On Learning Graphs with Edge-Detecting Queries

Hasan Abasi SHSON.TECH@GMAIL.COM
Technion, Haifa, Israel

Nader H. Bshouty BSHOUTY@CS.TECHNION.AC.IL
Technion, Haifa, Israel

Editors: Aurélien Garivier and Satyen Kale

Abstract

We consider the problem of learning a general graph $G = (V, E)$ using edge-detecting queries, where the number of vertices $|V| = n$ is given to the learner. The information theoretic lower bound gives $m \log n$ for the number of queries, where $m = |E|$ is the number of edges. In case the number of edges m is also given to the learner, Angluin-Chen's Las Vegas algorithm (Angluin and Chen, 2008) runs in 4 rounds and detects the edges in $O(m \log n)$ queries. In the harder case where the number of edges m is unknown, their algorithm runs in 5 rounds and asks $O(m \log n + \sqrt{m} \log^2 n)$ queries. They presented two open problems: *(i)* can the number of queries be reduced to $O(m \log n)$ in the second case, and, *(ii)* can the number of rounds be reduced without substantially increasing the number of queries (in both cases).

For the first open problem (when m is unknown) we give two algorithms. The first is an $O(1)$-round Las Vegas algorithm that asks $m \log n + \sqrt{m}(\log^{[k]} n) \log n$ queries for any constant k where $\log^{[k]} n = \log \overset{k}{\cdots} \log n$. The second is an $O(\log^* n)$-round Las Vegas algorithm that asks $O(m \log n)$ queries. This solves the first open problem for any practical n, for example, $n < 2^{65536}$. We also show that no deterministic algorithm can solve this problem in a constant number of rounds.

To solve the second problem we study the case when m is known. We first show that any non-adaptive Monte Carlo algorithm (one-round) must ask at least $\Omega(m^2 \log n)$ queries, and any two-round Las Vegas algorithm must ask at least $m^{4/3-o(1)} \log n$ queries on average. We then give two two-round Monte Carlo algorithms, the first asks $O(m^{4/3} \log n)$ queries for any n and m, and the second asks $O(m \log n)$ queries when $n > 2^m$. Finally, we give a 3-round Monte Carlo algorithm that asks $O(m \log n)$ queries for any n and m.

Keywords: Graph Learning, Group Testing, Edge-Detecting Queries, Monte Carlo Algorithm, Las Vegas Algorithm.

1. Introduction

We consider the problem of learning a general graph $G = (V, E)$ using edge-detecting queries. In edge-detecting queries the learning algorithm asks whether a subset of vertices $Q \subseteq V$ contains an edge in the graph G. That is, whether there are $u, v \in Q$ such that $\{u, v\} \in E$. The learner knows the number of vertices $|V| = n$.

Graph learning is a well-studied problem. It has been studied for general graphs, (Angluin and Chen, 2008) and also for specific graph families (i.e. matching, stars, cliques and other) see (Alon and Asodi, 2005; Alon et al., 2004). This problem has also been generalized to learning a hypergraph (where each edge consists of two or more vertices) see (Abasi et al., 2018, 2014; Angluin and

Chen, 2006). The motivation behind studying some graph families relevant to the problem above, was its various applications in different areas such molecular biology, chemistry and networks (Bouvel et al., 2005; Angluin et al., 2010). For example, the general graph case is motivated by problems from biology and chemistry where, given a set of molecules (chemicals), we need pairs that react with each other. In this case, the vertices correspond to the molecules (chemicals), the edges to the reactions, and the queries to experiments of putting a set of molecules (chemicals) together in a test tube and determining whether a reaction occurs. When multiple molecules (chemicals) are combined in one test tube, a reaction is detectable if and only if at least one pair of the molecules (chemicals) in the tube react. The task is to identify which pairs react using as few experiments as possible. One more example of a problem encountered by molecular biologists, is the problem of finding a hidden match, when applying multiplex PCR in order to close the gaps left in a DNA strand after shotgun sequencing. See (Alon and Asodi, 2005; Alon et al., 2004) and there references for more details.

