Tight query complexity bounds for learning graph partitions

Xizhi Liu XIZHI.LIU@WARWICK.AC.UK
Department of Mathematics, Statistics and Computer Science, University of Illinois at Chicago, USA.
Mathematics Institute and DIMAP, University of Warwick, Coventry, CV4 7AL, UK.

Sayan Mukherjee SAYAN@BLUEQAT.COM
Blueqat Research, Shibuya, Tokyo, Japan.

Editors: Po-Ling Loh and Maxim Raginsky

Abstract

Given a partition of a graph into connected components, the membership oracle asserts whether any two vertices of the graph lie in the same component or not. We prove that for $n \geq k \geq 2$, learning the components of an n-vertex hidden graph with k components requires at least $(k-1)n - \binom{k}{2}$ membership queries. Our result improves on the best known information-theoretic bound of $\Omega(n \log k)$ queries, and exactly matches the query complexity of the algorithm introduced by Reyzin and Srivastava (2007) for this problem. Additionally, we introduce an oracle, with access to which one can learn the number of components of G in asymptotically fewer queries than learning the full partition, thus answering another question posed by the same authors. Lastly, we introduce a more applicable version of this oracle, and prove asymptotically tight bounds of $\widetilde{\Theta}(m)$ queries for both learning and verifying an m-edge hidden graph G using it.

Keywords: graph learning, graph reconstruction, query complexity

1. Introduction

1.1. Background and Applications

A graph $G = (V, E)$ consists of a vertex set V and an edge set $E \subseteq \binom{V}{2}$. The field of graph learning deals with learning a hidden graph using queries to black-box oracles that reveal partial information about the graph. In several real world scenarios, learning a full graph by checking only pairwise adjacency is inefficient, and several oracles can speed up the process of learning by encoding more information per query. Different oracles could be useful or easier to implement in different scenarios. For example, in the context of trying to learn a hidden network graph, a *traceroute* query between a pair of vertices can give information about a shortest path between the vertices and their distance in the graph. In the context of bioinformatics, one can model a graph with vertices corresponding to chemicals, and two vertices are joined by an edge if they react when mixed together. In such a situation, oracles such as edge-detection and edge-counting can be implemented by mixing different sets of chemicals and measuring the intensity of reaction. As calling an oracle incurs cost, researchers try to estimate the *query complexity*: the least number of queries to the oracle required to learn a specific graph or graph property.

A separate type of problem that is also widely studied is the problem of graph verification. In this setting, we have a hidden graph $G = (V, E)$ and a known graph $\widehat{G} = (V, \widehat{E})$, and an oracle that reveals information about G. The main task in this area is to verify whether $G = \widehat{G}$ using as few oracle queries as possible. In the context of our paper, we assume $|\widehat{E}| = |E|$, as G and $\widehat{G}$ are trivially unequal if they do not have the same number of edges. Verification tasks are prevalent in real networks where it is important to make sure a recent snapshot of a network is accurate.

Due to their practical and theoretical importance, both graph learning and verification have garnered a lot of interest in recent years. Perhaps the first problem considered in the literature was the problem of learning a degree-bounded tree using the shortest path oracle (Hein, 1989; King et al., 2003). The shortest path oracle and the distance oracle were extensively studied in (Reyzin and Srivastava, 2007; Kannan et al., 2018; Rong et al., 2021). The current best upper bound on learning a connected bounded-degree graph on n vertices is due to Mathieu and Zhou (2013), where they provide a randomized $\widetilde{O}(n^{3/2})$-query algorithm to learn such a graph. Abrahamsen et al. (2016) prove the exact bound on the query complexity using a weaker oracle called the *betweenness oracle*, thus suggesting that the bound of $\widetilde{O}(n^{3/2})$ might be not tight. Parallel results in graph verification have also been obtained for the distance oracle (Kannan et al., 2015) and betweenness oracle (Janardhanan, 2017). Although there is still a gap between the lower bound of $\Omega(n)$ and upper bound of $\widetilde{O}(n^{3/2})$ for learning degree-bounded graphs using distance oracle, the gap was closed recently for random degree-bounded regular graphs by Mathieu and Zhou (2021).

Another well-studied oracle is the *edge-detection oracle*, which, given a set of vertices of the hidden graph G, tells if it is an independent set or not. Results on learning restricted classes of graphs using this oracle such as matchings (Alon et al., 2004), stars and cliques [Alon and Asodi (2005)], Hamiltonian cycles (Grebinski and Kucherov, 2000) were succeeded by a very general treatment by Angluin and Chen (2008). Using a recursive coloring argument, they show that $O(m \log n)$ edge-detection queries are sufficient to learn an arbitrary hidden graph. Reyzin and Srivastava (2007) consider and compare the shortest path, edge-detection and edge-counting queries and prove a variety of lower and upper bounds for learning partitions, trees and arbitrary graphs. In other related work, Beerliova et al. (2006) consider an oracle called the layered-graph oracle.

1.2. Our Results

1.2.1. Membership queries

Every graph G admits a partition of its vertex set into connected components. We say that a learner *learns the components of* G if it learns this partition. For an n-vertex hidden graph G with k components, one of the problems studied in (Reyzin and Srivastava, 2007) entails learning the components of G. More precisely, if α denotes the membership query given by

$$\alpha(u,v) = \begin{cases} 1, & \text{if } u \text{ and } v \text{ belong to the same component,} \\ 0, & \text{otherwise;} \end{cases}$$

then they demonstrate an algorithm to learn the components of G.

