Exact Learning of Weighted Graphs Using Composite Queries

Michael T. Goodrich[1], Songyu (Alfred) Liu[1], Ioannis Panageas[1]

[1]Department of Computer Science, University of California, Irvine, Irvine, CA, USA.

Contributing authors: goodrich@uci.edu; songyul4@uci.edu; ipanagea@uci.edu;

In this paper, we study the *exact learning* problem for weighted graphs, where we are given the vertex set, V, of a weighted graph, $G = (V, E, w)$, but we are not given E. The problem, which is also known as *graph reconstruction*, is to determine all the edges of E, including their weights, by asking queries about G from an oracle. As we observe, using simple shortest-path length queries is not sufficient, in general, to learn a weighted graph. So we study a number of scenarios where it is possible to learn G using a subquadratic number of composite queries, which combine two or three simple queries.

1 Introduction

The problem of exactly learning the edges of a graph from queries, which is also known as "graph reconstruction," is well-studied and well-motivated for both practical and theoretical reasons; e.g., see Mazzawi (2010); Abrahao et al. (2013); Assadi et al. (2021); Chen and Hwang (1989); Fang (1990); Shi and West (2001, 1999); Stegehuis and Weedage (2024); Choi (2013); Culberson and Rudnicki (1989); Kim (2012); Afshar et al. (2020); Mathieu and Zhou (2021); Jagadish and Sen (2013); Wang and Honorio (2017); Arunachaleswaran et al. (2023); Kannan et al. (2018); Afshar et al. (2022a,b); Afshar and Goodrich (2022); Eppstein et al. (2025). For instance, from a practical perspective, there are many contexts in which we know the vertices of a graph but must issue queries to learn its edges. Examples include social network science, where we know the members of a social network, but must issue queries to learn their relationships, such as friendship, collaboration, or even romantic partners; see, e.g., Yang et al. (2012); Stadtfeld and Pentland (2015); Backstrom and Kleinberg (2014).

Other practical examples include learning phylogenetic relationships through genetic tests Afshar et al. (2020), learning faults in circuit boards Chen and Hwang (1989); Fang (1990); Shi and West (2001), learning road networks from GPS traces Afshar et al. (2022b), and learning connections in a network, such as a peer-to-peer network or the Internet via routing queries Afshar et al. (2022a).

Furthermore, from the perspective of theoretical computer science, there are many interesting complexity theory and information theory questions that arise in graph reconstruction (e.g., see Mazzawi (2010); Stegehuis and Weedage (2024); Choi (2013); Afshar et al. (2020); Mathieu and Zhou (2021); Wang and Honorio (2017); Krivelevich and Zhukovskii (2025); Arunachaleswaran et al. (2023); Kannan et al. (2018); Afshar et al. (2022a,b); Afshar and Goodrich (2022)) including the following:

- What types of queries are sufficient to exactly learn all the edges in a graph and which ones are not?
- What kinds of queries can learn the edges of a graph using a subquadratic number of queries?
- What are the best algorithms for solving the graph reconstruction problem for a given set of queries?
- Are there non-trivial lower bounds for the number of queries needed to solve the graph reconstruction problem?
- What types of graphs are amenable to efficient exact learning algorithms?

Thus, there is ample practical and theoretical motivation for studying the graph reconstruction problem, where the vertices of the graph are known and we can only learn information about the edges through queries to an oracle with full knowledge of the graph. The goal is to find all the edges while minimizing the number of queries to the oracle.

Most of the existing literature on the graph reconstruction problem focuses on the case where the unknown graph is connected and unweighted, with notable results including work by Kannan, Mathieu, and Zhou Kannan et al. (2018); Mathieu and Zhou (2021). In this paper, we study the consequences of dropping these two assumptions, which leads to some interesting avenues of algorithmic exploration. For example, as we observe, previous types of queries that are sufficient to learn unweighted connected graphs turn out to be insufficient to learn all disconnected weighted graphs. Thus, we need new types of queries to solve the graph reconstruction problem for disconnected weighted graphs. Indeed, taking inspiration from how combinations of drugs can be used to treat diseases Foucquier and Guedj (2015), such as cancer and AIDS, the methods we explore in this paper involve solving the graph reconstruction problem through composite queries, which combine different types of queries in concert to efficiently learn the edges and their weights in a graph.

1.1 Queries

Let us therefore consider different types of queries that might be used to exactly learn a weighted graph efficiently. Assume that a given weighted graph, G, has n known vertices, m unknown edges, and let D denote the maximum degree of any vertex in the graph.

Wang and Honorio study the problem of reconstructing weighted directed trees using what the authors call "additive queries," which return the sum of the edge weights on the directed path between two given vertices and 0 otherwise Wang and Honorio (2017). Of course, unlike in trees, there can be multiple paths between two vertices in a general graph; hence, let us define the following analogous query for a possibly disconnected weighted graph:

- **Distance query**: define the query, q_d, to return the sum of the edge weights on a shortest weighted path between two given vertices and $+\infty$ if there is no path.

Note that we use $+\infty$ instead of 0 so that the distance returned by the query satisfies the triangle inequality. Kannan *et al.* Kannan et al. (2018) show that a connected unweighted graph can be learned using an oracle that returns an entire shortest path between a given pair of vertices, or an oracle that returns the distance of a shortest path between them, where distance is measured by the number of edges, which is also known as the "hop count." Unfortunately, as Figure 1 shows, the weighted shortest path may not correspond to the unweighted shortest path; hence, a solution to the weighted case cannot easily be derived from existing solutions to the unweighted case. Moreover, the distance query, q_d, only returns a single number, not an entire path in the graph.

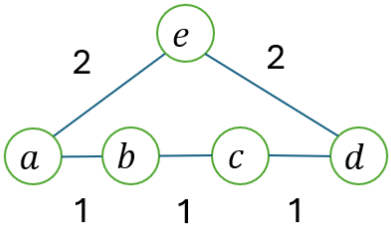

Fig. 1 The weighted shortest path and the unweighted shortest path from a to d are different. In a weighted graph, the bottom path is the shortest. However, if we ignore the weights and treat it as an unweighted graph, then the top path will be the shortest with only two hops.

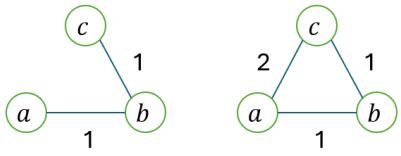

Fig. 2 Using query q_d alone, the two graphs will lead to the same query results for all pairs of vertices. The existence of the edge ac cannot be confirmed or denied. ac is called a transitive edge (Definition 1) in the graph on the right.

Indeed, there are weighted graphs that cannot be reconstructed using q_d queries alone. See Figure 2. Related to this issue, the notion of transitive edges was introduced to describe similar issues by Wang and Honorio Wang and Honorio (2017) and we redefine it here in our context.

Definition 1 Let $G = (V, E, w)$ be an undirected and weighted graph where $w(u, v)$ returns the weight of an edge, uv, and 0 if there is no edge. An edge uv is *transitive* if a path exists between u and v whose weighted length $l \leq w(u, v)$.

Transitive edges cannot be recovered by q_d alone. It might be possible to exclude such edges by imposing specific constraints on the graph. However, we want our algorithm to work on a more general class of graphs, so we need an additional type of query. In particular, we introduce the following additional type of query:

- **Edge-weight query**: the query, q_w, takes two vertices, u and v, and returns $w(u, v)$.

Using q_w alone, it is at least possible to reconstruct any weighted graph, but it is necessary to query every pair of vertices, leading to an $\Theta(n^2)$ brute-force solution. We refer to this algorithm as "EXHAUSTIVE-QUERY." To accommodate disconnected graphs and obtain better query complexities, therefore, we define yet another type of query and we consider layering the input graph based on weight thresholds. In network mapping, tools such as `traceroute` provide delay information, which can be seen as distances between vertices Afshar et al. (2022a); Kannan et al. (2018). Weights above a certain threshold can indicate high delay in the network.