For a general graph, the information theoretic lower bound gives $m\log n$ for the number of queries where $m = |E|$ is the number of edges. In case the number of edges m is also given to the learner, Angluin-Chen's Las Vegas algorithm (Angluin and Chen, 2008) runs in 4 rounds and detects the edges in $O(m\log n)$ queries. In the harder case where the number of edges m is unknown, their algorithm runs in 5 rounds and asks $O(m\log n + \sqrt{m}\log^2 n)$ queries. There have been two open problems: *(i)* can the number of queries be reduced to $O(m\log n)$ in the second case, and, *(ii)* can the number of rounds be reduced without substantially increasing the number of queries (in both cases).

For the first open problem (when m in unknown) we give two algorithms. The first is an $O(1)$-round algorithm that asks $m\log n + \sqrt{m}(\log^{[k]} n)\log n$ queries for any constant k. Here $\log^{[k]} n = \log \overset{k}{\cdots} \log n$. The second is an $O(\log^* n)$-round algorithm that asks $O(m\log n)$ queries. This solves the first open problem for any practical n, for example $n < 2^{65536}$. We also show that no deterministic algorithm can solve this problem in a constant number of rounds.

To solve the second problem we study the problem when m is known. We first show that any non-adaptive (one-round) Monte Carlo algorithm must ask at least $\Omega(m^2\log n)$ queries. We then give two two-round Monte Carlo algorithms, the first asks $O(m^{4/3}\log n)$ queries for any n and m, and the second asks $O(m\log n)$ queries when $n > 2^m$. Then we give a 3-round Monte Carlo algorithm that asks $O(m\log n)$ queries for any n and m. We also show that any two-round Las Vegas algorithm must ask at least $m^{4/3}$ queries on average and any deterministic two-round algorithm must ask at least $\Omega(m^2\log n)$ queries. Finally, we give a four-round deterministic algorithm that asks $m^{2+\epsilon}\log n$ queries for any constant ϵ. The question whether there is an $O(1)$-round deterministic algorithm that asks $O(m\log n)$ queries remains open.

Our results are summarized in the following two tables:

1.1. Results For Unknown m

	Lower Bound	Upper Bound	Poly. Time
LV& MC Randomized $O(1)$-Rounds	$m\log n$	$m\log n+$ $\sqrt{m}(\log^{[k]} n)\log n$	$m\log n+$ $\sqrt{m}(\log^{[k]} n)\log n$
LV&MC Randomized $O(\log^* n)$-Round	$m\log n$	$m\log n$	$m\log n$

1.2. Results For Known m

	Lower Bound	Upper Bound	Poly. Time
Non-Adaptive	Thm.5	⇒	Thm. 3
MC Randomized	$m^2 \log n$	$m^2 \log n$	$m^2 \log n$
Non-Adaptive	(D'yachkov and Rykov, 1983)	⇒	(Bshouty, 2015)
Deter. & LV Rand.	$\frac{m^3}{\log m} \log n$	$m^3 \log n$	$m^3 \log n$
Two Rounds		⇒	Thm.7&9
MC Randomized	OPEN	$m^{4/3} \log n$	$m^{4/3} \log n$
$n \to \infty$	$m \log n$	$m \log n$	$m \log n$
Two Rounds		⇒	Thm. 3
LV Randomized	OPEN	$m^2 \log n$	$m^2 \log n$
Two Rounds	Thm.19	Thm.16	
Deterministic	$\frac{m^2}{\log m} \log n$	$m^2 \log n$	OPEN
Three Rounds	I.T.	⇒	Thm. 8
MC Randomized	$m \log n$	$m \log n$	$m \log n$
Three Rounds	I.T.	⇒	Thm.7&11
LV Randomized	$m \log n$	$m^{4/3} \log n$	$m^{4/3} \log n$
$n \to \infty$	$m \log n$	$m \log n$	$m \log n$
Three Rounds			
Deterministic	OPEN	OPEN	OPEN
Four Rounds	I.T.	⇒	Thm. 8
LV Randomized	$m \log n$	$m \log n$	$m \log n$
Five Rounds		⇒	Thm.22
Deterministic	OPEN	$m^{2+\epsilon} \log n$	$m^{2+\epsilon} \log n$

