Tight Distance Query Reconstruction for Trees and Graphs Without Long Induced Cycles

Bastide, Paul; Groenland, Carla

DOI
10.1002/rsa.70017

Publication date
2025

Document Version
Final published version

Published in
Random Structures and Algorithms

RESEARCH ARTICLE OPEN ACCESS

Tight Distance Query Reconstruction for Trees and Graphs Without Long Induced Cycles

Paul Bastide[1] | Carla Groenland[2]

[1]LaBRI—Université de Bordeaux, Bordeaux, France | [2]TU Delft, Delft, the Netherlands

Correspondence: Paul Bastide (paul.bastide@ens-rennes.fr)

Received: 27 May 2024 | **Revised:** 2 May 2025 | **Accepted:** 4 June 2025

Funding: The authors received no specific funding for this work.

ABSTRACT

Given access to the vertex set V of a connected graph $G = (V, E)$ and an oracle that given two vertices $u, v \in V$, returns the shortest path distance between u and v, how many queries are needed to reconstruct E? Firstly, we show that randomized algorithms need to use at least $\frac{1}{200}\Delta n \log_\Delta n$ queries in expectation to reconstruct n-vertex trees of maximum degree Δ. The best previous lower bound (for graphs of bounded maximum degree) was an information-theoretic lower bound of $\Omega(n \log n / \log \log n)$. Our randomized lower bound is also the first to break through the information-theoretic barrier for related query models, including distance queries for phylogenetic trees, membership queries for learning partitions, and path queries in directed trees. Secondly, we provide a simple deterministic algorithm to reconstruct trees using $\Delta n \log_\Delta n + (\Delta + 2)n$ distance queries. This proves that our lower bound is optimal up to a multiplicative constant. We extend our algorithm to reconstruct graphs without induced cycles of length at least k using $O_{\Delta,k}(n \log n)$ queries. Our lower bound is therefore tight for a wide range of tree-like graphs, such as chordal graphs, permutation graphs, and AT-free graphs. The previously best randomized algorithm for chordal graphs used $O_\Delta(n \log^2 n)$ queries in expectation, so we improve by a $(\log n)$-factor for this graph class.

1 | Introduction

The distance query model has been introduced [1]. In this model, only the vertex set V of a hidden graph $G = (V, E)$ is known, and the aim is to reconstruct the edge set E via distance queries to an oracle. For a pair of vertices $(u, v) \in V^2$, the oracle answers the shortest path distance between u and v in G. The algorithm can select the next query based on the responses of earlier queries. If there is a unique graph consistent with the query responses, the graph has been reconstructed.

For a graph class $\mathcal{G}$ of connected graphs, we write $\mathcal{G}_n$ for the n-vertex graphs in $\mathcal{G}$. We say an algorithm reconstructs $\mathcal{G}$ if for every graph $G \in \mathcal{G}$ the distance profile obtained from the queries is unique to G within $\mathcal{G}$. Let $f(G, A)$ denote the

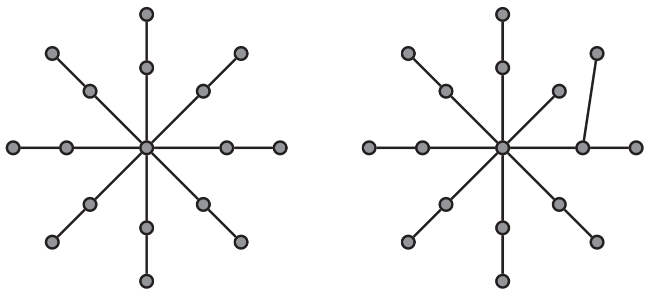


FIGURE 1 | To distinguish the tree on the left from all possible labelings of the tree on the right, $\Omega(n^2)$ queries are needed.

number of queries that a deterministic algorithm A takes until it has reconstructed the graph $G \in \mathcal{G}$. The *query complexity* of an algorithm A is defined as a function $f : \mathbb{N} \to \mathbb{N}$ where $f(n)$ is the maximum number of queries that A takes on an input graph $G \in \mathcal{G}_n$, i.e., $D(n) = \max_{G\in\mathcal{G}_n} f(G, A)$. A randomized algorithm is a probability distribution π over algorithms and its query complexity given by the expected number of queries $R(n) = \max_{G\in\mathcal{G}_n} \mathbb{E}_{A\sim\pi}[f(G, A)]$. The (randomized/deterministic) query complexity for reconstructing a graph class is now given by the complexity of the best (randomized/deterministic) algorithm.

Of course, by asking the oracle the distance between every pair (u, v) of vertices in G, we can completely reconstruct the edge set as $E = \{\{u, v\}|d(u, v) = 1\}$. This implies a trivial upper bound of $\binom{|V|}{2}$ on the query complexity. Unfortunately, this upper bound can be tight. For example, the clique K_n is difficult to distinguish from K_n minus an edge: If the missing edge is $\{u, v\}$, then the only query answer that is different is then one for the pair (u, v). Thus, all pairs need to be queried before K_n is reconstructed. The core of this problem is in fact high-degree vertices [2] (see Figure 1) and therefore we will restrict our attention to connected n-vertex graphs of maximum degree Δ, as has also been done in earlier work.

The best known lower bound (for bounded degree graphs) is from Reference [3]: By an information-theoretic argument, $\Omega_\Delta(n \log n/ \log\log n)$ queries are needed to reconstruct n-vertex trees of maximum degree Δ. Let us consider $\Delta = 10$ to sketch the idea of the proof. We use that the class C of n-vertex graphs of maximum degree 10 and diameter at most $\log n$ has size $\Omega(2^{n\log n})$. For any algorithm distinguishing graphs from C in N queries, no two distinct graphs from C can get the same responses to the first N queries. The diameter condition, ensures that (for graphs in C) every query answer is an element of $\{0, \dots, \lfloor \log n \rfloor\}$ and therefore can be encoded using at most $\log(\log n + 1)$ bits. Concatenating the answers to the first N queries, and using that the number of possible strings needs to be at least the number of graphs in the class, we find $2^{N \log(\log n+1)} = \Omega(2^{n\log n})$. This implies $N = \Omega(n \log n/ \log\log n)$.

Improving on such an information-theoretic lower bound is often difficult. More generally, randomized query complexity is infamously difficult to pinpoint: For example, state-of-the-art results are also far from tight bounds for the recursive majority function [4–6]. In the setting of the evasiveness conjecture, the oracle can answer *adjacency queries* (“given u, v, is $\{u, v\} \in E$?”) instead of distance queries. It has been shown already in 1975 [7] that for any fixed non-trivial monotone graph property of the graph (such as “does G have a triangle”), any deterministic algorithm needs $\Theta(n^2)$ on n-vertex graphs. At the same time, the best randomized query lower bound $\Omega(n^{4/3}(\log n)^{1/3})$ from Reference [8] is far from the best upper bound of $O(n^2)$. Even seemingly simple questions such as estimating the average degree of a graph using vertex degree queries require new probabilistic tools to achieve tight bounds [9, 10].

For n-vertex trees of maximum degree Δ, we achieve the correct dependency on n and Δ for both the randomized and deterministic query complexity using distance queries. Our first result towards this is the following lower bound.

Theorem 1.1. *Let $\Delta \geqslant 2$ and $n = 2c\Delta^k$ be integers, where $c \in [1, \Delta)$ and $k \geqslant 50(c \ln c + 3)$ is an integer. Any randomized algorithm requires at least $\frac{1}{50}\Delta n \log_\Delta n$ queries to reconstruct n-vertex trees of maximum degree $\Delta + 1$.*

Note that for any $n \geqslant 2$, there is a unique $(c, k) \in [1, \Delta) \times \mathbb{Z}_{\geqslant 0}$ with $n/2 = c\Delta^k$ so the only assumption in our lower bound is that n is sufficiently large compared to Δ. Our result allows Δ to grow slowly with n (e.g., $\Delta = O((\log n)^\alpha)$ with $\alpha \in (0, 1)$). Moreover, we allow Δ to be larger for specific values of n (e.g., $O(n^{1/150})$ for $c = 1$). We made no attempt to optimize the constant and slightly lowered the constant in the abstract to state the result for trees of maximum degree Δ instead of $\Delta + 1$.

This removes the $(1/\log\log n)$ factor compared to the information-theoretic lower bound. Moreover, we achieve both the correct dependence on n and the correct dependency on Δ (namely $\Delta/\log\Delta$) for the term in front of $n\log n$. This is shown by our following result.

Theorem 1.2. *For all $\Delta \geqslant 4$, there exists a deterministic algorithm to reconstruct trees of maximum degree at most Δ on n vertices using $\Delta n \log_\Delta n + (\Delta+2)n$ queries.*

For the class of trees of maximum degree 3, the algorithm from Theorem 1.2 for maximum degree $\Delta = 4$ is still optimal up to a multiplicative constant (by our Theorem 1.1 applied with $\Delta = 3$).

Theorems 1.1 and 1.2 show both the deterministic and randomized distance query complexity of n-vertex trees of maximum degree Δ are $\Theta(\Delta n \log_\Delta n)$ for various ranges of Δ. However, we expect that the randomized complexity will be slightly lower in terms of the multiplicative constant. Towards this, we also show that randomness can be exploited in our algorithm of Theorem 1.2 to use $\frac{1}{2}\Delta n \log_\Delta n + (\Delta + 1 + \log n)n$ queries in expectation.

Our algorithm extends to chordal graphs and beyond. A graph is called k-chordal if it has no induced cycles of length at least $k+1$. This gives an extension of chordal graphs (which are 3-chordal graphs). No (randomized nor deterministic) algorithms were previously known with $o(n^{3/2})$ query complexity for k-chordal n-vertex graphs for $k \geqslant 5$.

Theorem 1.3. *There exists a deterministic algorithm to reconstruct n-vertex k-chordal graphs of maximum degree at most Δ using $O_{\Delta,k}(n\log n)$ queries.*

Since permutation graphs and AT-free graphs are known to be 5-chordal and 6-chordal respectively (see [11, 12]), our results pinpoint the (randomized and deterministic) query complexity for those graph classes to $\Theta_\Delta(n\log n)$.

1.1 | Previous Algorithms and New Algorithmic Insight

Kannan, Mathieu, and Zhou [3, 13] designed a randomized algorithm with query complexity $\widetilde{O}_\Delta(n^{3/2})$, where the subscript denotes the constant may depend on Δ and $\widetilde{O}(f(n))$ is a short-cut for $O(f(n)\ \mathrm{polylog}(n))$. In the same article, they give randomized algorithms for chordal graphs and outerplanar graphs with a quasi-linear query complexity $O_\Delta(n\log^3 n)$. Rong, Li, Yang, and Wang [14] improved the randomized query complexity for chordal graphs to $O_\Delta(n\log^2 n)$. Their algorithm only requires a weaker type of oracle and applies to graphs without induced cycles of length at least 5. Our algorithm extends the class of graphs and shaves off a $\log n$-factor, achieving the best possible dependence on n by our Theorem 1.1. Low chordality has also been used in other works for designing efficient algorithms (e.g., routing schemes [15], computing maximal independent sets or maximal packings, etc. [16, 17]).

Most known algorithms with quasi-linear query complexity in the distance oracle setting are randomized, with a recent work giving a linear deterministic algorithm for interval graphs from Rong, Li, Yang, and Wang [14] as a notable exception.

We provide a new approach for exploiting separators, which also extends to various "tree-like" graphs. Our algorithm, restricted to the class of trees, is surprisingly simple: We compute a Breadth First Search (BFS) tree starting from a vertex v_0 and then inductively reconstruct the tree up to layer i. For each vertex in layer $i+1$, we use balanced separators to "binary search" its parent in layer i. The algorithm and its analysis are given in Section 3.1, and we extend it to k-chordal in Section 3.2 using structural graph theory insights.

1.2 | Lower Bound Technique

To prove our lower bounds, we first restrict our attention to reconstructing the labeling of the leaves of one particular tree. This reduces some "noise" and reduces the problem to its core. We prove a lower bound on the number of queries needed to reconstruct this labeling, and so the lower bound holds for any graph class that contains (all labelings of) this particular tree.

