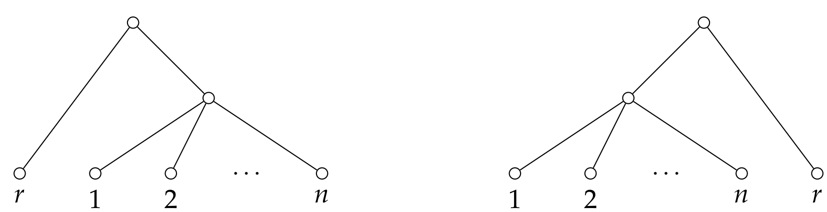


Figure 3: Rank instance

As the ordering on the left has $r - 1$ inversions and the one on the right $n - r$, the answer is $\min(r - 1, n - r)$. Thus, this instance of inversion minimization on trees is essentially equivalent to rank finding, which has query complexity exactly $n - 1$.

1.1 Main results

We prove that for many trees T, inversion minimization on T is nearly as hard as sorting. First, we exhibit a common structure that guarantees high complexity, namely a subtree that contains a fairly balanced fraction of the leaves. We make use of the following notation.

Definition 1.4 (leaf set, $\mathrm{L}(\cdot)$, and subtree)**.** For a tree T, the leaf set of T, denoted $\mathrm{L}(T)$, is the set of leaves of T. For a node v to T, T_v denotes the subtree of T rooted at v.

The quantitative statement references the gamma function Γ, which is a proxy for any convex real function that interpolates the factorial function on the positive integers. More precisely, we have that $\Gamma(n+1) = n!$ for every integer $n \geq 1$.

Theorem 1.5 (lower bound for general trees)**.** *Let T be a tree with n leaves, and v a node with $|\mathrm{L}(T_v)| = \ell$. The query complexity of inversion minimization on T is at least $\log_2(\Gamma(\frac{\ell(n-\ell)}{n} + 1))$. In particular, the complexity is at least $\log_2(\Gamma(\frac{k}{(k+1)^2} \cdot n + 1))$ where k denotes the degree of T.*

For trees of constant degree, Theorem 1.5 yields a lower bound that is as strong as the one for sorting up to a constant multiplicative factor. For the important case of binary trees (like the classification trees from the motivating example), we obtain a lower bound that is only a logarithmic additive term shy of the lower bound for sorting.

Theorem 1.6 (lower bound for binary trees)**.** *For binary trees T with n leaves, the query complexity of inversion minimization on T is at least $\log_2(n!) - O(\log n)$.*

The logarithmic loss can be reduced to a constant for certain restricted classes of trees. The full statement is somewhat technical. First, it assumes that the tree has no nodes of degree 1. This is without loss of generality, as we can short-cut all degree-1 nodes in the tree without affecting the minimum number of inversions. For example, trivial trees for inversion minimization have depth 1 without loss of generality. Second, the strength of the lower bound depends on the maximum size of a leaf child set, defined as follows.

Definition 1.7 (leaf child set, $\mathrm{LC}(\cdot)$)**.** The *leaf child set* $\mathrm{LC}(v)$ of a vertex v in a tree T is the set $\mathrm{LC}(v)$ of all the children of v that are leaves in T.

Most importantly, the result requires certain fragile parity conditions to hold. That said, there are interesting classes satisfying all requirements, and the bounds are very tight.

Theorem 1.8 (lower bound for restricted classes)**.** *Let T be a tree without nodes of degree 1 such that the leaf child sets have size at most k, at most one of them is odd, and if there exists an odd one, say $\mathrm{LC}(v^*)$, then all ancestors of v^* have empty leaf child sets. The query complexity of inversion minimization on T is at least $\log_2(n!) - O(k \log k)$. In particular, the lower bound applies to:*

- *perfect trees of even degree k, and*
- *full binary ($k = 2$) trees with at most one leaf without a sibling leaf.*

Recall that a tree of degree k is full if every node has degree 0 or k. It is perfect if it is full and all leaves have the same depth.

For the Mann–Whitney statistic, Theorem 1.5 provides an $\Omega(n \log n)$ lower bound for balanced instances, i. e., when a and b are $\Theta(n)$. For unbalanced instances there is a more efficient way to count cross inversions and thus evaluate the statistic: Sort the smaller of the two sides, and then do a binary search for each item of the larger side to find its position within the sorted smaller side so as to determine the number of cross inversions that it contributes. For $a \leq b$ the approach makes $b \log_2(a) + O(a \log a)$ comparisons. We establish a lower bound that shows the approach is optimal up to a constant multiplicative factor.

Theorem 1.9 (lower bound for counting cross inversions)**.** *Counting cross inversions from a set A of size a to a set B of size $b \geq a$ with respect to a ranking ρ of $X \doteq A \sqcup B$ requires $\Omega((a + b)\log(a))$ queries in the comparison-query model, as does inversion minimization on the tree of Figure 2.*

1.2 Techniques

We obtain our results by developing two new query lower bound techniques for problems Π in the comparison-query model, and then instantiating them to the problem Π_T of inversion minimization on a fixed tree T. Although some of our techniques extend to relations, we restrict attention to computational problems Π that are functions, just like the problem Π_T that we focus on.

Definition 1.10 (computational problem and algorithm in the comparison-query model)**.** A computational problem in the comparison-query model is a total function on the rankings of a set X. An algorithm in the comparison-query model can access an input ranking $\rho : X \to [n]$ using comparison queries: For given $x_1, x_2 \in X$, test if $\rho(x_1) < \rho(x_2)$.

Both of our techniques follow the common pattern of lower bounding the number of distinct execution traces that any algorithm for Π needs to have.

Definition 1.11 (execution trace, complexity measures $\mathrm{D}(\cdot)$ and $\mathrm{Q}(\cdot)$)**.** Consider an algorithm A for a problem Π in the comparison-query model. An execution trace of A is the sequence of comparisons that A makes on some input ρ, as well as the outcomes of the comparisons. The complexity $\mathrm{D}(\Pi)$ is the minimum over all possible algorithms for Π of the number of distinct traces the algorithm has over the set of all inputs ρ. The complexity $\mathrm{Q}(\Pi)$ is the minimum, over all possible algorithms for Π of the maximum number of comparisons that the algorithm makes over the set of all inputs ρ.

The complexity measure Q is what we refer to as query complexity. Since the maximum number of queries that an algorithm A makes is at least the base-2 logarithm of the number of execution traces, we have that $\mathrm{Q}(\Pi) \geq \log_2(\mathrm{D}(\Pi))$. Note that, in order to avoid confusion with the tree T specifying an instance of inversion minimization, we refrain from the common terminology of decision trees in the context of the complexity measure D. In those terms, we lower bound the number of leaves of any decision tree for Π, and use the fact that the depth of this binary decision tree is at least the base-2 logarithm of the number of leaves.

Both techniques proceed by considering the effect on the output of perturbations to the input ranking ρ that are hard for queries to observe. More specifically, we consider the following perturbations:

Definition 1.12 (adjacent-rank transposition, affected items)**.** An *adjacent-rank transposition* is a permutation τ of $[n]$ of the form $\tau = (r, r+1)$, where $r \in [n-1]$ and n denotes the number of items. Given τ and a ranking $\rho : X \to [n]$, the *affected items* are the two elements $x \in X$ for which $\tau(\rho(x)) \neq \rho(x)$, i. e., the items with ranks r and $r+1$ under ρ.

As with any permutation of the set of ranks, the effect of τ on a ranking ρ is the ranking $\tau\rho$. Adjacent-rank transpositions are the least noticeable perturbations one can apply to a ranking in the following sense: If two rankings differ by an adjacent-rank transposition, then the only query that distinguishes them is the query that compares the affected items.

Sensitivity. Our first technique turns this observation around to obtain a lower bound on query complexity. We adopt the terminology of sensitivity from Boolean query complexity.

Definition 1.13 (sensitivity, average sensitivity, $s(\cdot)$)**.** Let Π be a computational problem in the comparison-query model on a set X of items. For a fixed ranking ρ and adjacent-rank transposition τ, we say that Π is *sensitive* to τ at ρ if $\Pi(\rho) \neq \Pi(\tau\rho)$. The sensitivity of Π at ρ is the number of adjacent-rank transpositions τ such that Π is sensitive to τ at ρ. The *average sensitivity* of Π, denoted $s(\Pi)$, is the average sensitivity of Π at ρ when ρ is drawn uniformly at random from all rankings of X.

On input a ranking ρ, any algorithm for Π needs to make a number of queries that is at least the sensitivity of Π at ρ. Indeed, consider an adjacent-rank transposition τ to which Π is sensitive at ρ. If the algorithm does not make the query that compares the affected items, then it must output the same answer on input $\tau\rho$ as on input ρ. Since the value of Π differs on both inputs, this means the algorithm makes a mistake on at least one of the two. It follows that the average number of queries that any algorithm for Π makes is at least the average sensitivity $s(\Pi)$. A fortiori, $\mathrm{Q}(\Pi) \geq s(\Pi)$.

As sensitivity cannot exceed $n-1$, the best lower bound on query complexity that we can establish based on the above basic observation alone, is $n-1$. The following improvement yields a the lower bound $\mathrm{D}(\Pi) \geq n!/n = (n-1)!$, and therefore $\mathrm{Q}(\Pi) \geq \log_2(n!) - \log_2(n)$ for problems Π of maximum average sensitivity $s(\Pi) = n-1$. The argument hinges on an efficient encoding of rankings that share the same execution trace. See Section 3 for more details.

Lemma 1.14 (Sensitivity Lemma)**.** *For any problem Π in the comparison-query model with n items, $\mathrm{D}(\Pi) \geq \Gamma(s(\Pi)+2)/n$.*

The lower bound for general trees T in Theorem 1.5 and the strengthening for binary trees in Theorem 1.6 follow from corresponding lower bounds on the average sensitivity $s(\Pi_T)$. Theorem 1.5 only requires a short analysis to establish the sensitivity lower bound needed for the application of the Sensitivity Lemma; this illustrates the power of the lemma and of the lower bound technique. Theorem 1.6 requires a more involved sensitivity analysis, but then yields a very tight lower bound. Owing to the average-case nature of the underlying measure, the technique also exhibits some degree of robustness. For the particular problem of inversion minimization on trees, we show that small changes to the tree T do not affect the average sensitivity $s(\Pi_T)$ by much. See Section 4 and Section 5.

For sorting, counting inversions, and inversion parity, the average sensitivity reaches its maximum value of $n-1$, and Lemma 1.14 recovers the standard lower bounds up to a small loss. In contrast, for selection, the average sensitivity equals 1 for ranks 1 and n, and 2 for other ranks, so the bound from Lemma 1.14 is no good. This reflects that, just like in the Boolean setting, (average) sensitivity is sometimes too rough of a measure and not always capable of proving strong lower bounds. Our second technique looks at a more delicate structural aspect, which enables it to sometimes yield stronger lower bounds.

Permutahedron graph. Before introducing our second technique, we cast our first technique in graph theoretic terms. In fact, both our techniques can be expressed naturally in subgraphs of the graph with the rankings as vertices and adjacent-rank transpositions as edges. The latter graph can be viewed as the Cayley graph of the symmetric group with adjacent-rank transpositions as the generating set. It is also the edge graph of the permutahedron, the convex polytope spanned by all permutations of the vertex $(1, 2, \ldots, n)$ in $\mathbb{R}^n$. The permutahedron resides inside the hyperplane where the sum of the coordinates equals $\binom{n}{2}$, has positive volume inside that hyperplane, and can thus be represented naturally in dimension $n-1$; see Figure 4 for a rendering of the instance with $n = 4$ [9].

We think of coloring the vertices of the permutahedron with their values under Π and make use of the subgraph with the same vertex set but only containing the monochromatic edges, i. e., the edges whose end points have the same value under Π. We also consider the the complementary subgraph containing all bichromatic edges.

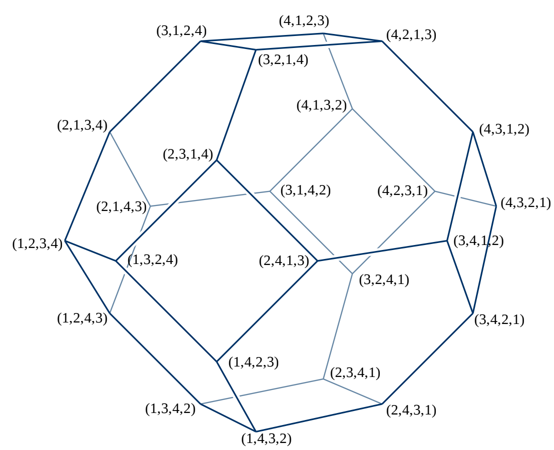


Figure 4: Permutahedron for $n = 4$ items

Definition 1.15 (permutahedron graph, $G(\cdot)$, $\overline{G}(\cdot)$)**.** Let Π be a computational problem in the comparison-query model on a set X of items. The *permutahedron graph* of Π, denoted $G(\Pi)$, has the rankings of X as vertices, and an edge between two rankings ρ_1 and ρ_2 if $\Pi(\rho_1) = \Pi(\rho_2)$ and there exists an adjacent-rank transposition such that $\rho_2 = \tau\rho_1$. The *complementary permutahedron graph* of Π, denoted $\overline{G}(\Pi)$, is defined similarly by replacing the condition $\Pi(\rho_1) = \Pi(\rho_2)$ by its complement, $\Pi(\rho_1) \neq \Pi(\rho_2)$.

Our first technique looks at degrees in the complementary permutahedron graph $\overline{G}(\Pi)$, and more specifically at the *average degree* $\deg_{\text{avg}}(\overline{G}(\Pi)) \doteq \mathbb{E}(\deg_{\overline{G}(\Pi)}(\rho))$, where the expectation is with respect to a uniform choice of the ranking ρ. Our second technique looks at the connected components of the permutahedron graph $G(\Pi)$.

Connectivity. Our second technique is reminiscent of a result in algebraic complexity theory, where the number of execution traces of an algorithm for a problem Π in the algebraic comparison-query model is lower bounded in terms of the number of connected components that Π induces in its input space $\mathbb{R}^n$ [1]. In the comparison-query setting, we obtain the following lower bound.

Lemma 1.16 (Connectivity Lemma). *For any problem Π in the comparison-query model,* $\mathrm{D}(\Pi)$ *is at least the number of connected components of* $G(\Pi)$*.*

The Connectivity Lemma allows for a simple and unified exposition of many of the known lower bounds. For counting inversions and inversion parity the argument goes as follows. Every adjacent-rank transposition changes the number of inversions by exactly one (up or down), and therefore changes the output of Π, so all $n!$ vertices in $G(\Pi)$ are isolated. This means that any algorithm for Π actually needs to sort and has to make at least $\log_2(n!)$ queries. See Section 7 for a proof of the Connectivity Lemma and more applications to classical problems, including the $\Omega(n)$ lower bound for median finding.

The Connectivity Lemma also enables us to establish strong lower bounds for inversion minimization on special types of trees T, namely those of Theorem 1.8 and the Mann–Whitney instances in Theorem 1.9, closely related to counting inversions. Both theorems involve an analysis of the size of the connected component of a random ranking ρ in $G(\Pi_T)$, and Theorem 1.8 uses the delicate parity conditions of its statement to keep $G(\Pi_T)$ as sparse as possible. See Section 8 for more details, including a more general property that guarantees the required sparseness (the partition property from Definition 8.1) and the resulting lower bound for any problem Π that satisfies the property (Lemma 8.4).

The Mann–Whitney setting illustrates well the relative power of our techniques. In the Mann–Whitney instances of inversion minimization, the leaves are naturally split between a subtree containing a of them and a subtree containing b of them. The argument behind Theorem 1.5 yields a lower bound of $\frac{ab}{a+b}$ on the sensitivity $s(\Pi_T)$. The true sensitivity is just $O(1)$ below the one for counting cross inversions, which is $\frac{2ab}{a+b}$. The resulting lower bounds on the query complexity in case $a \leq b$ are $\Theta(a \log a)$, which roughly account for sorting the smaller side but not for the $b \log_2(a)$ comparisons used in the subsequent binary searches for counting cross inversions. Our approach based on the Connectivity Lemma yields a lower bound that includes both terms. On the other hand, it is easier to estimate and obtain the lower bound via the Sensitivity Lemma than to argue the query lower bound via the Connectivity Lemma or from scratch.

1.3 Other results

Other modes of computation. We stated our lower bounds for the standard, deterministic mode of computation. Both of our techniques provide lower bounds for the number of distinct execution traces that are needed to cover all input rankings, irrespective of whether these execution traces derive from a single algorithm. Such execution traces can be viewed as certificates or witnesses for the value of Π on a given input ρ, or as valid execution traces of a *nondeterministic* algorithm for Π. We define the minimum number of traces needed to cover all input rankings for a problem Π as the nondeterministic complexity of Π and denote it by $\mathrm{N}(\Pi)$, along the lines of the Boolean setting [15]. All of our lower bounds on $\mathrm{D}(\Pi)$ actually hold for $\mathrm{N}(\Pi)$. See Remark 3.4 and Remark 7.2 for further discussion.

Since *randomized algorithms with zero error* are also nondeterministic algorithms, all of our lower bounds apply verbatim to the former mode of computation, as well. As for *randomized algorithms with bounded error*, we argue in Section 6 that our lower bounds on the query complexity of inversion minimization on trees that follow from the Sensitivity Lemma carry over modulo a small loss in strength. We do so by showing in general that high average sensitivity implies high query complexity against such algorithms (see Lemma 6.5).

The fact that our techniques yield lower bounds on $\mathrm{N}(\Pi)$ and not just $\mathrm{D}(\Pi)$ also explains why our approaches sometimes fail. For example, for the problem Π of finding the minimum of n items, a total of n certificates suffice and are needed, namely one for each possible item being the minimum. This means that our techniques cannot give a lower bound on the query complexity of Π that is better than $\log_2(n)$. In contrast, as reviewed in Section 2, $\mathrm{D}(\Pi) = 2^{n-1}$ and the number of queries needed is $n - 1$.

Cross-inversion distribution. As a technical result in the sensitivity analysis for inversion minimization on binary trees (Theorem 1.6), we need a strong upper bound on the probability that the number of cross inversions $\mathrm{XInv}_\rho(A, B)$ takes on any particular value when the ranking ρ of the set $X = A \sqcup B$ is chosen uniformly at random. This is the distribution of the Mann–Whitney statistic under the null hypothesis. Mann and Whitney [20] argued that it converges to a normal distribution with mean $\mu = ab/2$ and variance $\sigma^2 = ab(a + b + 1)/12$ as $a \doteq |A|$ and $b \doteq |B|$ grow large. Since the normal distribution has a maximum density of $1/(\sqrt{2\pi}\sigma)$, their result suggests that the maximum of the underlying probability distribution is $O(1/\sigma) = O(1/\sqrt{ab(a + b + 1)})$. Takács [28] managed to formally establish such a bound for all pairs (a, b) with $|a-b| = O(\sqrt{a + b})$, Stanley and Zanello [27] for all pairs (a, b) with $\min(a, b)$ bounded, and Melczer, Panova, and Pemantle [21] for all pairs (a, b) with $|a - b| \leq \alpha \cdot (a + b)$ for some constant $\alpha < 1$. However, these results do not cover all regimes and leave open a single bound of the same form that applies to all pairs (a, b), which is what we need for Theorem 1.6. We establish such a bound in Section 9. The counts of the rankings ρ with a particular value for $\mathrm{XInv}_\rho(A, B)$ appear as the coefficients of the *Gaussian polynomials*. Our bound can be stated equivalently as a bound on those coefficients.

1.4 Organization

We have organized the material so as to provide a shortest route to a full proof of Theorem 1.5. Here are the sections needed for the different main results:

- Theorem 1.5 (lower bound for general trees): 3, 4.

- Theorem 1.6 (lower bound for binary trees): 3, 5, 9.
- Theorem 1.8 (lower bound for restricted classes): 7, 8 up to 8.3 inclusive.
- Theorem 1.9 (lower bound for counting cross inversions): 7, 8 but not 8.2 nor 8.3.

In Section 2, we provide some background on known lower bounds in the comparison-query model, several of which are unified by the Sensitivity Lemma and Connectivity Lemma. In Section 6, we present our lower bounds against randomized algorithms with bounded error. The tight bound on maximum probability of the cross-inversion distribution is covered in Section 9. For completeness, we end in Section 10 with proofs of the results we stated on the Turing complexity of inversion minimization on trees.

2 The comparison-query model

In this section we provide an overview of known results and techniques for lower bounds in the comparison-query model. This section can be skipped without a significant loss in continuity.

Tight bounds have been established for problems like sorting, selection, and heap construction.

- We already discussed the central problem of sorting in Section 1.
- In *selection* we are told a rank r, and must identify the item with rank r. The query complexity is known to be $\Theta(n)$ [2, 7, 8]. There is also *multiple selection*, in which one is given multiple ranks $r_1, \ldots, r_k$, and must identify each of the corresponding items. The query complexity of multiple selection is likewise known up to a $\Theta(n)$ gap between the upper and lower bounds [16].
- In *heap construction* we must arrange the items as nodes in a complete binary tree such that every node has a rank no larger than its children. The query complexity is known to be $\Theta(n)$ [5, 11].

All the problems above can be cast as instantiations of a general framework known as *partial order production* [25]. Here, in addition to query access to the ranking ρ of the items, we are given n *slots* and regular access to a partial order $<_{\text{slot}}$ on the slots. The objective is to put each item into a slot, one item per slot, so that whenever two slots, s_1 and s_2, are related by $s_1 <_{\text{slot}} s_2$, we also have $\rho(s_1) < \rho(s_2)$. Sorting coincides with the case where $<_{\text{slot}}$ is a total order. In selection of rank r, there is a designated slot s^*, and there are exactly $r-1$ slots s with $s <_{\text{slot}} s^*$ and exactly $n-r$ slots s with $s^* <_{\text{slot}} s$; there are no other relations in $<_{\text{slot}}$ (see the example at the end of Section 7 for more details). Multiple selection is similar. For heap construction, $<_{\text{slot}}$ matches the complete binary tree arrangement.

Partial order production for a given $<_{\text{slot}}$ naturally decomposes into the same problem for each of the connected components of the undirected graph underlying $<_{\text{slot}}$. In the case of a single connected component, an elementary adversary argument shows that $\mathrm{Q}(\Pi) \geq n-1$: Any combination of less than $n-1$ queries to ρ leaves some pair of slots in $<_{\text{slot}}$ undetermined with respect to ρ. Another lower bound is the information-theoretic limit. For each way of putting items into slots, the number of input rankings ρ for which that way is a correct answer is bounded by $e(<_{\mathrm{slot}})$, the number of ways to extend $<_{\mathrm{slot}}$ to a total order. Therefore, there must be at least $n!/e(<_{\mathrm{slot}})$ distinct execution traces. Since each execution trace is determined by the outcomes of its queries, and each query has only two outcomes, we conclude that $\lambda(<_{\mathrm{slot}}) \doteq \log_2(n!/e(<_{\mathrm{slot}}))$ queries are necessary to solve partial order production. Complementing these lower bounds there exists an upper bound of $(1+o(1)) \cdot \lambda(<_{\mathrm{slot}}) + c \cdot (n-1)$ queries for some universal constant c [4]. For any instance Π with partial order $<_{\mathrm{slot}}$ it follows that $\mathrm{Q}(\Pi) = \Theta(\lambda(<_{\mathrm{slot}}) + n - \gamma(<_{\mathrm{slot}}))$, where $\gamma(<_{\mathrm{slot}})$ denotes the number of connected components of the undirected graph underlying $<_{\mathrm{slot}}$.

Not every problem of interest in the comparison model is an instance of partial order production. Here are a few examples.