2. Definitions and Preliminary Results

Let $G = (V, E)$ be a simple (contains no loop or multiple edges) undirected graph with $|V| = n$ vertices and $|E| \le m$ edges. We call G the *target m-graph*. The *learner* knows n. We study both cases where m is known to the learner and when it is not. The learner can ask an *edge-detecting queries* (or just a *query*) $Q \subseteq V$ to an *oracle* $\mathcal{O}_G$. The answer to the query Q is $\mathcal{O}_G(Q) =$"YES" if there are $u, v \in Q$ such that $\{u, v\} \in E$ and "NO" otherwise.

We say that a non-simple graph $G = (V, E)$ is an *m-Loops* if the graph contains at most m loops. That is, the graph contains at most m edges where each edge connects a vertex to itself. Learning m-Loop is equivalent to the problem of group testing (Du et al., 2000; Kwang-ming and Ding-zhu, 2006).

For a graph $G = (V, E)$ and a subset of vertices $V' \subseteq V$ we define the *neighbours of* V', $\Gamma_G(V')$, as the set of all vertices u in $V \backslash V'$ such that there is $v \in V'$ where $\{u, v\} \in E$. When $V' = \{v\}$ then we write $\Gamma_G(v)$ and call it the neighbours of v. We say that V' is an independent set if there are no edges between the vertices in V'.

We will denote by $[n] := \{1, 2, \ldots, n\}$ and $[m, n] = \{m, m + 1, \ldots, n\}$.

2.1. Preliminary Results for Randomized Algorithms

The following lemma is from (Angluin and Chen, 2008):

Lemma 1 *Let $G = (V, E)$ be the target graph with n vertices and m edges. Let $Q \subseteq V$ be a random query where for each vertex $i \in V$, i is included in Q independently with probability p. Let $\{u, v\} \notin E$. Then*

1. *If $|\Gamma_G(\{u, v\})| \le r$, then with probability at least $p^2(1 - rp - mp^2)$, $u, v \in Q$ and $\mathcal{O}_G(Q) =$“NO”.*
2. *If $|\Gamma_G(\{u, v\})| \ge r$, then with probability at most $p^2(1 - p)^r$, $u, v \in Q$ and $\mathcal{O}_G(Q) =$“NO”.*

Proof Let $\Gamma_G(\{u, v\}) = \{w_1, \ldots, w_{r'}\}$. Let $\{u_1, v_1\}, \ldots, \{u_{m-r'}, v_{m-r'}\}$ be all the edges of G that such that none of their endpoints are u or v. When $r' \le r$,

$$\begin{aligned}
\Pr[u, v \in Q \ \wedge\ \mathcal{O}_G(Q) = \text{“NO”}] &= \Pr[u, v \in Q] \Pr[\mathcal{O}_G(Q) = \text{“NO”} \mid u, v \in Q] \\
&= p^2 \Pr[(\forall i) w_i \notin Q \ \wedge\ (\forall j)\{u_j, v_j\} \not\subseteq Q] \\
&= p^2(1 - \Pr[(\exists i) w_i \in Q \ \vee\ (\exists j)\{u_j, v_j\} \subseteq Q]) \\
&\ge p^2(1 - r'p - (m - r')p^2) \\
&\ge p^2(1 - rp - mp^2).
\end{aligned}$$

and when $r' \ge r$,

$$\begin{aligned}
\Pr[u, v \in Q \ \wedge\ \mathcal{O}_G(Q) = \text{“NO”}] &= p^2 \Pr[(\forall i) w_i \notin Q \ \wedge\ (\forall j)\{u_j, v_j\} \not\subset Q] \\
&\le p^2 \Pr[(\forall i) w_i \notin Q] \\
&= p^2(1 - p)^{r'} \le p^2(1 - p)^r.
\end{aligned}$$