Dividing a weighted graph into layers based on the weights of the edges revealed interesting within-layer and across-layer connections between topology and edge weights in real-world datasets Bu et al. (2023). This motivates us to layer the input graph. Following the definition in Bu et al. (2023), layer W_{thr} of a graph G is defined as $G[w \geq W_{\text{thr}}]$, the subgraph of G including all vertices of G, but only edges with weight $w \geq W_{\text{thr}}$. W_{thr} is a weight threshold that will change in our algorithm. This subgraph may be disconnected. If $w_1 < w_2, G[w \geq w_1] \supseteq G[w \geq w_2]$. All the queries can take an additional parameter, W_{thr}, and give us information within this subgraph. This ability to query within a subgraph is essential because the distances between vertices in this subgraph may differ significantly from their distances in the original graph. There is no straightforward relationship between these two distances under our general assumptions on edge weights that we will explain in the next section.

Let $q_d(u, v, W_{\text{thr}})$ return the sum of the edge weights on a shortest weighted path between vertices u and v in $G[w \geq W_{\text{thr}}]$, and $+\infty$ if there is no such path. Similarly, let $q_w(u, v, W_{\text{thr}})$ return $w(u, v)$ in $G[w \geq W_{\text{thr}}]$ and 0 when there is no edge. Let EXHAUSTIVE-QUERY(V, W_{thr}) call $q_w(u, v, W_{\text{thr}})$ for every pair of vertices in V. To work with disconnected graphs, we need one more type of query, which can detect connected components efficiently. Relying solely on q_d for this purpose would result in prohibitively high query complexity. Suppose a graph has n vertices and k connected components.Liu and Mukherjee (2022) showed a lower bound of $kn - \binom{k+1}{2}$ queries and k can be as large as $\Theta(n)$ in our case. This necessitates another type of query. Liu and Mukherjee Liu and Mukherjee (2022) proposed the following oracle to learn k connected components using $\Theta(n \log k)$ queries: for any vertex u and set $S \subseteq V$ not containing u,

$$\alpha_m(u, S) = \begin{cases} 1 & \text{if for some } v \in S,\ u \text{ and } v \text{ belong to the same component,} \\ 0 & \text{otherwise.} \end{cases}$$

Incorporating the weight threshold, we have:

- **Component query**: let $q_c(u, S, W_{\text{thr}})$ return $\alpha_m(u, S)$ in $G[w \geq W_{\text{thr}}]$.

Our composite query q therefore, is a mixture of $q_w(u, v, W_{\text{thr}})$, $q_d(u, v, W_{\text{thr}})$, and $q_c(u, S, W_{\text{thr}})$. Every time we perform a query, we can choose the query type that is most important. In addition, we assume that edges in the original input graph have weight at least 1, and allow composite queries that fix $W_{\text{thr}} = 1$, in which case these queries return the corresponding quantities in the original input graph.

1.2 Related Work

As mentioned above, existing literature on the graph reconstruction problem tends to focus on connected unweighted graphs and the following query oracles defined for such graphs, with distance determined by hop count, e.g., see Kannan et al. (2018); Afshar et al. (2022a):

- An unweighted *distance* oracle in an unweighted graph returns the hop-count distance between a given pair of vertices.
- A *shortest-path* oracle returns an ordered list of the vertices of a shortest path between a given pair of vertices.
- A **k***th-hop* oracle returns the kth vertex on a shortest path between a given pair of vertices.

For example, Kannan, Mathieu, and Zhou Kannan et al. (2018) present an algorithm for reconstructing an unweighted, connected, bounded-degree graph using $\tilde{O}(n^{3/2})$ distance queries,[1] where one randomly selects centers and then partitions the graph into slightly overlapping Voronoi-cell subgraphs based on the chosen centers until the graph clusters were all small enough to perform an exhaustive query on each cluster. Afshar *et al.* Afshar et al. (2022b) proposed a similar approach with different Voronoi cell sizes and a query complexity that depended on the dual graph connecting neighboring cells. Mathieu and Zhou Mathieu and Zhou (2021) focused on random regular graphs, giving an algorithm that uses only $\tilde{O}(n)$ distance queries in this special case.

If we view the graph as an unknown non-negative $\binom{n}{2}$ dimensional vector, x_G, with supp (x_G) denoting the non-zero coordinates, the following queries can be defined Assadi et al. (2021):

- *Linear queries*: Given any non-negative $\binom{n}{2}$ dimension vector a_G, what is $a_G \cdot x_G$?
- *OR queries*: Given any subset S of the $\binom{n}{2}$ dimensions, is supp $(x_G) \cap S$ empty?

Linear queries are more general than cross-additive queries, which in weighted graphs return the sum of the weights of the edges crossing between two given sets of vertices, and Choi Choi (2013) presents one of the few existing results for reconstructing graphs with real weights using cross-additive queries.

A closely related, and more powerful query, is the general *additive* query. For any given set of vertices, the additive query returns the sum of the weights of the edges with both endpoints in the set.[2] Using this additive query, Mazzawi Mazzawi (2010) gives a reconstruction algorithm that matches the information-theoretic lower bound

[1] As is common, we use $\tilde{O}(*)$ to denote asymptotic bounds that ignore polylogarithmic factors.
[2] Note that this is not the same as the additive query defined by Wang and Honorio in the context of weighted trees Wang and Honorio (2017).

for graphs with positive integer weights. Alternatively, when the weights are bounded real numbers that are all positive (or negative), Kim Kim (2012) reduces the graph reconstruction problem using additive queries to a coin-weighing problem.

1.3 Our Results

In this paper, we study weighted graph reconstruction problems for graphs that are not necessarily connected and for connected graphs, and in both cases weights are real numbers in the range, $[1, W_{\max}]$. Table 1 shows our results. The first row where the weighted graph need not be connected is our main contribution. The second row is a natural generalization of the result presented in Kannan et al. (2018) and we provide the detailed analysis for this case in Section A. The main body of this paper will focus on the first case. The idea of our approach is to divide the graph into layers and use a Voronoi-cell approach to reconstruct each connected component in each layer. This way we will discover edges with weight in consecutive intervals, $[1, 2), [2, 4), [4, 8)$, and so on. Although an unweighted tree reconstruction algorithm based on layers was proposed in Bastide and Groenland (2025), it is not comparable to our approach. In Bastide and Groenland (2025), layers were defined to partition vertices based on their distances to the root of the tree. This is more natural in their context because trees admit a balanced vertex separator of size 1 Bastide and Groenland (2025). However, it is more convenient to define layers based on edges in our case.

We assume the input graph has m edges and maximum degree $D \gg 1$. We further assume that $m/D = \omega(1)$[3]. This condition of $m/D = \omega(1)$ is relatively mild. For instance, if $D = o(n)$ and the original input graph is connected, then $m \geq n - 1$ and the condition will be satisfied. For reconstructing graphs of unbounded degree using distance queries, Kannan *et al.* established an $\Omega(n^2)$ lower bound, demonstrating that bounded degree $D = o(n)$ is indeed a necessary assumption Kannan et al. (2018). We can further simplify the results when $D = O(\text{polylog}(n))$, an assumption made by Kannan *et al.* Kannan et al. (2018). However, we highlight the dependency on D in our query complexity results. If $W_{\max} = 1$, then the input graph is effecitvely unweighted, which implies a query complexity of $\tilde{O}(D^3 n^{3/2})$ when the input graph is connected, matching the complexity of the algorithm by Kannan *et al.*

Table 1 Our results for weighted graphs with maximum degree D. α is a positive constant to be defined in the next section.