- In *rank finding* there is a designated item x^*, and we have to compute its rank. The rank can be computed by comparing x^* with each of the $n-1$ other items. A similar elementary adversary argument as above shows that the query complexity is at least $n-1$.
- In *counting inversions* the items are arranged in some known order σ and the objective is to count the number of inversions of σ with respect to ρ. As we reviewed in Section 1, counting inversions has exactly the same query complexity as sorting.
- The problem of *inversion parity* is the same as counting inversions except that one need only count the number of inversions modulo 2. This problem, as well as counting inversions modulo m for any integer $m > 1$, also has exactly the same complexity as sorting.

For each of the three problems above, information theory does not provide a satisfactory lower bound. For example, in the inversion parity problem there are only two possible outputs, which yields a lower bound of $\log_2(2) = 1$. It so happens that for each of the preceding three examples, the query complexity is known quite precisely; however, the known arguments are rather problem-specific.

Inversion minimization on trees is another example that does not fit the framework of partial order generation, and for which information theory only yields a weak lower bound: $\log_2 \binom{n}{2} = 2\log_2(n) - \Theta(1)$. In contrast to the above examples, a strong lower bound does not seem to follow from a simple ad-hoc argument nor from a literal equivalence to sorting.

3 Sensitivity Lemma

In this section we develop Lemma 1.14. We actually prove a somewhat stronger version.

Lemma 3.1 (Strong Sensitivity Lemma). *Consider an algorithm A in the comparison-based model with n items, color each vertex of the permutahedron with its execution trace under A, and let $\overline{H}$ denote the subgraph with the same vertex set but only containing the bichromatic edges. The number of distinct execution traces of A is at least $g(\deg_{\mathrm{avg}}(\overline{H}) + 1)/n$, where $g : [1, \infty) \to \mathbb{R}$ is any convex function with $g(x) = x!$ for $x \in [n]$.*

The Sensitivity Lemma follows from Lemma 3.1 because the coloring with execution traces of an algorithm A for Π is a refinement of the coloring with Π, so every edge of the permutahedron that is bichromatic under Π is also bichromatic under A, and

$$s(\Pi) \doteq \mathbb{E}(\deg_{\overline{G(\Pi)}}(\rho)) \leq \mathbb{E}(\deg_{\overline{H}}(\rho)) \doteq \deg_{\text{avg}}(\overline{H}).$$

Provided g is nondecreasing, it follows that $\mathrm{D}(\Pi) \geq g(\deg_{\text{avg}}(\overline{H}) + 1)/n \geq g(s(\Pi) + 1)/n$.

In the Sensitivity Lemma we set $g(x) = \Gamma(x + 1)$. An optimal (but less elegant) choice for g is the piece-wise linear function that interpolates the prescribed values at the integral points in $[n]$, namely

$$g(x) \doteq (x - \lfloor x \rfloor) \cdot (\lceil x \rceil!) + (1 - (x - \lfloor x \rfloor)) \cdot (\lfloor x \rfloor!).$$

For the proof of Lemma 3.1 we take intuition from a similar result in the Boolean setting [23, Exercise 8.43], where the hypercube plays the role of the permutahedron in our setting.

Fact 3.2. *Let A be a query algorithm on binary strings of length n. Color each vertex of the n-dimensional hypercube by its execution trace under A, and let $\overline{H}$ denote the subgraph with the same vertex set but only containing the bichromatic edges. Then the number of distinct execution traces is at least $2^{\deg_{\text{avg}}(H)}$.*

One way to argue Fact 3.2 is to think of assigning a weight $w(x)$ to each $x \in \{0, 1\}^n$ so as to maximize the total weight on all inputs, subject to the constraint that the total weight on each individual execution trace is at most 1. Then the number of distinct execution traces must be at least the sum of all the weights. If the weight only depends on the degree, i. e., if we can write $w(x) = f(\deg_{\overline{H}}(x))$ for some function $f : [0, \infty) \to \mathbb{R}$, then we can lower bound the number k of distinct execution traces as follows:

$$k \geq \sum_x w(x) = \sum_x f(\deg_{\overline{H}}(x)) \geq 2^n \cdot f(\mathbb{E}(\deg(x))) = 2^n \cdot f(\deg_{\text{avg}}(\overline{H})), \tag{3.1}$$

where the last inequality holds provided f is convex.

In the Boolean setting, the set R of inputs $x \in \{0, 1\}^n$ with a particular execution trace forms a subcube of dimension $n - \ell$, where ℓ denotes the length of the execution trace, i. e., the number of queries. Each $x \in R$ has degree ℓ in H; this is because a change in a single queried position results in a different execution trace, and a change in an unqueried position does not. Therefore, a natural choice for the weight of $x \in R$ is $w(x) = f(\ell)$ where $f(x) = 1/2^{n-\ell}$. It satisfies the constraint that the total weight on R is (at most) one, and f is convex. We conclude by (3.1) that the number of distinct execution traces is at least $2^n \cdot f(\deg_{\text{avg}}(\overline{H})) = 2^{\deg_{\text{avg}}(\overline{H})}$, as desired.

Proof of Lemma 3.1. Let k denote the number of distinct execution traces of A, and let $R_1, \dots, R_k$ denote the corresponding sets of rankings. Following a similar strategy, we want to find a convex function $f : [0, \infty) \to \mathbb{R}$ such that the weight function $w(\rho) = f(\deg_{\overline{H}}(\rho))$ does not assign weight more than 1 to any one of the sets R_i. The following claim, to be proven later, is the crux of this.

Claim 3.3. *Let R denote the set of all rankings ρ that follow a particular execution trace on A, and let $d \in \{0, \dots, n - 1\}$. The number of rankings $\rho \in R$ with $\deg_{\overline{H}}(\rho) = d$ is at most $\frac{n!}{(d+1)!}$.*

Based on Claim 3.3, a natural choice for f is any convex function that satisfies $f(x) = \frac{1}{n}\frac{(x+1)!}{n!}$ for $x \in \{0, \ldots, n-1\}$. The factor of $\frac{1}{n}$ comes from the fact that there are n terms to sum together after the weights have been normalized. For every $i \in [k]$ we then have

$$\sum_{\rho \in R_i} w(\rho) = \sum_{d=0}^{n-1} \left|\{\rho \in R_i \,:\, \deg_{\overline{H}}(\rho) = d\}\right| \cdot f(d) \leq \sum_{d=0}^{n-1} \frac{1}{n}\frac{(d+1)!}{n!} \cdot \frac{n!}{(d+1)!} = \sum_{d=0}^{n-1} \frac{1}{n} = 1.$$

Similar to (3.1) we conclude

$$k \geq \sum_{i=1}^{k} \sum_{\rho \in R_i} w(\rho) = \sum_{\rho} w(\rho) = \sum_{\rho} f(\deg_{\overline{H}}(\rho)) \geq n! \cdot f(\mathbb{E}(\deg_{\overline{H}}(\rho))) = n! \cdot f(\deg_{\mathrm{avg}}(\overline{H})). \tag{3.2}$$

Setting $f(x) = \frac{1}{n}\frac{g(x+1)}{n!}$ turns the requirements for f into those for g in the statement of the lemma, and yields that $k \geq n! \cdot f(\deg_{\mathrm{avg}}(\overline{H})) = g(\deg_{\mathrm{avg}}(\overline{H}) + 1)/n$. □

We now turn to proving Claim 3.3. The comparisons and outcomes that constitute a particular execution trace of A can be thought of as directed edges between the items in X. We refer to the resulting digraph on the vertex set X as the comparison graph C. Since the outcomes of the comparisons are consistent with some underlying ranking, the digraph C is acyclic. The rankings in R are in one-to-one and onto correspondence with the linear orderings of the DAG C. For a given ranking $\rho \in R$, the degree $\deg_{\overline{H}}(\rho)$ equals the number of $r \in \{2, \ldots, n\}$ such that swapping ranks $r-1$ and r in ρ results in a ranking $\rho' = \tau\rho$ that is not in R, where τ denotes the adjacent-rank transposition $(r-1, r)$. The ranking ρ' not being in R means that it is inconsistent with the combined comparisons and outcomes of the underlying execution trace, which happens exactly when there is a path in C from the item $\rho^{-1}(r-1)$ of rank $r-1$ in ρ to the item $\rho^{-1}(r)$ with rank r in ρ. Thus, the degree $\deg_{\overline{H}}(\rho)$ equals the number of $r \in \{2, \ldots, n\}$ such that there is a path from $\rho^{-1}(r-1)$ to $\rho^{-1}(r)$ in C. See Figure 5 for an illustration, where a squiggly edge $u \rightsquigarrow v$ denotes that there exists a path from u to v in C. We only draw squiggly edges from one position to the next, so $\deg_{\overline{H}}(\rho)$ equals the number of squiggly edges in Figure 5.

$$\begin{array}{lccccccccc}
\text{rank} & 1 & 2 & 3 & 4 & 5 & \cdots & r-1 & r & \cdots & n \\
\rho^{-1} & \bullet & \circ \rightsquigarrow & \bullet \rightsquigarrow & \bullet & \circ & & \circ \rightsquigarrow & \bullet & & \circ
\end{array}$$

Figure 5: Ranking encoding

Our strategy is to give a *compressed encoding* of the rankings in R such that there is more compression as the number of squiggly edges increases. Our encoding is based on the well-known algorithm to compute a linear order of a DAG. Algorithm 1 provides pseudocode for the algorithm, which we refer to as BuildRanking.

In our formulation, BuildRanking is nondeterministic: There is a choice to make in step 3 for each $r = 1, \ldots, n$. The possible executions of BuildRanking are in one-to-one and onto correspondence with the linear orders of C, and thus with the rankings in R.

Algorithm 1 BuildRanking(C)

Input: DAG C on vertex set X
Output: ranking of X that is a linear order of C

1: $T \leftarrow \varnothing$
2: **for** $r = 1$ to n **do**
3: $\quad x \leftarrow$ arbitrary element of $S \doteq \{v \in X \mid \text{there is no } u \in X \setminus T \text{ with } u \rightsquigarrow v \text{ in } G\}$
4: $\quad \rho^{-1}(r) \leftarrow x$
5: $\quad T \leftarrow T \cup \{x\}$

Our encoding is a compressed description of how to make the decisions in BuildRanking such that the output is ρ. Note that if $\rho^{-1}(r-1) \rightsquigarrow \rho^{-1}(r)$, then the item x with rank r cannot enter the set S before iteration r. This is because before $\rho^{-1}(r-1)$ is removed from T at the end of iteration $r-1$, the edge $\rho^{-1}(r-1) \rightsquigarrow \rho^{-1}(r)$ prevents x from being in S. Thus, whenever $\rho^{-1}(r-1) \rightsquigarrow \rho^{-1}(r)$, the item $x = \rho^{-1}(r)$ is *lucky* in the sense that it gets picked in step 3 as soon as it enters the set S. In fact, the lucky items with respect to a ranking $\rho \in R$ are exactly those for which $\rho^{-1}(r-1) \rightsquigarrow \rho^{-1}(r)$ for some $r \in \{2, \ldots, n\}$, as well as the item $\rho^{-1}(1)$ with rank 1. In Figure 5 the lucky items are marked black. Their number equals $\deg_{\overline{H}}(\rho) + 1$.

In order to generate a ranking ρ using BuildRanking, it suffices to know:

(a) the lucky items (as a set, not their relative ordering), and

(b) the ordering of the non-lucky items (given which items they are).

This information suffices to make the correct choices in step 3 of Algorithm 1:

- If the set S contains a lucky item, there will be a unique lucky item in S; pick it as the element x.
- Otherwise, pick for x the first item in the ordering of the non-lucky items that is not yet in T. Such an element will exist, and all the items that come after it in the ordering are not yet in T either.

If ρ has degree $d = \deg_{\overline{H}}(\rho)$, then there are $d+1$ lucky items, so there are at most $\binom{n}{d+1}$ choices for (a), and at most $(n-d-1)!$ choices for (b), resulting in a total of at most $\binom{n}{d+1} \cdot (n-d-1)! = \frac{n!}{(d+1)!}$ choices. This proves Claim 3.3.

Remark 3.4. Suppose we allow an algorithm A to have multiple valid execution traces on a given input ρ, and let R_i denote the set of rankings on which the i-th execution trace is valid. The proof of Claim 3.3 carries over as it considers individually sets R_i, and only depends on the DAG that the comparisons in R_i induce. The rest of the proof of Lemma 3.1 carries through modulo the first equality in (3.2), which no longer holds as the sets R_i may overlap. However, the equality can be replaced by the inequality $\geq$, which does hold and is sufficient for the argument. This means that we can replace $\mathrm{D}(\Pi)$ in the statement of the Sensitivity Lemma by its nondeterministic variant $\mathrm{N}(\Pi)$.

4 Sensitivity approach for general trees

In this section we analyze the average sensitivity of the problem Π_T of inversion minimization on a tree T with a general shape. In Section 4.1 we show that the existence of a subtree containing a fair fraction of the leaves implies high sensitivity. The lower bound on query complexity for Π_T in Theorem 1.5 then follows from the Sensitivity Lemma. In Section 4.3 we prove that the average sensitivity measure is Lipschitz continuous. For the analysis, we make use of the decomposition of the objective of inversion minimization on trees mentioned earlier. We describe the decomposition in more detail in Section 4.2; it will be helpful in later parts of this paper, as well.

4.1 Subtree-induced sensitivity

We first introduce a sensitivity bound for inversion minimization based on the size of a subtree.

Lemma 4.1 (subtree-induced sensitivity)**.** *Consider a tree T with $n \doteq |\mathrm{L}(T)|$ leaves, and some node v in T with $\ell \doteq |\mathrm{L}(T_v)|$ leaves. We have*

$$s(\Pi_T) \geq \frac{\ell(n-\ell)}{n} - 1.$$

Note that v is not necessarily a direct child of the root, as shown in Figure 6.

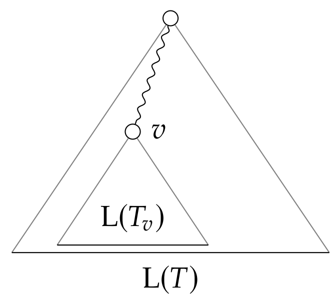


Figure 6: Subtree rooted at v

We now prove Lemma 4.1. Let ρ be a ranking of the leaves of T, and let $\sigma_{\min}$ be a tree ordering that minimizes the number of inversions with respect to ρ.

Claim 4.2. *ρ is sensitive to the transposition $\tau = (r, r+1)$ if $\sigma_{\min}(\rho^{-1}(r)) > \sigma_{\min}(\rho^{-1}(r+1))$.*

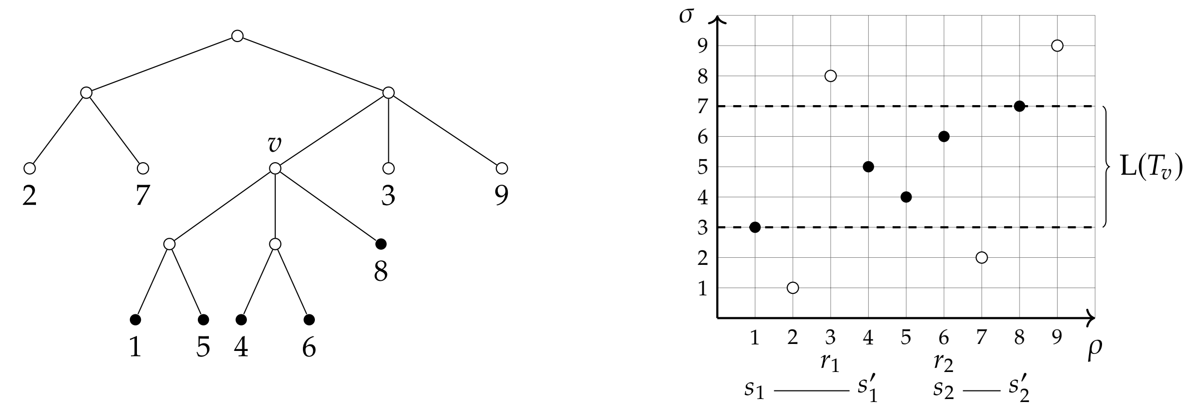
Proof. If $\sigma_{\min}(\rho^{-1}(r)) > \sigma_{\min}(\rho^{-1}(r+1))$, then $\mathrm{Inv}_{\tau\rho}(\sigma_{\min}) = \mathrm{Inv}_{\rho}(\sigma_{\min}) - 1$. By definition, $\mathrm{Inv}_{\rho}(\sigma_{\min}) = \mathrm{MinInv}(T, \rho)$, so this means that $\mathrm{MinInv}(T, \tau\rho) < \mathrm{MinInv}(T, \rho)$, or that ρ is sensitive to τ. □

In the case of general trees, a tree ordering σ that minimizes the number of inversions with respect to ρ is difficult to find (see the discussion on NP-hardness in Section 10). Our strategy is to find a lower bound on the number of r for which $\sigma(\rho^{-1}(r)) > \sigma(\rho^{-1}(r+1))$ that applies regardless of σ.

Claim 4.3. *For any ordering σ, the number of r such that $\sigma(\rho^{-1}(r)) > \sigma(\rho^{-1}(r+1))$ is at least one less than the number of s such that $\rho^{-1}(s) \in \mathrm{L}(T_v)$ and $\rho^{-1}(s+1) \notin \mathrm{L}(T_v)$.*

Proof. For all except at most one value of s (the maximum s for which $\rho^{-1}(s) \in \mathrm{L}(T_v)$), there exists a minimal $s' > s$ such that $\rho^{-1}(s') \in \mathrm{L}(T_v)$. We claim that at least one value of $r = s, \ldots, s'-1$ satisfies $\sigma(\rho^{-1}(r)) > \sigma(\rho^{-1}(r+1))$. If not, then σ would rank $\rho^{-1}(s), \rho^{-1}(s+1), \ldots, \rho^{-1}(s')$ in increasing order. Because σ is a tree ordering, the leaves of $L(T_v)$ must be mapped into a contiguous range by σ, as shown in Figure 7. However, we have $\rho^{-1}(s), \rho^{-1}(s') \in L(T_v)$ but $\rho^{-1}(s+1) \notin L(T_v)$, which violates this property since σ ranks a leaf outside $L(T_v)$ between two leaves inside $L(T_v)$.

Because each value of r is found between consecutive pairs of values in $L(T_v)$, the values of r are distinct. □



(a) Leaves in σ-order, labeled with ρ-ranks

(b) Corresponding plot of ρ, σ for each leaf

Figure 7: σ maps leaves of $L(T_v)$ in a contiguous range.

Claim 4.4. *Over a uniformly random ρ, the expected number of s such that $\rho^{-1}(s) \in L(T_v)$ and $\rho^{-1}(s+1) \notin L(T_v)$ is $\frac{\ell(n-\ell)}{n}$.*

Proof. For $s = 1, \ldots, n-1$, the probability that $\rho^{-1}(s) \in L(T_v)$ is $\frac{\ell}{n}$, and the probability that $\rho^{-1}(s+1) \notin L(T_v)$ given that $\rho^{-1}(s) \in L(T_v)$ is $\frac{n-\ell}{n-1}$. Using linearity of expectation on the indicator random variables for $\rho^{-1}(s) \in L(T_v)$ and $\rho^{-1}(s+1) \notin L(T_v)$, the expected number of s satisfying this property is

$$(n-1)\left(\frac{\ell(n-\ell)}{n(n-1)}\right) = \frac{\ell(n-\ell)}{n}.$$ □

Combining Claim 4.2, Claim 4.3, and Claim 4.4, we can conclude with Claim 4.1.

Bounded degree. We apply our analysis to the case of trees of degree k. Observe that for fixed n, Lemma 4.1 is strongest when $\ell = n/2$. Not every tree T has a subtree with exactly $n/2$ leaves, but Lemma 4.1 still gives a useful bound for subtrees that do not contain too few or too many leaves. In the case of trees of bounded degree, there always exists a subtree T_v that contains a fairly balanced fraction of the leaves. The following quantification is folklore, but we include a proof for completeness.

Fact 4.5. *If T is a tree of degree k with n leaves, there exists a node v in T such that $\ell \doteq |\mathrm{L}(T_v)| = \alpha \cdot n$, where $\frac{1}{k+1} \leq \alpha \leq \frac{k}{k+1}$.*

Proof. Let r be the root of T and construct a sequence $v_1 = r, v_2, v_3, \ldots$ such that v_i is a child of v_{i-1} that maximizes $\ell_i \doteq |\mathrm{L}(T_{v_i})|$, with ties broken arbitrarily. Notice that $\{\ell_i\}$ is a decreasing sequence, and since T has degree k, $\ell_i \leq k\ell_{i+1}$ for all i. We claim that some v_i in this sequence satisfies the conditions of the claim. If not, then for some i, $\ell_i > \frac{k}{k+1} \cdot n$ and $\ell_{i+1} < \frac{1}{k+1} \cdot n$, which contradicts the fact that $\ell_i \leq k\ell_{i+1}$. □

By choosing a subtree satisfying Fact 4.5, we can apply Lemma 4.1 and conclude that $s(\Pi_T) \geq \frac{k}{(k+1)^2} \cdot n - 1$. The Sensitivity Lemma then gives the "in particular" part of Theorem 1.5.

4.2 Decomposition of the objective function

For use in this section as well as later parts of the paper, we now explain how the objective of inversion minimization on trees decomposes. We introduce the notion of root inversion along the way, and observe the effect of adjacent-rank transpositions on the decomposition.

The objective $\mathrm{MinInv}(T, \rho)$ can be written as the sum of contributions from each of the individual nodes. A node v contributes those inversions that reside in the subtree T_v and go through the root v of T_v. We refer to them as the root inversions in T_v.

Definition 4.6 (root inversions, $\mathrm{RInv}(\cdot,\cdot,\cdot)$, $\mathrm{MinRInv}(\cdot,\cdot)$)**.** Given a tree T, a ranking ρ of the leaves of T, and an ordering σ of T, a root inversion of σ with respect to ρ is an inversion (ℓ_1, ℓ_2) of σ with respect to ρ for which the lowest common ancestor $\mathrm{LCA}(\ell_1, \ell_2)$ is the root of T. The number of root inversions of σ with respect to ρ in T is denoted by $\mathrm{RInv}(T, \rho, \sigma)$. The minimum number of root inversions in T with respect to ρ is denoted

$$\mathrm{MinRInv}(T, \rho) \doteq \min_{\sigma} \mathrm{RInv}(T, \rho, \sigma), \tag{4.1}$$

where σ ranges over all possible orderings of T.

The only aspect of the ordering σ of T_v that affects $\mathrm{RInv}(T_v, \rho, \sigma)$ is the relative order of the children of v. For a node v with k children $u_1, \ldots, u_k$, by abusing notation and using σ to also denote the ranking of the children induced by the ordering of the tree, we have

$$\mathrm{RInv}(T, \rho, \sigma) \doteq \sum_{1 \leq i < j \leq k} \mathrm{XInv}_\rho(L_{\sigma(i)}, L_{\sigma(j)}), \tag{4.2}$$

where L_i is a short-hand for the leaf set $L(T_{u_i})$. The contributions of the nodes can be optimized independently:

$$\mathrm{MinInv}(T, \rho) = \sum_{v} \mathrm{MinRInv}(T_v, \rho), \tag{4.3}$$

where v ranges over all nodes of T with degree $\deg_T(v) > 1$.

When we apply an adjacent-rank transposition τ to a ranking ρ, at most one of terms in the decomposition (4.3) can change, and the change is at most one unit. We capture this observation for future reference as it will be helpful in several sensitivity analyses.

Proposition 4.7. *Let ρ be a ranking of the leaf set X of a tree T, τ an adjacent-rank transposition, and ℓ_1 and ℓ_2 be the affected leaves. Then*

$$\mathrm{MinRInv}(T_v, \rho) = \mathrm{MinRInv}(T_v, \tau\rho)$$

for all nodes v in T except possibly $v = \mathrm{LCA}(\ell_1, \ell_2)$. Moreover, the difference is at most 1 in absolute value.