Our tree of interest is $T_{\Delta,k}$ depicted in Figure 2. It is the rooted tree of depth $k+1$ where the root (labeled by the empty string) has degree Δ, all vertices at layers $1, \ldots, k-1$ have degree $\Delta+1$, and all vertices on layer k have degree 2. Since the structure of the tree is fixed, what remains to be reconstructed is the labeling of the vertices. We moreover fix the

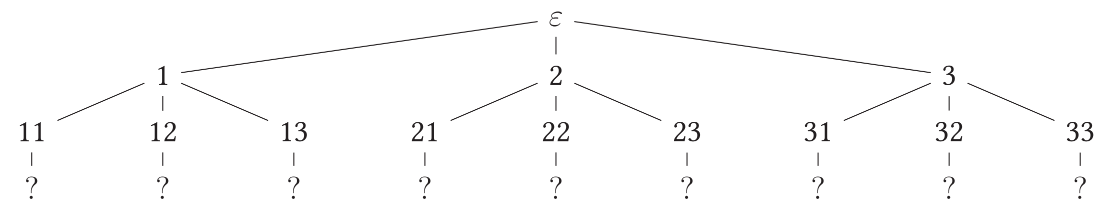


FIGURE 2 | An example of the tree $T_{\Delta,k}$ for $\Delta = 3$ and $k = 2$ is depicted with labels. The "?'s denote that the labels of the leaves are what needs to be reconstructed.

labels of all internal nodes, where the label of v incorporates information about the path from v to the root. We prove the following key lemma.

Lemma 1.4. *Let $\Delta \geqslant 2$, $k \geqslant 150$ be positive integers. Consider a labeling of the tree $T_{\Delta,k}$ with $N = \Delta^k$ leaves, where only the labels of the leaves are unknown. Any randomized algorithm requires at least $\frac{1}{11}\Delta N \log_\Delta N$ queries in expectation to reconstruct the labeling.*

Lemma 1.4 above only applies to specific values of N for readability purposes. An extended version is given in Section 4, which is needed to obtain a lower bound of all values of N in our applications of the lemma.

What remains for reconstruction in the lemma is to assign each leaf to its parent (after that, all adjacencies are determined). So the problem above reduces naturally to the problem of reconstructing a bijection $f : [\Delta]^k \to [\Delta]^k$, which is the viewpoint we will take. (We use the short-cut $[m] = \{1, \dots, m\}$.) To model this, we define two new reconstruction problems and oracles. We present next the simplest one, as we think it is natural and could be studied in its own right.

The aim is to reconstruct a bijective function $f : [\Delta]^k \to [\Delta]^k$ (where $N = \Delta^k = \Theta(n)$). The reader should see f as the function that maps the label of a leaf to the label of its parent in $T_{\Delta,k}$. We show that reconstructing f with distance queries is deeply linked with reconstructing f with the help of a new oracle, the *coordinate oracle*. This oracle answers the following two types of coordinate queries, where we use x_i to denote the i^{th} coordinate of $x \in [\Delta]^k$:

1. "is $f(a)_i = f(a')_i$?" for $a, a' \in [n]$ and $i \in [k]$, and
2. "is $f(a)_i = j$?" for $a \in [n]$, $j \in [\Delta]$ and $i \in [k]$.

Interestingly, instead of the usual number of queries, we can link the complexity of our original problem to the number of **NO** answers given by the coordinate oracle. The reduction goes via another function reconstruction problem with a more involved type of queries (see Section 4).

1.3 | Applications to Other Settings

The coordinate oracle presented above is of independent interest. Using the key lemma or intermediate results in this new setting, we deduce improved randomized lower bounds for various related reconstruction problems randomized lower bounds for other query models (see Section 4.4 for further details):

- betweenness queries in graphs [18] (also called separator queries [19]),
- path queries in directed graphs [20–22],
- membership queries for learning a partition [23–25],
- leaf-distance queries in phylogenetic trees [23, 26–29],
- comparison queries in tree posets [30].

Previous work in these settings found deterministic lower bounds or used information-theoretic arguments to obtain weaker randomized lower bounds. Our lower bound often matches the complexity of randomized algorithms in these settings, thereby settling those randomized query complexities (up to a multiplicative constant) as well.

1.4 | Roadmap

In Section 2, we set up our notation and give the relevant definitions. In Section 3, we present our deterministic algorithms for trees and k-chordal graphs, obtaining new upper bounds. In Section 4, we prove a matching randomized lower bound and discuss further applications of this lower bound. In Section 5 we conclude with some open problems.

2 | Preliminaries

In this paper, all graphs are simple, undirected, and connected except when stated otherwise. All logarithms in this paper are base 2, unless mentioned otherwise, where $\ln = \log_e$. For an integer $n \geqslant 1$, we write $[n] = \{1, \dots, n\}$.

For a graph G and two vertices $a, b \in V(G)$, we denote by $d_G(a, b)$ the length of a shortest path between a and b. For $G = (V, E)$, $A \subseteq V$ and $i \in \mathbb{N}$, we denote by $N_G^{\leqslant i}[A] = \{v \in V | \exists a \in A, d_G(v, a) \leqslant i\}$. We may omit the superscript when $i = 1$. We write $N_G(A) = N_G[A] \setminus A$ and use the shortcuts $N_G[u]$, $N_G(u)$ for $N_G[\{u\}]$, $N_G(\{u\})$ when u is a single vertex. We may omit the subscript when the graph is clear from the context.

2.1 | Distance Queries

We denote by $\textsc{Query}_G(u, v)$ the call to an oracle that answers $d_G(u, v)$, the distance between u and v in a graph G. For two sets A, B of vertices, we denote by $\textsc{Query}_G(A, B)$ the $|A| \cdot |B|$ calls to an oracle, answering the list of distances $d_G(a, b)$ for all $a \in A$ and all $b \in B$. We may abuse notation and write $\textsc{Query}_G(u, A)$ for $\textsc{Query}_G(\{u\}, A)$ and may omit G when the graph is clear from the context.

For a graph class $\mathcal{G}$ of connected graphs, we say an algorithm reconstructs the graphs in the class if for every graph $G \in \mathcal{G}$ the distance profile obtained from the queries does not belong to any other graph from $\mathcal{G}$. The *query complexity* is the maximum number of queries that the algorithm takes on an input graph from $\mathcal{G}$, where the queries are adaptive. For a randomized algorithm, the query complexity is given by the expected number of queries (with respect to the randomness in the algorithm).

2.2 | Tree Decomposition

A *tree decomposition* of a graph G, introduced by [31], is a tuple $(T, (B_t)_{t \in V(T)})$ where T is a tree and B_t is a subset of $V(G)$ for every $t \in V(T)$, for which the following conditions hold.

- For every $v \in V(G)$, the set of $t \in V(T)$ such that $v \in B_t$, is non-empty and induces a subtree of T.
- For every $uv \in E(G)$, there exists a $t \in V(T)$ such that $\{u, v\} \subseteq B_t$.

2.3 | Balanced Separators

For $\beta \in (0, 1)$, a *β-balanced separator* of a graph $G = (V, E)$ for a vertex set $A \subseteq V$ is a set S of vertices such that the connected components of $G[A \setminus S]$ are of size at most $\beta|A|$.

One nice property of tree decompositions is that they yield $\frac{1}{2}$-balanced separators.

Lemma 2.1 ([31]). *Let G be a graph, $A \subseteq V(G)$ and $(T, (B_t)_{t \in V(T)})$ a tree decomposition of G. Then there exists $u \in V(T)$ such that the bag B_u is a $\frac{1}{2}$-balanced separator of A in G.*

In fact, for trees (of treewidth 1), we can always find a $\frac{1}{2}$-balanced separator of size 1 (i.e., a single vertex). We refer to such a separator as a *vertex separator* or *separating vertex*.

3 | Improved Algorithm for Distance Reconstruction

This section presents our deterministic algorithms for the class of trees and k-chordal graphs. The lower bounds are given in Section 4.

3.1 | Optimal Algorithm for Tree Reconstruction

We first describe our algorithm for reconstructing trees, as it encapsulates most of the algorithmic ideas, while being fairly simple.

Theorem 1.2. *For all $\Delta \geqslant 4$, there exists a deterministic algorithm to reconstruct trees of maximum degree at most Δ on n vertices using $\Delta n \log_\Delta n + (\Delta + 2)n$ queries.*

Proof. Let T be a tree on n vertices, and let Δ be the maximum degree of T. Our algorithm starts as follows. We pick an arbitrary vertex $v_0 \in V(T)$ and will consider (for the analysis) the input tree T as rooted in v_0. We call $\textsc{Query}(v_0, V(T) \setminus \{v_0\})$. We define the i^{th}layer of T as $L_i = \{v \in V(T) | d(v_0, v) = i\}$. We proceed to reconstruct the graph induced by the first i layers by induction on i.

Note that $T[L_0] = (\{v_0\}, \emptyset)$ is immediately reconstructed. We fix an integer $i \geqslant 1$ and assume that the first $i-1$ layers are fully reconstructed (i.e., we discovered all the edges and non-edges of $T[L_0 \cup \cdots \cup L_{i-1}]$). Let $T_1 = T[L_0 \cup \cdots \cup L_{i-1}]$ be the already reconstructed subtree. We show how to reconstruct the edges between the $(i-1)^{\text{th}}$ layer and the i^{th}layer. Note that this suffices to reconstruct all the edges (since in a tree, edges can only be between consecutive layers).

Choose an arbitrary vertex $v \in L_i$. We first show that we can find the parent of v in L_{i-1} using $O(\Delta \log n)$ queries and then describe how to shave off an additional $(\log \Delta)$ factor.

The procedure goes as follows. As T_1 is a tree, it admits a $\frac{1}{2}$-balanced separator of size 1. Let s_1 be a vertex for which $\{s_1\}$ forms such a separator. We ask first $\textsc{Query}(v, N[s_1])$, where the neighborhood is taken in T_1. As T_1 is a tree, there is a unique path between any two vertices. Therefore, for $w \in N(s_1)$, the distance $d(v, w) = d(v, s_1) - 1$ if w lies on the shortest path from v to s_1 and $d(v, s_1) + 1$ otherwise. From this, we can infer the neighbor x of s_1 that is the closest to v as the one for which the answer is smallest (or find that s_1 is adjacent to v and finish). Moreover, the unique path from s_1 to v lives in the connected component T_2 of $T_1 \setminus \{s_1\}$ that contains x. In particular, T_2 contains the parent of v (see Figure 3).

We can repeat this process and construct two sequences $(T_j)_{j \in \mathbb{N}}$ and $(s_j)_{j \in \mathbb{N}}$, where T_{j+1} is the connected component of $T_j \setminus \{s_j\}$ containing the parent of v, and $s_j \in V(T_j)$ is chosen so that $\{s_j\}$ is a $\frac{1}{2}$-balanced separator of T_j. Once T_ℓ contains less than $\Delta + 1$ vertices for some ℓ or the vertex s_ℓ is identified as the parent of v, we finish the process[1]. By definition of $\frac{1}{2}$-balanced separator,

$$\forall j \in [\ell - 1], |T_{j+1}| \leqslant |T_j|/2 \quad \text{and thus} \quad \ell \leqslant \log n$$

If the process finished because T_ℓ has at most Δ vertices, we use at most Δ additional queries via $\textsc{Query}(T_\ell, v)$. We infer the parent of v from the result. For each $j \leqslant \ell$, we use at most $\Delta + 1$ queries to reconstruct T_{j+1} from T_j. Hence, we use $O(\Delta \log n)$ queries in total.

Taking a closer look at the process, at any step j, we can choose the order on the queries $\textsc{Query}(v, w)$ for $w \in N(s_j)$ and may not need to perform all the queries. Given a subtree S of T_1 on $b \geqslant 1$ vertices that contains the parent of v, we now show how to find the parent of v in $f(b) = \Delta \log_\Delta b + \Delta + 1$ queries (giving the desired improvement of a $(\log \Delta)$ factor). If S has at most $\Delta + 1$ vertices, we may simply $\textsc{Query}(v, S)$ and deduce the answer. Otherwise, let $s \in S$ be a $\frac{1}{2}$-balanced separator for S. This has at least two neighbors since S has at least $\Delta + 1$ vertices. We order the connected components of $S \setminus \{s\}$ by non-increasing size, and ask the queries in the same order: We start with $\textsc{Query}(v, w_1)$ for w_1, the neighbor of s which is in the largest component, then proceed to the neighbor of the second largest component, etc. We terminate when we find two different distances or have queried all the neighbors. In particular, we never perform $\textsc{Query}(v, s)$.