■

Lemma 2 *Let $G = (V, E)$ be the target graph with n vertices and m edges. Consider the algorithm in Figure 1 with*

$$t := t(n, m, r, p, \delta) = \frac{2 \ln n + \ln(1/\delta)}{p^2(1 - rp - mp^2)}.$$

Then for $E_r = \{\{u, v\} : |\Gamma_G(\{u, v\})| > r\}$,

1. *$E \subseteq E(H)$.*
2. *With probability at least $1 - \delta$, $E(H) \backslash (E \cup E_r) = \emptyset$, i.e., all the edges $\{u, v\} \in E(H) \backslash E$ satisfies $|\Gamma_G(\{u, v\})| > r$.*
3. *$\mathbf{E}\,[|E(H) \backslash (E \cup E_r)|] \le \delta$.*

Proof Consider the algorithm in Figure 1. An edge $\{i, j\}$ is removed from the graph H if and only if $i, j \in Q$ and $\mathcal{O}_G(Q) =$“NO”. Therefore, if $\{i, j\} \in E$ then $\{i, j\} \in E(H)$. This proves *1*.

We now prove *2*. Let $\{u, v\}$ be such that $|\Gamma(\{u, v\})| \le r$. The probability that $\{u, v\} \in E(H) \backslash E$ is the probability that for all the queries $Q^{(i)}$, $i = 1, \ldots, t$ in the algorithm, either $\{u, v\} \not\subseteq$

```
Algorithm

Learn First Round

$E(H) \leftarrow \{\{u,v\} \mid u,v \in V, u \neq v\}$
Repeat $t(n,m,r,p,\delta)$ times
    Choose $Q \subset V$ s.t. for all $i \in V$, $\Pr[i \in Q] = p$.
    Answer$\leftarrow \mathcal{O}_G(Q)$
    If Answer="NO" then
        For every $i,j \in Q$.
            $E(H) \leftarrow E(H)\backslash\{(i,j)\}$.
End Repeat
Output($H = (V, E(H))$)
```

Figure 1: First Round in the Algorithm.

$Q^{(i)}$ or $\mathcal{O}_G(Q^{(i)})$ ="YES". By Lemma 1, this probability is at most $(1-p^2(1-rp-mp^2))^t$. Therefore, the probability that there is $\{u,v\}$ such that $|\Gamma(\{u,v\})| \le r$ and $\{u,v\} \in E(H)\backslash E$ is at most

$$n^2(1-p^2(1-rp-mp^2))^t \le \delta.$$

The expected number of edges in $E(H)\backslash(E \cup E_r)$ is also at most $n^2(1-p^2(1-rp-mp^2))^t \le \delta$.

■

3. Non-Adaptive Algorithms when m is Known

In this section we study non-adaptive learning algorithms when m is known to the algorithm. We first give a polynomial time Monte Carlo algorithm that asks $O(m^2 \log n)$ queries and then prove the lower bound $\Omega(m^2 \log n)$ for the number of queries. For the non-adaptive deterministic algorithm, there is a polynomial time algorithm that asks $O(m^3 \log n)$ queries and it is known that the lower bound for the number of queries is $\Omega(m^3 \log n / \log m)$ (Abasi et al., 2018; Kwang-ming and Dingzhu, 2006). For Las Vegas algorithms, a non-adaptive algorithm must be deterministic because the success probability is 1. Therefore, the bounds for the number of queries of deterministic algorithms apply also for Las Vegas algorithms.

3.1. Upper Bound for Randomized Non-Adaptive Algorithms

We first state the following theorem:

Theorem 3 *There is a non-adaptive Monte Carlo randomized learning algorithm with $1/poly(n)$ error probability for m-Graph that asks $O(m^2 \log n)$ queries.*

There is a two-round Las Vegas randomized learning algorithm for m-Graph that the expected number of queries it asked is
$O(m^2 \log n)$.