Proof. Since the ranks of ℓ_1 and ℓ_2 under ρ are adjacent, for any leaf ℓ other than ℓ_1 and ℓ_2, the relative order of ℓ under ρ is the same with respect to ℓ_1 as it is with respect to ℓ_2. This means that the adjacent-rank transposition τ does not affect whether a pair of leaves constitutes an inversion unless that pair equals $\{\ell_1, \ell_2\}$. As a result, the only term on the right-hand side of (4.3) that can be affected by the transposition τ is the one corresponding to the node v, and it can change by at most one unit. □

4.3 Lipschitz continuity

Average-case notions typically do not change much under small changes to the input. This is indeed the case for the average sensitivity when "small" is interpreted as affecting few of the subtrees. The following lemma quantifies the property and can be viewed as a form of Lipschitz continuity.

Lemma 4.8. *Given a tree T, if a subtree T_{v^*} with ℓ leaves is replaced with a tree T'_{v^*} with the same number of leaves, resulting in the tree T', then*

$$|s(\Pi_T) - s(\Pi_{T'})| \leq \frac{\ell(\ell-1)}{n}.$$

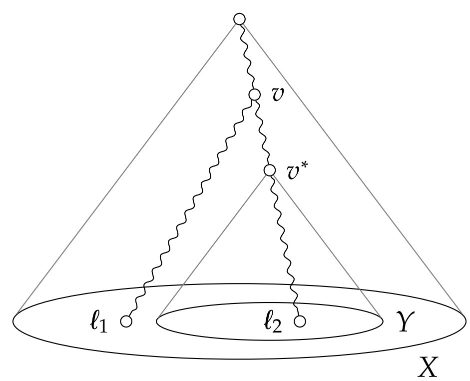


Figure 8: Effects of changing T_{v^*}

Proof. We think of the leaf sets of T and T' as being the same set $X = \mathrm{L}(T) = \mathrm{L}(T')$, and fix a ranking ρ of X. Consider an ordering of T and the ranking σ of X that it induces. Outside of T'_{v^*} we can order T' in the same way as T. Irrespective of how we order T' inside T'_{v^*}, the induced ranking σ' of X agrees with σ on all leaves in X except possibly those in $Y \doteq \mathrm{L}(T_{v^*}) = \mathrm{L}(T'_{v^*})$. Moreover, under both σ and σ', the set Y gets mapped to the same contiguous interval. It follows that for all pairs (ℓ_1, ℓ_2) of distinct leaves of which at least one lies outside of Y, (ℓ_1, ℓ_2) constitutes an inversion of σ with respect to ρ if and only if (ℓ_1, ℓ_2) constitutes an inversion of σ' with respect to ρ. For any node v outside of T_{v^*}, root inversions in T_v cannot involve leaves that are both in $Y \doteq \mathrm{L}(T_{v^*})$. See Figure 8 for an illustration. Thus, for such nodes v, $\mathrm{RInv}(T_v, \rho, \sigma) = \mathrm{RInv}(T'_v, \rho, \sigma')$. By taking the minimum over all orderings, we conclude:

Claim 4.9. $\mathrm{MinRInv}(T_v, \rho) = \mathrm{MinRInv}(T'_v, \rho)$ *holds for every node v outside of T_{v^*} (or equivalently, outside of T'_{v^*}).*

Consider a ranking ρ and an adjacent-rank transposition $\tau = (r, r+1)$. We claim that, unless $(\ell_1, \ell_2) \doteq (\rho^{-1}(r), \rho^{-1}(r+1)) \in Y \times Y$, Π_T is sensitive to τ at ρ if and only if $\Pi_{T'}$ is sensitive to τ at ρ. This is because by Proposition 4.7 the only term in the decomposition (4.3) of $\mathrm{MinInv}(T, \rho)$ that can be affected by τ is the contribution $\mathrm{MinRInv}(T_v, \rho)$ for $v = \mathrm{LCA}(\ell_1, \ell_2)$. If at least one of ℓ_1 or ℓ_2 is not inside T_{v^*}, then v is not inside T_{v^*} either, so by Claim 4.9, $\mathrm{MinRInv}(T_v, \rho) = \mathrm{MinRInv}(T'_v, \rho)$. By the same token, $\mathrm{MinRInv}(T_v, \tau\rho) = \mathrm{MinRInv}(T'_v, \tau\rho)$. It follows that $\mathrm{MinInv}(T, \rho) \neq \mathrm{MinInv}(T, \tau\rho)$ if and only if $\mathrm{MinInv}(T', \rho) \neq \mathrm{MinInv}(T', \tau\rho)$.

We bound the expected number of values of r for which $(\rho^{-1}(r), \rho^{-1}(r+1)) \in Y \times Y$ with $Y \doteq \mathrm{L}(T_v)$ when ρ is chosen uniformly at random. For $r \in [n-1]$, the probability that $\rho^{-1}(r) \in Y$ is $\frac{\ell}{n}$, and the probability that $\rho^{-1}(r+1) \in Y$ given that $\rho^{-1}(r) \in Y$ is $\frac{\ell-1}{n-1}$. Using linearity of expectation on the indicators, the expected number of said r is

$$(n-1)\left(\frac{\ell(\ell-1)}{n(n-1)}\right) = \frac{\ell(\ell-1)}{n}. \qquad \square$$

Lemma 4.8 helps to extend query lower bounds based on average sensitivity to larger classes. Suppose we have established a good lower bound on the sensitivity $s(\Pi_T)$ for a class C of trees. Consider a class C' obtained by taking a tree T in class C and replacing some of the subtrees T_v by other subtrees T'_v on the same number of leaves. For this new class C' the same lower bound on the sensitivity of inversion minimization applies modulo the Lipschitz loss. For example, Theorem 1.6 holds by virtue of a lower bound of the form $s(\Pi_T) \geq (n-1) - c\log(n)$ for every binary tree T with n leaves, where c is a universal constant. If we allow some of the subtrees of T to be replaced by, say freely arrangeable ones on the same leaves, applying Lemma 4.8 for each of the modified subtrees in sequence shows that the resulting new tree T' has

$$s(\Pi_{T'}) \geq s(\Pi_T) - \frac{\alpha n(\alpha n - 1)}{n} \geq (n-1) - c\log(n) - \alpha^2(n-1) = (1-\alpha^2)(n-1) - c\log(n),$$

where α denotes the fraction of leaves that belong to one of the replaced subtrees.

In fact, the notion of average sensitivity is robust with respect to the following, more refined type of surgery. From any tree T, let R be a connected subset of T that includes no leaves. Let T' be the subtree rooted at the LCA of R (T' contains all of R), and let $T'_1, \ldots, T'_k$ be the disjoint maximal subtrees of T' that are strictly below R. Let R' be any tree that has k leaves. Replace T' by R', and then replace the leaves of R' by $T'_1, \ldots, T'_k$.

The effect is that the region R has been "reshaped" to look like R', but the rest of T is unaffected. The cost of such a surgery is at most $(n-1)$ times the probability that a uniformly random pair of distinct leaves has their LCA in R. The bound follows from thinking of sensitivity as $(n-1)$ times the probability that a uniformly random edge in the full permutahedron is bichromatic. Provided the LCA of the affected leaves is outside R, then we get sensitivity before the surgery if and only if we get it after the surgery. Surgeries can be iterated, and the costs accumulate additively. In combination with our strong lower bound on the average sensitivity of binary trees (Lemma 5.7), this allows for a robust sense in which "mostly-binary" trees have high average sensitivity.

5 Refined sensitivity approach for binary trees

In this section we show how to refine the sensitivity approach for lower bounds on the query complexity of the problem Π_T of inversion minimization on trees in the important special case of binary trees T. In Section 5.1 we first develop a criterion for when a particular ranking ρ is sensitive to a particular adjacent-rank transposition τ. We then analyze the root sensitivity of binary trees in Section 5.2 and finally establish a strong lower bound on the average sensitivity in Section 5.3. An application of the Sensitivity Lemma then yields Theorem 1.6.

5.1 Sensitivity criterion

Recall the decomposition of the objective function $\mathrm{MinInv}(T, \rho)$ into contributions attributed to each node v of degree $\deg_T(v) > 1$, as given by (4.3) in Section 4.2. In the case of binary trees, the contribution of node v can be calculated simply as

$$\mathrm{MinRInv}(T_v, \rho) = \min(\mathrm{XInv}_\rho(L_1, L_2), \mathrm{XInv}_\rho(L_2, L_1)), \tag{5.1}$$

where u_1 and u_2 denote the two children of v, and $L_1 \doteq \mathrm{L}(T_{u_1})$ and $L_2 \doteq \mathrm{L}(T_{u_2})$ their leaf sets. This simplicity makes a precise analysis of sensitivity feasible, as we will see next.

For a given ranking ρ of T and a given adjacent-rank transposition τ, we would like to figure out the effect of τ on the objective $\mathrm{MinInv}(T, \cdot)$, in particular when $\mathrm{MinInv}(T, \tau\rho) = \mathrm{MinInv}(T, \rho)$. Let ℓ_{lo} and ℓ_{hi} denote the two leaves that are affected by the transposition τ on the ranking ρ, where the subscript "lo" indicates the lower of the two leaves with respect to ρ, and "hi" the higher of the two. Let v be the lowest common ancestor $\mathrm{LCA}(\ell_{\mathrm{lo}}, \ell_{\mathrm{hi}})$. We use the same subscripts "lo" and "hi" for the two children of v: u_{lo} denotes the child whose subtree contains ℓ_{lo}, and u_{hi} its sibling. Similarly, we denote by L_{lo} the leaf set of $T_{u_{\mathrm{lo}}}$, and by L_{hi} the leaf set of $T_{u_{\mathrm{hi}}}$. See Figure 9 for the subsequent analysis.

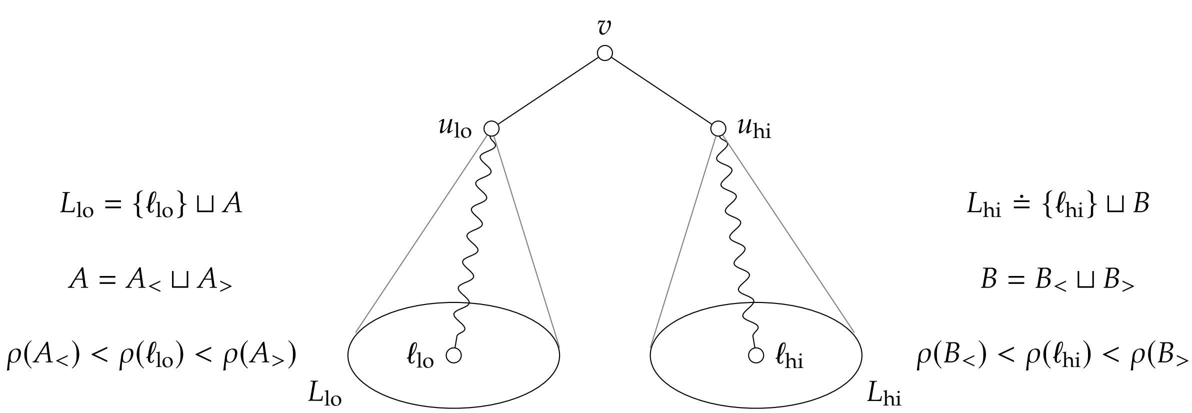


Figure 9: Sensitivity analysis for binary trees

By Proposition 4.7, the situation before and after the application of τ is as follows, where we abbreviate $x \doteq \mathrm{XInv}_\rho(L_{\mathrm{lo}}, L_{\mathrm{hi}})$ and $y \doteq \mathrm{XInv}_\rho(L_{\mathrm{hi}}, L_{\mathrm{lo}})$.

ranking	$\mathrm{RInv}(T_v, \cdot, \sigma(\ell_{\mathrm{lo}}) < \sigma(\ell_{\mathrm{hi}}))$	$\mathrm{RInv}(T_v, \cdot, \sigma(\ell_{\mathrm{hi}}) < \sigma(\ell_{\mathrm{lo}}))$	$\mathrm{MinRInv}(T_v, \cdot)$
ρ	x	y	$\min(x, y)$
$\tau\rho$	$y - 1$	$x + 1$	$\min(y - 1, x + 1)$

The objective function remains the same iff $\min(x, y) = \min(y - 1, x + 1)$, which happens iff $x - y = -1$, or equivalently iff

$$\mathrm{DInv}_\rho(L_{\mathrm{lo}}, L_{\mathrm{hi}}) = \mathrm{DInv}_\rho(\{\ell_{\mathrm{lo}}\}, \{\ell_{\mathrm{hi}}\}), \tag{5.2}$$

where we introduce the following short-hand:

Definition 5.1 (cross inversion difference, $\mathrm{DInv}_\cdot(\cdot, \cdot)$)**.** For a ranking ρ of a set X, and two subsets $A, B \subseteq X$,

$$\mathrm{DInv}_\rho(A, B) \doteq \mathrm{XInv}_\rho(A, B) - \mathrm{XInv}_\rho(B, A).$$

We can split L_{lo} as $L_{\mathrm{lo}} = \{\ell_{\mathrm{lo}}\} \sqcup A = A_< \sqcup \{\ell_{\mathrm{lo}}\} \sqcup A_>$, where $A_<$ contains all leaves in L_{lo} that ρ ranks before ℓ_{lo}, and $A_>$ contains all the leaves in L_{lo} that ρ ranks after ℓ_{lo}. We similarly split L_{hi}, as indicated in Figure 9. We have that

$$\mathrm{DInv}_\rho(L_{\mathrm{lo}}, L_{\mathrm{hi}}) = \mathrm{DInv}_\rho(\{\ell_{\mathrm{lo}}\}, \{\ell_{\mathrm{hi}}\}) + \mathrm{DInv}_\rho(\{\ell_{\mathrm{lo}}\}, B) + \mathrm{DInv}_\rho(A, \{\ell_{\mathrm{hi}}\}) + \mathrm{DInv}_\rho(A, B).$$

Since the ranks of ℓ_{lo} and ℓ_{hi} under ρ are adjacent, we have that $\mathrm{DInv}_\rho(\{\ell_{\mathrm{lo}}\}, B) = |B_<| - |B_>|$ and $\mathrm{DInv}_\rho(A, \{\ell_{\mathrm{hi}}\}) = |A_>| - |A_<|$. Plugging everything into (5.2) we conclude:

Proposition 5.2. *Let T be a binary tree, ρ a ranking of the leaves of T, τ an adjacent-rank transposition, ℓ_{lo} and ℓ_{hi} the two leaves affected by τ under ρ such that ρ ranks ℓ_{lo} before ℓ_{hi}. Referring to the notation in Figure 9, we have that*

$$\begin{aligned} \mathrm{MinInv}(T, \rho) = \mathrm{MinInv}(T, \tau\rho) &\Leftrightarrow \mathrm{DInv}_\rho(L_{\mathrm{lo}}, L_{\mathrm{hi}}) = \mathrm{DInv}_\rho(\{\ell_{\mathrm{lo}}\}, \{\ell_{\mathrm{hi}}\}) && (5.3)\\ &\Leftrightarrow \mathrm{DInv}_\rho(A, B) = |A_<| - |A_>| + |B_>| - |B_<|. && (5.4) \end{aligned}$$

5.2 Root sensitivity

Given a ranking ρ and an adjacent-rank transposition τ, we know by Proposition 4.7 that at most one of the terms in the decomposition (4.3) of $\mathrm{MinInv}(T, \rho)$ is affected by the transposition, namely $\mathrm{MinRInv}(T_v, \rho)$ where v is the lowest common ancestor of the affected leaves ℓ_{lo} and ℓ_{hi}. It follows that we can write the average sensitivity of $\Pi_T \doteq \mathrm{MinInv}(T, \cdot)$ as the following convex combination:

$$\begin{aligned} s(\Pi_T) &= (n-1) \cdot \mathbb{P}[\mathrm{MinInv}(T, \rho) \neq \mathrm{MinInv}(T, \tau\rho)] \\ &= (n-1) \sum_v \mathbb{P}[v = \mathrm{LCA}(\ell_{\mathrm{lo}}, \ell_{\mathrm{hi}})] \mathbb{P}[\mathrm{MinRInv}(T_v, \rho) \neq \mathrm{MinRInv}(T_v, \tau\rho) \mid v = \mathrm{LCA}(\ell_{\mathrm{lo}}, \ell_{\mathrm{hi}})], \end{aligned} \tag{5.5}$$

where the probability is over a uniformly random choice of the ranking ρ and the adjacent-rank transposition τ, and ℓ_{lo} and ℓ_{hi} denote the affected leaves. The conditional probability on the right-hand side of (5.5) only depends on the subtree T_v. The ranking ρ of all leaves induces a ranking ρ' of the leaves of T_v that is uniform under the conditioning. Similarly, the adjacent-rank transposition τ for ρ induces an adjacent-rank transposition τ' for ρ'; the distribution of τ' under the conditioning is independent of ρ' and uniform among all adjacent-rank transpositions such that the affected leaves live in subtrees of different children of v. Thus, the probability on the right-hand side of (5.5) coincides with the following notion for the subtree T_v.

Definition 5.3 (root sensitivity)**.** Let T be a tree. The *root sensitivity* of T is the probability that

$$\mathrm{MinRInv}(T_v, \rho) \neq \mathrm{MinRInv}(T_v, \tau\rho)$$

when ρ is a uniform random ranking of $\mathrm{L}(T)$, and τ a uniform random adjacent transposition with the condition that the affected leaves are in subtrees of different children of the root of T.

Note that the only nodes v that need to be considered in the sum on the right-hand side of (5.5) are those that can appear as the lowest common ancestor of two leaves, and such that T_v is not freely arrangeable. In the case of binary trees, this means that we only need to consider nodes v of degree 2 such that T_v contains more than 2 leaves. In this section we prove a strong lower bound on the root sensitivity of such trees T_v.

Consider the binary tree T with root v in Figure 9. The distribution underlying Definition 5.3 can be generated as follows: Pick a leaf on each side of the root v uniformly at random, and let ρ be a ranking of the leaves of T that is uniformly random on the condition that the selected leaves receive adjacent ranks; τ then is the adjacent-rank transposition that swaps the two selected leaves. The root sensitivity of T is the complement of the probability that the right-hand side of (5.4) holds under this distribution. Let us analyze the left-hand side of (5.4) further. As $A = A_< \sqcup A_>$ and $B = B_< \sqcup B_>$, we have that

$$\mathrm{DInv}_\rho(A, B) = \mathrm{DInv}_\rho(A_<, B_<) + \mathrm{DInv}_\rho(A_<, B_>) + \mathrm{DInv}_\rho(A_>, B_<) + \mathrm{DInv}_\rho(A_>, B_>).$$

By the defining properties of the sets involved (see Figure 9), we know that $\mathrm{DInv}_\rho(A_<, B_>) = -a_< b_>$ and $\mathrm{DInv}_\rho(A_>, B_<) = a_> b_<$, where $a_< \doteq |A_<|$, $a_> \doteq |A_>|$, $b_< \doteq |B_<|$, and $b_> \doteq |B_>|$. Thus, we can rewrite criterion (5.4) as:

$$\mathrm{DInv}_\rho(A_<, B_<) + \mathrm{DInv}_\rho(A_>, B_>) = a_< b_> - a_> b_< + a_< - a_> + b_> - b_<. \tag{5.6}$$

A critical observation that helps us to bound the probability of (5.6) is that, conditioned on all four values $a_\cdot$ and $b_\cdot$, the right-hand side of (5.6) is fixed, but the left-hand side still contains a lot of randomness. In fact, under the conditioning stated, the ranking that ρ induces on $A_< \sqcup B_<$ is still distributed uniformly at random, the same holds for the ranking that ρ induces on $A_> \sqcup B_>$, and both distributions are independent. This means that, under the same conditioning, the left-hand side of (5.6) has the same distribution as the sum $\mathrm{XInv}_{a_<, b_<} + \mathrm{XInv}_{a_>, b_>}$ of two independent random variables of the following type:

Definition 5.4 (cross inversion distribution, $\mathrm{XInv}_{\cdot,\cdot}$)**.** For nonnegative integers a and b, $\mathrm{XInv}_{a,b}$ denotes the random variable $\mathrm{XInv}(A, B)$ that counts the number of cross inversions from A to B, where A is an array of length a, B an array of length b, and the concatenation AB is a uniformly random permutation of $[a + b]$.

In Section 9 we establish the following upper bound on the probability that the number of cross inversions takes on any specific value.

Lemma 5.5. *There exists a constant C such that for all integers $a, b \geq 1$ and $0 \leq k \leq ab$,*

$$\mathbb{P}[\mathrm{XInv}_{a,b} = k] \leq \frac{C}{\sqrt{ab(a+b)}}. \tag{5.7}$$

Using Lemma 5.5 we can establish an upper bound of the same form as the right-hand side of (5.7) for the probability that (5.6) holds: For some constant C'

$$\mathbb{P}[\mathrm{XInv}_{a_<,b_<} + \mathrm{XInv}_{a_>,b_>} = a_< b_> - a_> b_< + a_< - a_> + b_> - b_<] \leq \frac{C'}{\sqrt{ab(a+b)}}, \tag{5.8}$$

where $a \doteq a_< + a_> \geq 1$ and $b \doteq b_< + b_> \geq 1$. We consider several cases based on the relative sizes of $a_<$ vs $a_>$, and $b_<$ vs $b_>$.

(i) In case both $a_< \geq \eta a$ and $b_< \geq \eta b$, the bound (5.8) follows from (5.7) as long as $C' \geq C/\eta^{3/2}$.

(ii) By switching the order in (i), the same holds true in case both $a_> \geq \eta a$ and $b_> \geq \eta b$.

(iii) In case $a_> \leq \eta a$ and $b_< \leq \eta b$, the left-hand side of (5.6) is at most $2\eta ab$, whereas the right-hand side is at least

$$(1-\eta)^2 ab - \eta^2 ab + (1-\eta)a - \eta a + (1-\eta)b - \eta b = (1-2\eta)(ab + a + b).$$

As long as $2\eta \leq 1 - 2\eta$, or equivalently, $\eta \leq 1/4$, this case cannot occur.

(iv) By switching the roles of A and B in (iii), the same holds true in case $b_> \leq \eta b$ and $a_< \leq \eta a$.

As long as $\eta \leq 1/2$, it holds that either $a_< \geq \eta a$ or $a_> \geq \eta a$, and either $b_< \geq \eta b$ or $b_> \geq \eta b$. Distributing the "and" over the "or", we obtain the four cases we considered, which are therefore exhaustive. We conclude that (5.8) holds for $C' = 4^{3/2}C$ whenever $a \doteq |A| \geq 1$ and $b \doteq |B| \geq 1$.

In the case where $a = 0$ and $b \geq 1$, the right-hand side of (5.4) vanishes, as do $|A_<|$ and $|A_>|$, so (5.4) holds if and only if $|B_<| = |B_>|$, or equivalently, the leaf ℓ_{hi} is ranked exactly in the middle of the leaf set L_{hi}. As the ranking ρ is chosen uniformly at random, this happens with probability $1/(b + 1)$ where $b \doteq |B| = |L_{\mathrm{hi}}| - 1$. The case where $a \geq 1$ and $b = 0$ is symmetric. The remaining case, $a = b = 0$, is one we do not need to consider as the tree T then only has two leaves. In all other cases we obtain a strong upper bound on the probability that (5.4) holds, and by complementation a strong lower bound on the root sensitivity. We capture the lower bound in the following single expression that holds for all cases under consideration.

Lemma 5.6. *There exists a constant c such that for every binary tree T with at leaves 3 leaves and a root of degree 2, the root sensitivity of T is at least*

$$1 - \frac{c}{\sqrt{n_1 n_2 (n_1 + n_2)}}, \tag{5.9}$$

where n_1 and n_2 denote the number of leaves in the subtrees rooted by the two children of the root.

5.3 Average sensitivity

We are now ready to establish that, except for trivial cases, the average sensitivity of a binary tree is close to maximal. The trivial cases are those where the tree has at most two leaves, in which case the sensitivity is zero.