Connected?	Queries	Query complexity
No	$q_w(u, v, W_{\text{thr}}), q_d(u, v, W_{\text{thr}}), q_c(u, S, W_{\text{thr}})$	$\tilde{O}\left(\left(1 + \frac{1}{\alpha} \log D\right) D^3 n^{3/2}\right)$
Yes	$q_w(u, v, 1), q_d(u, v, 1)$	$\tilde{O}(D^{3W_{\max}} n^{3/2})$

Our algorithms introduce a number of interesting techniques for solving the graph reconstruction problem for weighted and possibly disconnected graphs. For example, the approach of using queries with weight thresholds allows us to design efficient query algorithms even with edge weights that can vary significantly. Typically, when adapting

[3] $\omega(1)$ denotes a function that tends to infinity as the underlying parameter n tends to infinity.

algorithms from unweighted graphs to weighted graphs, the runtime becomes dependent on the aspect ratio, defined as the maximum weight divided by the minimum weight Bernstein et al. (2024). Therefore, it is desirable to have a graph with a small aspect ratio. Transforming a graph into another one with a smaller aspect ratio while preserving its shortest path structure is extremely difficult Bernstein et al. (2024). In contrast, our approach goes beyond adapting and makes it possible to get rid of this dependency. This stems from our probabilistic model where edge weights are randomly drawn from a distribution. Such a model may represent scenarios like road networks, where vertices correspond to intersections, edges to roads, and edge weights to traffic conditions. While the graph structure remains fixed, the traffic follows a probability distribution. This probabilistic model is a fundamental distinction between our work and that of Kannan et al. (2018). Building upon a specific edge weight distribution, we are able to eliminate the $\log_2 W_{\max}$ factor that appears in our algorithm, which is a key contribution of this paper. Furthermore, existing literature only uses one type of query to reconstruct graphs. A single type of query may have limitations but at the same time may excel at certain tasks. The constituent parts of our composite query either fail to reconstruct the graph or do so inefficiently. However, we can design a powerful algorithm by combining them.

2 Preliminaries

We largely follow the notation introduced by Kannan *et al.* Kannan et al. (2018), for the sake of consistency. Let $G = (V, E, w)$ be an undirected and weighted graph, where V is the vertex set, E is the edge set, and w is the weight function. When the context is clear, we also use w to denote the weight of a single edge. Let $d(u, v)$ denote the weighted shortest path distance of the underlying graph. The underlying graph can change since we work with different layers of the graph and different connected components. In most cases, the distance is the one returned by the query q_d and should be clear from the context. For a subset of vertices $S \subseteq V$ and a vertex $v \in V$, define $d(S, v) = \min_{s \in S} d(s, v)$. Define δ as the hop distance, i.e. the unweighted shortest path distance. For $v \in V$, define the neighborhood of v as $N(v) = \{u \in V : \delta(u, v) \leq 1\}$, and define the neighborhood of v within 2 hops as $N_2(v) = \{u \in V : \delta(u, v) \leq 2\}$.

For a pair of distinct vertices $u, v \in V$, we say that uv is an edge of G if and only if $uv \in E$. For a subset of vertices $S \subseteq V$, let $G[S]$ be the subgraph induced by S. A closely related notation is $G[w \geq W_{\text{thr}}]$, the edge induced subgraph of G including only edges with weight $w \geq W_{\text{thr}}$ where W_{thr} is a weight threshold. When we pass a vertex $s \in V$ and a subset $T \subseteq V$ to a query, we perform the query for every pair in their Cartesian product. When we pass two subsets of vertices, we also perform the query for every pair in their Cartesian product.

Recall that edges have real weights of at least 1. We explore two theoretical models regarding the weights, a general one and a specific one. When the input is a weighted graph or tree, existing literature assumes that the weights are fixed numbers Choi (2013); Wang and Honorio (2017). In contrast, we study a model where the structure of the input graph is fixed and the edge weights are sampled from a continuous distribution whose support is $[1, +\infty)$. We use $F(w)$ to denote its cumulative distribution

Algorithm 1 Layer-by-layer reconstruction (LBL-R)

function LBL-R(V)
  $E \leftarrow \{\}$
  for $j \leftarrow 0, 1, 2, \ldots, \lfloor \log_2 W_{max} \rfloor$ **do**
    $W_{\text{thr}} \leftarrow 2^j$
    $C \leftarrow$ FIND-CONNECTED-COMPONENTS(V, W_{thr})
    for $c \in C$ **do**
      if $|c| \leq n^{1/4}$ **then** ▷ A small connected component.
        $E_{jc} \leftarrow$ EXHAUSTIVE-QUERY(c, W_{thr})
      else
        $E_{jc} \leftarrow$ RECONSTRUCT(c, W_{thr})
      end if
      $E \leftarrow E \cup E_{jc}$
    end for
    if $\max_{c \in C} |c| \leq n^{1/4}$ **then** ▷ All the connected components are small.
      break
    end if
  end for
  return E
end function

function (CDF). Suppose the graph has m edges and we have m independent and identically distributed (i.i.d.) samples from this distribution. The maximum weight of the graph $W_{\max}$ is the largest order statistic and its CDF is $F(w)^m$. We assume that the observed value of this random variable in the input graph is known to the algorithms and we also denote it by $W_{\max}$. The distinction will be clear from the context.

In Section A, we make no further assumptions about the distribution of edge weights, only that they fall within the range $[1, W_{\max}]$. Section 3 maintains this range constraint but introduces a specific distribution for the weights. Experiments on real-world graphs suggest that edge weights follow a power law distribution (Pareto distribution) Kumar et al. (2020). Specifically, we use a Pareto distribution with lower threshold 1 and parameter $\alpha > 0$. Therefore, the edge weights are i.i.d. random variables with CDF

$$F(w) = 1 - w^{-\alpha}, \quad \text{for } w \in [1, +\infty).$$

If we want to make sure the variance exists, we can assume $\alpha > 2$. However, as long as it is a positive constant, our results will hold.

3 Queries with a Threshold

In this section, we design a layer-by-layer reconstruction (LBL-R) algorithm (Algorithm 1) that can reconstruct weighted graphs that are not necessarily connected using composite queries. We then prove its correctness and show that its query complexity is $\tilde{O}\left(\left(1 + \frac{1}{\alpha} \log D\right) D^3 n^{3/2}\right)$.

3.1 Description of LBL-R

First, we provide a high-level description of our algorithm. This algorithm takes as input the set V, which is all the vertices in the given graph, and recovers all the edges of the graph. It works iteratively. During iteration j, we set $W_{\text{thr}} = 2^j$ and find the connected components of $G[w \geq W_{\text{thr}}]$. If a connected component is small, then we can afford an exhaustive search. This will discover all of its edges with weight $w \geq W_{\text{thr}}$. If a connected component is large, we will call RECONSTRUCT, which has a lower query complexity than an exhaustive search, but is not guaranteed to find the same set of edges. The missing edges will be found in subsequent iterations. If all the connected components are small, we will exhaustively search all of them and this is the time to break out of the outer loop.

Remark 1 In fact, once EXHAUSTIVE-QUERY is called on a small connected component in iteration j, we do not need additional queries involving its vertices. It is possible to exclude these vertices after this iteration. The asymptotic query complexity will be the same, so we do not include this change for simplicity.

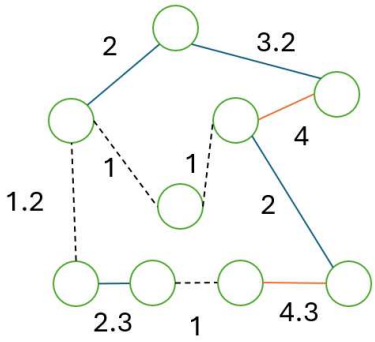

Fig. 3 Iteration 1 of LBL-R (Algorithm 1). $W_{\text{thr}} = 2$. Edges with weight $w < W_{\text{thr}}$ have been discovered and are represented by dashed lines. $G[w \geq W_{\text{thr}}]$ has three connected components, one of which only has one vertex. Edges with weight $w \in [W_{\text{thr}}, 2W_{\text{thr}})$ are guaranteed to be discovered and are represented by solid blue lines. Edges with weight $w \geq 2W_{\text{thr}}$ will be discovered in future iterations and are represented by solid orange lines.