Proof Consider the algorithm in Figure 1 with $r = m$, $p = 1/(2m)$ and $\delta = 1/poly(n)$. Then by Lemma 2, the number of queries is $t(n, m, r, p, \delta) = O(m^2 \log n)$. Again by Lemma 2, $E \subseteq E(H)$ and with probability at least $1 - \delta$ all the edges $\{u, v\} \in E(H)\backslash E$ satisfies $|\Gamma(\{u, v\})| > m + 1$. Since G has at most m edges we must have $|\Gamma(\{u, v\})| \le m + 1$ for any $u, v \in V$ and therefore with probability at least $1 - \delta$, $E(H) = E$.

For the Las Vegas algorithm we add another round that asks a query for each edge in $E(H)$. The number of expected edges is less than $m + 1$ and therefore the expected number of queries in the second round is less than $m + 1$. ■

3.2. Lower Bound for Randomized Non-Adaptive Algorithms

Lemma 4 *Any non-adaptive Monte Carlo randomized learning algorithm with error probability at most $1/2$ for m-Loop must ask at least $\Omega(m \log n)$ queries.*

Proof The number of m-loops with n vertices is $\binom{n}{m}$. Therefore, by the information theoretic lower bound and Lemma 2 in (Abasi et al., 2014) the result follows. ■

Theorem 5 *Any non-adaptive Monte Carlo randomized learning algorithm with error probability at most $1/4$ for m-Graph must ask at least $\Omega(m^2 \log n)$ queries.*

Proof Let $\mathcal{A}(s, \mathcal{O}_G)$ be a Monte Carlo randomized non-adaptive learning algorithm that learns m-Graph over the vertices $[n]$ with error probability at most $1/4$, where $s \in \{0, 1\}^*$ is a random seed and $\mathcal{O}_G$ is the query oracle to the target graph G. Let A_s be the set of queries that are asked when the random seed is s. Suppose, on the contrary, that for all s, $|A_s| \le cm^2 \log n$, for some $c = o(1)$. Let $A_{s,i}$, $i = 1, \dots, m/2$ be the set of queries Q in A_s that contains the vertex i and does not contain any of the vertices in $[m/2]\backslash\{i\}$. That is $Q \cap [m/2] = \{i\}$. Then, over the uniform distribution, $(m/2)\mathbf{E}_{i\in[m/2]}[|A_{s,i}|] \le |A_s| < cm^2 \log n$. Therefore, $\mathbf{E}_{i\in[m/2]}[|A_{s,i}|] \le 2cm \log n$. Thus, by Markov's inequality, we get $\Pr_{i\in[m/2]}[|A_{s,i}| \ge 8cm \log n] < 1/4$ and $\Pr_{i\in[m/2]}[|A_{s,i}| < 8cm \log n] > 3/4$.

Now for any $i \in [m/2]$ and $J = \{\{j_1\}, \dots, \{j_{m/2}\}\}$, $m/2 + 1 \le j_1 < j_2 < \cdots < j_{m/2} \le n$ we define the set of graphs $G_{J,i}(V, E_{J,i})$ where

$$E_{J,i} := \{\{i\} \times ([m/2]\backslash\{i\}) \cup \{\{i, j_1\}, \dots, \{i, j_{m/2}\}\} \mid m/2 + 1 \le j_1 < j_2 < \cdots < j_{m/2} \le n\}.$$

Notice that any query Q in $A_s\backslash A_{s,i}$ gives no information about the vertices $j_1, \dots, j_{m/2}$. That is because, either $i \notin Q$ and then the answer is "NO" or $i \in Q$ and $Q \cap ([m/2]\backslash\{i\}) \ne \emptyset$ and then the answer is "YES".

We now give an algorithm $\mathcal{B}$ that learns $m/2$-Loop over $n - m/2$ vertices that are labeled with $\{m/2, \dots, n\}$ with success probability at least $1/2$ using at most $4cm \log n$ queries. This gives a contradiction to the result of Lemma 4 and then the result follows.