Lemma 5.7. *The average sensitivity of Π_T for binary trees T with $n \geq 2$ leaves is $n - O(1)$.*

Proof. We use the expression (5.5) for the average sensitivity of Π_T, where v ranges over all nodes of degree 2 such that T_v contains as least two leaves. Consider a node v of degree 2 such that T_v contains $n_{v,1}$ leaves one one side and $n_{v,2}$ leaves on the other side, where $n_{v,1} + n_{v,2} \geq 3$. If we choose the ranking ρ and the adjacent-rank transposition τ uniformly at random, each of the $\binom{n}{2}$ pairs of leaves are equally likely to be the affected pair. As there are $n_{v,1} \cdot n_{v,2}$ choices that result in v as their lowest common ancestor, we have that $\mathbb{P}[v = \mathrm{LCA}(\ell_{\mathrm{lo}}, \ell_{\mathrm{hi}})] = \frac{2n_{v,1}n_{v,2}}{n(n-1)}$. Combining this with the root sensitivity lower bound given by (5.9), we have that

$$\begin{aligned} s(\Pi_T) &\geq (n-1) \sum_v \frac{2n_{v,1}n_{v,2}}{n(n-1)} \cdot \left(1 - \frac{c}{\sqrt{n_{v,1}n_{v,2}(n_{v,1}+n_{v,2})}}\right) \\ &= (n-1) - \frac{2c}{n} \sum_v \sqrt{\frac{n_{v,1}n_{v,2}}{n_{v,1}+n_{v,2}}}. \end{aligned}$$

The following claim then completes the proof. □

Claim 5.8. *There is a constant c' such that for all binary trees T with n leaves*

$$\sum_v \sqrt{\frac{n_{v,1}n_{v,2}}{n_{v,1}+n_{v,2}}} \leq c'n, \tag{5.10}$$

where the sum ranges over all nodes v of degree 2 such that T_v contains at least 3 leaves.

Proof of Claim 5.8. We use structural induction to prove a somewhat stronger claim, namely that

$$\sum_v \sqrt{\frac{n_{v,1}n_{v,2}}{n_{v,1}+n_{v,2}}} \leq c'n - d'\sqrt{n} \tag{5.11}$$

for some constants c' and d' to be determined. As the base case we consider binary trees T with at most two leaves. In this case, the left-hand side of (5.11) is zero and the right-hand side is non-negative provided $c' \geq d'$, so (5.11) holds.

For the inductive step, the case where the root of T has degree 1 immediately follows from the inductive hypothesis for the subtree T_u rooted by the child u of the root of T. The remaining case is where the root of T has degree 2. Let u_1 and u_2 be the two children for the root, $n_1 = |\mathrm{L}(T_{u_1})|$, and $n_2 = |\mathrm{L}(T_{u_2})|$. The sum on the left-hand side of (5.11) has three contributions: $\sqrt{\frac{n_1 n_2}{n_1+n_2}}$ from the root, and the contributions from T_{u_1} and T_{u_2}, to which we can individually apply the inductive hypothesis. This gives us an upper bound of

$$\sqrt{\frac{n_1 n_2}{n_1+n_2}} + (c'n_1 - d'\sqrt{n_1}) + (c'n_2 - d'\sqrt{n_2}) = \sqrt{\frac{n_1 n_2}{n_1+n_2}} + c'(n_1+n_2) - d'(\sqrt{n_1}+\sqrt{n_2}),$$

which we want to upper bound by

$$c'n - d'\sqrt{n} = c'(n_1+n_2) - d'\sqrt{n_1+n_2}.$$

Writing $n_1 = \alpha n$ for some $\alpha \in [0,1]$ and rearranging terms, the upper bound holds if and only if

$$\sqrt{\alpha(1-\alpha)} \leq d'(\sqrt{\alpha} + \sqrt{1-\alpha} - 1).$$

We claim that the upper bound holds for $d' = (\sqrt{2}+1)/2$. Let

$$F(\alpha) \doteq d'(\sqrt{\alpha} + \sqrt{1-\alpha} - 1) - \sqrt{\alpha(1-\alpha)}.$$

It suffices to show that $F(\alpha) \geq 0$. Since F is continuous on $[0,1]$, it attains a minimum on $[0,1]$. On $(0,1)$, F is differentiable. It can be verified that F' has a unique zero in $(0,1/2)$, which needs to be a maximum as F is increasing at $\alpha = 0$. By the symmetry $F(\alpha) = F(1-\alpha)$, it follows that the minimum of F on $[0,1]$ is attained at the midpoint $\alpha = 1/2$ or at one of the endpoint $\alpha = 0$ or $\alpha = 1$. At all three points $F(\alpha) = 0$. We conclude that (5.11) holds for any constants $d' \geq (\sqrt{2}+1)/2$ and $c' \geq d'$. ◻

6 Sensitivity approach for bounded error

In this section, we apply the sensitivity approach to obtain lower bounds on the query complexity of problems in the comparison-query model against randomized algorithms with bounded error. We derive a general result that query lower bounds against deterministic algorithms that are based on the Sensitivity Lemma also hold against bounded-error randomized algorithms with a small loss in strength. The approach works particularly well when we have linear lower bounds on the average sensitivity, in which case there is only a constant-factor loss in the strength of the query lower bound. Among others, this applies to the $\Omega(n \log n)$ query lower bound for inversion minimization on trees of bounded degree.

Our approach is based on Yao's minimax principle [29], which lower bounds worst-case complexity against randomized algorithms with bounded error by average-case complexity against deterministic algorithms with bounded distributional error. We view a deterministic algorithm with small distributional error for a problem Π as an exact deterministic algorithm for a slightly modified problem Π'. The idea is to then apply the sensitivity approach to Π', and capitalize on the closeness of the average sensitivities of Π and Π' to obtain a lower bound in terms of the sensitivity of Π. By using the Sensitivity Lemma as a black box, the approach yields a lower bound on the query complexity of bounded-error algorithms that is worst-case with respect to the input and with respect to the randomness, i. e., the lower bound holds for some input and some computation path on that input. By delving into the proof of the Sensitivity Lemma, we are able to obtain a lower bound that is worst-case with respect to the input but average-case with respect to the randomness, i. e., the lower bound holds for the expected number of queries on some input.[1]

We first define the notions of randomized complexity and distributional complexity.

Definition 6.1 (randomized query complexity, $\mathrm{RQ}_\cdot(\cdot)$, and distributional query complexity, $\mathrm{DistQ}_\cdot(\cdot,\cdot)$)**.** Let Π be a problem in the comparison-query model and $\varepsilon \in [0,1]$.

A randomized algorithm R for Π is said to have error ε if on every input ρ, the algorithm outputs $\Pi(\rho)$ with probability at least $1-\varepsilon$. The query complexity of R is the maximum, over all inputs ρ, of the expected number of queries that R makes on input ρ. The *ε-error randomized query complexity* of Π, denoted $\mathrm{RQ}_\varepsilon(\Pi)$, is the minimum query complexity of R over all ε-error randomized algorithms R for Π.

Let $\mathcal{D}$ be a probability distribution on the inputs ρ. A deterministic algorithm A for Π has error ε with respect to $\mathcal{D}$ if the probability that $A(\rho) = \Pi(\rho)$ is at least $1-\varepsilon$ where the input ρ is chosen according to $\mathcal{D}$. The query complexity of A with respect to $\mathcal{D}$ is the expected number of queries that A makes on input ρ when ρ is chosen according to $\mathcal{D}$. The *ε-error distributional query complexity* of Π with respect to $\mathcal{D}$, denoted $\mathrm{DistQ}_\varepsilon(\Pi,\mathcal{D})$, is the minimum query complexity of A with respect to $\mathcal{D}$ over all deterministic algorithms A for Π that have error ε with respect to $\mathcal{D}$.

The relationship between randomized complexity and distributional complexity is described by Yao's principle.

Lemma 6.2 (Yao's minimax principle [29])**.** *Let Π be a problem in the comparison-query model, $\varepsilon \in [0,1/2]$, and $\mathcal{D}$ a distribution on the inputs ρ.*

$$\mathrm{RQ}_\varepsilon(\Pi) \geq \frac{1}{2}\,\mathrm{DistQ}_{2\varepsilon}(\Pi,\mathcal{D}). \tag{6.1}$$

We now prove lower bounds on the distributional query complexity, and thus on randomized query complexity, of comparison-query problems Π based on average sensitivity bounds. For these bounds, we always set $\mathcal{D}$ to be the uniform distribution, the distribution underlying the notion of average sensitivity.

We start by studying average-case query complexity, i. e., zero-error distributional query complexity, and its relationship to the average sensitivity. We follow a strategy similar to the

[1]In fact, the approach yields a lower bound that is average-case with respect to the input (chosen uniformly at random) as well as the randomness. This follows because the proof of Yao's minimax principle allows us to replace the left-hand side of (6.1) by the average of the expected number of queries with respect to the distribution $\mathcal{D}$, which we pick to be uniform in our application of the principle.

one in the proof of the Sensitivity Lemma. Whereas a bound on deterministic complexity Q follows purely from the number of execution traces D, here, the execution traces are weighted by their depth and their probability of occurring.

Recall that g in the statement of the Strong Sensitivity Lemma denotes any convex function $g : [1, \infty) \to \mathbb{R}$ with $g(x) = x!$ for $x \in [n]$; for deriving the Sensitivity Lemma from the Strong Sensitivity Lemma we also need g to be nondecreasing. One such function is $g(x) = \Gamma(x+1)$. To prove a lower bound on the zero-error distributional complexity, we need the function g to be not only convex, but *log-convex*, i. e., $\log_2 g(x)$ needs to be convex. The function $g(x) = \Gamma(x+1)$ satisfies this constraint as well.

Proposition 6.3. *Let Π be a problem in the comparison-query model with n items, $\mathcal{D}$ the uniform distribution on the inputs ρ, and $g : [1, \infty) \to \mathbb{R}$ a nondecreasing log-convex function with $g(x) = x!$ for $x \in [n]$.*

$$\mathrm{DistQ}_0(\Pi, \mathcal{D}) \geq \log_2(g(s(\Pi) + 1)/n)$$

Proof. Let k be the number of distinct execution traces of a deterministic algorithm A for Π, and let $R_1, \ldots, R_k$ denote the corresponding sets of rankings. Interpreting A as a binary decision tree, let $d(R_i)$ be the depth of the execution trace corresponding to R_i. By Kraft's inequality,

$$\sum_{i=1}^{k} 2^{-d(R_i)} \leq 1.$$

Let $f(x) = \frac{1}{n}\frac{g(x+1)}{n!}$ and define the weight function $w(\rho) = f(\deg_{\overline{H}}(\rho))$, where $\overline{H}$ refers to the notation of the Strong Sensitivity Lemma: $\overline{H}$ denotes the subgraph of the full permutahedron that only consists of the bichromatic edges when the vertices are colored with their execution trace under A. By Claim 3.3, the sum of the weights of all rankings ρ in R_i is at most 1. Therefore,

$$\sum_{\rho} 2^{-d(\rho)} w(\rho) \leq 1.$$

Dividing both sides by $n!$ and taking the logarithm of both sides, we get that

$$\log_2 \mathbb{E}\left[2^{-d(\rho)} w(\rho)\right] \leq \log_2(1/n!), \tag{6.2}$$

where the expectation is with respect to a uniform distribution over the inputs ρ. By Jensen's inequality, since log is concave, we get

$$\log_2 \mathbb{E}\left[2^{-d(\rho)} w(\rho)\right] \geq \mathbb{E}\left[\log_2\left(2^{-d(\rho)} w(\rho)\right)\right] = \mathbb{E}[-d(\rho)] + \mathbb{E}[\log_2 w(\rho)],$$

which, in combination with (6.2), implies

$$\mathbb{E}[\log_2 w(\rho)] \leq \mathbb{E}[d(\rho)] + \log_2(1/n!).$$

Note that since g is log-convex, so is f. By applying Jensen's inequality again,

$$\mathbb{E}[\log_2 w(\rho)] = \mathbb{E}[\log_2 f(\deg_{\overline{H}}(\rho))] \geq \log_2 f(\mathbb{E}[\deg_{\overline{H}}(\rho)]),$$

implying

$$\log_2\left(\frac{1}{n}\cdot\frac{g(\mathbb{E}[\deg_{\overline{H}}(\rho)]+1)}{n!}\right) \leq \mathbb{E}[d(\rho)] + \log_2(1/n!),$$

or equivalently,

$$\mathbb{E}[d(\rho)] \geq \log_2(g(\mathbb{E}[\deg_{\overline{H}}(\rho)]+1)/n).$$

The result follows since A is an arbitrary deterministic algorithm for Π, $\mathbb{E}[d(\rho)]$ equals the query complexity of A with respect to the uniform distribution $\mathcal{D}$, $\mathbb{E}[\deg_{\overline{H}}(\rho)] \geq \mathbb{E}[\deg_{\overline{G(\Pi)}}(\rho)] = s(\Pi)$, and g is nondecreasing. □

Proposition 6.3 allows us to prove a lower bound on the ε-error distributional query complexity of Π with respect to the uniform distribution. In order to do so, we view a deterministic algorithm with distributional error ε for Π as an exact deterministic algorithm for a modified problem Π', apply Proposition 6.3, and lower bound the sensitivity of Π' in terms of the sensitivity of Π.

Proposition 6.4. *Let Π be a problem in the comparison-query model with n items, $\mathcal{D}$ the uniform distribution on the inputs ρ, $\varepsilon \in [0,1]$, and $g : [1,\infty) \longrightarrow \mathbb{R}$ a nondecreasing log-convex function with $g(x) = x!$ for $x \in [n]$.*

$$\mathrm{DistQ}_\varepsilon(\Pi,\mathcal{D}) \geq \log_2\left(\frac{g(s(\Pi)+1-2(n-1)\varepsilon)}{n}\right) \tag{6.3}$$

Proof. Consider any algorithm A with error ε for Π, or in other words, $\mathbb{P}[A(\rho) \neq \Pi(\rho)] \leq \varepsilon$. Let Π_A be the problem of determining the output of A. We prove that

$$s(\Pi_A) \geq s(\Pi) - 2(n-1)\varepsilon,$$

which implies the desired result by Proposition 6.3, since A is a deterministic algorithm for Π_A and g is nondecreasing.

Let G denote the full permutahedron graph for n items. We use the fact that $s(\Pi) = (n-1)\cdot\mathbb{P}_{e\in G}[e \in G(\Pi)]$, and similarly, $s(\Pi_A) = (n-1)\cdot\mathbb{P}_{e\in G}[e \in G(\Pi_A)]$, where all the underlying distributions are uniform. Suppose the endpoints of e are ρ_1 and ρ_2. Note that if $e \in G$ is picked uniformly at random, then the marginal distributions of both ρ_1 and ρ_2 are also uniform. If $A(\rho_1) = \Pi(\rho_1)$, $A(\rho_2) = \Pi(\rho_2)$, and $e \in G(\Pi_A)$, then $e \in G(\Pi)$, as well. By a union bound, the probability that $A(\rho_1) \neq \Pi(\rho_1)$ or $A(\rho_2) \neq \Pi(\rho_2)$ is at most 2ε.

$$\begin{aligned}\mathbb{P}_{e\in G}[e \in G(\Pi_A)] &\geq \mathbb{P}_{e\in G}[e \in G(\Pi)] - \mathbb{P}_{e\in G}[A(\rho_1) \neq \Pi(\rho_1) \text{ or } A(\rho_2) \neq \Pi(\rho_2)]\\ &\geq \mathbb{P}_{e\in G}[e \in G(\Pi)] - 2\varepsilon.\end{aligned}$$

Multiplying both sides by $n-1$ gives $s(\Pi_A) \geq s(\Pi) - 2(n-1)\varepsilon$. □

Since $s(\Pi) \leq n-1$, Proposition 6.4 only yields nontrivial lower bounds for small ε. In order to establish lower bounds for the standard $\varepsilon = 1/3$, we first reduce the error using standard techniques. Doing so such that the argument of g on the right-hand side of (6.3) remains $\Omega(s(\Pi))$, and picking $g(x) = \Gamma(x+1)$, we conclude:

Lemma 6.5 (Bounded-Error Sensitivity Lemma)**.** *For any problem Π in the comparison-query model with n items,*

$$\mathrm{RQ}_{1/3}(\Pi) = \Omega\left(\frac{s \log s}{\log(2n/s)}\right),$$

where $s \doteq s(\Pi)$.

Proof. By taking the majority vote of multiple independent runs and a standard analysis, e. g., based on Chernoff bounds, we have that $\mathrm{RQ}_\varepsilon(\Pi) = O(\log(1/\varepsilon))\,\mathrm{RQ}_{1/3}(\Pi)$ for any $\varepsilon \leq 1/3$. Combining this with Lemma 6.2 and Proposition 6.4, we have:

$$\mathrm{RQ}_{1/3}(\Pi) = \Omega\left(\frac{\mathrm{RQ}_\varepsilon(\Pi)}{\log(1/\varepsilon)}\right) = \Omega\left(\frac{\mathrm{DistQ}_{2\varepsilon}(\Pi)}{\log(1/\varepsilon)}\right) = \Omega\left(\frac{\log(g(s+1-4(n-1)\varepsilon)/n)}{\log(1/\varepsilon)}\right).$$

Setting ε such that $4n\varepsilon = s/2$ yields

$$\mathrm{RQ}_{1/3}(\Pi) = \Omega\left(\frac{\log(g(s/2+1)/n)}{\log(8n/s)}\right).$$

Picking $g(x) = \Gamma(x+1)$ and using the fact that $\Gamma(x) \geq \sqrt{2\pi x}\left(\frac{x}{\mathrm{e}}\right)^x$, we obtain

$$\mathrm{RQ}_{1/3}(\Pi) = \Omega\left(\frac{(s/2)\log(s/(2\mathrm{e})) - \log n}{\log(8n/s)}\right) = \Omega\left(\frac{s \log s}{\log(2n/s)}\right),$$

where the simplification can be verified by considering the cases of large s (say $s \geq \sqrt{n}$) and small s separately. ◻

We can apply Lemma 6.5 to the sensitivity lower bounds of Lemma 4.1 and produce randomized lower bounds for inversion minimization on bounded-degree trees. Using Fact 4.5 we obtain:

Theorem 6.6 (lower bound against bounded-error for inversion minimization on trees)**.** *Let T be a tree with $\deg(T) \leq k$. The query complexity of Π_T for bounded-error randomized algorithms is $\Omega(\frac{n\log(n/k)}{k\log(k)})$.*

7 Connectivity Lemma

In this section we establish Lemma 1.16 and use it to present some of the known lower bounds in a unified framework. We actually prove the following somewhat stronger result.

Lemma 7.1 (Strong Connectivity Lemma)**.** *Consider an algorithm A in the comparison-based model, color each vertex of the permutahedron with its execution trace under A, and let H denote the subgraph with the same vertex set but only containing the monochromatic edges. The number of distinct execution traces of A equals the number of connected components of H.*

The Connectivity Lemma follows from Lemma 7.1 because the coloring with execution traces of an algorithm A for Π is a refinement of the coloring with Π. Note that the counterpart of Lemma 7.1 in the Boolean setting is trivial. This is because an execution trace in the Boolean setting is specified by values for a subset of the input bits, so the set of inputs that follow a particular execution trace form a subcube of the hypercube, the Boolean counterpart of the permutahedron. Subcubes are trivially connected inside the hypercube. In the comparison-query model, the sets of inputs that follow a particular execution trace can be more complicated, and their connectedness is no longer trivial but still holds.

Proof of Lemma 7.1. Two rankings ρ_1 and ρ_2 that have distinct execution traces under A cannot be connected because any path between them needs to contain at least one bichromatic edge. For the remainder of the proof, we consider two rankings ρ_1 and ρ_2 that have the same execution trace under A, and construct a path from ρ_1 to ρ_2 in H.

If $\rho_1 = \rho_2$, we do not need to make any move and use an empty path.

Otherwise, there exists a rank $r < n$ such that ρ_1 and ρ_2 agree on ranks less than r and disagree on rank r. We have the following situation, where the item y_r with rank r under ρ_2, has rank $s > r$ under ρ_1.

$$
\begin{array}{r|ccccccccc}
\text{rank} & 1 & \cdots & r-1 & r & \cdots & s-1 & s & \cdots & n \\
\hline
\rho_1^{-1} & x_1 & \cdots & x_{r-1} & x_r & \cdots & x_{s-1} & x_s = y_r & \cdots & \\
 & = & = & = & \neq & & & & & \\
\rho_2^{-1} & y_1 & \cdots & y_{r-1} & y_r & \cdots & & & \cdots &
\end{array}
$$

Considering ranking ρ_1, we have that $\rho_1(x_{s-1}) = s-1 < s = \rho_1(x_s)$. Considering ranking ρ_2, since x_{s-1} differs from $y_i = x_i$ for every $i \in [r-1]$ and also differs from y_r, we have that $\rho_2(x_{s-1}) > r = \rho_2(y_r) = \rho_2(x_s)$. Thus, the relative ranks of x_{s-1} and x_s under ρ_1 and ρ_2 differ. As ρ_1 and ρ_2 have the same execution trace, this means that the algorithm does not compare x_{s-1} and x_s on either input, and on ρ_1 in particular. Let ρ'_1 be the ranking obtained from ranking ρ_1 by applying the adjacent-rank transposition $\tau = (s-1, s)$. Since the algorithm does not compare the affected items, the execution trace for ρ'_1 and ρ_1 are the same, so the edge from ρ'_1 to ρ_1 is monochromatic and in H. We use this edge as the first on the path from ρ_1 to ρ_2 in H. What remains is to find a path from ρ'_1 to ρ_2 in H. The situation is the same as the one depicted above but with r increased by one in case $s = r+1$, and with the same r and s decreased by one, otherwise. The proof then follows by induction on the ordered pair $(r, n-s)$. ◻

Remark 7.2. Suppose we allow an algorithm A to have multiple valid execution traces on a given input ρ, and let R denote the set of rankings on which a particular execution trace is valid. The construction in the proof of Lemma 7.1 yields a path in the permutahedron between any two rankings in R such that the path entirely stays within R. This means that we can replace $\mathrm{D}(\Pi)$ in the statement of the Connectivity Lemma by its nondeterministic variant $\mathrm{N}(\Pi)$.

The Connectivity Lemma captures all the prior lower bounds stated in Section 2 except the elementary adversary argument (which is also based on connectivity considerations, but in an undirected graph other than H, namely (X, E) where E denotes the queries the algorithm makes on a given input ranking ρ). It captures the information-theoretic lower bound because input rankings with different outputs cannot belong to the same connected component of H. We already explained in Section 1 how the Connectivity Lemma shows that counting inversions and inversion parity amount to sorting, and require at least $\log(n!)$ queries. We now illustrate its use for a classical problem that is easier than sorting, namely, median finding.

Let Π denote the selection problem with rank $r = \lceil n/2 \rceil$. For any ranking, the adjacent-rank transpositions τ that change the item with rank r are the two that involve rank r: $\tau = (r-1, r)$ and $\tau = (r, r+1)$. Those transpositions are the ones that correspond to missing edges in the permutahedron graph $G(\Pi)$. As a result, for any two rankings, there exists a path between them in $G(\Pi)$ if and only if they have the same median as well as the same set of items with rank less than r (and also the same set of items with rank greater than r). As there are n possibilities for the median and, for each median, $\binom{n-1}{r-1}$ possibilities for the set of items that have rank less than r, $G(\Pi)$ has $n \cdot \binom{n-1}{r-1}$ connected components. It follows that any algorithm for Π has at least $n \cdot \binom{n-1}{r-1} = \Omega(\sqrt{n} \cdot 2^n)$ distinct execution paths, and therefore needs to make at least $n + \frac{1}{2}\log(n) - O(1)$ queries.