Algorithm 1.1 (Reyzin and Srivastava (2007))

- *Start with a set S of already classified vertices and a set C of component representatives which are both initialized to $\{v_0\}$ for some arbitrary $v_0 \in V(G)$.*
- *For every unclassified vertex $v \in V(G) \setminus S$, sequentially query $\{\alpha(x,v) : x \in C\}$.*
- *If some query $\alpha(x,v)$ is true, add v to S. Otherwise, the component of v has not been discovered till now, so add v to both S and C.*
- *Repeat the procedure until $S = V(G)$.*

It can be seen that when k is small compared to n, Algorithm 1.1 uses $O(nk)$ many α-queries. In particular, if we consider a graph G which is the union of isolated vertices $\{1, \dots, k-1\}$ and a clique $K_{\{k,\dots,n\}}$, Proposition 2.1 demonstrates that Algorithm 1.1 uses $(k-1)n - \binom{k}{2}$ queries in the worst case.

However, the best known lower bound for this fundamental problem of learning components with α was an information-theoretic bound of $\Omega(n \log k)$ queries. To the knowledge of the authors, no proof of an $\Omega(nk)$ lower bound has been published till date since it was first posed by Reyzin and Srivastava (2007). Our main theorem is the first exact lower bound of $(k-1)n - \binom{k}{2}$ queries to this problem, when k is known to the algorithm.

Theorem 1.2 *Given any algorithm $\mathcal{A}$ that makes membership queries on a hidden graph G with n vertices and k components, there is an adversary that can force $\mathcal{A}$ to make at least $(k-1)n - \binom{k}{2}$ queries to learn the partition of G.*

We also prove an exact bound if the number of connected components in G is unknown.

Theorem 1.3 *Given any algorithm $\mathcal{A}$ that makes membership queries on a hidden graph G with n vertices, there is an adversary that can force $\mathcal{A}$ to make at least $kn - \binom{k+1}{2}$ queries to learn the partition of G, where k is the number of connected components in G.*

Remark 1.4 *When the number of components k of a hidden graph G is known to an algorithm, intuitively it should require fewer queries to learn all components of G. This is also reflected in the fact that our lower bound in Theorem 1.2 is $(n-k)$ lower than the lower bound in Theorem 1.3. According to Proposition 2.1, the bounds from both Theorems 1.2 and 1.3 are tight.*

We make a note here that Theorems 1.2 and 1.3, as well as the membership oracle is applicable to a much more general discrete setting: *learning partitions of any finite n-element set into k parts*. However, the problem of learning partitions is not well-represented in the literature and hence we use the more familiar and equivalent language of graph learning for presenting Theorems 1.2 and 1.3.

1.2.2. MULTIPLE-MEMBERSHIP QUERIES

Reyzin and Srivastava (2007) also posed a question on whether there is an oracle that has to be queried fewer times to learn the number of components in an n-vertex hidden graph than learning the components. For a vertex u and a set S of vertices not containing u, we define the multiple-membership query

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

Our second result gives a positive answer to their question:

Theorem 1.5 *For an n-vertex hidden graph G, learning the number of components of G can be done using $O(n)$ α_m-queries. However, learning all the components requires $\Theta(n \log k)$ α_m-queries.*

1.2.3. VERTEX-NEIGHBORHOOD DETECTION QUERIES

We modify α_m into a query that is more applicable in practical scenarios, which we call the vertex-neighborhood detection query β given by:

$$\beta(u, S) = \begin{cases} 1, & \text{if } u \text{ and } v \text{ are adjacent for some } v \in S, \\ 0, & \text{otherwise.} \end{cases}$$

β is useful in the setting of biochemistry where it is easy to detect a reaction between a fixed reagent and a set of other chemicals, and could be applicable to genome sequencing using polymerase chain reaction (PCR) tests [Bouvel et al. (2005); Chang et al. (2011)].

We analyze the problems of graph learning and graph verification using β-queries, and prove tight bounds for these problems (up to logarithmic factors):

Theorem 1.6 *Learning or verifying an n-vertex hidden graph on m edges requires $\Omega(m)$ β-queries. Conversely, learning such a graph can be done in $O(m \log n)$ queries, whereas verifying can be done in $O(m + n)$ queries.*

Remark 1.7 *As an immediate consequence to Theorem 1.6, note that all sparse families of graph which satisfy $m = O(n)$, such as trees, planar graphs and minor-free graphs, can be both learned and verified using $\widetilde{\Theta}(n)$ queries to β.*

This paper is organized as follows. In Section 2, we prove Theorems 1.2, 1.3 and 1.5. Section 3 presents our results on the β oracle, and proves Theorem 1.6. Finally, we make some concluding remarks in Section 4.

2. Graph partitions and the membership query

In this section, we consider the membership query given by $\alpha(u, v) = 1$ iff u and v belong to the same connected component. Our goal in this section is to prove Theorem 1.2.

2.1. Preliminaries

We briefly state some preliminary results before diving into the proof.

2.1.1. UPPER BOUND ON QUERY COMPLEXITY OF ALGORITHM 1.1

First, we give a quick demonstration of the upper bound of $(k-1)n - \binom{k}{2}$ of the query complexity of Algorithm 1.1.