Algorithm $\mathcal{B}$ chooses a random uniform $i \in [m/2]$ and runs algorithm $\mathcal{A}$. Suppose the set of $m/2$ loops of the target is $J = \{\{j_1\}, \dots, \{j_{m/2}\}\}$. The goal of algorithm $\mathcal{B}$ is to provide $\mathcal{A}$ answers to its queries as if the target is $G_{J,i}$. Therefore, for each query Q that $\mathcal{A}$ asks, if $i \notin Q$ then algorithm $\mathcal{B}$ returns the answer "NO" to $\mathcal{A}$ and if $i \in Q$ and $Q \cap ([m/2]\backslash\{i\}) \ne \emptyset$ then it returns the answer "YES". If $Q \cap [m/2] = \{i\}$ then algorithm $\mathcal{B}$ asks the query $Q \cup \{i\}$ and returns the

```
Algorithm

Learning the neighbours of $v$ in the independent set $I$

$N = I$
Repeat $t$ times
    Choose a random $Q \subseteq I$
        where for every $u \in I$, $\Pr[u \in Q] = p$.
    Answer$\leftarrow \mathcal{O}_G(Q \cup \{v\})$
    If Answer="NO" then $N \leftarrow N \backslash Q$
End Repeat
Output($N$)
```

Figure 2: An algorithm that given an independent set I in V, finds the vertices in I that are neighbours of v.

answer to $\mathcal{A}$. Algorithm $\mathcal{B}$ halts if the number of queries is more than $8cm \log n$ or $\mathcal{A}$ outputs a graph $H(V, E')$. If $E' = \{i\} \times ([m/2]\backslash\{i\}) \cup \{\{i, j_1'\}, \ldots, \{i, j_{m/2}'\}\}$ then algorithm $\mathcal{B}$ outputs $G(V, \{\{j_1'\}, \ldots, \{j_{m/2}'\}\})$ otherwise it returns an empty graph.

Algorithm $\mathcal{B}$ fails if the number of queries is greater than $8cm \log n$ or if algorithm $\mathcal{A}$ fails. The former happens with probability at most $1/4$ and the latter with probability at most $1/4$. Therefore the probability of failure of the algorithm is at most $1/2$. ■

4. Two and Three Round Randomized Learning - m is Known

In this section we study two-round randomized algorithms.

We first prove that there is a two-round Monte Carlo randomized learning algorithm (with $1/poly(n)$ error probability) for m-Graph that asks $O(m^{4/3} \log n)$ queries. Then we show that, for $n > m^m$, there is a two-round randomized Monte Carlo learning algorithm for m-Graph that asks $O(m \log n)$ queries.

For Las Vegas algorithms we prove the above query complexities for three-round algorithms. We then show that any two-round Las Vegas algorithm must ask at least $\Omega((m^{4/3} \log^{1/3} n)/(\log^{1/3} m))$ queries. For $m > (\log n)^{\omega(1)}$ this lower bound is $\Omega(m^{4/3-o(1)} \log n)$.

4.1. Learning the Neighbours in an Independent Set

Lemma 6 *Consider the Algorithm in Figure 2. Let $I \subset V$ be an independent set in G. Let $v \notin I$ be a vertex in G. For $p = 1/m$ and $t = 4m(\ln n + \ln(1/\delta))$ with probability at least $1 - \delta$ the output N of the algorithm satisfies $N = \Gamma(v) \cap I$. That is, N contains only the neighbours of v in I.*

Proof The output N is not the set of neighbors of v if and only if for some $u \notin \Gamma(v)$ each query Q in the algorithm satisfies: $u \notin Q$ or $\Gamma(v) \cap Q \neq \emptyset$. Therefore, the probability that output N is not the set of neighbors of v is less than

$$n(1-p(1-p)^m)^t \le n\left(1-\frac{1}{4m}\right)^t \le \delta.$$

■

4.2. Upper Bound for Randomized Two-Round Algorithm

Consider the algorithm in Figure 1 with $p < 1/(8\sqrt{m})$ and $r = 1/(2p)$. By Lemma 2, $E \subseteq E(H)$ and for $t = O((1/p)^2(\log n + \log(1/\delta)))$, with probability at least $1-\delta$, every edge $\{u,v\} \in E(H)\backslash E$ satisfies $\deg_G(u) + \deg_G(v) > r+1$. Assume for now that this is true with probability 1.