Next, we explain the subroutines used in our main algorithm. SAMPLE(W, s) receives a set of vertices W and a number s and returns a random subset of W by selecting each element independently with probability $s/|W|$ Kannan et al. (2018). If $|W| \leq s$, then SAMPLE(W, s) returns the set W. As a direct result of our component query, FIND-CONNECTED-COMPONENTS finds connected components of $G[w \geq W_{\text{thr}}]$ with query complexity $\tilde{O}(n)$ Liu and Mukherjee (2022). The other subroutine RECONSTRUCT can be treated as a black box algorithm for our purpose. RECONSTRUCT and its subroutines are adaptations of the algorithms presented in Kannan et al. (2018), using our specific queries. While our LBL-R algorithm builds upon this subroutine used for the unweighted, connected case in Kannan et al. (2018), we have introduced significant modifications to reconstruct a different class of graphs. When the parameter W_{thr} is set to 1, our RECONSTRUCT subroutine includes the

Algorithm 2 Finding connected components

function FIND-CONNECTED-COMPONENTS(V, W_{thr})
  Arbitrarily order $V = \{v_1, v_2, \ldots\}$
  Initialize $C_1 = \{v_1\}$, $k = 1$
  for $i = 2$ to $|V|$ **do**
    if $q_c(v_i, C_1 \cup \cdots \cup C_k, W_{\text{thr}}) = 0$ **then**
      $k \leftarrow k + 1$
      Add v_i to C_k
    else
      Find j such that $q_c(v_i, C_j, W_{\text{thr}}) = 1$ via binary search among $\{C_1, \ldots, C_k\}$
      Add v_i to its corresponding C_j
    end if
  end for
  return $\{C_1, \ldots, C_k\}$
end function

Algorithm 3 Finding $N_2(a)$

function FIND-NEIGHBORS(V, a, W_{thr})
  $N(a) \leftarrow \{v \in V : q_w(a, v, W_{\text{thr}}) \neq 0\}$ ▷ Find all neighbors of a.
  $N_2(a) \leftarrow N(a)$
  for $v \in N(a)$ **do** ▷ For each neighbor of a. At most D neighbors.
    $N(v) \leftarrow \{u \in V : q_w(u, v, W_{\text{thr}}) \neq 0\}$ ▷ Find all neighbors of v.
    $N_2(a) \leftarrow N_2(a) \cup N(v)$ ▷ Add these to $N_2(a)$.
  end for
  return $N_2(a)$
end function

original one in Kannan et al. (2018) as a special case. Now we briefly review this original version for completeness.

For a connected graph G_c with bounded degree, it first finds a set of vertices A and each vertex $a \in A$ will become the center of an extended Voronoi cell $D_a \subseteq V$. These cells are overlapping. Then an exhaustive search is performed within each cell and the union of vertex-induced subgraphs $\bigcup_{a \in A} G_c[D_a]$ will cover all the edges. Finally, because this whole procedure is randomized, its query complexity may differ. If the number of queries issued is too large, it will stop the ongoing procedure and retry.

We now consider the query complexity of RECONSTRUCT. The query complexity proofs in Kannan et al. (2018) can be directly extended to accommodate any metric between vertices of a connected graph with bounded degree. We present the following lemma without proof.

Lemma 1 (Extension of Section 2.2.2 in Kannan et al. (2018)) For connected graphs with n vertices and maximum degree D, the randomized algorithm RECONSTRUCT has expected query complexity $\tilde{O}(D^3 n^{3/2})$.

Algorithm 4 Selecting centers by estimation

function ESTIMATED-CENTERS(V, s, W_{thr})
  $n \leftarrow |V|$
  $A \leftarrow \emptyset$, $W \leftarrow V$
  $T \leftarrow K \cdot s \cdot \log n \cdot \log\log n$
  while $W \neq \emptyset$ **do**
    $A' \leftarrow$ SAMPLE(W, s)
    $q_d(A', V, W_{\text{thr}})$ ▷ Every pair in $A' \times V$ is queried.
    $A \leftarrow A \cup A'$
    for $w \in W$ **do**
      $X \leftarrow$ random multi-subset of V with T elements
      $q_d(X, w, W_{\text{thr}})$
      $\tilde{C}(w) \leftarrow |\{v \in X : d(w, v) < d(A, v)\}| \cdot n/T$
    end for
    $W \leftarrow \{w \in W : \tilde{C}(w) \geq 5n/s\}$
  end while
  return A
end function

Algorithm 5 Subroutine to reconstruct the graph once

function RECONSTRUCT-SUB(V, W_{thr})
  $n \leftarrow |V|, s \leftarrow D\sqrt{n}$
  $A \leftarrow$ ESTIMATED-CENTERS(V, s, W_{thr}) ▷ Every pair in $A \times V$ is queried.
  for $a \in A$ **do**
    $N_2(a) \leftarrow$ FIND-NEIGHBORS(V, a, W_{thr})
    $q_d(N_2(a), V, W_{\text{thr}})$ ▷ Every pair in $N_2(a) \times V$ is queried.
    for $b \in N_2(a)$ **do**
      $C(b) \leftarrow \{v \in V : d(b, v) < d(A, v)\}$
    end for
    $D_a \leftarrow \bigcup\{C(b) : b \in N_2(a)\} \cup N_2(a)$
    $E_a \leftarrow$ EXHAUSTIVE-QUERY(D_a, W_{thr})
  end for
  return $\bigcup_a E_a$
end function

The query complexity result assumes that n is large, so we do not apply RECONSTRUCT to all connected components. Note that the expectation is with respect to (wrt) the random choices in the algorithm and does not depend on the random weights of our input graph. Also, this lemma only states the query complexity and not the correctness. In fact, RECONSTRUCT by itself may not correctly reconstruct our input graph and a more detailed analysis is in the subsequent section.

Algorithm 6 Reconstructing the graph

```
function RECONSTRUCT(V, W_thr)
    n ← |V|
    Set query limit Q = O(D^3 · n^{3/2} · log^2 n · log log n)
    E ← RECONSTRUCT-SUB(V, W_thr)
    if RECONSTRUCT-SUB does not terminate after Q queries then
        Stop RECONSTRUCT-SUB
        Re-execute this algorithm from the beginning
    end if
    return E
end function
```

3.2 Correctness of LBL-R

To show the correctness of our algorithm LBL-R, we first analyze what happens in one iteration for one connected component. Our analysis focuses on RECONSTRUCT-SUB, which is the core procedure repeatedly executed by the RECONSTRUCT algorithm, and the result will directly extend to RECONSTRUCT.

Lemma 2 Suppose we set the threshold in the queries to W_{thr}. $G[w \geq W_{\text{thr}}]$ consists of one or more connected components. Let G_c be a connected component and let c be its vertices. $G_c[w < 2W_{\text{thr}}]$ is the subgraph of G_c induced by edges with weight $w < 2W_{\text{thr}}$. For function call RECONSTRUCT-SUB(c, W_{thr}), we have $G_c[w < 2W_{\text{thr}}] \subseteq \bigcup_{a \in A} G_c[D_a]$.

Proof Let d be the distance in the connected component G_c. For any two vertices in it, the distance between them in G_c equals their distance in $G[w \geq W_{\text{thr}}]$, which is what q_d would return with threshold W_{thr}.

Let uv be any edge of $G_c[w < 2W_{\text{thr}}]$. $d(u, v) = w(u, v) \in [W_{\text{thr}}, 2W_{\text{thr}})$. We want to show that $\exists a \in A$, such that $u, v \in D_a$. W.l.o.g., $d(A, u) \leq d(A, v)$. a is a vertex such that $d(a, u) = d(A, u)$. If there exists a shortest weighted path from a to u with one edge, then $u, v \in N_2(a) \subseteq D_a$.