Proposition 2.1 *Algorithm 1.1 uses at most $(k-1)n - \binom{k}{2}$ queries to learn the graph G consisting of isolated vertices $\{1, \ldots, k-1\}$ and a clique $K_{\{k,\ldots,n\}}$. If k is not known to the algorithm, it requires $(n-k)$ additional queries.*

Proof Suppose k is known. Without loss of generality assume that the algorithm determines the components of the $k-1$ isolated vertices using $1 + \cdots + (k-2) = \frac{1}{2}(k-2)(k-1)$ many membership queries first. Otherwise, if the large component is discovered earlier, our algorithm can learn G using fewer queries. To classify the remaining $(n-k+1)$ vertices into the k'th component, we only need to make sure that each of them does not share a component with $\{1, \ldots, k-1\}$. This requires $(k-1)(n-k+1)$ membership queries. Hence, the total number of α-queries required in this case equals

$$\frac{1}{2}(k-1)(k-2)+(k-1)(n-k+1)=(k-1)\left(n-\frac{k}{2}\right)=(k-1)n-\binom{k}{2},$$

as desired.

When k is not known, the last $(n-k+1)$ vertices each require not only $(k-1)$ queries each, but the vertices $\{k+1,\dots,n\}$ each need one additional query. The total number of queries made therefore, is $(n-k)$ more, proving the second part of our claim. ■

2.1.2. UNIQUELY k-COLORABLE GRAPHS

Our proof will be closely related to the following definition in graph theory.

Given a graph G, we say that it is k-colorable if there is a labelling $\chi : V(G) \to \{1,\dots,k\}$ such that the two endpoints of any edge receives distinct labels. Such a χ is also called a proper k-coloring. G is said to be *uniquely* k-colorable if it has a unique proper k-coloring χ (up to permutations of $\{1,\dots,k\}$). The following theorem gives a lower bound for the number of edges in a uniquely k-colorable graph on n vertices.

Theorem 2.2 (Truszczyński (1984); Shaoji (1990)) *A uniquely k-colorable graph on n vertices must have at least $(k-1)n-\binom{k}{2}$ edges.*

For the sake of completeness, we give an outline of the proof. We urge the reader to refer to the cited sources for further information on uniquely k-colorable graphs.

Proof (Sketch). Suppose G is a uniquely k-colorable graph on n vertices, and its color classes are $I_1,\dots,I_k$. Each I_i is an independent set, and the edges of G go across these color classes. Let us fix any pair $i \neq j$, and consider the bipartite graph $G[I_i \cup I_j]$. If $G[I_i \cup I_j]$ was disconnected, we could switch around the colors of these connected components to obtain a new proper k-coloring of G, a contradiction. Thus, $G[I_i \cup I_j]$ is a connected bipartite graph, implying there are at least $|I_i|+|I_j|-1$ edges between components I_i and I_j. It can then be seen that

$$\begin{aligned}|E(G)| &\geq \sum_{i<j} |I_i|+|I_j|-1\\ &= (k-1)(|I_1|+\cdots+|I_k|)-\binom{k}{2}\\ &= (k-1)n-\binom{k}{2}.\end{aligned}$$

■

We are ready to present the proofs of Theorems 1.2 and 1.3 in the following two sections.

2.2. Membership query

First we present our proof of Theorem 1.2. Recall that $\alpha(u,v)=1$ iff u and v belong to the same component in an n-vertex hidden graph with k components, and we are aiming to prove a lower bound of $(k-1)n-\binom{k}{2}$ queries on learning all its components.

2.2.1. PROOF OF THEOREM 1.2.

We will use the following definition in our proofs.

Definition 2.3 *Suppose that G is a k-colorable graph and $i, j \in V(G)$ are distinct vertices that are not adjacent in G. Then $\{i, j\}$ is called a k-separable pair if there exists a proper k-coloring $\chi\colon V(G) \to [k]$ of G such that $\chi(i) \neq \chi(j)$, and we call such a coloring χ an $\{i, j\}$-separating coloring. Otherwise, we call $\{i, j\}$ a k-inseparable pair.*

For example, the nonadjacent pair in K_{k+1}^- (the graph obtained from a $(k+1)$-clique by removing one edge) is k-inseparable. The following fact is easy to observe from the definition.

Fact 2.4 *Suppose that G is a k-colorable graph and $i, j \in V(G)$ are distinct vertices that are not adjacent in G. Then $\{i, j\}$ is a k-inseparable pair if and only if the graph G' obtained from G by adding the edge $\{i, j\}$ has chromatic number $k + 1$.*

Now we present our proof of Theorem 1.2. As mentioned earlier, our proof is adversarial. We (the adversary) start with initializing an empty auxiliary (simple) graph H with $|V(H)| = n$, and pick an arbitrary k-coloring $\chi : V(H) \to \{1, \dots, k\}$ of H. Let G be the hidden graph corresponding to H, where each color class corresponds to a partition.

Suppose now that $\mathcal{A}$ makes a query $\alpha(x, y)$. We respond to $\mathcal{A}$ and update the graph H and its k-coloring χ according to the following rules:

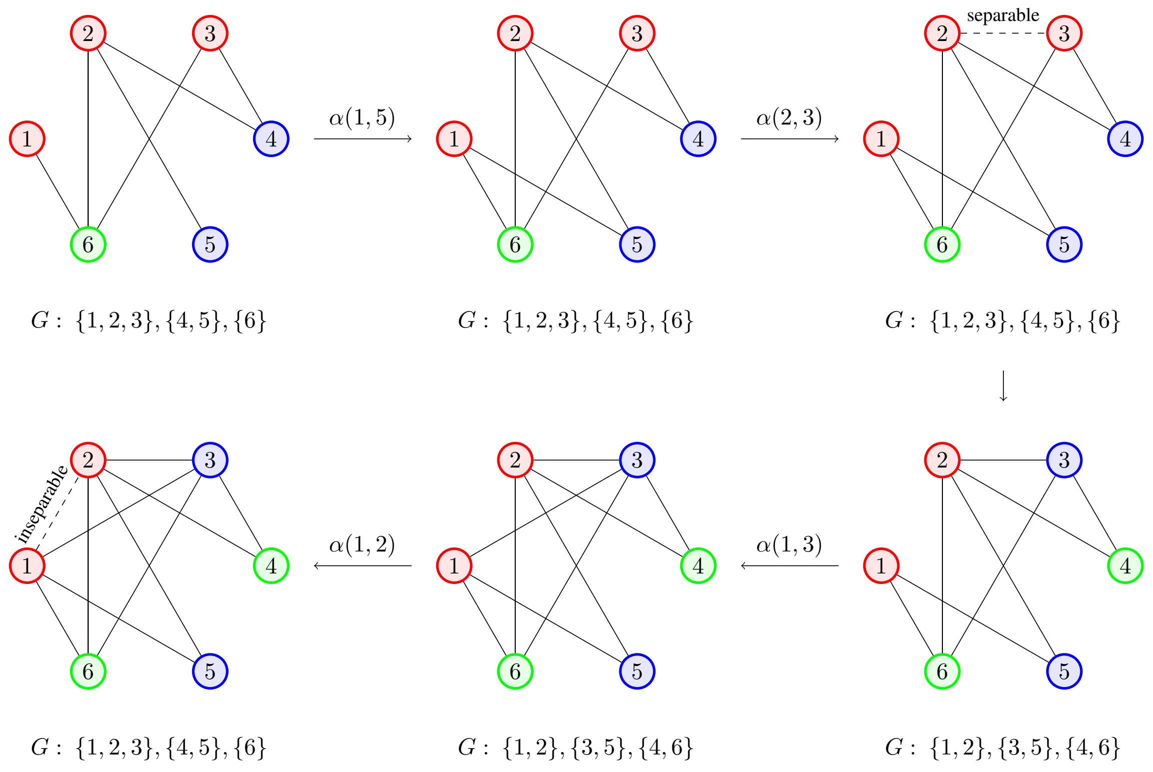
- If $\chi(x) \neq \chi(y)$, we add xy to $E(H)$ and reply "no" to the algorithm.
- If $\chi(x) = \chi(y)$ and $\{x, y\}$ is k-separable in H, we add xy to $E(H)$ and modify the coloring of G to a $\{x, y\}$-separating k-coloring, and reply "no".
- If $\chi(x) = \chi(y)$ and $\{x, y\}$ is k-inseparable, then we do not add xy to $E(H)$, and answer "yes".

We illustrate some intermediate steps in the evolution of H and χ against a sample algorithm $\mathcal{A}$ in Figure 2.1.

Observe that adding new edges into H does not change the inseparability of a pair $\{x, y\}$. Therefore, all k-inseparable pairs $\{x, y\}$ of H remain k-inseparable in the evolution of the algorithm. Further, by construction, every edge of H corresponds to a non-coincidence in G's components. Vertex pairs $\{x, y\}$ of H that have $\alpha(x, y) = 1$ are k-inseparable, and hence belong to the same color class for any proper k-coloring of H. Further, χ is always a proper k-coloring of H, and hence induces a partition of G into k components.

Now, suppose that $\mathcal{A}$ learns a unique partition of G into k components. This would mean that the coloring χ of the graph H must be a unique k-coloring. By Theorem 2.2, H has at least $(k-1)n - \binom{k}{2}$ edges, as desired. □

Remark 2.5 *We note that our adversary checks over all possible k-colorings of H, and hence does not have a poly(n, k) runtime. However, a slight modification to the argument above (given in Appendix A) is able to prove the existence of a poly(n, k)-time adversary that forces at least $\frac{1}{2}(n-k)(k-1)$ queries. It might be interesting to analyze the effect of limiting resources available to the adversary to prove different versions of this theorem.*



$\alpha(1,5)$
$\alpha(2,3)$
separable
$\alpha(1,3)$
$\alpha(1,2)$
inseparable
$G:\ \{1,2,3\},\{4,5\},\{6\}$
$G:\ \{1,2,3\},\{4,5\},\{6\}$
$G:\ \{1,2,3\},\{4,5\},\{6\}$
$G:\ \{1,2,3\},\{4,5\},\{6\}$
$G:\ \{1,2\},\{3,5\},\{4,6\}$
$G:\ \{1,2\},\{3,5\},\{4,6\}$

Figure 2.1: A sample set of queries and the corresponding evolution of H and the partition of G. Here $n = 6$ and $k = 3$. Our adversary responds "yes" only to $\alpha(1,2)$, and "no" to all other α-queries made by $\mathcal{A}$ in this example.

2.2.2. PROOF OF THEOREM 1.3

Now we prove Theorem 1.3. Our proof is basically a modification of the proof of Theorem 1.2: instead of a single auxiliary graph H with $|V(H)| = n$, now we will maintain two graphs H_1 and H_2 on the same vertex set V of size n. We start with the coloring $\chi\colon V \to \{1\}$, i.e. all vertices of V receive the color 1.
Suppose an algorithm $\mathcal{A}$ makes a query $\alpha(x, y)$. We then respond according to the rules below:

- If $\chi(x) \neq \chi(y)$, add xy to $E(H_1)$ and reply "no" to the algorithm.
- If $\chi(x) = \chi(y)$ and $\{x, y\}$ is k-separable in H_1, add xy to $E(H_1)$ and modify the coloring of H_1 to a $\{x, y\}$-separating coloring, and reply "no".
- If $\chi(x) = \chi(y)$ and $\{x, y\}$ is k-inseparable in H_1, add xy to $E(H_2)$ and answer "yes".