Fact 1 *Every edge $\{u,v\} \in E(H)\backslash E$ satisfies $\deg_G(u) + \deg_G(v) > r+1$.*

Figure 4 (in the appendix section) will help you in following the proof. Let $V_G := \{v|\{u,v\} \in E\}$, $V_H := \{v|\{u,v\} \in E(H)\}$. We partition the set of edges $E(H)\backslash E$ to three disjoint set $E_0 \cup E_1 \cup E_2$ where $E_i = \{\{u,v\} \in E(H)\backslash E\ :\ |\{u,v\} \cap V_G| = i\}$. Fact 1 immediately implies that

Fact 2 *$E_0 = \emptyset$. That is, every edge in $E(H)\backslash E$, at least has one endpoint in V_G.*

Let $u \in V_H \backslash V_G$. Then $\deg_G(u) = 0$. Therefore, by Fact 1,

Fact 3 *For any edge $\{u,v\} \in E_1$ where $u \in V_H\backslash V_G$ we have $\deg_G(v) > r+1$.*

Since the number of vertices in G that have degree greater than $r+1$ is less than $2m/(r+2) \le r/8$, the degree of each $u \in V_H\backslash V_G$ in the graph H is at most $r/8$.

Fact 4 *If $u \in V_H\backslash V_G$ then $\deg_H(u) \le r/8$. In particular, all the vertices of degree greater than $r/8$ are in V_G.*

Now take any edge $\{u',v'\} \in E(H)\backslash E$. By Fact 1, $\deg_G(u') + \deg_G(v') > r+1$ and therefore either $\deg_G(u') > r/2$ or $\deg_G(v') > r/2$. If $\deg_G(v') > r/2$ then $\deg_H(v') > r/2$ and by Fact 4, $v' \in V_G$. This with Fact 3 shows that

Fact 5 *Every edge $\{u',v'\} \in E_2$, one of its endpoints is in V_G and has degree at least $r/2$ in G and therefore also in H.*

Every edge $\{u',v'\} \in E_1$ has one endpoint in V_G, and a degree greater than $r+1$ in G and therefore also in H.

Denote by W the set of all vertices of degree greater than $r/2$ in H. Then

Fact 6 *The number of vertices of degree more than $r/2$ in H is at most $8m/r$. That is, $|W| \le 8m/r$.*

Proof Let u be a vertex in G of degree less than $r/4$. Then all its edges in H are in $E_2 \cup E$. This is because if it has an edge in E_1 then by Fact 5, its degree in G is more than $r+1$. If $\{u,v\} \in E_2$ then by Fact 5 the degree of v in G is at least $r/2$. Since the number of vertices in G of degree at least $r/2$ is at most $2m/(r/2) < r/4$ the degree of u in H is less than $r/4 + r/4 = r/2$. This shows that a vertex in H of degree at least $r/2$ is of degree at least $r/4$ in G. Therefore the number of such vertices is at most $2m/(r/4) = 8m/r$. ■

We now prove

Fact 7

$$|E_2| \le \frac{8m^2}{r}.$$

Proof By Fact 5, for an edges in E_2, one of its endpoint vertices is in V_G and has degree at least $r/2$ in G. There are at most $4m/r$ such vertices and each one can have at most $|V_G| \le 2m$ edges. ■

Now define for every vertex $w \in W$ the set I_w that contains all the neighbours $u \in \Gamma_H(w)$ where $\deg_H(u) \le r/8$ and $\Gamma_H(u) \cap \Gamma_H(w)$ contains only vertices of degree more than $r+1$ in H.