- If $d(v, w)$ for $w \in N(s)$ are all the same then s is the parent of v. We terminate and recognise s as the parent of v. We used at most $\Delta \leqslant f(b)$ queries.
- If we discover that $d(v, w) < d(v, w')$ for some $w, w' \in N(s)$, then s is not the parent of v. In fact, w is the vertex from $N[s]$ closest to v, and we recursively perform the same procedure on the subtree S' of $S \setminus \{s\}$ that contains w. Note that S' must contain the parent of v.

If we query 2 neighbors of s before detecting the component containing the parent of v, our next subtree S' satisfies $|S'| \leqslant |S|/2$ since $\{s\}$ is a $\frac{1}{2}$-balanced separator. If we query $m \geqslant 3$ neighbors of s before detecting the component containing

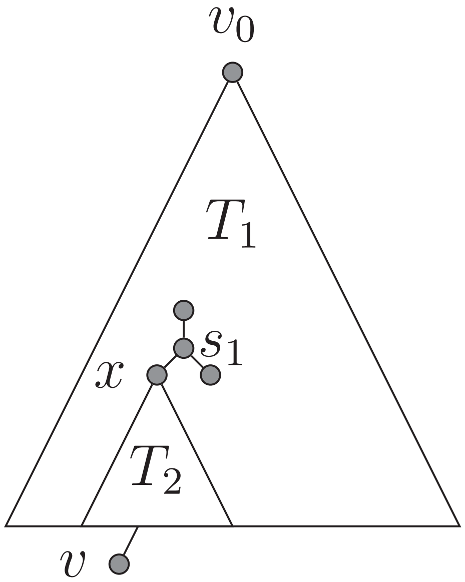


FIGURE 3 | The subtree T_2 contains the neighbor of v on a shortest path to s_1 and so contains the parent of v.

the parent of v, our next subtree S' satisfies $|S'| \leqslant |S|/m$ since there are $m-1$ components of $S \setminus \{s\}$ that are at least as large. Either way, we decrease the size of the tree by a factor of at least x if we perform x queries, where $x \in \{2, \dots, \Delta\}$.

We show by induction on b that the procedure described above uses at most $f(b) = \Delta \log_\Delta b + \Delta + 1$ queries. The claim is true when $b \leqslant \Delta + 1$. By the discussion above, for $b \geqslant \Delta + 2$, the process either finishes in Δ queries or uses $x + f(b')$ queries for some $b' \leqslant b/x$ and $x \in \{2, \dots, \Delta\}$. It thus suffices to show that

$$f(b/x) + x \leqslant f(b) \text{ for all } x \in \{2, \dots, \Delta\}$$

By definition, $f(b) - f(b/x) = \Delta \log_\Delta x$. We show that $\Delta \log_\Delta x \geqslant x$ for all $x \in \{2, \dots, \Delta\}$. By analyzing the derivative of $\Delta \log_\Delta x - x$ on the (real) interval $x \in \{2, \dots, \Delta\}$, we find that the minimum is achieved at $x = 2$ or $x = \Delta$. Since $\Delta \geqslant 4$, the minimum is achieved at $x = \Delta$, as desired.

With the improved procedure, we can reconstruct the edge from L_{j-1} to v in at most $\Delta \log_\Delta n + \Delta + 1$ queries. Repeating the same strategy to reconstruct the parent of every vertex, we obtain the edge set of T in at most

$$(n-1) + (n-1)(\Delta \log_\Delta n + (\Delta + 1)) \leqslant \Delta n \log_\Delta n + (\Delta + 2)n$$

queries. □

Even though we show in the next section that we cannot achieve a better dependency in (n, Δ) using randomization, we can improve the multiplicative constant.

Theorem 3.1. *For any $\Delta \geqslant 4$, there exists a randomized algorithm for reconstructing n-vertex trees of maximum degree at most Δ using $\frac{1}{2}\Delta n \log_\Delta + (\Delta + 2 + \log n)n$ queries in expectation.*

Proof. The algorithm works similarly to the algorithm from Theorem 1.2. We define the same layers and inductively reconstruct the graph induced on the first i layers. We find the parent of a vertex $v \in L_i$ via a similar sequence of separators $s_1, \dots, s_j$ and trees $T_1 \supseteq T_2 \supseteq \cdots \supseteq T_j$. The key difference is that when we wish to learn the vertex in $N[s_j]$ closest to v, then we perform QUERY(v, w) for $w \in N[s_j]$ in an order that is chosen at random. Suppose that $|T_j| = b$. We apply the following claim to the tree T_j and the vertex s_j (both of which the algorithm knows completely). □

Claim 3.2. Let T be a tree and let $t \in V(T)$. Let $a_1, \dots, a_k$ be the sizes of the components of $T \setminus \{t\}$ and let $v_1, \dots, v_k$ denote the neighbors of s_j in these components. There is a computable random order on $v_1, \dots, v_k$ such that the expected number of vertices placed before v_i is at most $\frac{1}{2}\frac{1}{a_i}(-a_i + \sum_{j=1}^k a_j)$ for all $i \in [k]$.

Proof. We generate the order by independently sampling $X_i \sim U[0, a_i]$ uniformly at random for all $i \in [k]$, where $[0, a_i]$ denotes the set of real numbers between 0 and a_i. Almost surely, $X_{\pi(1)} > \cdots > X_{\pi(k)}$ for some permutation π on the support $[k]$ and this gives us our desired random order.

We prove that the expected number of vertices placed before v_1 is at most $\frac{1}{2}\frac{a_2+\cdots+a_k}{a_1}$, and then the remaining cases will follow by symmetry. Let $I(x_1)$ denote the number of vertices placed before v_1 given that $X_1 = x_1$, i.e., the number of $i \in \{2, \dots, k\}$ such that $X_i > x_1$:

$$I(x_1) = \sum_{i=2}^{k} \text{Bern}\left(\max\left(\frac{a_i - x_1}{a_i}, 0\right)\right)$$

A Bernoulli random variable with probability p has expectation p. The expected number of vertices placed before v_1 is hence

$$\frac{1}{a_1}\int_0^{a_1} \mathbb{E}[I(x_1)]dx_1 = \sum_{i=2}^{k} \frac{1}{a_1}\int_0^{\min(a_1,a_i)} \left(1 - \frac{x_1}{a_i}\right)dx_1$$

We now show for all $i \in [2, k]$ that the ith summand is at most $\frac{1}{2}\frac{a_i}{a_1}$, which implies that the number of vertices placed before v_1 is indeed at most $\frac{1}{2}\frac{a_2+\cdots+a_k}{a_1}$. We compute

$$\frac{1}{a_1}\int_0^{\min(a_1,a_i)} \left(1 - \frac{x_1}{a_i}\right)dx_1 = \frac{\min(a_1, a_i)}{a_1}\left(1 - \frac{\min(a_1, a_i)}{2a_i}\right)$$

When $a_i \leqslant a_1$, the expression simplifies to $\frac{a_i}{a_1}\frac{1}{2}$ as desired. When $a_i \geqslant a_1$, the expression simplifies to $1 - \frac{1}{2}\frac{a_1}{a_i}$ which is at most $\frac{1}{2}\frac{a_i}{a_1}$ since

$$a_1 a_i \leqslant \frac{1}{2}a_i^2 + \frac{1}{2}a_1^2$$

□

By the claim, if the parent of v is in a component T_{j+1} of $T_j - s_j$ of size a, then we query at most $\frac{1}{2}(b - a)/a$ vertices in expectation before we query the neighbor of s_j in T_{j+1}. This means that for a size reduction of $x = b/a$, we perform approximately $\frac{1}{2}x$ queries in expectation, compared to x in our deterministic algorithm. Using linearity of expectation, we will repeat a similar calculation to the one done in the proof of Theorem 1.2.

We show that we reconstruct the edge to the parent of v using at most $\frac{1}{2}\Delta \log_\Delta n + \Delta + 1 + \log n$ queries in expectation. We show that once we have identified a subtree T_j containing the parent of v with $|T_j| = b$, we use at most $\Delta \log_\Delta b + \Delta + 1 + \log b$ queries in expectation to find the parent of v. Let s_j be a $1/2$-balanced separator of T_j. The first base case is given when s_j is the parent of v: In this case, we perform at most Δ queries (all neighbors of s_j) and find that s_j is the parent. The second base case is when $|T_j| = \Delta + 1$, in which case we query all distances between v and T_j and identify the parent of v.

We now assume that $b \geqslant \Delta + 2$ and that the parent of v is in a component T_{j+1} of $T_j \setminus s_j$. Let $a = |T_{j+1}|$. By the claim, we query at most $\frac{1}{2}(b - a)/a$ vertices in expectation before we query the neighbor w of s_j in T_{j+1}. We still need to query the distance from v to w. If w is the first vertex to be queried, then we need to query one more vertex. Since s_j is a $1/2$-balanced separator, this happens with probability at most $1/2$. So after at most $\frac{1}{2}\frac{b}{a} + 1$ in expectation, we find the neighbor of v is in T_{j+1} after which by induction we need another $\frac{1}{2}\Delta \log_\Delta a + \Delta + 1 + \log a$ queries in expectation. In total, we use at most

$$\frac{1}{2}\frac{b}{a} + 1 + \frac{1}{2}\Delta \log_\Delta a + \Delta + 1 + \log a$$

in expectation. We find that $\frac{b}{a} + \Delta \log_\Delta a \leqslant \Delta \log_\Delta b$ for all $\Delta \geqslant 4$ (same calculation as before) and $\log a + 1 \leqslant \log b$ since $a \leqslant b/2$. So we used at most $\Delta \log_\Delta b + \Delta + 1 + \log b$ queries in expectation, as claimed. □

3.2 | Optimal Algorithm for k-Chordal Graphs

Using additional structural analysis, we extend our algorithm from trees to k-chordal graphs: Graphs without induced cycles of length at least $k + 1$. In the simpler case of (3-)chordal graphs, (randomized) reconstruction from a quasi-linear number of queries was already known to be possible [13]. Besides extending the class of graphs, our algorithm shaves off a $(\log n)$ factor and is now optimal in n (the number of vertices of the input graph).

The core of the proof uses the same principles as for trees in Theorem 1.2: We reconstruct the edges of a vertex u to the previous layer, layer-by-layer and vertex-by-vertex. The two important ingredients are (1) a structural result on the neighborhood of a vertex (see Claim 3.4) and (2) the existence of "nice" balanced separators on the already reconstructed subgraph (see Claim 3.5). After removing the separator, we need to show that we can correctly determine the component that contains the neighborhood of the vertex u that we are currently considering. We also need to reconstruct the edges within the layer, but this turns out to be relatively easy.

Theorem 1.3. *There exists a deterministic algorithm to reconstruct n-vertex k-chordal graphs of maximum degree at most Δ using $O_{\Delta,k}(n\log n)$ queries.*

During the proof, we will need the following definition. The *treelength* of a graph G (denoted $\mathrm{tl}(G)$) is the minimal integer ℓ for which there exists a tree decomposition $(T,(B_t)_{t\in V(T)})$ of G such that $d_G(u,v)\leqslant \ell$ for every pair of vertices u,v that share a bag (i.e., $u,v\in B_t$ for some $t\in V(T)$). In particular, we will use the following result proved by Kosowski, Li, Nisse, and Suchan [32].

Lemma 3.3 ([32]). *For any $k\in\mathbb{N}$, any k-chordal graph has treelength at most k.*

Proof of Theorem 1.3. We start by fixing a vertex v_0 and asking $\textsc{Query}(V(G),v_0)$. From that, we reconstruct $L_i=\{v\in V(G)|d(v,v_0)=i\}$. We write $L_{\bowtie i}=\cup_{j\bowtie i}L_j$ for any relation $\bowtie\in\{\leqslant,<,>,\geqslant\}$.

The algorithm proceeds by iteratively reconstructing $G[L_{\leqslant i}]$ for increasing values of i. Note that since G has degree at most Δ there are at most $\Delta^{2\Delta k}$ vertices in $L_{\leqslant 2\Delta k}$, the vertices at distance at most $2\Delta k$ from v_0. We ask $\textsc{Query}(L_{\leqslant 2\Delta k},L_{\leqslant 2\Delta k})$ which represent at most $O_{\Delta,k}(1)$ queries and allows us reconstruct completely the adjacencies inside $L_{\leqslant 2\Delta k}$.