Note that similar to before, adding new edges into H_1 does not change the inseparability of a pair $\{x, y\}$. Hence, if $\mathcal{A}$ learns the hidden graph G when the algorithm stops, the graph H_1 must be a uniquely k-colorable graph. This implies

$$|E(H_1)| \geq (k-1)n - \binom{k}{2}.$$

Additionally, we claim the following about H_2.

Claim 2.6 *When $\mathcal{A}$ stops, the graph H_2 must have at most k connected components.*

Proof Let us assume that the unique k-partition corresponding to H_1 is $\mathcal{P}_k = \{V_1, \dots, V_k\}$. Suppose on the contrary that H_2 has m connected components and $m \geq k+1$, say with vertex sets $C_1, \dots, C_m$. As every edge of H_2 corresponds to an inseparable pair, for every $i \in [m]$, there exists $j \in [k]$ such that $C_i \subset V_j$. In other words, the m-partition $\mathcal{P}_m = \{C_1, \dots, C_m\}$ is a refinement of P_k. The contradiction is that $\mathcal{A}$ cannot distinguish the m-partition $\mathcal{P}_m$ from the k-partition $\mathcal{P}_k$. Therefore, H_2 has at most k connected components. ■

As an immediate corollary, we obtain $|E(H_2)| \geq n - k$. Therefore, the algorithm requires at least

$$|E(H_1)| + |E(H_2)| \geq (k-1)n - \binom{k}{2} + n - k = kn - \binom{k+1}{2}$$

many queries, completing the proof of Theorem 1.3.

2.3. Multiple-membership query

Now we turn our attention to the problem of learning both k and individual components using multiple-membership queries.

2.3.1. PROOF OF THEOREM 1.5

Recall that $\alpha_m(u, S) = 1$ if and only if there is a $v \in S$ such that u and v belong to the same component. There are three assertions in Theorem 1.5, and we prove each of them below.

- **Part 1. Learning k:** First, we demonstrate an algorithm that learns the number of components of an n-vertex hidden graph G using $O(n)$ queries to α_m. Start with $S = V(G)$. While there is a vertex $v \in S$ such that $\alpha_m(v, S \setminus \{v\}) = 1$, we delete v from S. When this algorithm terminates, we end up with an independent set S whose each vertex lies in a different component. Conversely, as every other vertex $v \in V(G)$ lies in the same component as some vertex in G, $|S|$ will equal the number of components of G. Therefore, the above algorithm learns k using $O(n)$ many α_m-queries. ■

- **Part 2. Lower bound on learning components:** Next, we claim that learning all components of G requires at least $\Omega(n \log k)$ queries. This proof is information-theoretic. The total number of partitions of n vertices into k parts is $\Omega(k^n)$, and hence any algorithm making α_m-queries must have depth $\Omega(n \log k)$. ■

- **Part 3. Upper bound on learning components:** Note that the function learnComponentsMMQ learns the components of the hidden graph G using multiple-membership queries.

  The binary search in the **else** statement works because α_m can query v_i with a union of several C_j's at once. This step requires $O(\log k)$ queries, and therefore our algorithm has query complexity of $O(n \log k)$. ■

This completes the proof of Theorem 1.5, and hence α_m is an oracle that can learn the number of components k in a hidden graph with fewer queries than learning the components. □

```
Function learnComponentsMMQ
Input  : Vertex set $V(G) = \{v_1, \ldots, v_n\}$, Oracle $\alpha_m$.
Output: Partition $V(G) = C_1 \sqcup \cdots \sqcup C_k$ such that each $C_i$ is a connected component.
Initialize $C_1 = \{v_1\}$, $k = 1$;
for $i = 2$ to $n$ do
    if $\alpha_m(v_i, C_1 \cup \cdots \cup C_k) = 0$ then
        k += 1 ;
        Add $v_i$ to $C_k$;
    end
    else
        Find $j$ such that $\alpha_m(v_i, C_j) = 1$ via binary search among $\{C_1, \ldots, C_k\}$;
        Add $v_i$ to its corresponding $C_j$;
    end
end
return $\{C_1, \ldots, C_k\}$
```

3. The vertex-neighborhood detection query

For the remainder of the paper, we consider the vertex-neighborhood detection query given by $\beta(v, S) = 1$ iff there is some edge from v to some vertex in S. We now prove tight bounds of $\widetilde{\Theta}(m)$ for both learning and verifying graphs (with m edges) using β-queries.

3.1. Proof of Theorem 1.6

Now we move onto analyzing the problems of graph learning and verification using the oracle β. Let us take a hidden graph G on n vertices and m edges. It is clear that β can only detect one edge at a time, and hence both learning and verifying a hidden graph using β would trivially require at least $\Omega(m)$ queries. Hence, it suffices to prove upper bounds of $O(m + n)$ for verification and $O(m \log n)$ for learning G.

3.1.1. VERIFICATION USING β

For the verification problem, we have a graph $\widehat{G}$ with $V(\widehat{G}) = V(G) = V$ that is known to us. We verify each edge $uv \in E(\widehat{G})$ individually by checking $\beta(u, \{v\}) = 1$, and this requires m queries.

Next, we verify the non-edges of $\widehat{G}$. Fix a vertex v and compute its neighborhood $N_{\widehat{G}}(v) = \{u \in V : uv \in E(\widehat{G})\}$. Note that if $\widehat{G}$ was the same as G, we would have

$$\beta\Big(v, V \setminus (N_{\widehat{G}}(v) \cup \{v\})\Big) = 0.$$

Since checking all non-edges through a vertex takes a single β-query, we can verify all non-edges of $\widehat{G}$ using n queries. Hence, verifying all edges and non-edges of G using can be done using $O(m + n)$ β-queries. ■

3.1.2. LEARNING USING β

Although slightly more involved, our algorithm for learning a hidden graph G is also based on divide-and-conquer. The main step consists of devising a recursive method (which we call findNeighbors) that, for a fixed vertex $v \in V$, learns all neighbors (and non-neighbors) of v.