Otherwise, any shortest weighted path from a to u has at least two edges. Let b be the vertex two hops away from a on one such path. $d(b, u) = d(a, u) - d(a, b) = d(A, u) - d(a, b) < d(A, u)$. By the triangle inequality,

$$
\begin{aligned}
& d(b, v) \leq d(b, u) + d(u, v) && (1)\\
& = d(A, u) - d(a, b) + d(u, v) && (2)\\
& \leq d(A, v) - d(a, b) + d(u, v) && (3)\\
& < d(A, v) - 2W_{\text{thr}} + 2W_{\text{thr}} && (4)\\
& = d(A, v) && (5)
\end{aligned}
$$

Therefore, $u, v \in C(b)$. Since $b \in N_2(a), u, v \in D_a$. □

We can show that RECONSTRUCT will discover edges with weights $w \in [W_{\text{thr}}, 2W_{\text{thr}})$ within a single connected component during one iteration of our main algorithm. When the threshold in the queries is W_{thr}, the algorithm might also discover edges with weight $w \geq 2W_{\text{thr}}$ but there is no guarantee.

Now we can show the correctness of LBL-R (Algorithm 1). During any iteration, for a connected component, if the EXHAUSTIVE-QUERY case is executed, we will discover its edges with weight $w \geq W_{\text{thr}}$, which is a superset of $[W_{\text{thr}}, 2W_{\text{thr}})$. Otherwise, we will discover its edges with weight $w \in [W_{\text{thr}}, 2W_{\text{thr}})$. The union of these intervals will cover all the possible weights when the outer loop finishes. If the break statement is triggered, EXHAUSTIVE-QUERY will discover all edges with weight $w \geq W_{\text{thr}}$. The correctness of LBL-R (Algorithm 1) thus follows.

Theorem 3 (Main correctness) *The output of LBL-R is all edges of G.*

Proof We claim that when starting iteration j, we have discovered all edges with weight $w < 2^j$. So even though the queries are restricted to $G[w \geq 2^j]$, we will not miss any edges because of it. If the algorithm breaks out of the main loop during iteration j, then EXHAUSTIVE-QUERY will be performed for all connected components, and it will discover all edges with weight $w \geq 2^j$, so together all edges will be found. If the break statement is never triggered, the outer loop has enough iterations to cover all the edges.

The proof of the claim is by induction. The base case is that when starting iteration 0, we have discovered all edges with weight $w < 2^0$. This is a vacuous truth. The inductive hypothesis is that when starting iteration j, we have discovered all edges with weight $w < 2^j$. We want to show that when starting iteration $j + 1$, we will find all edges with weight $w < 2^{j+1}$.

For any edge $e = uv$ with weight w such that $2^j = W_{\text{thr}} \leq w < 2W_{\text{thr}}$, we want to show this edge will be discovered during iteration j. Since $e \in G[w \geq 2^j]$, it will be in one of the connected components. EXHAUSTIVE-QUERY or RECONSTRUCT will find it by the lemma above. □

3.3 Query Complexity of LBL-R

As previously indicated, the central challenge of our analysis lies in eliminating the $\log_2 W_{\max}$ factor in LBL-R (Algorithm 1). The lemmas and proofs that accomplish this constitute the primary technical contribution of our work and distinguish us from the prior work by Kannan et al. (2018). We first consider the query complexity for one iteration of our main algorithm. The query complexity to find all the connected components is much smaller than the other steps since we have component queries at our disposal. Then, we compute the query complexity to process all connected components. Finally, we analyze when the break statement will be triggered.

Let us focus on the inner for loop in LBL-R (Algorithm 1). For some connected components, we will perform an exhaustive search, and for the rest, we will call RECONSTRUCT. The sizes of these connected components are carefully chosen so that for each group of components, the query complexity has a small upper bound.

Fact 4 Let $n_1, \ldots, n_m > 0, p > 1$. $\sum_{i=1}^{m} n_i^p \leq (\sum_{i=1}^{m} n_i)^p$.

Lemma 5 (Query complexity for all connected components) The inner for loop in LBL-R has expected query complexity $\tilde{O}(D^3 n^{3/2})$. The expectation is wrt the random choices in the RECONSTRUCT algorithm and this result holds for all input graphs.

Proof An exhaustive search will only be performed on connected components with at most $n^{1/4}$ vertices, requiring $O(n^{1/2})$ queries each. There are at most n connected components, resulting in $O(n^{3/2})$ queries. Suppose the rest of the connected components have $n_1, \ldots, n_l$ vertices. $\sum_{i=1}^{l} n_i \leq n$. Calling RECONSTRUCT on these connected components has query complexity

$$\sum_{i=1}^{l} \tilde{O}(D^3 n_i^{3/2}) = \tilde{O}(D^3 \sum_{i=1}^{l} n_i^{3/2}) \leq \tilde{O}(D^3 (\sum_{i=1}^{l} n_i)^{3/2}) \leq \tilde{O}(D^3 n^{3/2}) \tag{6}$$

Assuming $D \gg 1$, the overall expected query complexity is $\tilde{O}(D^3 n^{3/2})$. □

If we directly incorporate the outer loop, the overall query complexity will depend on $\lfloor \log_2 W_{max} \rfloor$. This is undesirable since W_{max} can be quite large. However, we can leverage our edge weight distribution to eliminate this term and significantly improve the query complexity. Recall that throughout this section, edge weights are sampled from a Pareto distribution. For a particular input graph, all its weights are fixed. The distribution specifies all the weight configurations that this graph may have. The following lemma shows that under our assumptions, the maximum weight is $\Omega(D^{\frac{1}{\alpha}})$ asymptotically almost surely (a.a.s.).[4] While a stronger result could be established, this lemma is sufficient for our subsequent analysis and effectively demonstrates that the maximum weight is large. Recall that $m/D = \omega(1)$, the weights are m i.i.d. samples, and $W_{\max}$ is their maximum. Since the Pareto distribution has a heavy tail, when we draw sufficiently many samples from it, the maximum will be arbitrarily larger than $D^{\frac{1}{\alpha}}$ a.a.s.

Lemma 6 (Maximum weight) Let $w^* = D^{\frac{1}{\alpha}} > 1$, then for any constant $c > 1$, $\lim_{n \to +\infty} \Pr(W_{\max} > cw^*) = 1$.

Proof

$$\Pr(W_{\max} > cw^*) = 1 - \Pr(W_{\max} \leq cw^*) = 1 - \left(F(cw^*)\right)^m \tag{7}$$

We want to show that $q \coloneqq (F(cw^*))^m$ tends to 0. Let $K = c^{-\alpha}$. Since $cw^* > 1$,

$$1 > 1 - F(cw^*) = (cw^*)^{-\alpha} = c^{-\alpha}(w^*)^{-\alpha} = K/D > 0 \tag{8}$$

$$q = (1 - K/D)^m = \left((1 - K/D)^D\right)^{m/D} \tag{9}$$

If D is a constant, then $(1 - K/D)^D$ is a constant in $(0, 1)$. If $D = \omega(1)$, then by the definition of the exponential function as a product limit, $(1 - K/D)^D \to \exp(-K)$ is also a constant in $(0, 1)$. In both cases, $\lim_{n \to +\infty} (1 - K/D)^D \in (0, 1)$. Since $m/D = \omega(1)$, $\left((1 - K/D)^D\right)^{m/D}$ approaches 0. Thus, q approaches 0. □

[4] a.a.s. means the statement holds true with probability tending to one as the number of vertices tends to infinity.

At any iteration j, the weight of an edge may or may not be below the threshold. For any edge, the probability that it is in $G[w \geq W_{\text{thr}}]$ is $p \coloneqq \Pr(w \geq W_{\text{thr}}) \coloneqq T(W_{\text{thr}})$. We notice the similarity between this and the binomial random graph $G(n,p)$ Janson et al. (2000). In $G(n,p)$, there are n vertices, and any vertex can connect to all other vertices with probability p. The size of connected components in $G(n,p)$ has been well studied. In our model, we can show an analogous result which allows us to analyze at which iteration the break statement will be triggered.