Suppose that we reconstructed $G_1:=G[L_{\leqslant i-1}]$ for some $i\geqslant 2\Delta k$ and we again want to reconstruct the two edge sets

$$E_{i-1,i}=\{uv\in E(G)|u\in L_{i-1},\ v\in L_i\}$$

and

$$E_{i,i}=\{uv\in E(G)|u,v\in L_i\}$$

We call $H_1=G[L_{\leqslant i-1-k}]$ the core of G_1. We need a lemma that implies that neighborhoods are not spread out too much in G_1.

Claim 3.4. For all $u\in L_i$ and $v,w\in N(u)\cap L_{i-1}$, $d_{G_1}(v,w)\leqslant\Delta k$.

Proof. Let $v,w\in N(u)\cap L_{i-1}$ and let P be a shortest vw-path in G_1. If $V(P)\cap N(u)=\{v,w\}$, then the vertex set $V(P)\cup\{u\}$ induces a cycle in G, and so $|V(P)|\leqslant k$ (else the k-chordality would be contradicted). For the same reason, P can have at most $k-1$ consecutive vertices outside of $N(u)$. Since u has at most Δ neighbors, it follows that $d_{G_1}(v,w)\leqslant\Delta k$. □

Since G has treelength at most k by Lemma 3.3, it has a tree decomposition $(T',\mathcal{B}')$ such that all bags $B'\in\mathcal{B}'$ satisfy $d_G(u,v)\leqslant k$ for all $u,v\in B'$. In particular, the bags have size at most Δ^k+1.

Restricting this tree decomposition to G_1 shows that G_1 admits a tree decomposition $(T,\mathcal{B})$ such that each $B\in\mathcal{B}$ has size at most Δ^k+1 and $d_G(u,v)\leqslant k$ for all $u,v\in B$. When a bag $B\in\mathcal{B}$ contains at least one vertex v of the core H_1, then in fact $B\subseteq V(G_1)$ and $d_{G_1}(u,v)\leqslant k$ for all $u\in B$. In particular, $d_{G_1}(u,v)\leqslant 2k$.

We have already reconstructed G_1, so in particular, we know $N^{\leqslant k}[v]$ for all $v\in H_1$. Therefore, we can construct such a tree decomposition $(T,\mathcal{B})$ of G_1 such that each $B\in\mathcal{B}$ has size at most Δ^k+1 and for any bag $B\in\mathcal{B}$, if $v\in H_1\cap B$, then $d_{G_1}(u,v)\leqslant k$ for all $u\in B$.

Fix $u\in L_i$. We describe an algorithm to reconstruct $N(u)\cap L_{i-1}$. The algorithm recursively constructs a sequence of connected graphs $(G_j)_{j=0}^{\ell}$ and a sequence of separators $(S_j)_{j=1}^{\ell}$ for some $\ell\leqslant\lceil\log(n)\rceil$, such that S_j is a $\frac{1}{2}$-balanced separator of G_j, $S_j\subseteq L_{\leqslant i-\Delta k-1}$ and $N(u)\cap L_{i-1}\subseteq V(G_j)$.

We first prove the following claim that we use to find our sequence of separators (S_j).

Claim 3.5. For n large enough compared to Δ and k and any set of vertices $A \subseteq V(G_1)$ with $|A| \geqslant \log n$, there exists a bag B of T such that B is a $\frac{1}{2}$-balanced separator of A, and B is contained in $L_{\leqslant i-\Delta k-1}$.

Proof. Let T be rooted in a bag that contains v_0. By Lemma 2.1, there is a bag B of T that forms a $\frac{1}{2}$-balanced separator for A (i.e., all connected components of $G_1[A \setminus B]$ are of size at most $|A|/2$). We choose such a bag B of minimum depth (in T). We need to show B is contained in $L_{\leqslant i-\Delta k-1}$.

If B contains v_0, then $B \subseteq G[L_{\leqslant k}]$. Since $i \geqslant 2\Delta k$, we are done in this case.

Suppose now that $v_0 \notin B$ and let B' be the parent of B. By definition, B' is not a $\frac{1}{2}$-balanced separator of A. If B contains a vertex v of $L_{\leqslant i-1-k-\Delta k}$, then also $v \in L_{\leqslant i-1-k} = V(H_1)$ and so all vertices of B are at distance at most k from v. This implies that $B \subseteq L_{\leqslant i-1-\Delta k}$, as desired.

If not, B has no vertex in $L_{\leqslant i-1-k-\Delta k}$ so $B \subseteq L_{>i-(\Delta+1)k-1}$. Since G_1 is connected, $B \cap B' \neq \emptyset$. The same diameter argument gives that $B' \subseteq L_{>i-(\Delta+2)k-1}$. If C is a component of $G_1 \setminus B'$ that does not contain v_0, then the shortest path from any $v \in C$ to v_0 in G_1 (of length at most i) must go through B' (at distance at least $i-(\Delta+2)^k-1$ from v_0). In particular, all such components are contained in $N^{((\Delta+2)^k+1)}(B')$ and so the total size is at most $\Delta^{(\Delta+2)k+1}|B'| = O_{k,\Delta}(1)$. For n sufficiently large, this is at most $\frac{1}{2}\log n$.

On the other hand, as B' is not a $\frac{1}{2}$-balanced separator, there exist a component A' of $G_1[A \setminus B']$ with $|A'| > |A|/2$ and so $|A'| > \frac{1}{2}\log n$. We found above that A' then must contain v_0. Since B does not contain v_0, A' is contained in the component of $G_1[A \setminus B]$ containing v_0. This yields a contradiction with the fact that B is a $\frac{1}{2}$-balanced separator of A. □

Suppose that we have defined G_j for some $j \geqslant 1$ and let us describe how to define G_{j+1}. If $|G_j| \leqslant \log n$, we ask QUERY$(u, V(G_j))$ and output $N(u) \cap V(G_j)$. Otherwise, we let S_j be the bag found in Claim 3.5 when applied to $A = V(G_j)$. Then $S_j \subseteq L_{\leqslant i-\Delta k-1}$ and it is a $\frac{1}{2}$-balanced separator of G_j. Since it is a bag of T and contained in H_1, we find the size of the bag is at most $\Delta^k + 1$ and $d_{G_1}(u,v) \leqslant k$ for all $u, v \in S_j$. We ask QUERY$(N[S_j], u)$ and let G_{j+1} be a component of $G_j \setminus S_j$ that contains a vertex from $\arg\min_{x \in N[S_j]} d_G(x,u)$. Then, we increase j by one and repeat the same procedure.

We now prove the correctness of the algorithm presented above.

We first argue that $N(u) \cap L_{i-1}$ is contained in a unique connected component of $G_j \setminus S_j$. Since every separator is included in $L_{\leqslant i-\Delta k-1}$, we find $d_{G_1}(u, S_\ell) \geqslant \Delta k + 1$ for all $\ell \leqslant j$. By Claim 3.4, all vertices in $N(u) \cap L_{i-1}$ are connected via paths in G_1 of length at most Δk and by the observation above, these paths avoid all separators so will be present in a single connected component of $G_j \setminus S_j$. The only thing left to prove is the following claim.

Claim 3.6. G_{j+1} is the component of $G_j \setminus S_j$ that contains $N(u) \cap L_{i-1}$.

Proof. Let $x \in N[S_j]$ such that $d_G(x,u)$ is minimized and let H_x be the connected component of x in $G_j \setminus S_j$. We will prove that H_x contains $N(u) \cap L_{i-1}$. In particular, this implies that $G_{j+1} = H_x$ (a priori, G_{j+1} could be a different component for another minimiser than x), so that G_{j+1} contains $N(u) \cap L_{i-1}$, as desired.

Suppose towards a contradiction that $N(u) \cap L_{i-1}$ is instead contained in a different connected component H. We will find an induced cycle of length at least $k+1$, depicted in Figure 4.

Let P be a shortest path in G from x to u.

Let P' be a path from u to S_j with all internal vertices in H. Such a path exists since G_j is connected.

Let P'' be a shortest path in G between a neighbor of x in S_j to the endpoint of P' in S_j. As S_j is a bag contained in H_1, any two vertices in S_j are within distance k in G_1. So $P'' \subseteq L_{\leqslant i-2k-2}$ (we may assume $\Delta \geqslant 4$).

Let y be the first vertex on the path P (from x to u) that lies in L_i (such a vertex must exist since the path does not have internal vertices in S_j by choice of x and since H_x contains no neighbors of u).

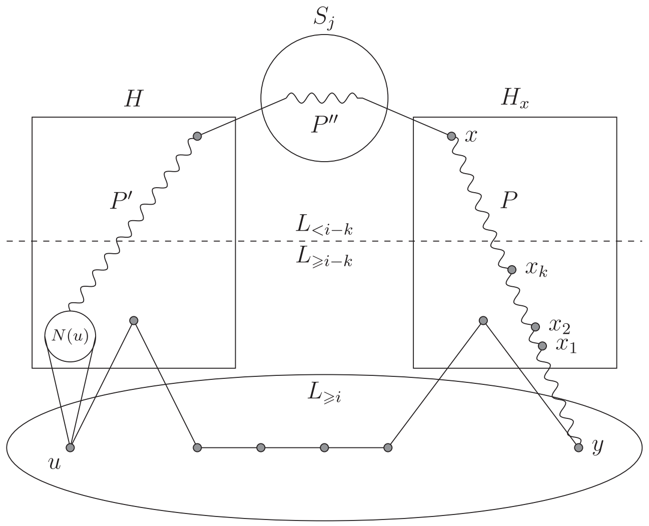


FIGURE 4 | This figure depicts a possible configuration in the proof of Claim 3.6 for which we end up with a contradiction by finding a large induced cycle.

Let $x_1, \dots, x_k$ be the k vertices before y in P. Note that none of the x_i can be adjacent to or part of $P' \cup P''$ (since they are in $H_x \cap L_{\geqslant i-k}$). Let G' be the graph obtained from $G[P \cup P'' \cup P']$ by contracting $P'' \cup P' \cup (P \setminus \{x_1, \dots, x_k\})$ to a single vertex p. Note that the selected vertex set is indeed connected and that the resulting graph has vertex set $\{x_1, \dots, x_k, p\}$. Since P was a shortest path in G, the vertex set $\{x_1, \dots, x_k\}$ still induces a path, and it suffices to argue about the adjacencies of p. Via edges of P, the vertex p is adjacent to x_1 and x_k. If p was adjacent to x_i for some $i \in [2, k-1]$, then there must be a vertex $y \in P'' \cup P' \cup (P \setminus \{x_1, \dots, x_k\})$ adjacent to x_i. But a case analysis shows this is not possible, because the only vertices adjacent to x_i in P are x_{i+1} and x_{i-1} since P is a shortest path; we already argued that $y \notin P' \cup P''$. We hence found an induced cycle of length $k+1$, a contradiction. □

We now show how to reconstruct all edges in $E_{i,i}$ incident to u.

Claim 3.7. If $x \in N(u) \cap L_{i-1}$ and $y \in N(v) \cap L_{i-1}$ for some $uv \in E_{i,i}$, then $d_{G[L_{\leqslant i-1}]}(x, y) \leqslant 2\Delta k$.

Proof. Since $|N[\{u, v\}]| \leqslant 2\Delta$, it suffices to prove that $d_{G_1}(x, y) \leqslant k$ when the shortest path P in G_1 between x and y avoids other vertices from $N[\{u, v\}]$. As we argued in Claim 3.4, this is true when x or y is a neighbor of both u and v (else we create an induced cycle of length at least $k+1$). So we may assume that $x \in N(u) \setminus N(v)$ and $y \in N(v) \setminus N(u)$. But now $P \cup \{u, v\}$ is an induced cycle of length at least $k+1$. □

Let G_1' be the graph obtained from G_1 by adding the vertices in L_i and the edges in $E_{i,i-1}$. Our algorithm already reconstructed G_1'. If $uv \in E_{i,i}$, then applying Claim 3.7 to vertices $x, y \in L_{i-1}$ on the shortest paths from u, v to the root v_0 respectively, we find that $d_{G_1'}(u, v) \leqslant 2\Delta k + 2$. For each $u \in L_i$, we ask QUERY$(u, N_{G_1'}^{\leqslant 2\Delta k+2}(u) \cap L_i)$ and we record the vertices v for which the response is 1. Those are exactly the vertices adjacent to u. Per vertex $u \in L_i$, this takes at most $\Delta^{2\Delta k+3}$ queries.