Function findNeighbors(v, S).

Input : Vertex v, Set S, Hidden graph G, Oracle β.
Result: Mark all neighbors of v in S blue, and non-neighbors red.

```
if $|S| = 1$ then
    if $\beta(v, S) = 1$ then
        Mark the vertex of $S$ blue and terminate;
    end
    else
        Mark the vertex of $S$ red and terminate;
    end
end
Divide $S$ into two (approximately) equal parts $S_1 \sqcup S_2$;
if $\beta(v, S_1) = 1$ then
    findNeighbors($v, S_1$);
end
else
    Mark all vertices of $S_1$ red and terminate;
end
if $\beta(v, S_2) = 1$ then
    findNeighbors($v, S_2$);
end
else
    Mark all vertices of $S_2$ red and terminate;
end
```

Let us first analyze the query complexity of findNeighbors.

Claim 3.1 *Suppose ℓ is the number of neighbors of v in S. Then, findNeighbors(v, S) takes at most $O(\ell \log |S|)$ queries to mark all neighbors of v in S red and non-neighbors blue.*

Proof (Claim 3.1): We take a look at the recursion tree T for findNeighbors(v, S), and color its nodes red or blue as follows. The root node corresponds to the computation of findNeighbors(v, S), and so we keep the set S in it. If $\beta(v, S) = 1$, we color the root node blue, otherwise we color it red. In the next level, the node containing S has two children: S_1 and S_2 corresponding to the equipartition of S. Let us color S_1 blue if $\beta(v, S_1) = 1$, and red otherwise. Similarly, we color S_2, and continue this coloring scheme down the entire recursion tree. A sample T and its coloring is depicted in Figure 3.1.

Let T_b be the subtree with blue vertices. It is clear that each leaf of T_b corresponds to a vertex marked blue by findNeighbors, implying that T_b is a tree with ℓ leaves and depth at most $\log |S|$.

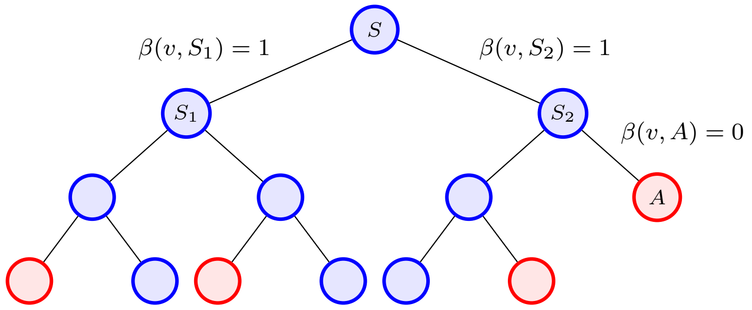


Figure 3.1: A sample recursion tree T corresponding to findNeighbors(v, S)

Therefore,

$$|V(T_b)| \leq O(\ell \log |S|).$$

Now we take a closer look at the red vertices of T. By construction, each red vertex is a leaf, i.e. the computation stops at these vertices. This means that the parents of red vertices are always blue. Further, two red nodes cannot be siblings, as if $\beta(v, A) = \beta(v, B) = 0$, then $\beta(v, A \cup B) = 0$, implying that the computation would have stopped at the parent of A and B. Thus, every red vertex has a ***unique*** blue parent. This implies that the number of red vertices in T is bounded above by $|V(T_b)|$, leading to

$$|V(T)| \leq 2|V(T_b)| \leq O(\ell \log |S|),$$

completing the proof of Claim 3.1. ■

Therefore, to learn G, we can run findNeighbors$(v, V \setminus \{v\})$ over all vertices v. Observe that the total number of queries made is at most

$$\sum_{v \in V(G)} O(\deg(v) \log n) = O(m \log n),$$

thus proving the required upper bound. ■

4. Conclusion and future work

In this paper, we demonstrated a fundamental and exact lower bound on the problem of learning partitions using membership queries (which we called α), filling a long lasting gap in the literature. We generalized the membership oracle to take subsets of vertices as one of its inputs (which we called α_m), and demonstrated how learning the number of components can be done using fewer α_m-queries than the components themselves. In the second section, we also demonstrated a powerful oracle (which we called β) that can be used to efficiently learn and verify sparse graphs.

It would be very interesting to see other oracles which can exploit structural properties of sparse graphs, such as the existence of small separators. Graph families that admit the existence of small separators include a vast array of graphs such as planar graphs Lipton and Tarjan (1979); Alon et al. (1994), bounded genus graphs Gilbert et al. (1984), minor-free graphs Reed and Wood (2009);

Wulff-Nilsen (2011), etc. Further, it should be possible to extend these results to learning and verification of graphs with polynomially bounded expansion Nešetřil and De Mendez (2008); Dvorák and Norin (2016), and we leave this as a future avenue of investigation.

Acknowledgments

The authors are immensely thankful to Mano Vikash Janardhanan for bringing the membership oracle to their attention, and for several fruitful discussions on graph learning theory. Research of the first author was supported by ERC Advanced Grant 101020255.