Lemma 7 (Part of Theorem 5.4 in Janson et al. (2000)) In a binomial random graph $G(n,p)$, if $np = c \in (0,1)$, then a.a.s. the largest connected component has $O(\log(n))$ vertices.

Lemma 8 (Largest connected component) In our model, if $Dp = c \in (0,1)$, then a.a.s. the largest connected component has $O(\log(n))$ vertices.

Proof The proof is similar to the proof of Theorem 5.4 in Janson et al. (2000). We reveal the component structure step by step, using the following procedure. Choose a vertex v, find all neighbors $v_1, \ldots, v_r$ of v in $G[w \geq W_{\text{thr}}]$ based on the randomly generated weights, and mark v as *saturated*. Then, generate all vertices $\{v_{11}, \ldots, v_{1s}\}$ from $N(v_1) \setminus \{v, v_1, \ldots, v_r\}$ which are adjacent to v_1 in $G[w \geq W_{\text{thr}}]$, so v_1 becomes saturated, and continue this process until all vertices in the component containing v are saturated. $N(v_1)$ is all neighbors of v_1 in the original input graph G.

The number X_i of new vertices we add to the component in the i-th step, provided m of its elements have already been found, has binomial distribution $\text{Bi}(n_i, p)$ where $n_i = |N(v_i) \setminus \{v, v_1, \ldots v_{m-1}\}|, p = \Pr(w \geq W_{\text{thr}}) = T(W_{\text{thr}})$.

The probability that a given vertex v belongs to a component of size at least $k = k(n)$ is bounded from above by the probability that the sum of $k = k(n)$ random variables X_i is at least $k-1$. Furthermore, the conditional distribution of X_i can be bounded from above by X_i^+, where all X_i^+ are independent random variables following binomial distribution $\text{Bi}(D,p)$ and the bound is in terms of stochastic domination. We have $\sum_{i=0}^{k-1} X_i^+ \sim \text{Bi}(kD,p)$. By additive Chernoff bounds, for n large enough the probability that there is a component of size at least $k \geq 3\log n/(1-c)^2$ is bounded from above by

$$n \Pr\left(\sum_{i=0}^{k-1} X_i^+ \geq k-1\right) \tag{10}$$

$$= n \Pr\left(\sum_{i=0}^{k-1} X_i^+ \geq ck + (1-c)k - 1\right) \tag{11}$$

$$\leq n \exp\left(-\frac{((1-c)k-1)^2}{2(ck + ((1-c)k-1)/3)}\right) \tag{12}$$

$$\leq n \exp\left(-\frac{(1-c)^2 k}{2}\right) \tag{13}$$

$$= o(1) \tag{14}$$

Taking the complement, the probability that all components have size $O(\log(n))$ is bounded from below by $1 - o(1)$. □

Now we can prove our main lemma in this section. The result suggests that in LBL-R (Algorithm 1) the outer loop is in fact superficial. We do not need to finish the outer loop to fully recover the underlying graph. This will allow us to get rid of the query complexity's dependency on the maximum weight and obtain an efficient algorithm. The proof outline is as follows: W_{thr} increases with each iteration, and since weights follow a Pareto distribution, $p = \Pr(w \geq W_{\text{thr}}) = T(W_{\text{thr}})$ will decrease. Once Dp falls below 1, we can apply the lemma above. Consequently, the break statement in LBL-R (Algorithm 1) will be triggered no later than this point and we can calculate when this occurs.

Lemma 9 (Early termination) Under the following assumptions:

1. $m/D = \omega(1)$.
2. Edge weights are i.i.d. random variables with CDF

$$F(w) = 1 - w^{-\alpha} \quad \text{for } w \in [1, \infty), \text{where } \alpha > 0 \text{ is a constant,}$$

and the maximum of these random variables is $W_{\max}$.

We have the following result: a.a.s. the break statement in LBL-R will terminate the outer loop at least i iterations early for any natural number i.

Proof We want to find the critical value W_{thr} such that $Dp = DT(W_{\text{thr}}) = 1$. The unique solution to this equation is $w^* = D^{\frac{1}{\alpha}}$. Since $D > 1$, we have $T(w^*) = 1/D \in (0, 1), w^* > 1$. Suppose k is the first iteration such that $W_{\text{thr}} = 2^k > w^*$. This is not possible at iteration 0 so $k > 0$. Early termination happens at or before iteration k a.a.s.

Without the break statement, the outer loop would terminate at iteration $j = \lfloor \log_2 W_{max} \rfloor$. Let $k' = \log_2 w^* = \frac{1}{\alpha} \log_2 D, j' = \log_2 W_{max}$. Since $k \leq k' + 1, j \geq j' - 1$, the number of iterations we skip is at least $j - k \geq j' - k' - 2$ a.a.s. Next, we study the distribution of $I \coloneqq j' - k' - 2$. $\forall i \in \mathbb{N}$, let $c = 2^{i+2} > 1$.

$$\Pr(I > i) = \Pr\left(2^I > 2^i\right) = \Pr\left(W_{\max}/4w^* > 2^i\right) = \Pr\left(W_{\max} > cw^*\right) \tag{15}$$

Now we can invoke Theorem 6 and complete the proof. □

Theorem 10 (Query complexity) *For weighted graph reconstruction using composite queries, there is a randomized algorithm with query complexity* $\tilde{O}\left(\left(1 + \frac{1}{\alpha} \log D\right) D^3 n^{3/2}\right)$.

Proof The query complexity to find connected components is dominated by the other steps of LBL-R (Algorithm 1). The inner for loop has expected query complexity $\tilde{O}(D^3 n^{3/2})$. The early termination happens no later than iteration k a.a.s. where $k \leq k' + 1, k' = \frac{1}{\alpha} \log_2 D$ as in the proof of Theorem 9. The outer for loop will be executed at most $k' + 1$ times a.a.s. Thus, the overall expected query complexity is $\tilde{O}\left(\left(1 + \frac{1}{\alpha} \log D\right) D^3 n^{3/2}\right)$ a.a.s. The expectation is wrt the random choices in the algorithm and the a.a.s. part is wrt the randomness of the edge weights. □

4 Conclusion

In this paper, we demonstrate that for weighted and potentially disconnected graphs with bounded degree whose weights follow a Pareto distribution, combining distance queries with edge-weight and component queries enables efficient graph reconstruction. An interesting direction for future work would be to establish lower bound results when only two of the three query types are allowed for interesting subclasses of graphs. For instance, if a graph has no transitive edges, can edge-weight queries be omitted? Similarly, is it possible to reconstruct subclasses of graphs without component queries? As previously noted, $\Omega(n^2)$ distance queries are necessary to identify all connected components in a general graph. However, stronger assumptions on the graph structure and edge weights might lead to better lower bounds.

Acknowledgements. Michael T. Goodrich was supported by NSF grant 2212129. Ioannis Panageas was supported by NSF grant CCF-2454115.

Supplementary information. Not applicable.

Declarations

Not applicable.