As a side note, this example clarifies a subtlety in the equivalence between ordinary selection and the instantiation of partial order production that is considered equivalent to selection. Whereas selection of rank r ordinarily requires outputting only the item of rank r, the instantiation of partial order production additionally requires partitioning the remaining items according to whether their ranks are less than or greater than r. The above analysis implies that it is impossible for the algorithm to know the item of rank r without also knowing how to partition the remaining items into those of rank less than and greater than r. It follows that, in the comparison-based model, ordinary selection and the instantiation of partial order production are equivalent.

8 Connectivity approach

This section covers the connectivity approach for obtaining query lower bounds in the comparison-query model. Our main focus is the problem Π_T of inversion minimization on a fixed tree T, for which we derive very strong query lower bounds in the case of the special types of trees in Theorem 1.8. Some parts of the analysis carry through for a broader class of problems Π, namely those that satisfy a certain partition property. We first develop the property and apply the Connectivity Lemma to a general problem Π with the property. We then present sufficient conditions for the problem Π_T to have the property and perform a detailed analysis, leading to Theorem 1.8. Finally, we apply the same ideas to the problem of counting cross inversions, for which we obtain the query lower bound of Theorem 1.9, as well as to the closely related problem of inversion minimization on the Mann–Whitney trees of Figure 2.

8.1 Partition property

In order to obtain good lower bounds on $\mathrm{D}(\Pi)$ using the Connectivity Lemma, it is sufficient to find good upper bounds on the size of the connected components of a typical vertex in $G(\Pi)$. For the problem Π_T, we can assume without loss of generality that T has no internal nodes of degree 1, i. e., no nodes with exactly one child. With that assumption, Π_T is insensitive to any adjacent-rank transposition τ at a ranking ρ for which the affected leaves are siblings in T. Thus, the corresponding edges from the permutahedron are always present in $G(\Pi_T)$. From the perspective of ensuring small connected components in $G(\Pi_T)$, the ideal situation would be if there were no other edges in $G(\Pi_T)$. That is to say, Π_T is *sensitive* at ρ to *every* adjacent-rank transposition τ *except* when the affected leaves are siblings. We will investigate conditions on T that guarantee this situation in the next two subsections. In this subsection, we analyze the size of the connected components of $G(\Pi_T)$ when T is of the desired type, and use it obtain a query lower bounds via the Connectivity Lemma. Our analysis applies more generally to any problem Π with the following property.

Definition 8.1 (partition property)**.** A computational problem Π in the comparison-query model on a set X of n items has the *partition property* if the set X can be partitioned into sets X_i such that for any ranking ρ of X and adjacent-rank transposition $\tau = (r, r+1)$ with $r \in [n-1]$, $\Pi(\rho) = \Pi(\tau\rho)$ if and only if $\rho^{-1}(r)$ and $\rho^{-1}(r+1)$ belong to the same partition class X_i. If every partition class X_i has size at most k, we say that Π has the partition property with class size at most k.

In other words, a problem Π has the partition property if the underlying universe can be partitioned in such a way that adjacent-rank transpositions that do not change the answer are exactly those whose affected items fall within the same partition class. In the case of the problem Π_T, the partition classes X_i correspond to the leaf child sets $\mathrm{LC}(v)$ from Definition 1.7, where v ranges over the leaf parents.

Let us investigate the size of the connected components of $G(\Pi)$ when Π satisfies the partition property. Consider a walk in $G(\Pi)$. As the only steps we can take correspond to adjacent-rank transpositions τ that swap elements in the same partition class, the sets X_i remain invariant, irrespective of the ranking ρ we start from. Depending on ρ, there may be more structure inside each partition class X_i; the set X_i may be broken up into smaller subsets that are each invariant. For our analysis, we list the elements of each partition class in order of increasing rank under ρ, and include an edge between elements that have successive ranks. We introduce the term "successor graph" to capture this structure, viewed as a graph with the ranks as vertices.

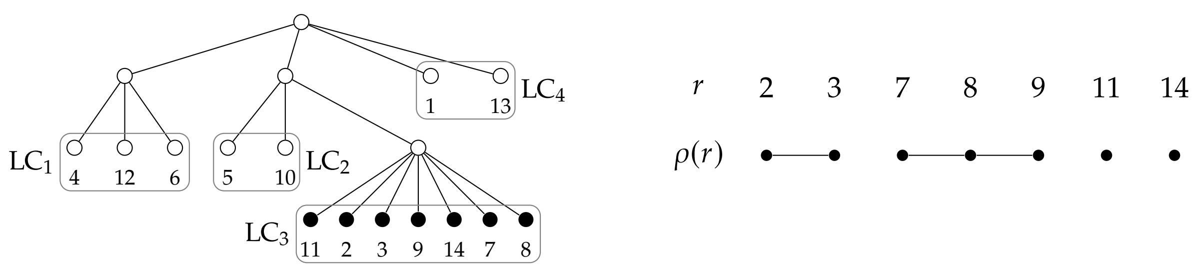
Definition 8.2 (successor graph, $S(\cdot,\cdot)$)**.** Let Π be a computational problem in the comparison-query model on a set X of n items, and ρ a ranking of X. The successor graph of Π on ρ, denoted $S(\Pi, \rho)$, has vertex set $[n]$ and contains all edges of the form $(r, r+1)$ with $r \in [n-1]$ such that $\Pi(\rho) = \Pi(\tau\rho)$, where τ denotes the adjacent-rank transposition $(r, r+1)$.

We have the following connection.

Proposition 8.3. *Let Π be a computational problem in the comparison-query model on the set X, and let ρ be a ranking of X. If Π has the partition property, then the connected component of ρ in $G(\Pi)$ has size $\prod_j(n_j!)$, where the n_j's denote the sizes of the connected components of $S(\Pi, \rho)$.*

Proof. The connected components of the successor graph $S(\Pi, \rho)$ correspond to subsets of the classes X_i that each remain invariant under walks in $G(\Pi)$. Within each of the subsets, independently for each subset, every possible ordering can be realized by such walks. This is because for any adjacent-rank transposition τ, the successor graphs $S(\Pi, \rho)$ and $S(\Pi, \tau\rho)$ are the same, and every ordering can be realized by a sequence of swaps of adjacent elements. It follows that the number of rankings that can be reached from ρ in $G(\Pi)$ equals the product over all connected components of $S(\Pi, \rho)$ of the number of possible orderings of the elements in the connected component. □

Figure 10 depicts an example for a problem of type Π_T and a partition consisting of 4 classes, namely the leaf child sets $\mathrm{LC}_1, \mathrm{LC}_2, \mathrm{LC}_3$ and LC_4. The tree T and ranking ρ are represented in Figure 10a. Figure 10b represents the part of the successor graph $S(\Pi_T, \rho)$ involving the leaf child set LC_3 and illustrates the subpartitioning into invariant subsets.



(a) Leaf child sets and leaf ranks under ρ

(b) Inside leaf child set LC_3

Figure 10: Connected component analysis

If each of the partition classes X_i has size at most k, then each of the connected components of $S(\Pi, \rho)$ has size $n_j \leq k$, irrespective of ρ. The maximum value that $\prod_j(n_j!)$ can take under the constraints $\sum_j n_j = n$ and $n_j \leq k$ is no more than $(k!)^{n/k}$. By the Connectivity Lemma, we conclude that $\mathrm{D}(\Pi) \geq n!/(k!)^{n/k}$, and that the query complexity is at least $\log_2(n!) - O(n\log(k))$.

We can do better by observing that, for a random ranking ρ, the number of adjacent-rank transpositions τ that do not jump from one partition class to another is not much larger than the average size of the partition classes.

Lemma 8.4 (lower bound for problems with the partition property)**.** *Let Π be a computational problem in the comparison-query model on a set of size n. If Π satisfies the partition property with class size at most k, then then* $\mathrm{D}(\Pi) \geq n!/(2(k!)^2)$.

Proof. For any rank $r \in [n-1]$, the probability that $\rho^{-1}(r)$ and $\rho^{-1}(r+1)$ belong to the same partition class equals $\sum_i \frac{|X_i|}{n}\frac{|X_i|-1}{n-1}$, which is at most $\frac{k-1}{n-1}$ provided each partition class X_i has size at most k. It follows that the expected number of adjacent-rank transpositions τ that do not change partition class, is at most $k-1$, so for a fraction at least half of the rankings ρ the number is at most $2(k-1)$.

The number of adjacent-rank transpositions τ that do not change partition class for a given ranking ρ equals the number of edges in the successor graph $S(\Pi, \rho)$. In terms of the sizes n_j, the number equals $\sum_j (n_j - 1)$. We are considering rankings ρ for which the sum is at most $2(k-1)$. The maximum of $\prod_j (n_j!)$ under the constraints that $\sum_j (n_j - 1) \leq 2(k-1)$ and that each individual $n_j \leq k$, is reached when two of the n_j's equal k and the rest are 1. Thus, if each of the partition classes X_i are of size at most k, for a fraction at least half of the rankings ρ, the size of the connected component of ρ in $G(\Pi)$ is at most $(k!)^2$. It follows that the number of connected components of $G(\Pi)$ is at least $n!/(2(k!)^2)$. The Connectivity Lemma then yields the claimed lower bound on $\mathrm{D}(\Pi)$. □

Lemma 8.4 yields a lower bound of $\log(n!) - O(k \log(k))$ on the query complexity of Π whenever Π satisfies the partition property with class size at most k.

Next we turn to sufficient conditions on the tree T that guarantee the partition property for Π_T. For didactic reasons we first develop the conditions for binary trees, and then generalize them to arbitrary trees.

8.2 Binary trees

In the case of binary trees T, the sensitivity analysis of Section 5.1 leads to a simple sufficient condition for the partition property to hold for Π_T. Recall that we are assuming without loss of generality that T has no internal nodes of degree 1, which in the case of binary trees is equivalent to saying that the tree is full: Every internal node has the maximum degree of 2.

Consider criterion 5.3 in Proposition 5.2. The right-hand side is always -1. As for the left-hand side, we know the following.

Fact 8.5. *For all disjoint sets $A, B \subseteq X$ and any ranking ρ of X, $\mathrm{DInv}_\rho(A, B) = |A| \cdot |B| \bmod 2$.*

Proof. As every pair in $A \times B$ constitutes a cross-inversion for either A to B, or B to A, we have $\mathrm{XInv}_\rho(A, B) + \mathrm{XInv}_\rho(B, A) = |A| \cdot |B|$. Thus,

$$\begin{aligned}
\mathrm{DInv}_\rho(A, B) &\doteq \mathrm{XInv}_\rho(A, B) - \mathrm{XInv}_\rho(B, A) \\
&= (\mathrm{XInv}_\rho(A, B) + \mathrm{XInv}_\rho(B, A)) - 2\,\mathrm{XInv}_\rho(B, A) \\
&= |A| \cdot |B| - 2\,\mathrm{XInv}_\rho(B, A). \qquad (8.1)
\end{aligned}$$

As $\mathrm{XInv}_\rho(B, A)$ in an integer, the claim follows. □

Fact 8.5 implies that whenever at least one of the leaf sets L_{lo} or L_{hi} is of even cardinality, then (5.3) fails to hold, and Π_T is sensitive to the underlying τ at ρ. Thus, we can guarantee that Π_T satisfies the partition property provided that for any two siblings u_1 and u_2 in T that are not both leaves, at least one of $|\mathrm{L}(T_{u_1})|$ or $|\mathrm{L}(T_{u_2})|$ is even. We refer to the latter condition as the *product condition*. In trees without nodes of degree 1, the product condition can be expressed alternately in terms of the leaf child sets. We state and prove the result for arbitrary trees as it will help us in the next subsection to generalize the analysis.

Proposition 8.6. *Let T be a tree without nodes of degree 1. The following two conditions are equivalent:*

(a) For any two siblings u_1 and u_2 that are not both leaves, at least one of $|\mathrm{L}(T_{u_1})|$ or $|\mathrm{L}(T_{u_2})|$ is even.

(b) At most one leaf child set is odd, and if there exists a node v^* with an odd leaf child set $\mathrm{LC}(v^*)$, then all ancestors of v^* have an empty leaf child set.

In the case of binary trees, (b) can be simplified to: At most one leaf has a non-leaf sibling.

Proof. We establish the two directions of implication separately.

⇒: We argue the contrapositive. Suppose that at least two of the leaf child sets are odd. Start with the root of T as the node v, and iterate the following: If v has a child u such that T_u contains at least two nodes with an odd leaf child set, replace v by such a child u. When the process ends, one of the following situations applies:

- There are two distinct children u_1 and u_2 of v that are not leaves and each contain a single node with an odd leaf child set. In this case both T_{u_1} and T_{u_2} contain an odd number of leaves, violating (a).
- There exists a unique child u_1 of v that is not a leaf and contains a single node with an odd leaf child set, and v itself has an odd number of leaf children. In this case, setting u_2 to any one leaf child of v (which exists as their number is odd), leads to a violation of (a).

Next, suppose that there exists a unique node v^* that has an odd leaf child set, and that an ancestor v of v^* has a leaf child u_1. Setting u_2 to the child of v that contains v^* in its subtree, yields a violation of (a) as T_{u_2} contains an odd number of leaves.

⇐: If neither u_1 nor u_2 are leaves, the first condition of (b) guarantees that at most one of T_{u_1} or T_{u_2} contains an odd number of leaves. If u_1 is a leaf and u_2 is not, then the second condition of (b) implies that T_{u_2} cannot contain a node with an odd leaf child set, and therefore has an even number of leaves.

In the case of binary trees, the first condition of (b) implies the second one, which can therefore be dropped from the equivalence statement. Moreover, for binary trees the first condition of (b) can be expressed as: At most one leaf has a non-leaf sibling. □

By Proposition 8.6, in a binary tree the condition that at most one leaf has no sibling is equivalent to the product condition, which implies the partition property of Π_T, so the lower bound of Lemma 8.4 applies. This establishes Theorem 1.8 in the case of binary trees.

As a side note, Fig. 11 shows the simplest example of a full binary tree T and a ranking ρ for which there exists an adjacent-rank transposition τ to which Π_T is insensitive at ρ while the affected leaves are not siblings. The tree has two leaves without siblings, namely 1 and 2. The adjacent-rank transposition $(3,4)$ acts on nodes that are not siblings, but leaves the minimum number of inversions at 4.

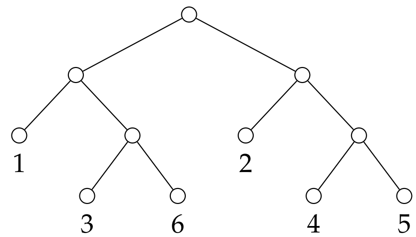


Figure 11: Tree insensitive to non-sibling transposition

8.3 General trees

For general trees T the sensitivity analysis of Π_T becomes more complicated than for binary trees, and we do not know of a simple sensitivity criterion like Proposition 5.2, but we can nevertheless extend the result for binary trees to arbitrary trees with similar constraints. For a given ranking ρ of $\mathrm{L}(T)$ and a given adjacent-rank transposition τ, we would like to figure out the effect of τ on the objective $\mathrm{MinInv}(T,\cdot)$, in particular when $\mathrm{MinInv}(T,\tau\rho) = \mathrm{MinInv}(T,\rho)$. Recall the decomposition (4.3) of $\mathrm{MinInv}(T,\cdot)$ from Section 4.2. By Proposition 4.7 the only term on the right-hand side of (4.3) that can be affected by the transposition τ is

$$\mathrm{MinRInv}(T_v,\rho) \doteq \min_{\sigma} \mathrm{RInv}(T,\rho,\sigma)$$

corresponding to the node v that is the least common ancestor $\mathrm{LCA}(\ell_{\mathrm{lo}},\ell_{\mathrm{hi}})$ of the two leaves ℓ_{lo} and ℓ_{hi} that are affected by τ under ρ. In Section 5 we considered the two possible relative orderings σ_1 and σ_2 of the children of v, and derived a criterion for when the lowest cost does not change under τ. More precisely, when

$$\min(\mathrm{RInv}(T,\rho,\sigma_1),\mathrm{RInv}(T,\rho,\sigma_2)) = \min(\mathrm{RInv}(T,\tau\rho,\sigma_1),\mathrm{RInv}(T,\tau\rho,\sigma_2)). \tag{8.2}$$

There are two complications in generalizing this approach from binary to general trees.

- The expression (4.2) for $\mathrm{RInv}(T,\rho,\sigma)$ involves multiple terms instead of just one as in (5.1). This complicates probabilistic analyses like the one we did in Section 5 because the difference in cost of the two relative orderings of two children is also affected by parts of the tree outside of their combined subtrees. The issue did not matter for the analysis in Section 4. We will be able to manage it here, as well.

- There now are not just two but multiple possible orderings σ, and it is not clear what pairs (σ_1,σ_2) we need to impose (8.2) on in order to guarantee that $\mathrm{MinRInv}(T_v,\rho) = \mathrm{MinRInv}(T_v,\tau\rho)$ but no more.

In Section 4 we circumvented the second issue by only considering sensitivities that decrease the objective function, and establishing a lower bound on their occurrence independent of the ordering σ. Here we are also able to handle the second issue by shooting for a sufficient condition for sensitivity rather than a criterion. We do so by requiring that for no pair of distinct orderings σ_1 and σ_2, condition (8.2) holds (unless the two affected leaves are siblings). Similar to the case of binary trees, we guarantee that (8.2) fails based on parity considerations given the product condition.

For the analysis we again assume without loss of generality that T has no nodes of degree 1. We use the same notation as in Section 5: Let ℓ_{lo} denote the affected leaf that is smaller with respect to ρ, and ℓ_{hi} the other affected leaf. Let L_i denote the leaf set $L_i \doteq \mathrm{L}(T_{u_i})$, where $u_1, \ldots, u_k$ are the children of $v = \mathrm{LCA}(\ell_{\text{lo}}, \ell_{\text{hi}})$. We also write u_{lo} for the child of v that contains ℓ_{lo} in its subtree, and L_{lo} for the leaf set of the subtree rooted at u_{lo}, and define ℓ_{hi}, u_{hi}, and L_{hi} similarly. See Figure 12 for a sketch of the setting.

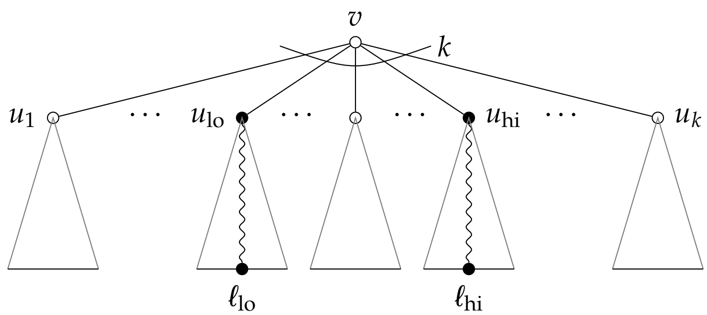


Figure 12: Sensitivity for general trees

We slightly abuse notation and let σ denote both the ordering of the entire tree T as well as the ranking of the children of v. By the analysis of Section 5, we have that for two orderings σ_1 and σ_2 the situation (8.2) can only occur if $\sigma_1(u_{\text{lo}}) < \sigma_1(u_{\text{hi}})$, $\sigma_2(u_{\text{lo}}) > \sigma_2(u_{\text{hi}})$, and

$$\mathrm{RInv}_\rho(T_v, \rho, \sigma_1) - \mathrm{RInv}_\rho(T_v, \rho, \sigma_2) = -1 = \mathrm{DInv}_\rho(\{\ell_{\text{lo}}\}, \{\ell_{\text{hi}}\}). \tag{8.3}$$

For any two disjoint sets of leaves, (8.1) lets us write

$$\mathrm{XInv}_\rho(A, B) = \frac{1}{2}\,\mathrm{DInv}_\rho(A, B) + \frac{1}{2}|A| \cdot |B|. \tag{8.4}$$

Applying (8.4) to all the terms involved in (4.2), we have

$$\begin{aligned}
\mathrm{RInv}_\rho(T_v, \rho, \sigma) &= \sum_{1 \le i < j \le k} \mathrm{XInv}_\rho(L_i, L_j) \cdot \mathbb{I}[\sigma(i) < \sigma(j)] \\
&\quad + \sum_{1 \le i < j \le k} \mathrm{XInv}_\rho(L_j, L_i) \cdot \mathbb{I}[\sigma(i) > \sigma(j)] \\
&= \frac{1}{2} \sum_{1 \le i < j \le k} \left(\mathrm{DInv}_\rho(L_i, L_j) \cdot (-1)^{\mathbb{I}[\sigma(i > \sigma(j)]} + |L_1| \cdot |L_2| \right) \\
\mathrm{RInv}_\rho(T_v, \rho, \sigma_1) - \mathrm{RInv}_\rho(T_v, \rho, \sigma_2) &= \sum_{\substack{1 \le i < j \le k \\ \sigma_1(i) < \sigma_2(j) \\ \sigma_2(i) > \sigma_2(j)}} \mathrm{DInv}_\rho(L_i, L_j) + \sum_{\substack{1 \le i < j \le k \\ \sigma_1(i) > \sigma_2(j) \\ \sigma_2(i) < \sigma_2(j)}} \mathrm{DInv}_\rho(L_i, L_j)
\end{aligned}$$

By combining the last equation with (8.3) and separating out the term for $(i, j) = (\mathrm{lo}, \mathrm{hi})$, we obtain the following necessary condition for (8.2) to hold:

$$\mathrm{DInv}_\rho(L_{\mathrm{lo}}, L_{\mathrm{hi}}) - \mathrm{DInv}_\rho(\{\ell_{\mathrm{lo}}\}, \{\ell_{\mathrm{hi}}\}) = -\sum_{\substack{1 \le i < j \le k \\ \sigma_1(i) < \sigma_2(j) \\ \sigma_2(i) > \sigma_2(j) \\ (i,j) \neq (\mathrm{lo},\mathrm{hi})}} \mathrm{DInv}_\rho(L_i, L_j) + \sum_{\substack{1 \le i < j \le k \\ \sigma_1(i) > \sigma_2(j) \\ \sigma_2(i) < \sigma_2(j)}} \mathrm{DInv}_\rho(L_i, L_j). \quad (8.5)$$

In order for Π_T to have the partition property, it suffices to ensure that (8.5) fails whenever u_{lo} and u_{hi} are not both leaves. By (8.1) each of the terms $\mathrm{DInv}_\rho(L_i, L_j)$ in (8.5) has the same parity as $|L_i| \cdot |L_j|$. Since $\mathrm{DInv}_\rho(\{\ell_{\mathrm{lo}}\}, \{\ell_{\mathrm{hi}}\})$ is odd, it follows that (8.5) fails whenever at most one of the leaf sets L_i involved is odd, which is condition (a) in Proposition 8.6. Switching to the equivalent condition (b) from Proposition 8.6 allows us to conclude via Lemma 8.4:

Theorem 8.7. *Let T be a tree without nodes of degree 1 such that the leaf child sets have size at most k, at most one of them is odd, and if there exists an odd one, say $\mathrm{LC}(v^*)$, then all ancestors of v^* have empty leaf child sets. Then $\mathrm{D}(\Pi_T) \geq n!/(2(k!)^2)$.*

Theorem 1.8 follows by taking the base-2 logarithm of the bound.

8.4 Counting cross inversions and evaluating the Mann–Whitney statistic

We now apply the connectivity approach that we captured in Proposition 8.3 to the problem Π_{XInv} of computing the number of cross inversions between two disjoint sets A and B with respect to a ranking ρ of $X = A \sqcup B$. Note that this problem is a refinement of evaluating the Mann–Whitney statistic, or equivalently, of inversion minimization on the tree T of Figure 2: Any algorithm that solves Π_{XInv} with q queries, can be transformed into an algorithm for Π_T with q queries, namely by transforming the output y of the algorithm for Π_{XInv} to $\min(y, |A| \cdot |B| - y)$. Viewed in the contrapositive, a lower bound for Π_{XInv} is easier to obtain than one for Π_T on the tree T of Figure 2. We first establish a lower bound for Π_{XInv} and then see how it extends to Π_T.