References

Mikkel Abrahamsen, Greg Bodwin, Eva Rotenberg, and Morten Stöckel. Graph reconstruction with a betweenness oracle. In *33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

Noga Alon and Vera Asodi. Learning a hidden subgraph. *SIAM Journal on Discrete Mathematics*, 18(4):697–712, 2005.

Noga Alon, Paul Seymour, and Robin Thomas. Planar separators. *SIAM Journal on Discrete Mathematics*, 7(2):184–193, 1994.

Noga Alon, Richard Beigel, Simon Kasif, Steven Rudich, and Benny Sudakov. Learning a hidden matching. *SIAM Journal on Computing*, 33(2):487–501, 2004.

Dana Angluin and Jiang Chen. Learning a hidden graph using o (logn) queries per edge. *Journal of Computer and System Sciences*, 74(4):546–556, 2008.

Zuzana Beerliova, Felix Eberhard, Thomas Erlebach, Alexander Hall, Michael Hoffmann, Mat Mihal'ak, and L Shankar Ram. Network discovery and verification. *IEEE Journal on selected areas in communications*, 24(12):2168–2181, 2006.

Mathilde Bouvel, Vladimir Grebinski, and Gregory Kucherov. Combinatorial search on graphs motivated by bioinformatics applications: A brief survey. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 16–27. Springer, 2005.

Huilan Chang, Hong-Bin Chen, Hung-Lin Fu, and Chie-Huai Shi. Reconstruction of hidden graphs and threshold group testing. *Journal of combinatorial optimization*, 22(2):270–281, 2011.

Zdenek Dvorák and Sergey Norin. Strongly sublinear separators and polynomial expansion. *SIAM Journal on Discrete Mathematics*, 30(2):1095–1101, 2016.

John R Gilbert, Joan P Hutchinson, and Robert Endre Tarjan. A separator theorem for graphs of bounded genus. *Journal of Algorithms*, 5(3):391–407, 1984.

Vladimir Grebinski and Gregory Kucherov. Optimal reconstruction of graphs under the additive model. *Algorithmica*, 28(1):104–124, 2000.

Jotun J Hein. An optimal algorithm to reconstruct trees from additive distance data. *Bulletin of mathematical biology*, 51(5):597–603, 1989.

Mano Vikash Janardhanan. Graph verification with a betweenness oracle. In *International Conference on Algorithmic Learning Theory*, pages 238–249. PMLR, 2017.

Sampath Kannan, Claire Mathieu, and Hang Zhou. Near-linear query complexity for graph inference. In *International Colloquium on Automata, Languages, and Programming*, pages 773–784. Springer, 2015.

Sampath Kannan, Claire Mathieu, and Hang Zhou. Graph reconstruction and verification. *ACM Transactions on Algorithms (TALG)*, 14(4):1–30, 2018.

Valerie King, Li Zhang, and Yunhong Zhou. On the complexity of distance-based evolutionary tree reconstruction. In *SODA*, volume 3, pages 444–453, 2003.

Richard J Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

Claire Mathieu and Hang Zhou. Graph reconstruction via distance oracles. In *International Colloquium on Automata, Languages, and Programming*, pages 733–744. Springer, 2013.

Claire Mathieu and Hang Zhou. A simple algorithm for graph reconstruction. In *29th Annual European Symposium on Algorithms (ESA 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

Jaroslav Nešetřil and Patrice Ossona De Mendez. Grad and classes with bounded expansion ii. algorithmic aspects. *European Journal of Combinatorics*, 29(3):777–791, 2008.

Bruce Reed and David R Wood. A linear-time algorithm to find a separator in a graph excluding a minor. *ACM Transactions on Algorithms (TALG)*, 5(4):1–16, 2009.

Lev Reyzin and Nikhil Srivastava. Learning and verifying graphs using queries with a focus on edge counting. In *International Conference on Algorithmic Learning Theory*, pages 285–297. Springer, 2007.

Guozhen Rong, Wenjun Li, Yongjie Yang, and Jianxin Wang. Reconstruction and verification of chordal graphs with a distance oracle. *Theoretical Computer Science*, 859:48–56, 2021.

Xu Shaoji. The size of uniquely colorable graphs. *Journal of Combinatorial Theory, Series B*, 50 (2):319–320, 1990.

M. Truszczyński. Some results on uniquely colourable graphs. In *Finite and infinite sets, Vol. I, II (Eger, 1981)*, volume 37 of *Colloq. Math. Soc. János Bolyai*, pages 733–748. North-Holland, Amsterdam, 1984.