References

Abrahao B, Chierichetti F, Kleinberg R, et al (2013) Trace complexity of network inference. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, Chicago Illinois USA, KDD '13, pp 491–499, https://doi.org/10.1145/2487575.2487664

Afshar R, Goodrich MT (2022) Exact learning of multitrees and almost-trees using path queries. Latin American Symposium on Theoretical Informatics pp 293–311. https://doi.org/10.48550/arXiv.2208.04216, arXiv:2208.04216 [cs]

Afshar R, Goodrich MT, Matias P, et al (2020) Reconstructing Biological and Digital Phylogenetic Trees in Parallel. In: Grandoni F, Herman G, Sanders P (eds) 28th Annual European Symposium on Algorithms (ESA 2020), Leibniz International Proceedings in Informatics (LIPIcs), vol 173. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp 3:1–3:24, https://doi.org/10.4230/LIPIcs.ESA.2020.3

Afshar R, Goodrich MT, Matias P, et al (2022a) Mapping Networks via Parallel kth-Hop Traceroute Queries. In: Berenbrink P, Monmege B (eds) 39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022), Leibniz International Proceedings in Informatics (LIPIcs), vol 219. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp 4:1–4:21, https://doi.org/10.4230/LIPIcs.STACS.2022.4

Afshar R, Goodrich MT, Ozel E (2022b) Efficient Exact Learning Algorithms for Road Networks and Other Graphs with Bounded Clustering Degrees. In: Schulz C, Uçar B (eds) 20th International Symposium on Experimental Algorithms (SEA 2022), Leibniz International Proceedings in Informatics (LIPIcs), vol 233. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp 9:1–9:18, https://doi.org/10.4230/LIPIcs.SEA.2022.9

Arunachaleswaran ER, De A, Kannan S (2023) Reconstructing ultrametric trees from noisy experiments. In: Proceedings of the 34th International Conference on Algorithmic Learning Theory. PMLR, pp 82–114

Assadi S, Chakrabarty D, Khanna S (2021) Graph Connectivity and Single Element Recovery via Linear and OR Queries. In: Mutzel P, Pagh R, Herman G (eds) 29th Annual European Symposium on Algorithms (ESA 2021), Leibniz International Proceedings in Informatics (LIPIcs), vol 204. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp 7:1–7:19, https://doi.org/10.4230/LIPIcs.ESA.2021.7

Backstrom L, Kleinberg J (2014) Romantic partnerships and the dispersion of social ties: a network analysis of relationship status on facebook. In: 17th ACM Conference on Computer Supported Cooperative Work & Social Computing, CSCW, pp 831—-841, https://doi.org/10.1145/2531602.2531642, URL https://doi.org/10.1145/2531602.2531642

Bastide P, Groenland C (2025) Tight Distance Query Reconstruction for Trees and Graphs Without Long Induced Cycles. Random Structures & Algorithms 66(4):e70017. https://doi.org/10.1002/rsa.70017

Bernstein A, Bodwin G, Wein N (2024) Are There Graphs Whose Shortest Path Structure Requires Large Edge Weights? In: Guruswami V (ed) 15th Innovations in Theoretical Computer Science Conference (ITCS 2024), Leibniz International Proceedings in Informatics (LIPIcs), vol 287. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp 12:1–12:22, https://doi.org/10.4230/LIPIcs.ITCS.2024.12

Bu F, Kang S, Shin K (2023) Interplay between topology and edge weights in real-world graphs: Concepts, patterns, and an algorithm. Data Mining and Knowledge Discovery 37(6):2139–2191. https://doi.org/10.1007/s10618-023-00940-w

Chen C, Hwang F (1989) Detecting and locating electrical shorts using group testing. IEEE Transactions on Circuits and Systems 36(8):1113–1116. https://doi.org/10.1109/31.192423

Choi SS (2013) Polynomial time optimal query algorithms for finding graphs with arbitrary real weights. In: Proceedings of the 26th Annual Conference on Learning Theory. PMLR, pp 797–818

Culberson JC, Rudnicki P (1989) A fast algorithm for constructing trees from distance matrices. Information Processing Letters 30(4):215–220. https://doi.org/10.1016/0020-0190(89)90216-0

Eppstein D, Goodrich MT, Liu S (2025) Bandwidth vs BFS Width in Matrix Reordering, Graph Reconstruction, and Graph Drawing. In: Benoit A, Kaplan H, Wild S, et al (eds) 33rd Annual European Symposium on Algorithms (ESA 2025), Leibniz International Proceedings in Informatics (LIPIcs), vol 351. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp 69:1–69:17, https://doi.org/10.4230/LIPIcs.ESA.2025.69

Fang SC (1990) Detecting electrical shorts on printed circuit boards. International Journal of Production Research 28(6):1031–1037. https://doi.org/10.1080/00207549008942773

Foucquier J, Guedj M (2015) Analysis of drug combinations: Current methodological landscape. Pharmacology Research & Perspectives 3(3):e00149. https://doi.org/https://doi.org/10.1002/prp2.149, URL https://bpspubs.onlinelibrary.wiley.com/doi/abs/10.1002/prp2.149, https://bpspubs.onlinelibrary.wiley.com/doi/pdf/10.1002/prp2.149

Jagadish M, Sen A (2013) Learning a bounded-degree tree using separator queries. In: Jain S, Munos R, Stephan F, et al (eds) Algorithmic Learning Theory. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 188–202

Janson S, Łuczak T, Ruciński A (2000) Random Graphs. Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & sons, Inc, New York Chichester Weinheim

Kannan S, Mathieu C, Zhou H (2018) Graph reconstruction and verification. ACM Transactions on Algorithms 14(4):1–30. https://doi.org/10.1145/3199606

Kim JH (2012) Finding weighted graphs by combinatorial search. Arxivorg abs/1201.3793. arXiv:1201.3793 [cs, math]

Krivelevich M, Zhukovskii M (2025) Reconstructing Random Graphs from Distance Queries. In: Benoit A, Kaplan H, Wild S, et al (eds) 33rd Annual European Symposium on Algorithms (ESA 2025), Leibniz International Proceedings in Informatics (LIPIcs), vol 351. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp 30:1–30:17, https://doi.org/10.4230/LIPIcs.ESA.2025.30

Kumar R, Liu P, Charikar M, et al (2020) Retrieving top weighted triangles in graphs. In: Proceedings of the 13th International Conference on Web Search and Data Mining. ACM, Houston TX USA, WSDM '20, pp 295–303, https://doi.org/10.1145/3336191.3371823

Liu X, Mukherjee S (2022) Tight query complexity bounds for learning graph partitions. In: Proceedings of Thirty Fifth Conference on Learning Theory. PMLR, pp 167–181

Mathieu C, Zhou H (2021) A Simple Algorithm for Graph Reconstruction. In: Mutzel P, Pagh R, Herman G (eds) 29th Annual European Symposium on Algorithms (ESA 2021), Leibniz International Proceedings in Informatics (LIPIcs), vol 204. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp 68:1–68:18, https://doi.org/10.4230/LIPIcs.ESA.2021.68

Mazzawi H (2010) Optimally reconstructing weighted graphs using queries: (extended abstract). In: Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, Proceedings, pp 608–615, https://doi.org/10.1137/1.9781611973075.51

Shi W, West DB (1999) Diagnosis of wiring networks: An optimal randomized algorithm for finding connected components of unknown graphs. SIAM Journal on Computing 28(5):1541–1551. https://doi.org/10.1137/S0097539795288118

Shi W, West DB (2001) Structural diagnosis of wiring networks: Finding connected components of unknown subgraphs. SIAM Journal on Discrete Mathematics 14(4):510–523. https://doi.org/10.1137/S0895480100371286

Stadtfeld C, Pentland AS (2015) Partnership Ties Shape Friendship Networks: A Dynamic Social Network Study. Social Forces 94(1):453–477. https://doi.org/10.1093/sf/sov079, URL https://doi.org/10.1093/sf/sov079, https://academic.oup.com/sf/article-pdf/94/1/453/6873143/sov079.pdf

Stegehuis C, Weedage L (2024) Reconstruction of geometric random graphs with the simple algorithm. Arxivorg abs/2407.18591. https://doi.org/10.48550/arXiv.2407.18591, arXiv:2407.18591 [cs, math]

Wang Z, Honorio J (2017) Reconstructing a bounded-degree directed tree using path queries. URL https://arxiv.org/abs/1606.05183, arXiv:1606.05183

Yang SH, Smola AJ, Long B, et al (2012) Friend or frenemy? predicting signed ties in social networks. In: 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR, pp 555—-564, https://doi.org/10.1145/2348283.2348359, URL https://doi.org/10.1145/2348283.2348359

Appendix A Queries without a Threshold

In this section, we will show that it is difficult to efficiently reconstruct a weighted graph without applying different weight thresholds in the queries. We design a no-threshold reconstruction (NT-R) algorithm that can reconstruct connected weighted graphs using distance and edge-weight queries. For all queries, we fix $W_{\text{thr}} = 1$ and always work with the original input graph so there is effectively no threshold. We will then show the correctness of our algorithm and establish that its query complexity is $\tilde{O}(D^{3W_{\max}} n^{3/2})$. If we compare this and the $\tilde{O}\left(\left(1 + \frac{1}{\alpha}\log D\right) D^3 n^{3/2}\right)$ result we showed, we can see that the query complexity now has an exponent that depends on $W_{\max}$, and by Theorem 6, this will significantly increase the complexity.