One can think of Π_{XInv} as inversion minimization on the Mann–Whitney tree without allowing swapping the two children of the root. As a result, the problem Π_{XInv} is sensitive to *every* adjacent-rank transposition between non-siblings, and therefore automatically satisfies the partition property (with A and B being the partition classes), so Proposition 8.3 applies. In contrast, the problem Π_T of minimizing inversions on the Mann–Whitney tree may not have the partition property. This is why analyzing Π_{XInv} is a bit simpler, and why we handle it first.

Let $a \doteq |A|$ and $b \doteq |B|$. By the partition property, the average sensitivity of Π_{XInv} equals $\frac{2ab}{a+b}$, which via the Sensitivity Lemma yields a query lower bound of $\Omega(a \log(a))$ for $a \leq b$. To obtain the stronger lower bound of $\Omega((a + b) \log(a))$ we need a more detailed analysis of the connectivity of the permutahedron graph $G(\Pi_{\mathrm{XInv}})$.

For a given ranking ρ, let $x_1, \ldots, x_a$ be the elements of A listed in increasing order, and similarly for $y_1, \ldots, y_b$ for the elements of B. We define $m_1, \ldots, m_{b+1}$ such that for each i, m_i is the number of elements of A between y_{i-1} and y_i. (Here, y_0 and y_{b+1} serve as sentinels with an infinitely low and infinitely high rank.) Similarly, we define $n_1, \ldots, n_{a+1}$ as the number of elements in B between successive elements of A. The numbers m_i and n_i are the sizes of the connected components of the successor graph $S(\Pi_{\mathrm{XInv}}, \rho)$ (possibly with some additional zeroes). By Proposition 8.3, the connected component of ρ in $G(\Pi_{\mathrm{XInv}})$ has size

$$(m_1)! \cdots (m_{b+1})!(n_1)! \cdots (n_{a+1})!. \tag{8.6}$$

Depending on the values of $m_1, \ldots, m_{b+1}, n_1, \ldots, n_{a+1}$, some connected components may be much larger than others. We apply the Connectivity Lemma in a similar way as in Lemma 8.4 and only count the rankings ρ that are in small connected components, which are the rankings for which $m_1, \ldots, m_{b+1}, n_1, \ldots, n_{a+1}$ are bounded. Let m^* and n^* be the minimum integers for which

$$\mathbb{P}[m_1, \ldots, m_{b+1} \leq m^*] \geq \frac{3}{4} \quad \text{and} \quad \mathbb{P}[n_1, \ldots, n_{a+1} \leq n^*] \geq \frac{3}{4}.$$

By a union bound, the probability that both of these events hold is at least $1/2$. In other words, there are least $(a+b)!/2$ rankings ρ for which $m_1, \ldots, m_{b+1} \leq m^*$ and $n_1, \ldots, n_{a+1} \leq n^*$.

Proposition 8.8.

$$\mathrm{D}(\Pi_T) \geq \frac{(a+b)!}{2(m^*)!^{a/m^*}(n^*)!^{b/n^*}}. \tag{8.7}$$

Proof. We consider the rankings for which $m_1, \ldots, m_{b+1} \leq m^*$ and $n_1, \ldots, n_{a+1} \leq n^*$. We first argue the following upper bound on the size (8.6) of the connected component in $G(\Pi_{\mathrm{XInv}})$ of any such ranking ρ:

$$(m_1)! \cdots (m_{b+1})!(n_1)! \cdots (n_{a+1})! \leq (m^*)^{a/m^*}(n^*)^{b/n^*}. \tag{8.8}$$

For nonnegative integers n, $n!^{1/n}$ is increasing. This can be seen by noticing that $\log(n!^{1/n})$ is the average of $\log(1), \ldots, \log(n)$. As a result, for $i \in [b+1]$, $(m_i)!^{1/m_i} \leq (m^*)!^{1/m^*}$, or equivalently, $(m_i)! \leq (m^*)^{m_i/m^*}$. Using the fact that $m_1 + \cdots + m_{b+1} = a$,

$$(m_1)! \cdots (m_{b+1})! \leq (m^*)!^{(m_1 + \cdots + m_{b+1})/m^*} = (m^*)!^{a/m^*}.$$

We can apply similar reasoning to get $(n_1)! \cdots (n_{a+1})! \leq (n^*)!^{b/n^*}$. From this, we conclude that the size of the connected components among the rankings under consideration is at most the right-hand side of (8.8).

Since there are at least $(a+b)!/2$ of the rankings under consideration, we derive the stated bound on $\mathrm{D}(\Pi_T)$ by applying the Connectivity Lemma. □

Now, we find concrete bounds on m^* and n^*.

Proposition 8.9.

$$\max\left(1, \frac{a}{b+1}\right) \leq m^* \leq \frac{a+b}{b} \ln(4(b+1)).$$

Symmetrically,

$$\max\left(1, \frac{b}{a+1}\right) \leq n^* \leq \frac{a+b}{a} \ln(4(a+1)).$$

Proof. We first prove the upper bound on m^*. Let k be a positive integer. We compute the probability, over an average ranking ρ, that $m_i > k$ for a specific i. Notice that there is a one-to-one correspondence between the ranking ρ and the corresponding sequence of nonnegative integers $m_1, \ldots, m_{b+1}$ such that $m_1 + \cdots + m_{b+1} = a$, because the ranks of B can be uniquely recovered as $m_1 + 1, m_1 + m_2 + 2, \ldots$, and the remaining ranks form A. By stars and bars, there are $\binom{a+b}{b}$ such sequences. Now, if $m_i > k$, then $m_i - (k+1)$ is an arbitrary nonnegative integer, and $m_1 + \cdots + (m_i - (k+1)) + \cdots + m_{b+1} = a - k - 1$. By stars and bars, there are $\binom{a+b-k-1}{b}$ such sequences. Therefore, the probability that $m_i > k$ is $\binom{a+b-k-1}{b} / \binom{a+b}{b}$. Continuing,

$$\mathbb{P}[m_i > k] = \frac{(a+b-k-1)\cdots(a-k)}{(a+b)\cdots(a+1)} \leq \left(\frac{a+b-k-1}{a+b}\right)^b \leq \exp\left(-b \cdot \frac{k+1}{a+b}\right),$$

where the last step uses the bound $1 + x \leq \exp(x)$. By a union bound and taking the complement,

$$\mathbb{P}[m_1, \ldots, m_{b+1} \leq k] \geq 1 - (b+1)\exp\left(-b \cdot \frac{k+1}{a+b}\right). \tag{8.9}$$

If k is such that the right-hand side of (8.9) is at least $3/4$, we know that $m^* \leq k$. Solving for k yields the stated bound on m^*.

For the lower bounds, $m^* \geq 1$ because at least one of $m_1, \ldots, m_{b+1}$ is at least 1, and $m^* \geq \frac{a}{b+1}$ because $m_1 + \cdots + m_{b+1} = a$, and m^* is greater than or equal to the average term in the sum. □

The first part of Theorem 1.9 now comes from taking the logarithm of $\mathrm{D}(\Pi_{\mathrm{XInv}})$ in Proposition 8.8 and using the bounds in Proposition 8.9.

Proof of the first part of Theorem 1.9. We mainly make use of the following approximation based on Stirling's formula.

$$\ln(n!) = \left(n + \frac{1}{2}\right)\ln(n) - n + O(1). \tag{8.10}$$

In order to estimate $\ln \mathrm{D}(\Pi_{\mathrm{XInv}})$, we need to estimate $\ln((a+b)!) - \frac{a}{m^*}\ln(m^*) - \frac{b}{n^*}\ln(n^*)$. By (8.10),

$$\ln((a+b)!) = \left(a + b + \frac{1}{2}\right)\ln(a+b) - (a+b) + O(1) \tag{8.11}$$

$$\frac{a}{m^*}\ln((m^*)!) = \left(a + \frac{a}{2m^*}\right)\ln(m^*) - a + \frac{a}{m^*}O(1) \tag{8.12}$$

$$\frac{b}{n^*}\ln((n^*)!) = \left(b + \frac{b}{2n^*}\right)\ln(n^*) - b + \frac{b}{n^*}O(1) \tag{8.13}$$

We can use the lower bounds in Proposition 8.9, namely that $m^* \geq 1$ and $n^* \geq \frac{b}{a+1}$, to simplify the occurrences of m^* and n^* in the denominators of (8.12) and (8.13). Therefore,

$$\begin{aligned}\ln \mathrm{D}(\Pi_{\mathrm{XInv}}) &\geq a\ln\left(\frac{a+b}{m^*}\right) + b\ln\left(\frac{a+b}{n^*}\right) - \frac{a}{2}\ln(m^*) - \frac{a+1}{2}\ln(n^*) + \frac{1}{2}\ln(a+b) - O(a) \\ &= a\ln\left(\frac{a+b}{m^*\sqrt{m^* n^*}}\right) + \left(b + \frac{1}{2}\right)\ln\left(\frac{a+b}{n^*}\right) - O(a).\end{aligned}$$

Using the upper bounds in Proposition 8.9, absorbing low-order terms, and using the condition that $a \leq b$, we get

$$\log \mathrm{D}(\Pi_{\mathrm{XInv}}) \geq \Omega\left(a \log\left(\frac{b\sqrt{ab}}{a+b}\right) + b\log(a)\right) \geq \Omega(a\log(\sqrt{ab}/2) + b\log(a)) = \Omega((a+b)\log(a)).$$

□

Evaluating the Mann–Whitney statistic. We now argue how the second part of Theorem 1.9 follows, i. e., that the lower bound of $\Omega((a+b)\log(a))$ holds for inversion minimization on the Mann–Whitney tree T of Figure 2 with $a \leq b$. We do so by tweaking our lower bound argument for Π_{XInv} to the setting of Π_T.

How does the permutahedron graph for Π_T relate to the one for Π_{XInv}? The problem Π_T is a coarsening of the problem Π_{XInv}: Output values y and $ab - y$ for Π_{XInv} are both mapped to $\min(y, ab-y)$ under Π_T. This means that all edges present in $G(\Pi_{\mathrm{XInv}})$ are also present in $G(\Pi_T)$, but there may be more, and some of the connected components in $G(\Pi_{\mathrm{XInv}})$ corresponding to output value y, may be merged in $G(\Pi_T)$ with some of the connected components of $G(\Pi_{\mathrm{XInv}})$ corresponding to output value $ab - y$. However, by the reasoning behind Proposition 4.7, edges in $G(\Pi_{\mathrm{XInv}})$ can only go between rankings whose value under Π_{XInv} differ by at most one. This means that the above merging of connected components can only happen if the difference between y and $ab - y$ is 1, i. e., for the values $\lfloor ab/2 \rfloor$ and $\lceil ab/2 \rceil$, and only if ab is odd. In fact, this is exactly the situation that we analyzed in Figure 9, where v coincides with the root of T.

If we ignore the rankings with value $\lfloor ab/2 \rfloor$ or $\lceil ab/2 \rceil$ under Π_{XInv}, our lower bound argument for Π_{XInv} carries over verbatim to Π_T, except that on the right-hand side of Proposition 8.8 the factor of $\frac{1}{2}$ is replaced by $\frac{1}{2} - W$, where W represents the fraction of rankings with value $\lfloor ab/2 \rfloor$ or $\lceil ab/2 \rceil$ under Π_{XInv}. Lemma 5.5 tells us that $W \leq 2C/\sqrt{ab(a+b)}$, where C denotes the constant from the lemma. Thus, we obtain a lower bound for $\mathrm{D}(\Pi_T)$ that is a negligible fraction smaller than the one for $\mathrm{D}(\Pi_{\mathrm{XInv}})$. Taking logarithms, we obtain the same lower bound for the query complexity up to an additive term. In particular, we obtain a query lower bound of $\Omega((a+b)\log(a))$ for Π_T in case $a \leq b$. This is the second part of Theorem 1.9.

9 Cross-inversion distribution

In this section we prove the upper bound we need for the proof of Theorem 1.6 in Section 5.2, namely Lemma 5.5. Recall that $\mathrm{XInv}_{a,b}$ denotes a random variable that counts the number of cross inversions $\mathrm{XInv}(A, B)$ from A to B, where A is an array of length a, B an array of length b, and the concatenation AB is a random permutation of $[a+b]$. Lemma 5.5 states that for all positive integers a and b, $\mathrm{XInv}_{a,b}$ takes on no value with probability more than $C/\sqrt{ab(a+b)}$, where C is a universal constant.

9.1 Approach

We establish Lemma 5.5 by going through the characteristic function of $\mathrm{XInv}_{a,b}$.

Definition 9.1 (characteristic function, $\varphi_{\cdot,\cdot}$). The *characteristic function* φ_Y of a random variable Y is the function $\varphi_Y(t) : \mathbb{R} \to \mathbb{C} : t \mapsto \mathbb{E}(\mathrm{e}^{itY})$. We denote by $\varphi_{a,b}$ the characteristic function of $\mathrm{XInv}_{a,b}$.

The characteristic function is well-defined for any random variable Y and the distribution function of Y can always be retrieved from it (see, e.g., [24, Chapter VI]). In the case of integer-valued random variables like $\mathrm{XInv}_{a,b}$, the characteristic function is periodic with period 2π and the distribution function can be retrieved via the inverse Fourier transform of the characteristic function. In particular, the probabilities of $\mathrm{XInv}_{a,b}$ can be expressed as the following integrals.

Fact 9.2. *For any positive integers a, b and integer k*

$$\mathbb{P}[\mathrm{XInv}_{a,b} = k] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \varphi_{a,b}(t)\mathrm{e}^{-itk}\, dt. \tag{9.1}$$

Proof. Observe that the characteristic function $\varphi_{a,b}$ of $\mathrm{XInv}_{a,b}$ is a polynomial in $z = \mathrm{e}^{it}$, where the coefficient of degree k equals the probability of the outcome k.[2] Formula (9.1) then follows by linearity from

$$\int_{-\pi}^{\pi} z^d \mathrm{e}^{-ikt}\, dt = \int_{-\pi}^{\pi} \mathrm{e}^{i(d-k)t}\, dt = \begin{cases} 2\pi & d = k \\ 0 & d \neq k. \end{cases}$$

□

The following lemma represents the essence of the proof of Lemma 5.5.

Lemma 9.3. *Then there exists a constant C such that for all integers a, b with $b \geq a \geq 2$*

$$\int_{-\pi}^{\pi} |\varphi_{a,b}(t)|\, dt \leq \frac{C}{b\sqrt{a}}. \tag{9.2}$$

where $\varphi_{a,b}(t) \doteq \mathbb{E}(\mathrm{e}^{it\,\mathrm{XInv}_{a,b}})$.

Proof of Lemma 5.5. By symmetry, it suffices to consider the case where $a \leq b$. In the case where $a = 1$, the distribution of $\mathrm{XInv}_{a,b}$ is uniform over $\{0, \ldots, b\}$, so the maximum probability is $\frac{1}{b+1} \leq C/\sqrt{ab(a+b)}$ for any constant $C \geq \sqrt{2}/2$. Otherwise, we have

$$\mathbb{P}[\mathrm{XInv}_{a,b} = k] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \varphi_{a,b}(t)\mathrm{e}^{-itk}\, dt \leq \frac{1}{2\pi} \int_{-\pi}^{\pi} |\varphi_{a,b}(t)|\, dt \leq \frac{C}{2\pi b\sqrt{a}} \leq \frac{C}{\sqrt{ab(a+b)}}$$

by (9.1) and Lemma 9.3. □

[2] After multiplication by $\binom{a+b}{a}$, the resulting polynomial is known as the Gaussian polynomial with parameter (a, b).

To establish Lemma 9.3, as $|\varphi_{a,b}(t)|$ is an even function, it suffices to take the integral (9.2) over the domain $[0, \pi]$ and multiply by two:

$$\int_{-\pi}^{\pi} |\varphi_{a,b}(t)|\, dt = 2 \int_{0}^{\pi} |\varphi_{a,b}(t)|\, dt \tag{9.3}$$

We divide the domain of integration on the right-hand side of (9.3) into two regions: one close to zero, and the rest. The integrand is well-behaved in the center near zero, with it being approximated accurately by a Gaussian curve. It is harder to analyze the behavior of the function away from zero. In this region, a pole reduction lemma (captured by Lemma 9.13) that hinges on a combinatorial matching result (Lemma 9.15), plays a crucial role in eliminating most of the messy behavior of the function and still providing an effective bound.

We first derive an expression for the characteristic function of $\mathrm{XInv}_{a,b}$ in Section 9.2, bound the central part of the integral in Section 9.3, the peripheral part in Section 9.4, and conclude with the pole reduction lemma in Section 9.5.

9.2 The characteristic function $\varphi_{a,b}$

Our derivation of the characteristic function $\varphi_{a,b}$ of $\mathrm{XInv}_{a,b}$ is based on a connection between cross inversions and inversions in arrays.

Fact 9.4. *Let A and B be arrays, and let AB be the concatenation of A with B. Then*

$$\mathrm{Inv}(AB) = \mathrm{Inv}(A) + \mathrm{Inv}(B) + \mathrm{XInv}(A, B).$$

Proof. Any inversion in AB is either between two elements of A, two elements of B, or one element of A and one element of B. In each case, the inversion is counted in $\mathrm{Inv}(A)$, $\mathrm{Inv}(B)$, or $\mathrm{XInv}(A, B)$, respectively. □

We also make use of the following handy property of characteristic functions.

Fact 9.5. *For independent random variables Y, Z,*

$$\varphi_{Y+Z}(t) = \varphi_Y(t)\, \varphi_Z(t).$$

Proof. $\varphi_{Y+Z}(t) = \mathbb{E}(\mathrm{e}^{it(Y+Z)}) = \mathbb{E}(\mathrm{e}^{itY}\mathrm{e}^{itZ}) = \mathbb{E}(\mathrm{e}^{itY})\mathbb{E}(\mathrm{e}^{itZ}) = \varphi_Y(t)\, \varphi_Z(t)$. □

Let Inv_a be the random variable that counts the number $\mathrm{Inv}(A)$ of inversions in an array A that is a uniform permutation of $[a]$, and let $\varphi_a(t)$ be the characteristic function of Inv_a.

Claim 9.6.

$$\varphi_a(t) = \prod_{k=1}^{a} \left(\frac{\mathrm{e}^{it(k-1)/2}}{k} \cdot \frac{\sin(kt/2)}{\sin(t/2)} \right) \tag{9.4}$$

Proof. Consider the process of placing the elements $1, \dots, a$ one by one, each time placing each new element between two elements or on some end of the array, to form an array A.

For $k = 1, \dots, a$, consider the random variable that counts the number of new inversions formed with k when k is placed. First of all, when k is placed, the number of new inversions is equal to the number of elements to the left of k at the time of placement (only the elements $1, \dots, k-1$ have been placed at this point). This means the random variable has a uniform distribution over $\{0, \dots, k-1\}$, which we denote by U_{k-1}. Furthermore, this situation applies regardless of the placement of the other elements, so this random variable U_{k-1} is independent from all other previous random variables U_{j-1} with $j < k$.

Therefore, Inv_a can be written as the following sum of independent variables:

$$\mathrm{Inv}_a = U_0 + \cdots + U_{a-1}.$$

We can use Fact 9.5 to calculate the characteristic function:

$$\varphi_a(t) = \prod_{k=1}^{a} \mathbb{E}[\mathrm{e}^{itU_{k-1}}] = \prod_{k=1}^{a} \left(\frac{1}{k} \sum_{m=0}^{k-1} \mathrm{e}^{itm} \right),$$

from which (9.4) follows as $\sum_{m=0}^{k-1} \mathrm{e}^{itm} = \frac{\mathrm{e}^{itk}-1}{\mathrm{e}^{it}-1} = \frac{\mathrm{e}^{itk/2}(\mathrm{e}^{itk/2}-\mathrm{e}^{-itk/2})}{\mathrm{e}^{it/2}(\mathrm{e}^{ik/2}-\mathrm{e}^{-it/2})} = \mathrm{e}^{it(k-1)/2} \cdot \frac{2i\sin(tk/2)}{2i\sin(t/2)}$. ☐

Consider a random permutation of $[a+b]$, let A be the array consisting of the first a elements, and B the array consisting of the remaining b. Then $\mathrm{Inv}(A)$, $\mathrm{Inv}(B)$, $\mathrm{Inv}(A, B)$, and $\mathrm{XInv}(A, B)$ have the same distributions as Inv_a, Inv_b, Inv_{a+b}, and $\mathrm{XInv}_{a,b}$, respectively. Moreover, the values of $\mathrm{Inv}(A)$, $\mathrm{Inv}(B)$, and $\mathrm{XInv}(A, B)$ are independent. Hence, by Fact 9.4 and Fact 9.5 we have

$$\varphi_{a+b}(t) = \varphi_a(t)\, \varphi_b(t)\, \varphi_{a,b}(t),$$

or

$$\varphi_{a,b}(t) = \frac{\varphi_{a+b}(t)}{\varphi_a(t)\, \varphi_b(t)}.$$

By (9.4), the arithmetic sum formula applied to the exponents of e, and algebraic simplifications we conclude:

Proposition 9.7. *For integers $n \geq 0$, let $s_n(t) = \prod_{k=1}^{n} \frac{\sin(kt)}{k}$. Then*

$$\varphi_{a,b}(t) = \mathrm{e}^{itab} \frac{s_{a+b}(t/2)}{s_a(t/2)s_b(t/2)} \text{ and } |\varphi_{a,b}(t)| = \left| \frac{s_{a+b}(t/2)}{s_a(t/2)s_b(t/2)} \right|.$$

9.3 Center bound

For the first piece of the integral on the right-hand side of (9.3), we integrate $|\varphi_{a,b}(t)|$ over the interval $[0, 2\pi/(a+b)]$. For the sake of convenience, we substitute t with $2t$ in order to avoid the denominator of 2 in the sine terms of the integrand.

Lemma 9.8 (center bound). *For integers $b \geq a \geq 2$,*

$$\int_0^{\frac{2\pi}{a+b}} |\varphi_{a,b}(t)|\, dt = 2 \int_0^{\frac{\pi}{a+b}} |\varphi_{a,b}(2t)|\, dt = O\left(\frac{1}{b\sqrt{a}}\right).$$

We can write

$$|\varphi_{a,b}(2t)| = \prod_{k=1}^{a} \frac{k}{b+k} \cdot \frac{\sin((b+k)t)}{\sin(kt)}$$

as every term in the product on the right-hand side is nonnegative on this interval. We start with the following estimates.

Claim 9.9. *For positive integers $k \leq b$ and $x \in [0, \pi/(b+k)]$,*

$$\frac{k}{b+k} \cdot \frac{\sin((b+k)x)}{\sin(kx)} \leq 1 - \frac{b^2}{2\pi^2} x^2.$$

To prove this claim, we first prove two trigonometric bounds. Refer to Figure 13 for a plot of the functions and bounds.

Claim 9.10. *For positive integers k, and $x \in [0, \pi/k]$,*

$$\sin(kx) \leq kx - \frac{(kx)^3}{\pi^2}.$$

Proof. Let $y = kx$. It is enough to argue that $\sin(y) \geq y - \frac{y^3}{\pi^2}$ in the range $y \in [0, \pi]$.

Let $f(y) = \sin(y) - y + \frac{y^3}{\pi^2}$. Notice that $f(0) = f(\pi) = 0$ and $f'(\pi) > 0$. We will argue that there is a unique point $y^* \in (0, \pi)$ such that $f'(y^*) = 0$, which will ensure that $f(y) \leq 0$ for all $y \in [0, \pi]$.