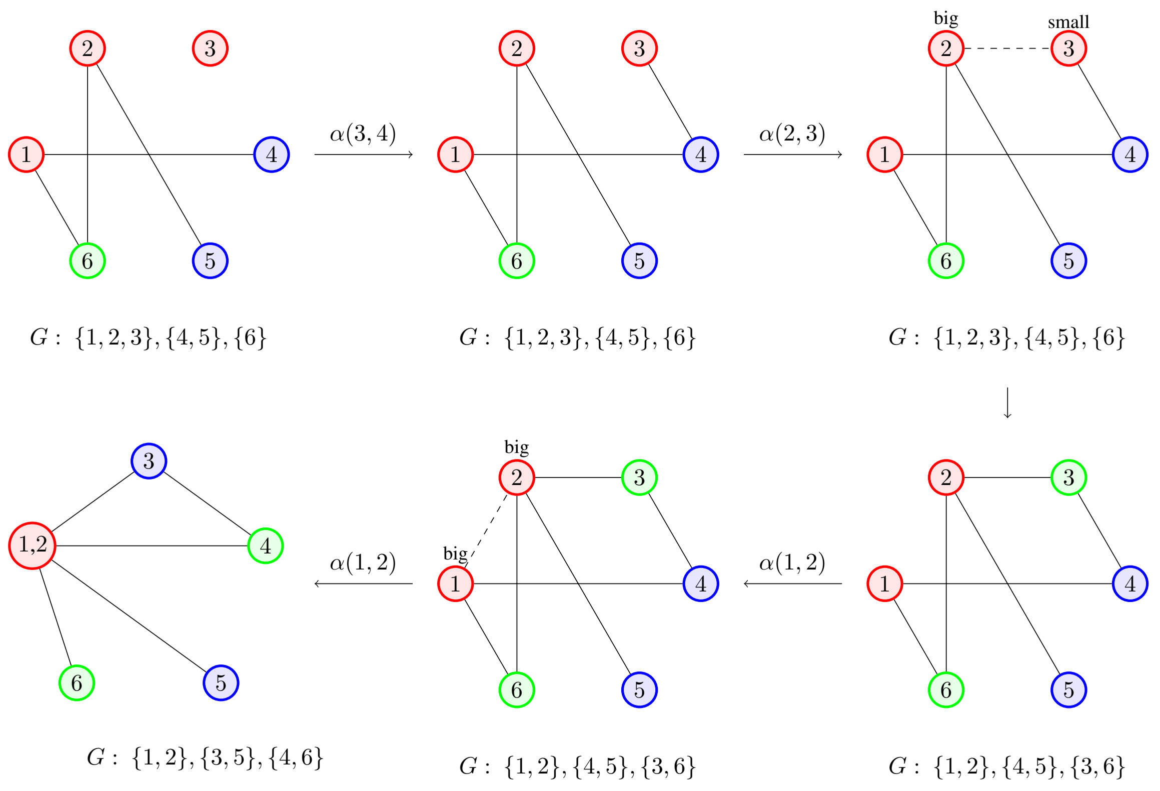
Christian Wulff-Nilsen. Separator theorems for minor-free and shallow minor-free graphs with applications. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 37–46. IEEE, 2011.

Appendix A. Polynomial-time adversary against Algorithm 1.1

Our goal in this section is to prove a slight modification of Theorem 1.2. We prove that by sacrificing a constant factor of $\frac{1}{2}$, it is possible to demonstrate an adversary that runs in $\text{poly}(n, k)$ time:

Proposition A.1 *Given any algorithm $\mathcal{A}$ that makes membership queries on a hidden graph G with n vertices and k components, there is a $\text{poly}(n, k)$-time adversary that can force $\mathcal{A}$ to make at least $\frac{1}{2}(n-k)(k-1)$ α-queries to learn the partition of G.*

Proof As in the original proof, we initialize an empty graph H with $|V(H)| = n$, and pick any k-coloring $\chi : V(H) \to \{1, \dots, k\}$ of H. Let G be the hidden graph corresponding to H, where each color class corresponds to a partition. For the sake of simplicity, we shall call vertices of H with degree at least $k-1$ *big*, and those with degree less than $k-1$ *small*.



$\alpha(3,4)$
$\alpha(2,3)$
$\alpha(1,2)$
$\alpha(1,2)$
big
small
big
big
$G:\ \{1,2,3\},\{4,5\},\{6\}$
$G:\ \{1,2,3\},\{4,5\},\{6\}$
$G:\ \{1,2,3\},\{4,5\},\{6\}$
$G:\ \{1,2\},\{3,5\},\{4,6\}$
$G:\ \{1,2\},\{4,5\},\{3,6\}$
$G:\ \{1,2\},\{4,5\},\{3,6\}$

Figure A.1: A sample set of queries and the corresponding evolution of H and the partition of G. Here $n = 6$ and $k = 3$, and vertices of degree at least 2 are "big". Our adversary responds "yes" to $\alpha(1,2)$ and "no" to every other query made by $\mathcal{A}$.

Suppose now that $\mathcal{A}$ makes a query $\alpha(x, y)$. We respond to $\mathcal{A}$ as follows:

- If $\chi(x) \neq \chi(y)$, we add xy to $E(H)$ and reply "no" to the algorithm.
- If $\chi(x) = \chi(y)$ and either x or y is *small*, say x, then we note that x has at most $k-2$ neighbors and hence two admissible colors. We then modify $\chi(x)$ to its other admissible color different from $\chi(y)$, add the edge xy to $E(H)$, and reply "no".

- Finally, if $\chi(x) = \chi(y)$ and both x and y are *big*, we identify (i.e., contract) vertices x and y. In terms of G, this would mean x and y belong to the same component.

We illustrate some intermediate steps in the evolution of H and χ in Figure A.1.

Note that χ is always a proper k-coloring of H, and hence induces a partition of G into k components. Suppose that $\mathcal{A}$ learns a unique partition of G into k components. Then, there are two cases to consider:

- **Case 1. The adversary contracts at least $\frac{n}{2}$ vertices:** In this case, as each contracted vertex has degree at least $k-1$, $\mathcal{A}$ must have made at least $\frac{n}{2}(k-1)$ queries.
- **Case 2. The adversary contracts less than $\frac{n}{2}$ vertices:** In this case, $\mathcal{A}$ can learn a unique partition of G iff χ is a unique k-coloring of H. Since H started with n vertices, H is a graph on at least $\frac{n}{2}$ vertices which is uniquely k-colorable. By Theorem 2.2, H has at least $\frac{n}{2}(k-1) - \binom{k}{2}$ edges.

In either case, $\mathcal{A}$ makes at least $\frac{n}{2}(k-1) - \binom{k}{2} = \frac{1}{2}(n-k)(k-1)$ queries, as desired. ■