A.1 Description of NT-R

First, we describe our algorithm. NT-R is RECONSTRUCT (Algorithm 6) modified. It takes as input the set V, which is all the vertices in the input graph, and $W_{\text{thr}} = 1$, and recovers all the edges of the graph. The changes are as follows. In RECONSTRUCT, set $Q = \tilde{O}(D^{3W_{\max}} n^{3/2})$. In its subroutine RECONSTRUCT-SUB (Algorithm 5), we set s to a different value and the exact formula is in the appendix. In addition, we replace $N_2(a) \leftarrow \text{FIND-NEIGHBORS}(V, a, W_{\text{thr}})$ by $N_2(a) \leftarrow \bar{B}(a, 2W_{\max}) = \{v \in V : d(a, v) \leq 2W_{\max}\}$, i.e., the closed ball centered at a with radius $2W_{\max}$ using metric d. We call this modified subroutine of NT-R NT-RS. Note that $\bar{B}(a, 2W_{\max})$ can be directly calculated because when the algorithm needs it, every pair in $A \times V$ has been queried. $\forall a \in A, \forall v \in V, d(a, v)$ is available.

A.2 Correctness of NT-R

The correctness is a result of the changes we made and the intuition is as follows. $\bar{B}(a, 2W_{\max})$ includes all the vertices two hops away from a and potentially much more. When we compute the extended Voronoi cell D_a, it will take the union of $C(b)$ over more vertices b. Since each cell is larger, we will discover more edges. More precisely, we have the following lemma.

Lemma 11 (Correctness) For NT-RS, $G \subseteq \bigcup_{a \in A} G[D_a]$ where $G[D_a]$ is the subgraph of G induced by vertices in D_a.

A.3 Query Complexity of NT-R

Without the weight thresholds, the query complexity will depend on $W_{\max}$ and this term is large. If we assume a specific distribution for the weights, we can further analyze the distribution of $W_{\max}$ and how it affects the query complexity. In general, we have the following:

Theorem 12 (Query complexity) *For connected weighted graph reconstruction using distance and edge-weight queries, there is a randomized algorithm with query complexity* $\tilde{O}\left(n^{3/2} D^{3W_{max}}\right)$.

Appendix B Deferred Proofs from Section A

B.1 Correctness

Lemma 13 (Theorem 11 restated) For NT-RS, $G \subseteq \bigcup_{a \in A} G[D_a]$ where $G[D_a]$ is the subgraph of G induced by vertices in D_a.

Proof $\forall uv = e \in E$, we want to show that $\exists a \in A$, such that $u, v \in D_a$. W.l.o.g., $d(A,u) \leq d(A,v)$. a is a vertex such that $d(a,u) = d(A,u)$. If $d(a,u) \leq W_{\max}$, then $d(a,v) \leq d(a,u) + d(u,v) \leq 2W_{\max} \Rightarrow v \in \bar{B}(a, 2W_{\max})$.

Otherwise, $d(a,u) > W_{\max}$. We claim that on any shortest weighted path from a to u, there is a vertex b such that $d(a,b) \in (W_{\max}, 2W_{\max}]$. It is possible that $b = u$. $d(b,u) = d(a,u) - d(a,b) = d(A,u) - d(a,b) < d(A,u)$. By the triangle inequality,

$$
\begin{aligned}
& d(b,v) \leq d(b,u) + d(u,v) && \text{(B1)} \\
& \leq d(b,u) + W_{\max} && \text{(B2)} \\
& = d(A,u) - d(a,b) + W_{\max} && \text{(B3)} \\
& \leq d(A,v) - d(a,b) + W_{\max} && \text{(B4)} \\
& < d(A,v) && \text{(B5)}
\end{aligned}
$$

Therefore, $u, v \in C(b)$. Since $b \in \bar{B}(a, 2W_{\max}), u, v \in D_a$.

To prove our claim, imagine arranging such a path on an x-axis so that a is at the origin and all edges are on its right. Check the vertices on this path from left to right. Suppose b is the first vertex such that $d(a,b) > W_{\max}$. We want to show that b also satisfies the requirement $d(a,b) \leq 2W_{\max}$ and is the vertex we want. If not, $d(a,b) > 2W_{\max}$. Suppose b^- is the vertex on the left of b. Since b is the first vertex such that $d(a,b) > W_{\max}$, $d(a,b^-) \leq W_{\max}$. $d(b^-,b) = d(a,b) - d(a,b^-) > W_{\max}$. This contracts the maximum weight.

□

B.2 Query Complexity

To prove the query complexity of NT-R, we first recall the following lemma in Kannan et al. (2018).

Lemma 14 (Rephrasing of Lemma 2.3 in Kannan et al. (2018)) Let k in ESTIMATED-CENTERS be some well-chosen constant. With probability at least $1/4$, the algorithm ESTIMATED-CENTERS (Algorithm 4) terminates after $O\left(sn \log^2 n \log\log n\right)$ queries, and outputs a set $A \subseteq C$, such that $|A| \leq 12s \log n$ and $|C(w)| \leq 6n/s$ for every $w \in V$ where $C(w) = \{v \in V : d(w,v) < d(A,v)\}$.

Theorem 15 (Theorem 12 restated) *For connected weighted graph reconstruction using distance and edge-weight queries, there is a randomized algorithm with query complexity* $\tilde{O}\left(n^{3/2} D^{3W_{max}}\right)$.

Proof We first analyze the query complexity of NT-RS. $\forall v \in \bar{B}(a, 2W_{\max})$, $d(a, v) \leq 2W_{\max}$. Since edge weights are at least 1, the number of edges on a shortest weighted path from a to v is at most $2W_{\max}$. Since the graph has maximum degree D,

$$\left|\bar{B}(a, 2W_{\max})\right| \leq \sum_{i=0}^{2W_{\max}} D^i = \frac{D^{2W_{\max}+1} - 1}{D - 1} \coloneqq b \tag{B6}$$

By the lemma above, with probability at least 1/4, $|D_a| \leq b + \frac{6n}{s}b$ and the nested for loops in Algorithm 5 has query complexity

$$\sum_{a \in A} (\left|\bar{B}(a, 2W_{\max})\right| n + |D_a|^2) \leq 12s \log n \left(bn + \left(b + \frac{6n}{s} b \right)^2 \right) \tag{B7}$$

ESTIMATED-CENTERS has query complexity $O(sn \log^2 n \log\log n)$. When $s = \sqrt{bn}$, the overall complexity is $O\left(n^{3/2} b^{1/2} \log n \left(37b + \log n \log\log n\right)\right)$ with probability at least 1/4. Assuming $D \gg 1$, we can further simplify $b = D^{2W_{\max}}$. By the expectation of a geometric random variable, the expected query complexity of Algorithm 6 is $\tilde{O}\left(n^{3/2} b^{3/2}\right) = \tilde{O}\left(n^{3/2} D^{3W_{\max}}\right)$. The expectation is wrt the randomized algorithm. □