The query complexity of reconstructing $E_{i-1,i}$ is $O_{k,\Delta}(\log n |L_i|)$ as there are at most $\log n$ iterations (using the fact that the $(S_j)_j$ are $\frac{1}{2}$-balanced separators) and in each iteration we do $O_{k,\Delta}(\log n)$ queries per vertex $u \in L_i$. To reconstruct $E_{i,i}$, we use $O_{k,\Delta}(1)$ queries per vertex of L_i. Therefore, the total query complexity of the algorithm is $\sum_i O_{k,\Delta}(|L_i| \log n) = O_{k,\Delta}(n \log n)$.

4 | Lower Bounds for Randomized Tree Reconstruction

In this section, we show that the algorithm presented in the previous section is optimal in terms of the dependency on n and Δ, even when randomization is allowed.

Theorem 1.1. *Let $\Delta \geqslant 2$ and $n = 2c\Delta^k$ be integers, where $c \in [1, \Delta)$ and $k \geqslant 50(c \ln c + 3)$ is an integer. Any randomized algorithm requires at least $\frac{1}{50}\Delta n \log_\Delta n$ queries to reconstruct n-vertex trees of maximum degree $\Delta + 1$.*

Note that, for constant c, Δ could even be a small polynomial in n. Any "algorithm" is allowed to be randomized unless specified to be deterministic.

4.1 | Reconstructing Functions From the Coordinate Oracle

To prove the lower bound, we reduce to a natural function reconstruction problem that could be of independent interest. Let $\Delta \geqslant 3$, $k \geqslant 1$ and $n = c\Delta^k$ be integers, where $c \in [1, \Delta)$.

Let $A = [n]$ and $B = [\Delta]^k$. Suppose that $f : A \to B$ is an unknown function that we want to reconstruct. For $b \in B$ and $i \in [k]$, we write b_i for the value of the ith coordinate of b.

The *coordinate oracle* can answer the following two types of queries:

- *Type 1*. $\textsc{Query}_1^c(a, b, i)$ for $a \in A$, $b \in [\Delta]$ and $i \in [k]$ answers **YES** if $f(a)_i = b$ and **NO** otherwise.
- *Type 2*. $\textsc{Query}_2^c(a, a', i)$ for $a, a' \in A$ and $i \in [k]$ answers **YES** if $f(a)_i = f(a')_i$ and **NO** otherwise.

In the case of the coordinate oracle, we will count the number of queries for which the answer is **NO** instead of the number of queries.

We say that $f : A \to B$ is a *balanced function* if for every $b \in B$, $|f^{-1}(b)| = c$ for some integer $c \geqslant 1$.

Our main result on function reconstruction from a coordinate oracle is the following.

Theorem 4.1. *Let $\Delta \geqslant 3$, $c \leqslant \Delta - 1$ and $k \geqslant 50(c \ln c + 2)$ be positive integers and let $n = c\Delta^k$. Any algorithm reconstructing $f : [n] \to [\Delta]^k$ using the coordinate oracle, in the special case where f is known to be a balanced function, has at least $\frac{1}{11}\Delta nk$ queries answered* **NO** *in expectation.*

To prove Theorem 4.1, we first study the query complexity in the general case, when no restriction is put on f. Using Yao's minimax principle [33], studying the expected complexity of a randomized algorithm can be reduced to studying the query complexity of a deterministic algorithm on a randomized input.

Lemma 4.2 (Corollary of Yao's minimax principle [33]). *For any distribution D on the inputs, for any randomized algorithm M, the expected query complexity of M is at least the average query complexity of the best deterministic algorithm for input distribution D.*

We will apply Yao's principle for D the uniform distribution and the query complexity measuring the number of queries answered **NO**. We combine this with the following lemma.

Lemma 4.3. *Let n, k and Δ be integers. For any deterministic algorithm R using the coordinate oracle and $f : [n] \to [\Delta]^k$ sampled u.a.r., the probability that R reconstructs f in at most $\frac{1}{10}\Delta nk$ queries answered* **NO** *is at most $e^{-\frac{1}{50}nk}$.*

We first deduce our main theorem on function reconstruction from the two lemmas above.

Proof of Theorem 4.1. Let M be a deterministic algorithm that reconstructs balanced functions using the coordinate oracle. We first extend M to an algorithm $\widetilde{M}$ that reconstructs all functions (among all functions) while the number of **NO** answers remains the same if the input is balanced. The algorithm $\widetilde{M}$ first performs the same queries as M does, until it either has no balanced candidates or a single balanced candidate f compatible with the answers so far. In the former case, it reconstructs the function by brute-force. In the second case, it performs $\textsc{Query}_1^c(a, f(a)_i, i)$ for all $a \in A$ and $i \in [k]$ to verify that indeed the input is f. If the input is indeed f, we have now distinguished f among all functions (rather than all balanced functions) without additional **NO** answers. If any of the queries answer **NO**, we again have no balanced candidates left and may perform the brute-force approach again.

We will show that, when restricted to balanced functions, $\widetilde{M}$ has an average query complexity (in terms of the number of **NO** answers) greater than $\frac{1}{11}\Delta nk$. Since M has the same number of **NO** answers as $\widetilde{M}$ on balanced inputs, it has the same average query complexity as $\widetilde{M}$. Using Yao's principle (Lemma 4.2), it then follows that any randomized algorithm that reconstructs balanced functions has at least $\frac{1}{11}\Delta nk$ queries answered **NO** in expectation.

By Lemma 4.3, there are at most $|B|^n e^{-\frac{1}{50}nk}$ functions $f : A \to B$ for which $\widetilde{M}$ reconstructs f in less than $\frac{1}{10}\Delta nk$ queries. On the other hand, the number of balanced functions from A to B is the following multinomial coefficient $\binom{n}{c,\dots,c} = \frac{n!}{(c!)^n}$. In particular, there are at least $\binom{n}{c,\dots,c} - (n/c)^n e^{-\frac{1}{50}nk}$ balanced functions for which $\widetilde{M}$ requires at least $\frac{1}{10}\Delta nk$ queries. This means that the average query complexity of $\widetilde{M}$ is at least

$$\frac{\binom{n}{c,\dots,c} - (n/c)^n e^{-\frac{1}{50}nk}}{\binom{n}{c,\dots,c}} \frac{1}{10}\Delta nk = \frac{n! - (c!)^n (n/c)^n e^{-\frac{1}{50}nk}}{n!}\frac{1}{10}\Delta nk \geqslant \frac{1}{11}\Delta nk$$

since,

$$(c!)^n (n/c)^n e^{-\frac{1}{50}nk} = \left(\frac{n}{e}\right)^n \left(\frac{ec!}{c} e^{-\frac{1}{50}k}\right)^n \leqslant n^n e^{-\frac{51}{50}n} \leqslant \frac{1}{100} n!$$

using for the first inequality that $k \geqslant 50(c \ln c + 2)$ and for the second that $n \geqslant 2^{51}$. □

Proof of Lemma 4.3. Let R be a deterministic algorithm that uses the coordinate oracle to reconstruct functions. Let F_t denote the set of possible functions $f : A \to B$ that are consistent with the first t queries done by R. (This depends on the input function $g : A \to B$, but we leave this implicit.) For $a \in A$ and $i \in [k]$, let

$$J^t_{a,i} = \{j \in [\Delta] | f(a)_i = j \text{ for some } f \in F_t\}$$

Note that all values $j_1, j_2 \in J^t_{a,i}$ are equally likely in the sense that there is an equal number of $f \in F_t$ with $f(a)_i = j_1$ as with $f(a)_i = j_2$. The algorithm R will perform the same t queries for all $f \in F_t$. In particular, if $g : A \to B$ was chosen uniformly at random, then after the first t queries all $f \in F_t$ are equally likely (as input function) and in particular $g(a)_i$ is uniformly distributed over $J^t_{a,i}$, independently of the sets $J^t_{a',i'}$ for $(a', i') \neq (a, i)$. This is the part for which we crucially depend on the fact that we allow all functions $f : A \to B$ and not just bijections (where there may be dependencies between the probability distributions of $g(a)$ and $g(a')$ for distinct $a, a' \in A$).

We say that the t^{th} query of the algorithm is *special* if

- it is a Type 1 query $\text{QUERY}^c_1(a, b, i)$ and $|J^t_{a,i}| \geqslant \Delta/2$, or
- it is a Type 2 query $\text{QUERY}^c_2(a, a', i)$ and either $|J^t_{a,i}|$ or $|J^t_{a',i}|$ is at least $\Delta/2$.

Let T denote the number of **NO**answers to *special* queries that R does to the coordinate oracle until it has reconstructed the input function. We let $Y_i = 1$ if the answer of the i^{th} special query is **YES**, and 0 otherwise. So $\sum_{i=1}^T Y_i$ denotes the number of special queries with answer **YES**.

At the start of the algorithm $J^0_{a,i} = [\Delta]$ for all $a \in A$ and $i \in [k]$. Thus, to reconstruct the function, the pair (a, i) is either (1) involved in a special query with answer is **YES** or (2) involved in $\Delta/2$ special queries for which the answer is **NO**. Since any query involves at most two elements of A, we deduce that

$$|A|k/2 = nk/2 \leqslant \left(T - \sum_{i=1}^T Y_i\right)\frac{2}{\Delta} + \sum_{i=1}^T Y_i$$

We aim to prove that if $g : A \to B$ is sampled uniformly at random, then with high probability $T = T(g) \geqslant \frac{1}{10}\Delta nk$. To do so, we consider a simplified process and a random variable τ which is stochastically dominated by T (i.e., for any $x \in \mathbb{R}^+$, $\mathbb{P}(T \leqslant x) \leqslant \mathbb{P}(\tau \leqslant x)$). Let us consider an infinite sequence of i.i.d. random variables $X_1, X_2, X_3, \dots \sim$ Bernouilli$(2/\Delta)$. Note that

$$H(t) = \left(t - \sum_{i=1}^t X_i\right)\frac{2}{\Delta} + \sum_{i=1}^t X_i = \left(1 - \frac{2}{\Delta}\right)\sum_{i=1}^t X_i + \frac{2t}{\Delta}$$

is increasing in t. Let τ be the first integer t for which $H(t) \geqslant \frac{1}{2} n \log_\Delta n$.

If g is sampled uniformly at random, then the jth special query (say involving $a \in A$ and $i \in [k]$ with $|J^t_{a,i}| \geqslant \Delta/2$) has answer **YES** with probability

$$\mathbb{P}(Y_i = 1) \leqslant \frac{2}{\Delta} = \mathbb{P}(X_i = 1)$$

This is because all values of $J^t_{a,i}$ are equally likely for $g(a)_i$ (and independent of the value of $g(a')_i$ or b_i for $b \in B$ and $a' \in A$). This inequality holds independently of the values of $(Y_1, \dots, Y_{i-1})$. This implies that, for any $t \in \mathbb{N}^+$ and any $x \in \mathbb{R}^+$,

$$\mathbb{P}\left(\sum_{i=1}^{t} X_i \leqslant x\right) \leqslant \mathbb{P}\left(\sum_{i=1}^{t} Y_i \leqslant x\right)$$

Therefore,

$$\mathbb{P}\left(\left(1 - \frac{2}{\Delta}\right)\sum_{i=1}^{t} X_i + \frac{2t}{\Delta} \leqslant x\right) \leqslant \mathbb{P}\left(\left(1 - \frac{2}{\Delta}\right)\sum_{i=1}^{t} Y_i + \frac{2t}{\Delta} \leqslant x\right)$$

From this we can conclude that $\mathbb{P}(T \leqslant x) \leqslant \mathbb{P}(\tau \leqslant x)$, thus T stochastically dominates τ.