We can calculate that

$$\begin{aligned} f'(y) &= \cos(y) - 1 + \frac{3y^2}{\pi^2} \\ &= \frac{3y^2}{\pi^2} - 2\sin^2\left(\frac{y}{2}\right). \end{aligned}$$

So $f'(y) = 0$ if and only if $\sin(y/2) = \pm(\sqrt{6}/\pi) \cdot (y/2)$, which is satisfied by one unique point $y^* \in (0, \pi)$. □

Claim 9.11. *For positive integers k, and $x \in (0, \pi/2k]$,*

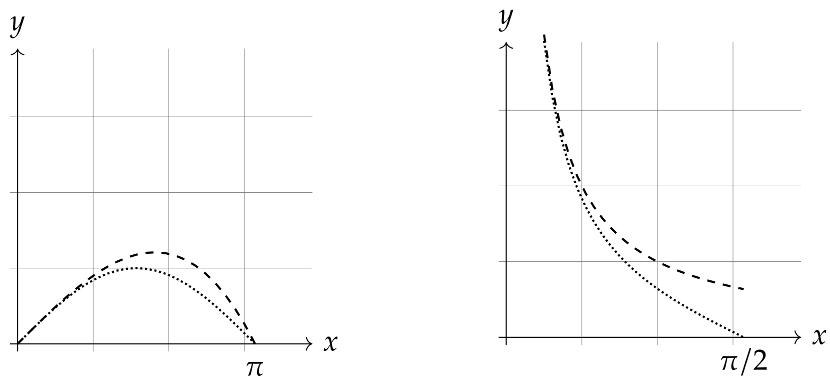
$$\cot(kx) \leq \frac{1}{kx}.$$

Proof. Let $y = kx$. We will prove that $\cot(y) \leq \frac{1}{y}$ for all $y \in (0, \pi/2]$. It is enough to prove that $y\cos(y) \leq \sin(y)$ for all $y \in [0, \pi/2]$.

The latter inequality follows because both sides are zero when $y = 0$, and the derivative of the left hand side is bounded above by the derivative of the right hand side when $y \in [0, \pi/2]$:

$$\cos(y) - y\sin(y) \leq \cos(y).$$

□



(a) Claim 9.10

(b) Claim 9.11

Figure 13: Plots of Trigonometric Bounds.
Trigonometric functions are dotted, upper bounds are dashed.

Now we finish the proof of Claim 9.9.

Proof. Notice that

$$\begin{aligned}\frac{\sin((b+k)x)}{\sin(kx)} &= \frac{\sin(kx)\cos(bx) + \sin(bx)\cos(kx)}{\sin(kx)} \\ &= \cos(bx) + \cot(kx)\sin(bx).\end{aligned}$$

Of course, $\cos(bx) \leq 1$. Furthermore, from Claim 9.10 and Claim 9.11, we can see that in this domain of x, $\sin(bx) \leq bx - \frac{b^3x^3}{\pi^2}$ and $\cot(kx) \leq \frac{1}{kx}$. Additionally, $\sin(bx) \geq 0$. Therefore, using the fact that $k \leq b$,

$$\begin{aligned}\frac{k}{b+k} \cdot \frac{\sin((b+k)x)}{\sin(kx)} &\leq \frac{k}{b+k}\left(1 + \frac{1}{kx}\left(bx - \frac{b^3x^3}{\pi^2}\right)\right) \\ &\leq 1 - \frac{b^3}{(b+k)\pi^2}x^2 \leq 1 - \frac{b^2}{2\pi^2}x^2.\end{aligned}$$

□

From this, we can now prove Lemma 9.8.

Proof. Recall that on this interval

$$|\varphi_{a,b}(2t)| = \prod_{k=1}^{a} \frac{k}{b+k} \cdot \frac{\sin((b+k)t)}{\sin(kt)}.$$

Claim 9.9 applies on all t in the domain because $\pi/(b+a) \leq \pi/(b+k)$ for all k. Therefore,

$$\int_0^{\frac{\pi}{a+b}} |\varphi_{a,b}(2t)|\, dt \leq \int_0^{\frac{\pi}{a+b}} \left(1 - \frac{b^2t^2}{2\pi^2}\right)^a dt \leq \int_0^{\frac{\pi}{a+b}} \exp\left(-\frac{ab^2t^2}{2\pi^2}\right) dt = O\left(\frac{1}{b\sqrt{a}}\right).$$

Here, we use the fact that $1 - x \leq \exp(-x)$ for all x, and the Gaussian integral: the integral of $\exp(-t^2)$ over $\mathbb{R}$ is constant, and by scaling the argument, the integral of $\exp(-ct^2)$ over $\mathbb{R}$ is a constant factor of $c^{-1/2}$ for any parameter c. □

9.4 Peripheral bound

We now bound $|\varphi_{a,b}(t)|$ in the region away from 0, namely, the interval $[2\pi/(a+b), \pi]$. As for the other part of the integral on the right-hand side of (9.3), we substitute t with $2t$ in order to avoid the denominator of 2 in the sine terms of the integrand.

Lemma 9.12 (peripheral bound)**.** *For integers $b \geq a \geq 2$,*

$$\int_{\frac{2\pi}{a+b}}^{\pi} |\varphi_{a,b}(t)|\, dt = 2\int_{\frac{\pi}{a+b}}^{\frac{\pi}{2}} |\varphi_{a,b}(2t)|\, dt = O\left(\frac{1}{b\sqrt{a}}\right).$$

In this region, the main problem is that the denominator of $|\varphi_{a,b}(2t)|$ often goes to zero, which could potentially blow up the integrand. However, the terms in the numerator always cancel out these blowups. The following lemma will be our main tool for bounding $|\varphi_{a,b}(2t)|$ in this region; it will allow terms in the numerator to cancel out bad terms in the denominator.

Lemma 9.13 (pole reduction)**.** *For every $t \in \mathbb{R}$, there exists a bijection $\beta_t : \{1, \ldots, a\} \to \{b+1, \ldots, b+a\}$ (depending on t) such that for every $k = 1, \ldots, a$,*

$$\left|\frac{1}{k}\sin(kt)\right| \geq \left|\frac{1}{\beta_t(k)}\sin(\beta_t(k)t)\right|.$$

A basic bound can be found by applying Lemma 9.13 on $|\varphi_{a,b}(2t)|$ for $k = 2, \ldots, a$, resulting in an upper bound of $1/b\sin(t)$. This bound is usable, but is weak on points closer to 0. To remedy this, we can divide the domain of integration into multiple parts, where the points closer to 0 can safely include more terms in the denominator.

Let $2 \leq n \leq a$ be an integer. We split the domain of integration into three intervals: $[\frac{\pi}{a+b}, \frac{\pi}{2n}]$, $[\frac{\pi}{2n}, \frac{\pi}{2} - \frac{\pi}{2n}]$, and $[\frac{\pi}{2} - \frac{\pi}{2n}, \frac{\pi}{2}]$. By selecting a good value of n, we can get a reasonable upper bound on this region. Here, we will make use of Lemma 9.13 and the following linear approximation to sine:

Fact 9.14. *For a positive integer k and $t \in [0, \pi/2k]$,*

$$\sin(kt) \geq \frac{2kt}{\pi}.$$

Proof. Notice that $\sin(kt) = \frac{2kt}{\pi}$ when $t = 0$ and $t = \frac{\pi}{2k}$. This fact then follows since sine is concave on this interval. $\square$

Region I. The first region of integration is $[\pi/(a+b), \pi/(2n)]$.

$$\begin{aligned}
\int_{\frac{\pi}{a+b}}^{\frac{\pi}{2n}} |\varphi_{a,b}(2t)|\, dt &\leq \int_{\frac{\pi}{a+b}}^{\frac{\pi}{2n}} \frac{n!}{\beta_t(1)\cdots\beta_t(n)} \left|\frac{\sin(\beta_t(1)t)\cdots\sin(\beta_t(n)t)}{\sin(t)\cdots\sin(nt)}\right| dt \\
&\leq \frac{n!}{b^n} \int_{\frac{\pi}{a+b}}^{\frac{\pi}{2n}} \left|\frac{1}{\sin(t)\cdots\sin(nt)}\right| dt \\
&\leq \frac{1}{b^n} \int_{\frac{\pi}{a+b}}^{\frac{\pi}{2n}} \left(\frac{\pi}{2}\right)^n \frac{1}{t^n}\, dt \\
&\leq \frac{1}{b^n} \cdot \left(\frac{\pi}{2}\right)^n \cdot \frac{1}{n-1} \cdot \left(\frac{a+b}{\pi}\right)^{n-1} \\
&\leq \frac{1}{b^n} \cdot \left(\frac{\pi}{2}\right)^n \cdot \frac{1}{n-1} \cdot \left(\frac{2b}{\pi}\right)^{n-1} \\
&\leq \frac{\pi}{2b(n-1)}.
\end{aligned}$$

The first two steps involve applying Lemma 9.13 on all $k \in \{n+1, \ldots, a\}$, and then using the fact that $|\sin(x)| \leq 1$ and $\beta_t(k) \geq b$. The third step uses Fact 9.14 for $k \in \{1, \ldots, n\}$, which applies on the interval $[\pi/(a+b), \pi/(2n)]$ for these values of k. From there, we bound with the left limit of integration.

Region II. We now bound $|\varphi_{a,b}(2t)|$ on the interval $[\frac{\pi}{2n}, \frac{\pi}{2} - \frac{\pi}{2n}]$. We can use Lemma 9.13 to eliminate all terms except $k \in \{1, 2\}$ this time.

$$\begin{aligned}
\int_{\frac{\pi}{2n}}^{\frac{\pi}{2}-\frac{\pi}{2n}} |\varphi_{a,b}(2t)|\, dt &\leq \int_{\frac{\pi}{2n}}^{\frac{\pi}{2}-\frac{\pi}{2n}} \frac{2}{\beta_t(1)\,\beta_t(2)} \left|\frac{\sin(\beta_t(1)t)\sin(\beta_t(2)t)}{\sin(t)\sin(2t)}\right| dt \\
&\leq \frac{2}{b^2} \int_{\frac{\pi}{2n}}^{\frac{\pi}{2}-\frac{\pi}{2n}} \left|\frac{1}{\sin(t)\sin(2t)}\right| dt.
\end{aligned}$$

Because $|\sin(t)|$ is increasing here and $|\sin(2t)|$ is symmetric about $\frac{\pi}{4}$, the value of the integral on the interval $[\frac{\pi}{2n}, \frac{\pi}{4}]$ exceeds the value on the interval $[\frac{\pi}{4}, \frac{\pi}{2} - \frac{\pi}{2n}]$. Using Fact 9.14,

$$\frac{2}{b^2} \int_{\frac{\pi}{2n}}^{\frac{\pi}{4}} \left|\frac{1}{\sin(t)\sin(2t)}\right| dt \leq \frac{1}{b^2} \int_{\frac{\pi}{2n}}^{\frac{\pi}{4}} \left(\frac{\pi}{2}\right)^2 \frac{1}{t^2}\, dt \leq \frac{\pi^2}{4b^2} \cdot \frac{2n}{\pi} = \frac{\pi n}{2b^2}.$$

We have now established that

$$\int_{\frac{\pi}{2n}}^{\frac{\pi}{2}-\frac{\pi}{2n}} |\varphi_{a,b}(2t)|\, dt \leq \frac{\pi n}{b^2}.$$

Region III. Notice that $|\sin(t)|$ is increasing on the interval $[\frac{\pi}{2}-\frac{\pi}{2n}, \frac{\pi}{2}]$, so we can bound $|\sin(t)|$ by $|\sin(\frac{\pi}{2}-\frac{\pi}{2n})|$. Similar to before, the first step follows from Lemma 9.13, this time applied to $k \in \{2, \ldots, a\}$.

$$\begin{aligned}\int_{\frac{\pi}{2}-\frac{\pi}{2n}}^{\frac{\pi}{2}} |\varphi_{a,b}(2t)|\, dt &\leq \int_{\frac{\pi}{2}-\frac{\pi}{2n}}^{\frac{\pi}{2}} \frac{1}{\beta_t(1)} \left|\frac{\sin(\beta_1(t))}{\sin(t)}\right| dt \leq \int_{\frac{\pi}{2}-\frac{\pi}{2n}}^{\frac{\pi}{2}} \frac{1}{|b\sin(t)|}\, dt \\ &\leq \frac{\pi}{2n} \cdot \frac{1}{b\sin(\frac{\pi}{2}-\frac{\pi}{2n})} \leq \frac{\pi}{2n} \frac{1}{b(1-\frac{1}{n})} = \frac{\pi}{2b(n-1)}.\end{aligned}$$

Overall bound. Summing the above bounds, we can deduce that

$$\int_{\frac{\pi}{a+b}}^{\frac{\pi}{2}} |\varphi_{a,b}(2t)|\, dt \leq \frac{\pi}{b(n-1)} + \frac{\pi n}{b^2}.$$

By choosing $n = \lceil \sqrt{a}\, \rceil$ (keeping in mind that $b \geq a \geq 2$), we can deduce Lemma 9.12.

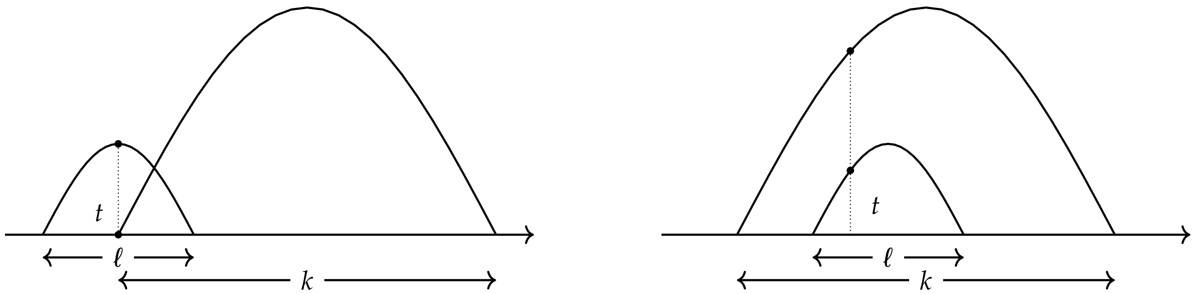
9.5 Pole reduction

Finally, we prove the pole reduction lemma (Lemma 9.13). The essence is an interval matching strategy capture Lemma 9.15. Here is the intuition.

Recall that we want to upper bound factors of the form $|\frac{\sin(\ell t)}{\ell}|$ by rescaled versions $|\frac{\sin(kt)}{k}|$ of the same pattern, where $\ell \in \{b+1, \ldots, b+a\}$ and $k \in \{1, \ldots, a\}$ are matched. The matching definitely needs to avoid situations like in Figure 14a, where $|\frac{\sin(kt)}{k}|$ vanishes at the point t while $|\frac{\sin(\ell t)}{\ell}|$ does not. Ideally, the period of the k-scaled version that contains the point t encloses the period of the ℓ-scaled version that contains t, like in Figure 14b. As long as the pattern is convex, this ensures that the k-scaled version is larger than the ℓ-scaled version everywhere on the encompassed period. (We will formally prove this in Claim 9.22.) Thus, if at every point t, we can set up a matching such that the enclosing relationship holds for all matched pairs, we are home free. Lemma 9.15 below does exactly this.

Let us first introduce some notation. For every real number t and positive integer k, there is a unique integer n such that t is contained in the half-open interval $[n/k, (n+1)/k)$. We call this interval the *k-interval* of t. For positive integers k, ℓ, we can say that the k-interval of t *encloses* the ℓ-interval of t if the ℓ-interval of t is a subset of the k-interval of t. We use the shorthand that k encloses ℓ at t.

Lemma 9.15 (interval matching). *Let a, b be positive integers. For any real t, there exists a bijection β_t between $\{1, \ldots, a\}$ and $\{b+1, \ldots, b+a\}$ such that for all $k = 1, \ldots, a$, k encloses $\ell = \beta_t(k)$ at t.*



(a) Bad case: no interval enclosure

(b) Good case: interval enclosure

Figure 14: Enclosing intervals are needed for Lemma 9.13.

Note that the bijection can be different depending on t. In fact, this is necessary as otherwise ℓ would need to be a multiple of k, which is not possible with a bijection between the sets $\{1, \ldots, a\}$ and $\{b+1, \ldots, b+a\}$.

We can interpret Lemma 9.15 as a matching on a bipartite graph by using Hall's marriage lemma.

Lemma 9.16 (Hall's marriage lemma)**.** *Let G be a bipartite graph with partitions L, R. For any $A \subseteq L$, let $N(A)$ be the set of all vertices in R with at least one neighbor in A. The graph G admits a perfect matching if and only if for all such A, $|N(A)| \geq |A|$.*

For $A \subseteq \{1, \ldots, a\}$, let $N_t(A)$ be the set of all $\ell \in \{b+1, \ldots, b+a\}$ such that there exists $k \in A$ where k encloses ℓ at t. To produce the desired bijection β_t in Lemma 9.15, it is sufficient to prove that $|N_t(A)| \geq |A|$ for all t and A.

There are some values of ℓ that are always contained in $N_t(A)$ regardless of the value of t. Let $N(A)$ be the set of all $\ell \in \mathbb{N}$ such that for all t, there exists $k \in A$ where k encloses ℓ (this k can vary depending on t). As $N(A) \cap \{b+1, \ldots, b+a\} \subseteq N_t(A)$, it is sufficient to prove that $|N(A) \cap \{b+1, \ldots, b+a\}| \geq |A|$ for all A and apply Theorem 9.16 to prove Lemma 9.15.

Example 9.17. $5 \in N(\{2, 3\})$.

Proof. We only consider $t \in [0, 1)$ for clarity. When $t \in [0, 0.4)$, the 2-interval for t is $[0, 0.5)$, while the 5-interval for t is either $[0, 0.2)$ or $[0.2, 0.4)$, which means 2 encloses 5. When $t \in [0.4, 0.6)$, the 3-interval for t is $[1/3, 2/3)$, which encloses the 5-interval $[0.4, 0.6)$. When $t \in [0.6, 1)$, the 2-interval is $[0.5, 1)$ while the 5-interval is either $[0.6, 0.8)$ or $[0.8, 1)$, so 2 encloses 5. These enclosures are shown in Figure 15, where the marked 5-intervals are contained within the respectively marked 2 or 3-intervals.

For all t, either 2 encloses 5, or 3 encloses 5. In other words, $5 \in N(\{2, 3\})$. $\square$

We first consider a different characterization of $N(A)$.

Claim 9.18. *Let ℓ be a positive integer. Then $\ell \in N(A)$ if and only if every open interval $I \subset \mathbb{R}$ that contains a fraction of denominator k for every $k \in A$ must also contain a fraction of denominator ℓ. (These fractions do not have to be distinct or reduced.)*

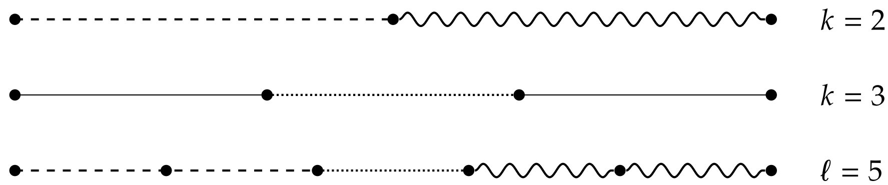


Figure 15: Example 9.17

The idea is that if no $k \in A$ encloses ℓ for some t, then the endpoints of each k-interval form a fraction of denominator k contained strictly within the ℓ-interval of t. As a result, we have a contiguous interval I containing a fraction of denominator k, and I is contained strictly within the ℓ-interval of t. As such, I cannot contain a fraction of denominator ℓ. This is illustrated in Figure 16. The formal proof also considers the edge cases involving the endpoints of the intervals.

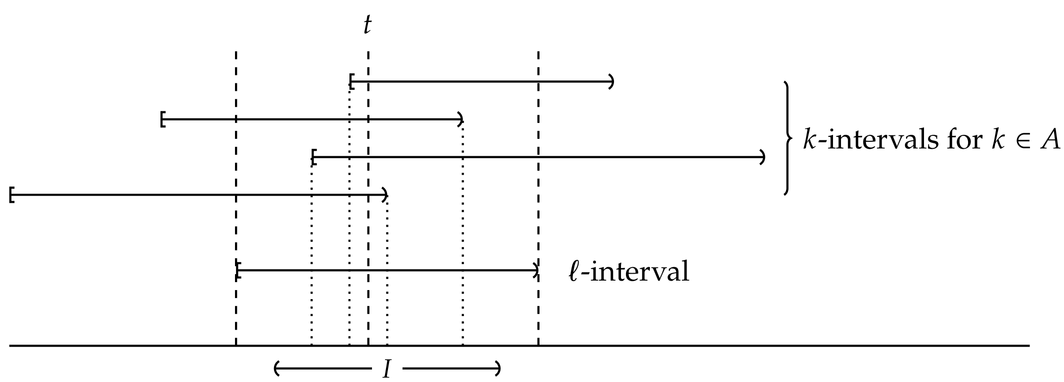


Figure 16: I contains a fraction of denominator k for all k, but no fraction of denominator ℓ.

For Claim 9.18, we prove the following proposition.

Proposition 9.19. *The following statements are equivalent for any half-open interval* $[c, d)$ *and positive integer* k*:*

(1) $[c, d)$ *is contained within a* k*-interval.*

(2) (c, d) *is contained within a* k*-interval.*

(3) (c, d) *contains no fraction of denominator* k*.*

Proof. (1) $\implies$ (2). This follows as (c, d) is a subset of $[c, d)$.

(2) $\implies$ (3). Suppose (c, d) is contained in the k-interval $[n/k, (n+1)/k)$. The interval (c, d) cannot contain a fraction of denominator k, otherwise said fraction would be strictly between n/k and $(n+1)/k$.

(3) $\implies$ (1). Let n/k be the largest fraction of denominator k less than or equal to c. It must be true that $(n+1)/k \geq d$, since $(n+1)/k$ cannot be contained in (c, d). Therefore, $[c, d)$ is contained in the k-interval $[n/k, (n+1)/k)$. ◻

From this, we can prove Claim 9.18.

Proof. By definition, the condition that $\ell \in N(A)$ is that any ℓ-interval J is contained in some k-interval for some $k \in A$. By condition (2) in Proposition 9.19, this is equivalent to saying $\mathrm{Int}(J)$ is contained in some k-interval, where $\mathrm{Int}(J)$ is the interior of J. This is true if and only if any open subinterval I of $\mathrm{Int}(J)$ is contained in some k-interval. Equivalently, if I is an open interval that is not contained in any k-interval, then I is not contained in any ℓ-interval. Using condition (3) in Proposition 9.19, this is finally equivalent to the condition that if I contains a fraction of denominator k for every $k \in A$, then I contains a fraction of denominator ℓ. ◻

The characterization in Claim 9.18 allows us to prove the following key claim about $N(A)$.

Claim 9.20. *If $\ell_1, \ell_2 \in N(A)$, then $\ell_1 + \ell_2 \in N(A)$.*

Proof. By Claim 9.18, any interval I that contains a fraction of denominator k for every $k \in A$ must also contain two fractions x/ℓ_1 and y/ℓ_2. For positive integers a, b, c, d, define the *mediant* of a/b and c/d to be $(a+c)/(b+d)$. Then the mediant is always between the two fractions. In other words, if $a/b \leq c/d$,

$$\frac{a}{b} \leq \frac{a+c}{b+d} \leq \frac{c}{d}.$$

This fact can be proven with elementary algebra, as both sides are equivalent to $a/b \leq c/d$.

From this, we see that the mediant $(x+y)/(\ell_1+\ell_2)$ is contained in I, as it is between two elements of I. As this applies to every such I, we can use Claim 9.18 to conclude that $\ell_1 + \ell_2 \in N(A)$.

Note that x/ℓ_1 and y/ℓ_2 do not have to be distinct. ℓ_1 and ℓ_2 might be equal, or $x/\ell_1 = y/\ell_2$. ◻

As k encloses k for every t, we have that $A \subseteq N(A)$. Let $\mathrm{Sums}(A)$ be the set of positive integers that can be written as the sum of not necessarily distinct elements of A. Claim 9.20 implies that $\mathrm{Sums}(A) \subseteq N(A)$.