If $\tau \leqslant \frac{1}{10}\Delta nk$, then using the definition of τ we find that

$$\left(\frac{1}{10}\Delta nk - \sum_{i=1}^{\tau} X_i\right)\frac{2}{\Delta} + \sum_{i=1}^{\tau} X_i \geqslant \frac{1}{2}nk$$

which implies

$$\sum_{i=1}^{\tau} X_i \geqslant \left(1 - \frac{2}{\Delta}\right)\sum_{i=1}^{\tau} X_i \geqslant \frac{3}{10}nk$$

Let $x = \frac{1}{10}\Delta nk$. We compute $\mathbb{E}\left[\sum_{i=1}^{x} X_i\right] = \frac{2}{\Delta}x = \frac{1}{5}nk$. Using Chernoff's inequality (see, e.g., [34]). we find

$$\mathbb{P}(\tau \leqslant x) \leqslant \mathbb{P}\left(\sum_{i=0}^{x} X_i \geqslant \left(1 + \frac{1}{2}\right)\frac{1}{5}nk\right) \leqslant \exp\left(-\left(\frac{1}{2}\right)^2 \frac{1}{5}nk \Big/ \left(2 + \frac{1}{2}\right)\right)$$

Since $\frac{1}{2}\frac{1}{2}\frac{1}{5}\frac{2}{5} = \frac{1}{50}$, this proves $\mathbb{P}(T \leqslant x) \leqslant \mathbb{P}(\tau \leqslant x) \leqslant e^{-\frac{1}{50}nk}$. In particular, the probability that at most $\frac{1}{10}\Delta k$ queries are used is at most $e^{-\frac{1}{50}nk}$, as desired. □

4.2 | Reconstructing Functions From the Word Oracle

Let once again $A = [n]$ and $B = [\Delta]^k$. We next turn our attention to reconstructing functions $f : A \to B$ from a more complicated oracle that we use as a stepping stone to get to distance queries in trees. For $b \in B$, we write $b_{[i,j]} = (b_i, b_{i+1}, \dots, b_j)$. It will also be convenient to define $b_\emptyset$ as the empty string. The **word oracle** can answer the following two types of questions.

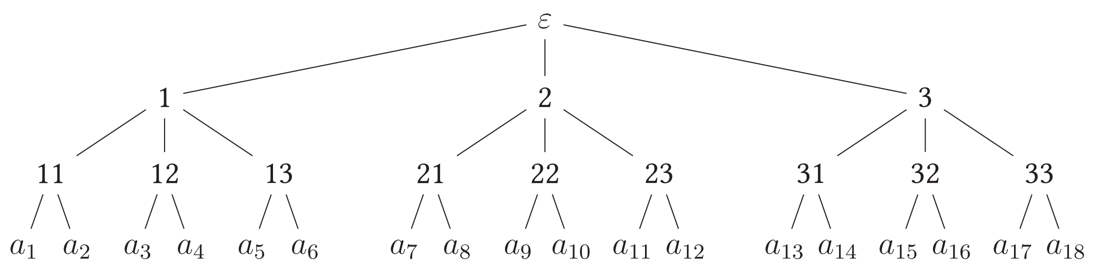
Type 1. $\text{Query}^w_1(a, b)$ for $a \in A$ and $b \in B$, answers the largest $i \in \{0, \dots, k\}$ with $f(a)_{[1,i]} = b_{[1,i]}$.

Type 2. $\text{Query}^w_2(a, a')$ for $a, a' \in A$, answers the largest $i \in \{0, \dots, k\}$ with $f(a)_{[1,i]} = f(a')_{[1,i]}$.

By studying the number of queries for the word oracle and the number of **NO** answers for the coordinate oracle, we can link the two reconstruction problems as follows.

Lemma 4.4. *For all positive integers Δ, k and n, for any algorithm M using the word oracle that reconstructs functions $f : A \to B$ in at most $q(f)$ queries in expectation, there exists an algorithm M' using the coordinate oracle that reconstructs functions $f : [n] \to [\Delta]^k$ such that at most $q(f)$ queries are answered* **NO** *in expectation.*

Proof. Given an algorithm M using the word oracle, we build a new algorithm M' using the coordinate oracle. We do so query-by-query. If M asks $\text{Query}^w_1(a, b)$, then M' performs a sequence of queries $\text{Query}^c_1(a, b_1, 1), \text{Query}^c_1(a, b_2, 2), \dots, \text{Query}^c_1(a, b_{i+1}, i + 1)$, where $i \in \{0, \dots, k - 1\}$ is the largest for which $f(a)_{[1,i]} =$



ε
1
2
3
11
12
13
21
22
23
31
32
33
a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10} a_{11} a_{12} a_{13} a_{14} a_{15} a_{16} a_{17} a_{18}

FIGURE 5 | Example of the tree $T_{c,\Delta,k}$ constructed in the proof of Theorem 1.1 for $\Delta = 4$, $c = 2$ and $k = 2$ with the labeling ℓ of the internal nodes.

$b_{[1,i]}$. Note that the sequence indeed simulates a query of the word oracle, yet the coordinate oracle answers **NO** at most once (on the $(i+1)$th query).

Queries of Type 2 can be converted analogously. This way, for every input f, the natural "coupling" of the randomness in M and M' ensures that the number of **NO** answers to M' is stochastically dominated by the number of queries to M. In particular, the expected number of **NO** answers given by M' is upper bounded by $q(f)$, the expected number of queries to M. □

Lemma 4.4 and Theorem 4.1 now give the following result.

Corollary 4.5. *Let $\Delta \geqslant 3$, $c \leqslant \Delta - 1$ and $k \geqslant 50(c \ln c + 2)$ be positive integers. Let $n = c\Delta^k$. Any algorithm reconstructing $f : [n] \to [\Delta]^k$ using the word oracle, in the special case where f is known to be balanced, needs at least $\frac{1}{11}\Delta nk$ queries in expectation.*

4.3 | Reducing Tree Reconstruction to Function Reconstruction

To prove Theorem 1.1 we consider a specific tree $T_{c,\Delta,k}$ (with $c \leqslant \Delta$): The tree of depth $k+1$ where each node at depth at most $k-1$ has exactly Δ children and each node at depth k has exactly c children (see Figure 5). Theorem 1.1 is an almost direct consequence of the following lemma.

Lemma 1.4. *Let $\Delta \geqslant 2$, $k \geqslant 150$ be positive integers. Consider a labeling of the tree $T_{\Delta,k}$ with $N = \Delta^k$ leaves, where only the labels of the leaves are unknown. Any randomized algorithm requires at least $\frac{1}{11}\Delta N \log_\Delta N$ queries in expectation to reconstruct the labeling.*

Proof. We consider $T = T_{c,\Delta,k}$ and let L be the set of leaves of T, and let P be the set of parents of the leaves. The tree T has $n = \sum_{i=0}^{k} \Delta^i + c\Delta^k$ nodes and $N = c\Delta^k$ leaves. Let $p : L \to P$ be the bijection that sends each leaf to its direct parent. We label internal nodes as follows. The root is labeled $\emptyset$ (the empty string) and if a node v has label ℓ and has Δ children, then we order the children $1, \ldots, \Delta$ and we label the child i with label obtained from concatenation $\ell + (i)$. We put such labels on all internal nodes.

Let I denote the set of internal nodes and let $\ell(v)$ denote the label of $v \in I$. Let $f : L \to [\Delta]^k$ be the bijection that sends a leaf $u \in L$ to the label $\ell(p(u)) \in [\Delta]^k$ of its direct parent.

We consider the trees that have a fixed labeling (as described above) for a node in I and every possible permutation of the labeling of the leaves. All possible bijections $f : L \to [\Delta]^k$ appear among the trees that we are considering. To reconstruct the tree, we in particular recover the corresponding bijection f. Distance queries between internal vertices always give the same response and can be ignored. We show that the other queries are Type 1 and Type 2 queries in disguise.

- For $a \in L$ and $b \in I$ the distance between a and b is given as follows. Let $z \in I$ be the nearest common ancestor of a and b, and say z has depth i and b has depth j. The distance between a and b is $1 + (k-i) + (j-i)$. The values of k and j do not depend on f but the value of i is exactly given by $\max\{s : f(a)_{[0,s]} = b_{[0,s]}\}$, the answer to the corresponding type 1 query of (a, b) to the word oracle. To be precise, since b may have a length shorter than i, the query $\text{QUERY}_1^w(a, b')$ where $b'_s = b_s$ for all $s \in [|b|]$ and $b'_s = 0$ otherwise, gives the desired information.

- For $a, a' \in L$, the distance between a and a' is given by $2(1 + (k - i))$ for i the answer of a Type 2 $\textsc{Query}_2^w(a, a')$ to the word oracle.

This shows that we reduce an algorithm to reconstruct the labeling of the leaves from q distance queries to an algorithm that reconstructs functions $f : L \to [\Delta]^k$ from q queries to the word oracle. By Theorem 4.1, since $k \geqslant 50(c \ln c + 2)$, we need at least $\frac{1}{11}\Delta N k$ queries. □

We are now ready to deduce the main result of this section.

Theorem 1.1. *Let $\Delta \geqslant 2$ and $n = 2c\Delta^k$ be integers, where $c \in [1, \Delta)$ and $k \geqslant 50(c \ln c + 3)$ is an integer. Any randomized algorithm requires at least $\frac{1}{50}\Delta n \log_\Delta n$ queries to reconstruct n-vertex trees of maximum degree $\Delta + 1$.*

Proof. Let $\Delta, n \geqslant 2$ be integers. We write $n = 2c\Delta^k$ for $c \in [1, \Delta)$ and k an integer. (When $n/2 \geqslant 1$, there is a unique pair $(c, k) \in [1, \Delta) \times \mathbb{Z}_{\geqslant 0}$ with $n/2 = c\Delta^k$.)

Suppose that $k \geqslant 50(c \ln c + 3)$. In particular, $\Delta \geqslant 2$ implies $k \geqslant \lfloor \log_\Delta n - 1 \rfloor$. The tree $T = T_{\lfloor c \rfloor, \Delta, k}$ considered in Lemma 1.4has maximum degree $\Delta + 1$, $N = \lfloor c \rfloor \Delta^k$ leaves and $n' = \sum_{i=0}^{k} \Delta^i + \lfloor c \rfloor \Delta^k$ vertices, where

$$n/4 \leqslant N \leqslant n' \leqslant 2c\Delta^k = n$$

For $\Delta \geqslant 2$ and $c \geqslant 1$, if $n \geqslant 2\Delta^{50(c \ln c+3)}$ then $N \geqslant \frac{n}{4} \geqslant \Delta^{50(c \ln c+2)}$. So we may apply Lemma 1.4 and find that at least

$$\frac{1}{11}\Delta N \lfloor \log_\Delta N - 1 \rfloor \geqslant \frac{1}{44}\Delta n(\log_\Delta n - 4)$$

queries are required. As $\log_\Delta n \geqslant 150$, we find that this is at least $\frac{1}{50}\Delta n \log_\Delta n$. □

Corollary 4.6. *Any randomized algorithm requires at least $\frac{1}{50}\Delta n \log_\Delta n$ distance queries to reconstruct n-vertex trees of maximum degree $\Delta + 1 \geqslant 3$ if $n \geqslant 2\Delta^{50(\Delta \ln \Delta+3)}$.*

4.4 | Randomized Lower Bounds for Related Models

We next show that our result implies various other new randomized lower bounds. Although we state these results with a weaker assumption on n for readability reasons, we remark that our more precise set-up (allowing Δ to be a small polynomial in n for specific values of n) also applies here.

4.4.1 | Betweenness Queries

A *betweenness query* answers for three vertices (u, v, w) whether v lies on a shortest path between u and w. Using three distance queries to (u, w), (u, v) and (v, w), you can determine whether v lies on a shortest path between u and w, so the betweenness oracle is weaker (up to multiplicative constants). It has been shown in Reference [18] that randomized algorithms can obtain a similar query complexity for betweenness queries as was obtained for distance queries by [3]. Moreover, a randomized algorithm for 4-chordal graphs has been given that uses a quasi-linear number of queries to a betweenness oracle [35]. A deterministic algorithm using $\widetilde{O}(\Delta n^{3/2})$ betweenness queries has been given for trees, as well as an $\Omega(\Delta n)$ lower bound [19]. Our randomized lower bound from Theorem 1.1 immediately extends to this setting.

Corollary 4.7. *Any randomized algorithm requires at least $\frac{1}{150}\Delta n \log_\Delta n$ betweenness queries to reconstruct n-vertex trees of maximum degree $\Delta + 1 \geqslant 3$ if $n \geqslant 2\Delta^{50(\Delta \ln \Delta+3)}$.*

4.4.2 | Path and Comparison Queries

Given two nodes i, j in a directed tree, a *path query* answers whether there exists a directed path from i to j. Improving on work from Reference [22], it was shown in Reference [21] that any algorithm needs $\Omega(n \log n + n\Delta)$ to reconstruct a directed tree on n nodes of maximum degree Δ. When we consider a directed rooted tree in which all edges are directed from parent to child, then path queries are the same as *ancestor queries*: Given u, v in a rooted tree, is u an ancestor of v? We apply this to the tree $T_{c,\Delta,h}$ from Lemma 1.4 for which the labels of all internal vertices are fixed, but the labels of the leaves are unknown. Path queries (u, v) only give new information if v is a leaf and u is an internal vertex. But this is weaker than distance queries, since we can obtain the same information by asking the distance between u and v. This means that we can redo the calculation from the proof of Theorem 1.1 (applying Lemma 1.4) to lift the lower bound to path queries.

Corollary 4.8. *Any randomized algorithm requires at least* $\frac{1}{50}\Delta n \log_\Delta n$ *path queries to reconstruct* n*-vertex directed trees of maximum degree* $\Delta + 1 \geqslant 3$ *if* $n \geqslant 2\Delta^{50(\Delta \ln \Delta + 3)}$.

A randomized algorithm using $O(n \log n)$ path queries on bounded-degree n-vertex trees has been given in Reference [21], but their dependency on Δ does not seem to match our lower bound. We remark that, besides query complexity, works on path queries such as [21, 22, 36] have also studied the round complexity (i.e., the number of rounds needed when queries are performed in parallel).

The same ideas apply to lift our lower bound to one for reconstructing tree posets $(T, >)$ from *comparison queries*, which answer, for given vertices u, v of the tree, whether $u < v$, $v < u$ or $u || v$.

Corollary 4.9. *Any randomized algorithm requires at least* $\frac{1}{50}\Delta n \log_\Delta n$ *comparison queries to reconstruct* n*-vertex tree posets of maximum degree* $\Delta + 1 \geqslant 3$ *if* $n \geqslant 2\Delta^{50(\Delta \ln \Delta + 3)}$.

This improves on the lower bound of $\Omega(\Delta n + n \log n)$ from Reference [30] and matches (up to a $(C \log \Delta)$ factor) the query complexity of their randomized algorithm.

4.4.3 | Membership Queries for Reconstructing Partitions

The (n, k)-partition problem was introduced by King, Zhang and Zhou [23]. Given n elements which are partitioned into k equal-sized classes, the partition needs to be determined via queries of the form "Are elements a and b in the same class?". They used the adversary method to prove $\Omega(nk)$ queries are needed by any deterministic algorithm. Liu and Mukherjee [25] studied this problem phrased as learning the components of a graph via membership queries (which answer whether given vertices lie in the same component or not) and provide an exact deterministic lower bound of $(k-1)n - \binom{k}{2}$ for deterministic algorithms. It indeed seems natural that randomized algorithms need αkn queries for some constant α in this setting. Nonetheless, the best lower bound for a randomized algorithm seems to be the information-theoretic lower bound of $\Omega(n \log k)$. Our next result remedies this gap in the literature.

Corollary 4.10. *Let* $\varepsilon > 0$. *Any randomized algorithm requires at least* $\frac{1}{11}nk$ *membership queries to solve the* (n, k)*-partition problem if* $n \geqslant k^{1+\varepsilon}$ *is sufficiently large.*

Proof. Since we plan to apply Lemma 4.3, we will write $\Delta = k$.

First, note that we can see the (n, Δ)-partition problem using *membership* queries as reconstructing a balanced function using only Type 2 queries to the coordinate oracle. Formally, if $f : [n] \to [\Delta]$ is the function which associates an element $a \in [n]$ to the index $i \in [\Delta]$ of the part that contains a (out of the Δ parts in the partition), then a membership query between $a, a' \in [n]$ is exactly equivalent to the coordinate query $\text{QUERY}_2^c(a, a')$ applied the function f. Once we reconstructed the partition, we can retrieve the index of each part using $\Delta^2 = o(n\Delta)$ queries of Type 1 to the coordinate oracle. Therefore, it suffices to show that at least $\frac{1}{11}\Delta n$ queries are needed in expectation to reconstruct a balanced function using the coordinate oracle.

Applying Lemma 4.3 with $k = 1$, we find that when f is sampled uniformly at random (among all functions $g : [n] \to [\Delta]$), the probability that a given randomized algorithm uses less than $\frac{1}{10}n\Delta$ queries is at most $e^{-\frac{1}{50}n}$. In particular, the number of balanced functions reconstructed in less than $\frac{1}{10}n\Delta$ queries is upper bounded by $\Delta^n e^{-\frac{1}{50}n}$. We compare this number to the total number $\binom{n}{n/\Delta, \dots, n/\Delta} = \frac{n!}{(n/\Delta)!^\Delta}$ of balanced functions:

$$\frac{\Delta^n e^{-\frac{1}{50}n}}{\frac{n!}{(n/\Delta)!^\Delta}} \leqslant \Delta^n e^{-\frac{1}{50}n} \frac{(2\pi n/\Delta)^{\Delta/2}(n/(e\Delta))^n}{\sqrt{2\pi n}(n/e)^n} e^{\Delta/(12n/\Delta)}$$
$$= \frac{1}{\sqrt{2\pi n}}(2\pi n/\Delta)^{\Delta/2}\exp(\Delta^2/(12n) - n/50)$$

Here we used that for all $n \geqslant 1$,

$$\sqrt{2\pi n}(n/e)^n < n! < \sqrt{2\pi n}(n/e)^n e^{1/(12n)}$$

Since $\Delta \leqslant n^{1/(1+\epsilon)}$ for some $\epsilon > 0$, the fraction tends to 0, so in particular becomes smaller than $\frac{1}{100}$ when n is sufficiently large (depending on ϵ). This implies that the expected number of queries to reconstruct a Δ-balanced function is at least $\frac{99}{100}\frac{1}{10}\Delta n \geqslant \frac{1}{11}n\Delta$. By the discussion at the start, we find the same lower bound for the (n, Δ)-partition problem. □

The same lower bound holds when queries of the form "Is element a in class i?" are also allowed. We expect that our methods can be adapted to handle parts of different sizes and that our constant $\frac{1}{11}$ can be easily improved.

Randomized algorithms have been studied in a similar setting by Lutz, De Panafieu, Scott, and Stein [24] under the name *active clustering*. They provide the optimal average query complexity when the partition is chosen uniformly at random among all partitions. They also study the setting in which a partition of n items into k parts is chosen uniformly at random and allow queries of the form "Are items i and j in the same part?". However, there is a key difference: The set of answers the algorithm receives needs to distinguish the partition from any other partition (including those with a larger number of parts). In this setting, the following algorithm is optimal. Order the items $1, \dots, n$. For $i = 1, \dots, n$, query item i to items $j = 1, \dots, i$ in turn if the answer to the query "Are items i and j in the same part?" is not yet known. It follows from Reference [24, Lemma 9] that this algorithm has the lowest possible expected number of queries. The expected number of queries used is at most

$$(1 + 2 + 3 + \cdots + k)\frac{1}{k}n = \frac{k+1}{2}n$$

In particular, for $i = \Omega(\log n)$ queries, with high probability, there are items in k different parts among the first $i - 1$ items. Since the number of parts is not "known," the algorithm will use k queries for item i if it is in the "last part," and so the complexity is $(\frac{k+1}{2} + o(1))n$ as $n \to \infty$. The same algorithm would use $(\frac{k+1}{2} - \frac{1}{k} + o(1))n$ queries when the number of parts may be assumed to be at most k, in which case queries "to the last part" are never needed. So, the assumption on whether the number of parts is known changes the query complexity. Together with the parts "being known to be exactly balanced," this introduces additional dependencies in the (n, k)-partition problem that our analysis had to deal with.

4.4.4 | Phylogenetic Reconstruction

This setting comes from biology. Reconstructing a phylogenetic tree has been modeled via what we call *leaf-distance* queries (similarity of DNA) between leaves of the input tree [23, 27, 37]. Although very similar, the query complexities of the phylogenetic model and the distance query model on trees are not directly related. In the phylogenetic model, the set of leaves is already known, and the leaf-distance queries are only possible between leaves. Moreover, we consider a phylogenetic tree to be reconstructed once we know all the pairwise distances between the leaves. For example, if the input tree is a path on n vertices, then in the phylogenetic setting we receive only two leaves and are finished once we query their distance, whereas in the distance reconstruction setting it takes $\Omega(n)$ queries to determine the exact edge set.

Improving on various previous works [26, 28, 29], King, Zhang, and Zhou [23, Theorem 32] showed that any deterministic algorithm reconstructing phylogenetic trees of maximum degree Δ with N leaves needs at least $\Omega(\Delta N \log_\Delta N)$ leaf-distance queries. Deterministic algorithms achieving this complexity are also known [27].

For randomized algorithms, the previous best lower bound was the information-theoretic lower bound of $\Omega(N \log N / \log\log N)$. We provide the following randomized lower bound from Lemma 1.4, which is tight up to a multiplicative constant.

Theorem 4.11. *Let $\Delta \geqslant 2$, $c \leqslant \Delta - 1$ and $k \geqslant 50(c \ln c + 2)$ be positive integers. Let $N = c\Delta^k$. Any randomized algorithm reconstructing phylogenetic trees of maximum degree $\Delta + 1$ with N leaves needs at least $\frac{1}{20}\Delta N \log_\Delta N$ leaf-distance queries in expectation.*

Proof. Let $T = T_{c,\Delta,k}$ be the tree considered in Lemma 1.4 with N leaves.

Suppose, towards a contradiction, that we could obtain the pairwise distances between the leaves of this tree in $\frac{1}{20}\Delta N \log_\Delta N$ leaf-distance queries in expectation. We show that, from this, we can recover the labels of the leaves of T using only $\Delta^2 N \leqslant \frac{1}{30}\Delta N \log_\Delta N$ additional distance queries, contradicting Lemma 1.4 since $\frac{1}{20} + \frac{1}{50} \leqslant \frac{1}{11}$ and $\log_\Delta N \geqslant k \geqslant 50$.

We proceed by induction on k, the depth of the tree T. When $k = 0$, $T_{c,\Delta,0}$ is a star with c leaves. There is nothing to prove, as the parent of each leaf is known to be the root. Suppose $k \geqslant 1$, and that the claim has already been shown for smaller values of k. We define an equivalence relation on the set of leaves: For $u_1, u_2 \in L$, $u_1 \sim u_2$ if and only if $d(u_1, u_2) < 2k$. This is an equivalence relation with Δ equivalence classes, as $d(u_1, u_2) < 2k$ if and only if u_1 and u_2 have a child of the root as common ancestor.

Let $u_1, u_2, \dots, u_\Delta$ be arbitrary representatives of each of the Δ equivalence classes. (Note that we can select these since we already know the distances between the leaves.) Let r denote the root of T. We ask $\text{QUERY}(u_i, N(r))$ for all $i \in [\Delta]$. From this, we can deduce the common ancestor among the children of the root for each of the classes. It is the unique neighbor of r lying on a shortest path from u_i to r. Let V_i denote the set of all the leaves that have the i^{th} neighbor of r as common ancestor. We also define T_i to be the subtree rooted in the i^{th} neighbor of r. We remark that it is now sufficient to solve Δ subproblems of reconstructing T_i knowing each V_i leaf matrix. By the induction hypothesis, each subproblem is solvable in $|V(T_i)|\Delta^2 = \frac{N-1}{\Delta}\Delta^2 = (N-1)\Delta$ queries. Therefore, in total this algorithm uses $(N-1)\Delta^2 + \Delta^2 = \Delta^2 N$ distance queries. □

5 | Open Problems

We presented new algorithms for reconstructing classes of graphs with bounded maximum degree from a quasi-linear number of distance queries and gave a new randomized lower bound of $\frac{1}{200}\Delta n \log_\Delta n$ for n-vertex trees of maximum degree Δ. This lower bound is now also the best lower bound for the class of bounded degree graphs, while the best-known randomized algorithm uses $\widetilde{O}_\Delta(n^{3/2})$ queries [13]. The main open question is to close the gap between our lower bound and this upper bound. In particular, it would be interesting to see if the quasi-linear query complexity achieved for various classes of graphs can be extended to all graphs of bounded maximum degree.