Claim 9.21. *For nonnegative integers n,*

$$|\mathrm{Sums}(A) \cap \{n+1, \dots, n+a\}| \geq |A|.$$

Proof. Let $m = \max(A)$. Because $A \subseteq \mathrm{Sums}(A)$, $\mathrm{Sums}(A)$ contains every positive integer congruent to an element of A modulo m, since we can repeatedly add m to any element of A. As $a \geq m$, the set $\{n+1, \dots, n+a\}$ contains at least one element for every residue mod m. Therefore, $\mathrm{Sums}(A)$ contains at least $|A|$ elements of $\{n+1, \dots, n+a\}$. ◻

Putting it all together, we have that

$$|A| \leq |\mathrm{Sums}(A) \cap \{b+1, \dots, b+a\}| \leq |N(A) \cap \{b+1, \dots, b+a\}| \leq |N_t(A)|,$$

which proves Lemma 9.15 by our previous reasoning. To finish, we need to show that this implies Lemma 9.13.

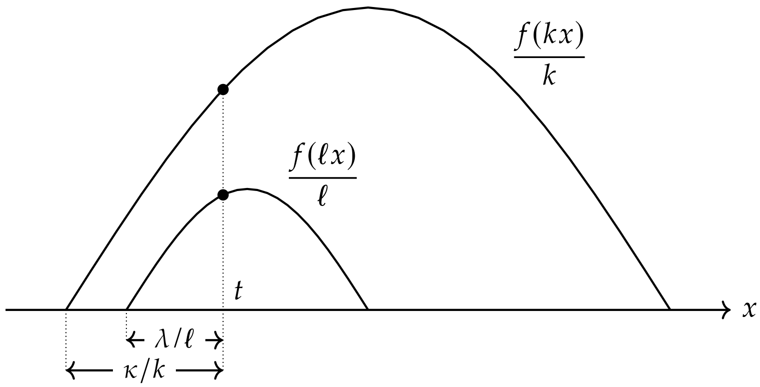


Figure 17: Proof of Claim 9.22: $\lambda/\ell \leq \kappa/k$

Claim 9.22. *Lemma 9.15 implies Lemma 9.13.*

Proof. We prove the following more general claim. Let $f : \mathbb{R} \to \mathbb{R}$ be a function that has period 1, and additionally, f is concave on $[0, 1]$ and $f(0) = f(1) = 0$. For $t \in [0, 1]$, let β_t be a bijection that satisfies the conditions of Lemma 9.15. Then for any $k \in \{1, \ldots, a\}$, let $\ell = \beta_t(k)$. We seek to prove that

$$\frac{f(kt)}{k} \geq \frac{f(\ell t)}{\ell}$$

for all $k = 1, \ldots, a$. Notice that $f(t) = |\sin(t)|$ satisfies all the conditions of the claim, albeit after scaling the period from π to 1.

To prove the general claim, let $\kappa = kt - \lfloor kt \rfloor$ and $\lambda = \ell t - \lfloor \ell t \rfloor$, noting that $\kappa, \lambda \in [0, 1]$. By the enclosing property, it is true that $\kappa/k \geq \lambda/\ell$, since they represent the distance from t to the left endpoints of the k and ℓ-intervals, respectively, and k encloses ℓ at t. This can be seen in Figure 17.

Suppose $\kappa \leq \lambda$. We have

$$f(kt) = f(\kappa) \geq \frac{\kappa}{\lambda} f(\lambda) \geq \frac{k}{\ell} f(\lambda) = \frac{k}{\ell} f(\ell t).$$

The first step follows from the periodicity of f, the second from the concavity of f on $[0, 1]$ and the fact that $f(0) = 0$ (the point $(\kappa, f(\kappa))$ is above the line segment connecting $(\lambda, f(\lambda))$ and $(0, 0)$), the third from the aforementioned inequality $\kappa/k \geq \lambda/\ell$, and the last from the periodicity of f again.

If $\lambda < \kappa$, we can instead consider the function $\widetilde{f}(t) = f(1 - t)$. Here, $\widetilde{\kappa} = 1 - \kappa$ and $\widetilde{\lambda} = 1 - \lambda$, and we can use our previous reasoning. This proves the general claim. □

10 Turing complexity

This section serves as a supplementary discussion on the time (and space) complexity of algorithms for computing the minimum number of inversions in trees. From the analysis in Section 4.2, computing $\mathrm{MinInv}(T, \rho)$ involves computing the sum of $\mathrm{MinRInv}(T_v, \rho)$ independently for each $v \in T$:

$$\mathrm{MinInv}(T, \rho) = \sum_v \mathrm{MinRInv}(T_v, \rho).$$

In order to compute $\mathrm{MinRInv}(T_v, \rho)$ for each v, we need to check the orderings σ of the children $u_1, \ldots, u_k$ of v and determine the minimum number of cross inversions between the corresponding leaf sets $L_1, \ldots, L_k$ in the order given by σ:

$$\mathrm{RInv}(T_v, \rho, \sigma) \doteq \sum_{1 \le i < j \le k} \mathrm{XInv}_\rho(L_{\sigma(i)}, L_{\sigma(j)}).$$

A natural approach for computing $\mathrm{MinInv}(T, \rho)$ consists of two phases:

1. In a first phase, we compute XInv_ρ between the leaf sets of any pair of siblings in T. This can be done in a bottom-up fashion similar to mergesort. More precisely, for a node v with sorted leaf sets $L_1, \ldots, L_k$, the cross inversions between every pair of leaf sets can be calculated using a merge operation in $O(k \cdot (|L_1| + \cdots + |L_k|))$ time, and sorts the concatenation of the leaf sets for use in future steps. Each leaf of depth d appears in d operations, which gives the total runtime of $O(\deg(T) \cdot d_{\mathrm{avg}}(T) \cdot n)$.

2. In a second phase, we check, for each node v independently, which ordering σ of the children of v minimizes $\mathrm{RInv}(T_v, \rho, \sigma)$. We then output the sum of the values $\mathrm{MinRInv}(T_v, \rho)$.

Exhaustively testing all orderings σ of the k children of v to compute $\mathrm{MinRInv}(T_v, \rho)$ takes $O(k \cdot k!)$ time, as $O(k)$ time is required to calculate $\mathrm{RInv}(T_v, \rho, \sigma)$, for each σ, from the precomputed values of XInv. This results in a total of $O(n \cdot \deg(T)!)$ time for the second phase, and an overall running time of $O((\deg(T)! + \deg(T) \cdot d_{\mathrm{avg}}(T)) \cdot n)$.

For any constant bound on the degree of the tree T, the basic algorithm runs in time polynomial in n. The dependency of the running time on the degree can be improved. One way to do so is by reducing the problem of the second phase to the closely-related and well-studied problem of computing a minimum arc feedback set of a weighted directed graph.

Definition 10.1 (Minimum Feedback Arc Set)**.** Given a directed graph $G = (V, E)$ with an ordering σ on the vertices $v_1, \ldots, v_n$, a *feedback arc* is an edge e_k from v_i to v_j such that $\sigma(v_i) > \sigma(v_j)$. The minimum feedback arc set problem is finding the minimum number of feedback arcs induced by any ordering σ. In *weighted* minimum feedback arc set, each edge from v_i to v_j has a weight w_{ij}, and the objective is to minimize $\sum_{e \in E} e_{ij} \cdot \mathbb{I}[\sigma(v_i) > \sigma(v_j)]$.

We can encode the problem of computing $\mathrm{MinRInv}(T_v, \rho)$ as an instance of weighted minimum arc feedback set, where each edge of the graph G has a positive weight. If the leaf sets of the children of v are $L_1, \ldots, L_k$, we construct a graph G with k vertices $v_1, \ldots, v_k$. For each pair of vertices v_i and v_j, if $\mathrm{XInv}_\rho(L_i, L_j) < \mathrm{XInv}_\rho(L_j, L_i)$, we add an edge from v_i to v_j of weight $\mathrm{XInv}_\rho(L_j, L_i) - \mathrm{XInv}_\rho(L_i, L_j)$. We can extract the value of $\mathrm{MinRInv}(T_v, \rho)$ from the weight of the minimum feedback arc set.

As a consequence, we can use existing efficient algorithms for weighted minimum arc feedback set to construct algorithms for inversion minimization on trees that are more efficient than the basic algorithm we described. [3] gives two exact algorithms for weighted minimum arc feedback set. One algorithm [3, Algorithm 1] is based on the Held-Karp algorithm for the traveling salesman problem [13]; it uses dynamic programming to achieve a time complexity of $\Theta(n^2 2^n)$ and a space complexity of $\Theta(2^n)$ for a graph of n vertices. Another algorithm [3, Algorithm 2] uses a divide and conquer approach that achieves a time complexity of $O(\mathrm{poly}(n) \cdot 4^n)$, but has the advantage of only needing polynomial space.

Algorithm 2 MinRInv(T_v, ρ), Dynamic Programming

Input: Tree T_v with child leaf sets $L_1, \ldots, L_k$, ranking ρ
  Initialize $\textsc{Cost}[S]$, where S is over all subsets of $\{1, \ldots, k\}$.
  $\textsc{Cost}[\varnothing] \leftarrow 0$
  for i from 1 to k **do**
    for all sets S of size i **do**
      $\textsc{Cost}[S] \leftarrow \min_{s \in S}(\textsc{Cost}(S \setminus \{s\}) + \sum_{j \in S, j \neq s} \mathrm{XInv}_\rho(L_j, L_s)))$
  return $\textsc{Cost}[\{1, \ldots, k\}]$.

An adaptation of the dynamic programming algorithm for calculating MinRInv is given in Algorithm 2. Using this subroutine for computing MinRInv, we can improve the time complexity of our basic algorithm to $O((\deg(T)^2 2^{\deg(T)} + \deg(T) \cdot d_{\mathrm{avg}}(T)) \cdot n)$.

The improved running time is still not efficient for trees with unrestricted degree. In fact, the problem is NP-hard.

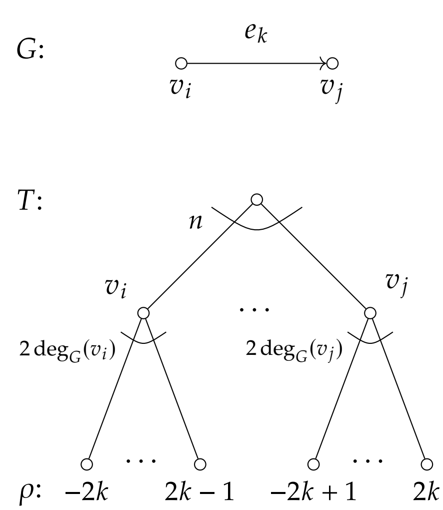


Figure 18: Encoding of an edge e_k

Proposition 10.2. *Computing* $\mathrm{MinInv}(T, \rho)$ *is NP-hard.*

Proof. We show a reduction from the known NP-hard problem "Minimum feedback arc set" [17].

For a graph G with n vertices $v_1, \ldots, v_n$ and m directed edges $e_1, \ldots, e_m$, we construct a depth-2 tree T and a ranking ρ of its leaves. We will assume that G has no isolated vertices; this goes without loss of generality as isolated vertices can be dropped from an instance of minimum arc feedback set without affecting the answer. We also assume that between any two vertices at most one of the two directed edges is present; this is also without loss of generality since dropping the edges in case both are present reduces the answer by one. Finally, for ease of notation, we allow the ranking ρ to be an injective mapping into the integers; this can be changed easily by replacing each integer by its rank in the range.

In the first layer, the root of T has n children corresponding to $v_1, \ldots, v_n$. The second layer has leaves with ranks encoding the edges of G. For each edge e_k going from v_i to v_j, we add two leaves under v_i with ranks $-2k$ and $2k-1$, and two leaves under v_j with ranks $-2k+1$ and $2k$, as shown in Figure 18. All ranks are distinct.

Consider the number of inversions in T induced by an ordering σ of $v_1, \ldots, v_n$. For each edge e_k, the number of inversions between the leaves of rank $-2k$ and $2k-1$ and the leaves of rank $-2k+1$ and $2k$ is 1 if $\sigma(v_i) < \sigma(v_j)$ and 3 if $\sigma(v_i) > \sigma(v_j)$. These four leaves also form $8(m-1)$ inversions with all other leaves, keeping in mind that these inversions are counted twice when summed up over all edges.

Therefore, the minimum number of inversions in T is given by

$$\mathrm{MinInv}(T, \rho) = \min_{\sigma} \left(4m(m-1) + m + 2 \cdot \sum_{e_k \in E} \mathbb{I}[\sigma(v_i) > \sigma(v_j)] \right).$$

The size of the minimum arc feedback set is precisely $\min_{\sigma} \left(\sum_{e_k \in E} \mathbb{I}[\sigma(v_i) > \sigma(v_j)] \right)$, which can be extracted from $\mathrm{MinInv}(T, \rho)$ with straightforward calculations. This completes the reduction. □

Approximation Algorithms. If we relax our requirements to an approximate answer, we can approximate MinRInv in polynomial time using existing approximation algorithms for weighted minimum feedback arc set. The best known such algorithm achieves an approximation ratio of $O(\log n \log\log n)$ on a graph with n vertices [10]. Adapting this algorithm for minimizing inversions in trees produces an approximation factor of $O(\log(\deg(T)) \log\log(\deg(T)))$ for $\mathrm{MinInv}(T, \rho)$. Under the unique games conjecture, there does not exist a constant-factor approximation algorithm for minimum feedback arc set on arbitrary digraphs [12].

In the special case of tournament graphs, which have exactly one edge of weight 1 between every pair of vertices, there are efficient constant factor approximation algorithms for minimum arc feedback set. Some of these also apply to weighted tournaments, where for every pair of vertices v_i, v_j, the nonnegative edge weights w_{ij}, w_{ji} satisfy $w_{ij} + w_{ji} = 1$. This case corresponds to the scenario of computing $\mathrm{MinRInv}(T_v, \rho)$ where all leaf sets L_i of siblings have the same size. [18] gives an algorithm with runtime $O^*(2^{O(\sqrt{\mathrm{OPT}})})$, given that the optimal answer is OPT. [19] also gives an approximation algorithm in the case where $w_{ij} + w_{ji} \in [b, 1]$ for some $b > 0$: For any $\epsilon > 0$, the algorithm produces a $(1+\epsilon)$-approximation of OPT in time $n 2^{\tilde{O}(1/(\epsilon b)^{12})}$. For the problem of computing $\mathrm{MinRInv}(T_v, \rho)$, the parameter b represents the ratio between the smallest and largest possible values of $|L_i| \cdot |L_j|$.

Wilcoxon test. As a final remark we point out an alternate way of computing $\Pi_T(\rho)$ in the special case of the Mann–Whitney trees of Figure 2. The number of cross inversions $\mathrm{XInv}_\rho(A, B)$ can be written in terms of the rank sum $W_B \doteq \sum_{y \in B} \rho(y)$ as follows, where $a \doteq |A|$ and $b \doteq |B|$:

$$\mathrm{XInv}_\rho(A, B) = ab + \frac{b(b+1)}{2} - W_B. \tag{10.1}$$

The quantity W_B is known as the Wilcoxon rank-sum statistic for differences between random variables. Because of the relationship (10.1) the Wilcoxon test is equivalent in power to the Mann–Whitney test. However, the evaluation based on the efficient computation of cross inversions (especially in the case of unbalanced set sizes a and b) is superior to the evaluation based on the rank sum S_B, as the latter presumes sorting the combined set $X = A \sqcup B$.

Acknowledgements

We would like to thank Greta Panova, Robin Pemantle, and Richard Stanley for pointers regarding Gaussian polynomials, Stasys Jukna for answering questions about the complexity measure N, and the anonymous reviewers and copy editors for helpful suggestions. We appreciate the partial support for this research by the U.S. National Science Foundation under Grants No. 2137424 and 2312540.

References

[1] MICHAEL BEN-OR: Lower bounds for algebraic computation trees. In *Proc. 15th STOC*, pp. 80–86. ACM Press, 1983. [doi:10.1145/800061.808735] 10

[2] MANUEL BLUM, ROBERT W. FLOYD, VAUGHAN PRATT, RONALD L. RIVEST, AND ROBERT E. TARJAN: Time bounds for selection. *J. Comput. System Sci.*, 7(4):448–461, 1973. [doi:10.1016/S0022-0000(73)80033-9] 12

[3] HANS L. BODLAENDER, FEDOR V. FOMIN, ARIE M. C. A. KOSTER, DIETER KRATSCH, AND DIMITRIOS M. THILIKOS: A note on exact algorithms for vertex ordering problems on graphs. *Theory Computing Sys.*, 50(3):420–432, 2012. [doi:10.1007/s00224-011-9312-0] 57

[4] JEAN CARDINAL, SAMUEL FIORINI, GWENAËL JORET, RAPHAËL M. JUNGERS, AND J. IAN MUNRO: An efficient algorithm for partial order production. *SIAM J. Comput.*, 39(7):2927–2940, 2010. [doi:10.1137/090759860] 13

[5] SVANTE CARLSSON AND JINGSEN CHEN: The complexity of heaps. In *Proc. 3rd Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA'92)*, pp. 393–402. SIAM, 1992. ACM DL. 12

[6] RICHARD DEGERMAN: Ordered binary trees constructed through an application of Kendall's tau. *Psychometrika*, 47(4):523–527, 1982. [doi:10.1007/BF02293713] 2, 3, 4

[7] DORIT DOR AND URI ZWICK: Selecting the median. *SIAM J. Comput.*, 28(5):1722–1758, 1999. [doi:10.1137/S0097539795288611] 12

[8] DORIT DOR AND URI ZWICK: Median selection requires $(2 + \epsilon)n$ comparisons. *SIAM J. Discr. Math.*, 14(3):312–325, 2001. [doi:10.1137/S0895480199353895] 12

[9] DAVID EPPSTEIN: A permutohedron, 2007. LINK, see also Wikipedia article, accessed 11-07-2024. 9

[10] GUY EVEN, JOSEPH NAOR, BARUCH SCHIEBER, AND MADHU SUDAN: Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(2):151–174, 1998. [doi:10.1007/pl00009191] 58

[11] GASTON H. GONNET AND J. IAN MUNRO: Heaps on heaps. *SIAM J. Comput.*, 15(4):964–971, 1986. [doi:10.1137/0215068] 12

[12] VENKATESAN GURUSWAMI, JOHAN HÅSTAD, RAJSEKAR MANOKARAN, PRASAD RAGHAVENDRA, AND MOSES CHARIKAR: Beating the random ordering is hard: Every ordering CSP is approximation resistant. *SIAM J. Comput.*, 40(3):878–914, 2011. [doi:10.1137/090756144] 58

[13] MICHAEL HELD AND RICHARD M. KARP: A dynamic programming approach to sequencing problems. *J. SIAM*, 10(1):196–210, 1962. [doi:10.1137/0110015] 57

[14] IVAN HU, DIETER VAN MELKEBEEK, AND ANDREW MORGAN: Query complexity of inversion minimization on trees. In *Proc. 34th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA'23)*, pp. 2836–2866. SIAM, 2023. [doi:10.1137/1.9781611977554.ch107] 1

[15] STASYS JUKNA, ALEXANDER A. RAZBOROV, PETR SAVICKÝ, AND INGO WEGENER: On P versus NP ∩ co-NP for decision trees and read-once branching programs. *Comput. Complexity*, 8(4):357–370, 1999. [doi:10.1007/s000370050005] 11

[16] KANELA KALIGOSI, KURT MEHLHORN, J. IAN MUNRO, AND PETER SANDERS: Towards optimal multiple selection. In *Proc. 32nd Internat. Colloq. on Automata, Languages, and Programming (ICALP'05)*, pp. 103–114. Springer, 2005. [doi:10.1007/11523468_9] 12

[17] RICHARD M. KARP: Reducibility among combinatorial problems. In RAYMOND E. MILLER AND JAMES W. THATCHER, editors, *Complexity of Computer Computations*, pp. 85–103. Plenum Press/Springer, 1972. Available at ResearchGate. [doi:10.1007/978-1-4684-2001-2_9] 57

[18] MAREK KARPINSKI AND WARREN SCHUDY: Faster algorithms for feedback arc set tournament, Kemeny rank aggregation and betweenness tournament. In *Proc. Internat. Symp. on Algorithms and Computation (ISAAC'10)*, pp. 3–14. Springer, 2010. [doi:10.1007/978-3-642-17517-6_3] 58

[19] CLAIRE KENYON-MATHIEU AND WARREN SCHUDY: How to rank with few errors. In *Proc. 39th STOC*, pp. 95–103. ACM Press, 2007. [doi:10.1145/1250790.1250806] 58

[20] HENRY B. MANN AND DONALD R. WHITNEY: On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Stat.*, 18(1):50–60, 1947. JSTOR. 11

[21] STEPHEN MELCZER, GRETA PANOVA, AND ROBIN PEMANTLE: Counting partitions inside a rectangle. *SIAM J. Discr. Math.*, 34(4):2388–2410, 2020. [doi:10.1137/20M1315828] 11

[22] GEORGE A. MILLER: Psychological method to investigate verbal concepts. *J. Mathematical Psychology*, 6(2):169–191, 1969. [doi:10.1016/0022-2496(69)90001-7] 2

[23] RYAN O'DONNELL: *Analysis of Boolean Functions*. Cambridge Univ. Press, 2014. [doi:10.1017/CBO9781139814782] 14

[24] ALFRÉD RÉNYI: *Probability Theory*. North-Holland 1970, Dover 2007. 44

[25] ARNOLD SCHÖNHAGE: The production of partial orders. *Astérisque*, 38–39:229–246, 1976. NumDam. 12

[26] IGOR S. SERGEEV: On the upper bound of the complexity of sorting. *Comput. Math. and Math. Physics*, 61(2):329–346, 2021. [doi:10.1134/S0965542521020111] 5

[27] RICHARD P. STANLEY AND FABRIZIO ZANELLO: Some asymptotic results on q-binomial coefficients. *Annals of Combinatorics*, 20(3):623–634, 2016. [doi:10.1007/s00026-016-0319-8] 11

[28] LAJOS TAKÁCS: Some asymptotic formulas for lattice paths. *J. Stat. Planning and Inference*, 14(1):123–142, 1986. [doi:10.1016/0378-3758(86)90016-9] 11

[29] ANDREW CHI-CHIN YAO: Probabilistic computations: Toward a unified measure of complexity. In *Proc. 18th FOCS*, pp. 222–227. IEEE Comp. Soc., 1977. [doi:10.1109/SFCS.1977.24] 27, 28

AUTHORS

Ivan Hu
Ph. D. student
Department of Computer Science
University of Wisconsin – Madison
Madison, Wisconsin, USA
ilhu@wisc.edu
https://pages.cs.wisc.edu/~ihu/

Dieter van Melkebeek
Professor
Department of Computer Science
University of Wisconsin – Madison
Madison, Wisconsin, USA
dieter@cs.wisc.edu
https://pages.cs.wisc.edu/~dieter/

Andrew Morgan
Software engineer
Google
amorgan@cs.wisc.edu
https://pages.cs.wisc.edu/~amorgan/

ABOUT THE AUTHORS

IVAN HU is a third year Ph. D. student at the University of Wisconsin–Madison under the supervision of Dieter van Melkebeek. He is currently studying complexity theory.

DIETER VAN MELKEBEEK received his Ph. D. from the University of Chicago, under the supervision of Lance Fortnow. His thesis was awarded the ACM Doctoral Dissertation Award. After postdocs at DIMACS and the Institute for Advanced Study, he joined the faculty at the University of Wisconsin-Madison, where he currently is a full professor. His research interests include the power of randomness, lower bounds for NP-complete problems, and connections between derandomization and lower bounds.

ANDREW MORGAN received his Ph. D. in 2022 from the University of Wisconsin–Madison under the supervision of Dieter van Melkebeek. After graduating, he became a software engineer at Google.