Problem 5.1. Does there exist a randomized algorithm that reconstructs an n-vertex graph of maximum degree Δ using $\widetilde{O}_\Delta(n)$ distance queries in expectation?

From a more practical viewpoint, studying the query complexity of reconstructing scale-free networks is of high importance as it is the class of graphs that best describes real-world networks like the Internet. Graphs in this class have vertices of large degree; therefore, recent theoretical works (including this one) do not directly apply. In particular, not even an $o(n^2)$ algorithm is known in this setting.

Problem 5.2. How many distance queries are needed for reconstructing scale-free networks?

We showed in this paper that *deterministic* algorithms with good query complexity exist for specific classes of graphs. One of the classes of graphs that does not fit in the scope of this paper but is known to have an efficient randomized algorithm is the class of bounded degree outerplanar graphs [3]. Note that outerplanar graphs also have a nice separator structure. For example, there is always a $\frac{1}{2}$-balanced separator which induces a path (see, e.g., [38, Proposition 6]).

Problem 5.3. Does there exist a deterministic algorithm that reconstructs an n-vertex outerplanar graph of maximum degree Δ in $\widetilde{O}_\Delta(n)$ distance queries?

We provided various randomized lower bounds in Section 4 that are tight up to a multiplicative constant. We expect that determining the exact constant for the optimal expected number of queries for reconstructing n-vertex trees of maximum degree Δ for the entire range of (n, Δ) may be complicated. We believe that for deterministic algorithms, when n is large compared to Δ (and takes particularly nice forms, e.g., $1 + \Delta + \cdots + \Delta^k$ for some integer $k \geqslant 1$), determining the correct constant could be achievable (yet would require new ideas). In particular, our simple algorithm may be even optimal up to an additive constant. We set the following challenge towards this.

Problem 5.4. Is there a constant $c > \frac{1}{2}$ such that for all $\Delta \geqslant 3$, there are infinitely many values of n for which any deterministic algorithm that reconstructs n-vertex trees of maximum degree Δ needs at least $c\Delta n \log_\Delta n$ queries?

We showed that the randomized and deterministic query complexities have the same dependence on n and Δ, and in that sense, randomness does not help much. If the answer to the question above is positive, it shows that randomness does at least make some difference.

A lower bound for trees immediately implies a lower bound for any class containing trees (such as k-chordal graphs), but a better lower bound could hold for bounded-degree graphs. In our algorithm for k-chordal graphs, we did not try to optimize the dependency on the maximum degree Δ and k, and the dependency on k can probably be improved upon. If one believes the answer to Problem 5.1 to be positive, then the dependency on k should be at most poly-logarithmic. It would be interesting to see if some dependency on k is needed. To ensure the lower bound needs to exploit cycles due to our $\Delta n \log_\Delta n$ algorithm for trees, we pose the following problem.

Problem 5.5. Is it true that for some fixed values of k and Δ and all n sufficiently large, any algorithm reconstructing k-chordal graphs on n vertices of maximum degree Δ requires at least $10^6 \Delta n \log_\Delta n$ queries?

Finally, we believe the problems of reconstructing functions from the coordinate or the word oracle are of independent interest. There are various variations that can be considered to which our methods may extend. For example, it is also natural to consider (bijective) functions $f : A \to B$ where both A and B are the same product of sets of different sizes (e.g., $A = B = [a_1] \times [a_2] \times \cdots \times [a_k]$).

Acknowledgments

We would like to thank Claire Mathieu for helpful suggestions and Guillaume Chapuy for helpful discussions regarding randomized lower bounds for the (n, k)-partition problem, which inspired our randomized lower bound proofs. We are also grateful to Jatin Yadav for pointing us to an issue we did not address in an earlier version, and we thank an anonymous referee for pointing out that our upper bound in Theorem 1.2 is different when $\Delta = 3$.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Endnotes

[1] If desired, we may define $T_j = T_\ell$ and $s_j = s_\ell$ for all $j \geqslant \ell$.

References

1. Z. Beerliova, F. Eberhard, T. Erlebach, et al., "Network Discovery and Verification," *IEEE Journal on Selected Areas in Communications* 24, no. 12 (2006): 2168–2181.

2. L. Reyzin and N. Srivastava, "Learning and Verifying Graphs Using Queries With a Focus on Edge Counting," in *Algorithmic Learning Theory: 18th International Conference, ALT 2007, Sendai, Japan, October 1-4, 2007. Proceedings 18* (Springer, 2007), 285–297.

3. S. Kannan, C. Mathieu, and H. Zhou, "Near-Linear Query Complexity for Graph Inference," in *International Colloquium on Automata, Languages, and Programming (ICALP)* (Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015), 773–784.

4. M. Göös and T. S. Jayram, "A Composition Theorem for Conical Juntas," in *31st Conference on Computational Complexity (CCC 2016)* (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016).

5. N. Leonardos, "An Improved Lower Bound for the Randomized Decision Tree Complexity of Recursive Majority," in *Automata, Languages, and Programming*, ed. F. V. Fomin, R. Freivalds, M. Kwiatkowska, and D. Peleg (Springer Berlin Heidelberg, 2013), 696–708.

6. F. Magniez, A. Nayak, M. Santha, J. Sherman, G. Tardos, and D. Xiao, "Improved Bounds for the Randomized Decision Tree Complexity of Recursive Majority," *Random Structures & Algorithms* 48, no. 3 (2016): 612–638.

7. R. Rivest and J. Vuillemin, "A Generalization and Proof of the Aanderaa-Rosenberg Conjecture," in *Proceedings of the Seventh Annual ACM Symposium on Theory of Computing (STOC)* (Association for Computing Machinery (ACM), 1975), 6–11.

8. A. Chakrabarti and S. Khot, "Improved Lower Bounds on the Randomized Complexity of Graph Properties," in *28th International Colloquium on Automata, Languages and Programming (ICALP)* (Springer Verlag, 2001), 285–296.

9. U. Feige, "On Sums of Independent Random Variables With Unbounded Variance, and Estimating the Average Degree in a Graph," in *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing* (Association for Computing Machinery (ACM), 2004), 594–603.

10. O. Goldreich and D. Ron, "Approximating Average Parameters of Graphs," *Random Structures & Algorithms* 32, no. 4 (2008): 473–493.

11. D. G. Corneil, S. Olariu, and L. Stewart, "Asteroidal Triple-Free Graphs," *SIAM Journal on Discrete Mathematics* 10, no. 3 (1997): 399–430.

12. Y. Dourisboure and C. Gavoille, "Tree-Decompositions With Bags of Small Diameter," *Discrete Mathematics* 307, no. 16 (2007): 2008–2029.

13. C. Mathieu and H. Zhou, "Graph Reconstruction via Distance Oracles," in *International Colloquium on Automata, Languages, and Programming (ICALP)* (Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2013), 733–744.

14. G. Rong, W. Li, Y. Yang, and J. Wang, "Reconstruction and Verification of Chordal Graphs With a Distance Oracle," *Theoretical Computer Science* 859 (2021): 48–56.

15. Y. Dourisboure, "Compact Routing Schemes for Generalised Chordal Graphs," *Journal of Graph Algorithms and Applications* 9, no. 2 (2005): 277–297.

16. P. Gartland, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, and P. Rzażewski, "Finding Large Induced Sparse Subgraphs in C_t-Free Graphs in Quasipolynomial Time," in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing* (Association for Computing Machinery (ACM), 2021), 330–341.

17. M. Pilipczuk, M. Pilipczuk, and P. Rzażewski, "Quasi-Polynomial-Time Algorithm for Independent Set in Pt-Free Graphs via Shrinking the Space of Induced Paths," in *Symposium on Simplicity in Algorithms (SOSA)* (SIAM, 2021), 204–209.

18. M. Abrahamsen, G. Bodwin, E. Rotenberg, and M. Stöckel, "Graph Reconstruction With a Betweenness Oracle," in *33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016)* (Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016), 5:1–5:14.

19. M. Jagadish and A. Sen, "Learning a Bounded-Degree Tree Using Separator Queries," in *Algorithmic Learning Theory (ALT)*, ed. R. M. SanjayJain, F. Stephan, and T. Zeugmann (Springer, 2013), 188–202.

20. R. Afshar, M. Goodrich, P. Matias, and M. Osegueda, "Reconstructing Binary Trees in Parallel," in *32nd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)* (Association for Computing Machinery (ACM), 2020), 491–492.

21. R. Afshar, M. T. Goodrich, P. Matias, and M. C. Osegueda, "Reconstructing Biological and Digital Phylogenetic Trees in Parallel," in *28th Annual European Symposium on Algorithms (ESA 2020)* (Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020), 3:1–3:24.

22. Z. Wang and J. Honorio, "Reconstructing a Bounded-Degree Directed Tree Using Path Queries," in *57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)* (IEEE, 2019), 506–513.

23. V. King, L. Zhang, and Y. Zhou, "On the Complexity of Distance-Based Evolutionary Tree Reconstruction," in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (Society for Industrial and Applied Mathematics, 2003), 444–453.

24. Q. Lutz, E. de Panafieu, A. Scott, and M. Stein, "Active Clustering for Labeling Training Data," in *Advances in Neural Information Processing Systems (NIPS)* (NeurIPS Foundation, 2021), 8469–8480.

25. X. Liu and S. Mukherjee, "Tight Query Complexity Bounds for Learning Graph Partitions," in *Proceedings of 35th Conference on Learning Theory (COLT)* (ML Research Press, 2022), 167–181.

26. G. S. Brodal, R. Fagerberg, C. N. S. Pedersen, and A. Ostlin, "The Complexity of Constructing Evolutionary Trees Using Experiments," *BRICS Report Series* 8, no. 1 (2001), https://www.brics.dk/RS/01/1/BRICS-RS-01-1.pdf.

27. J. Hein, "An Optimal Algorithm to Reconstruct Trees From Additive Distance Data," *Bulletin of Mathematical Biology* 51, no. 5 (1989): 597–603.

28. M.-Y. Kao, A. Lingas, and A. Ostlin, "Balanced Randomized Tree Splitting With Applications to Evolutionary Tree Constructions," in *Symposium on Theoretical Aspects of Computer Science (STACS)*, ed. C. Meinel and S. Tison (Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 1999), 184–196.

29. S. Kannan, E. Lawler, and T. Warnow, "Determining the Evolutionary Tree Using Experiments," *Journal of Algorithms* 21, no. 1 (1996): 26–50.

30. J. Roychoudhury and J. Yadav, "Efficient Algorithms for Sorting in Trees" (2022), arXiv:2205.15912.

31. N. Robertson and P. Seymour, "Graph Minors. II. Algorithmic Aspects of Tree-Width," *Journal of Algorithms* 7, no. 3 (1986): 309–322.

32. A. Kosowski, B. Li, N. Nisse, and K. Suchan, "*k*-Chordal Graphs: From Cops and Robber to Compact Routing via Treewidth," *Algorithmica* 72, no. 3 (2015): 758–777.

33. A. C.-C. Yao, "Probabilistic Computations: Toward a Unified Measure of Complexity," in *18th Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE Computer Society, 1977), 222–227.

34. M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis* (Cambridge University Press, 2017).

35. G. Rong, Y. Yang, W. Li, and J. Wang, "A Divide-and-Conquer Approach for Reconstruction of $C_{\geq 5}$-Free Graphs via Betweenness Queries," *Theoretical Computer Science* 917 (2022): 1–11.

36. R. Afshar and M. T. Goodrich, "Exact Learning of Multitrees and Almost-Trees Using Path Queries," in *LATIN 2022: Theoretical Informatics*, ed. A. Castañeda and F. Rodríguez-Henríquez (Springer, 2022), 293–311.

37. M. Waterman, T. Smith, M. Singh, and W. A. Beyer, "Additive Evolutionary Trees," *Journal of Theoretical Biology* 64, no. 2 (1977): 199–213.

38. E. Diot and C. Gavoille, "Path Separability of Graphs," in *Frontiers in Algorithmics*, ed. D.-T. Lee, D. Z. Chen, and S. Ying (Springer Berlin Heidelberg, 2010), 262–273.