Fact 8 *I_w is an independent set.*

Proof If I_w contains $\{u,v\} \in E(H)$ then $\deg_H(u), \deg_H(v) < r/8$. Since $\Gamma_H(u) \cap \Gamma_H(w)$ contains v, we get $\deg_H(v) > r+1$. This gives a contradiction. ■

We now show

Fact 9 *We have*

$$E_1 \subseteq E_W := \bigcup_{w \in W} \{\{w,u\} \mid u \in I_w\}$$

Proof If $\{w,u\} \in E_1$ then, by the definition of E_1 and Fact 5 and 4, one of the vertices, say w, is in V_G and is in W and the other vertex, u, is in $V_H \backslash V_G$ and has degree at most $r/8$ in H. Now we show that $u \in I_w$. If $u \notin I_w$ then $\Gamma_H(u) \cap \Gamma_H(w)$ contains a vertex v of degree less or equal to $r+1$. If $v \in V_H \backslash V_G$ then $\{u,v\}$ is an edge in E_0 and we get a contradiction to Fact 2. Therefore $v \in V_G$ then since $\{v,u\}$ is an edge in E_1 and $\deg_H(u) \le r/8$ we must have $\deg_H(v) \ge r+1$ and again we get a contradiction. ■

Since $U := E(H) \backslash E_W \subseteq E \cup E_2$, by Fact 7, we get

Fact 10

$$|U| = |E(H) \backslash E_W| \le m + \frac{8m^2}{r}.$$

Therefore, we first find W and for each $w \in W$ learn the neighbours of w in I_w. This eliminates all the edges of w that are not in the graph. In particular, it removes all the edges in E_1 and some of those in E_2. Then for each edge in $U = E(H) \backslash E_W$ we ask a query.

We now can prove

```
Algorithm

Learn Second Round

1.  $W = \{u \mid \{u,v\} \in E(H)$ and $\deg(u) > r/2\}$; $U = E(H) \backslash E_W$.
2.  For every $w \in W$ do
3.      Learn the Neighbours of w in the independent set $I_w$
4.      Let $N_w$ be the output.
5.      Remove from $E(H)$ all $\{u,w\}$ where $w \in I_w \backslash N_w$.
6.  For every $\{u,v\} \in U$
7.      Answer$\leftarrow$ $\mathcal{O}_G(\{u,v\})$
8.      If Answer="NO" then $E(H) \backslash \{\{u,v\}\}$.
6.  Output($H = (V, E(H))$)
```

Figure 3: Second Round in the Algorithm.

Theorem 7 *There is a two-round Monte Carlo randomized learning algorithm for m-Graph with $1/poly(n)$ error probability that asks $O(m^{4/3}\log n)$ queries.*

There is a three-round Las Vegas randomized learning algorithm for m-Graph that asks $O(m^{4/3}\log n)$ expected number of queries.

Proof Consider the First round in Figure 1 and the second round in Figure 3. We choose $p = 1/m^{2/3}$ and $\delta = 1/poly(n)$. Then $r = m^{2/3}/2$ and $t = O(m^{4/3}\log n)$.

By Fact 6, $|W| \le 8m/r$. By Lemma 6 finding the neighbours of each $w \in W$ takes $O(m\log n)$ queries. Therefore the total number of queries for steps 1-5 in the algorithm is $O((m^2/r)\log n) = O(m^{4/3}\log n)$. By Fact 10 the number of queries in steps 6-8 is at most $m + 8m^2/r = O(m^{4/3})$. ■

4.3. Randomized Three Rounds Monte Carlo Algorithm

In this section we show that when m is known to the algorithm then there is a three-round Monte Carlo algorithm that asks $O(m\log n)$ queries.

In (Angluin and Chen, 2008), Angluin and Chen gave a three-round Monte Carlo randomized algorithm that asks $O(m\log n + \sqrt{m}\log^2 n)$ queries. We give here a three-round algorithm that asks $O(m\log n + m^{1.5})$ queries. Both results imply

Theorem 8 *There is a three-round Monte Carlo randomized learning algorithm for m-Graph with $1/poly(n)$ error probability that asks $O(m\log n)$ queries.*

There is a four-round Las Vegas randomized learning algorithm for m-Graph that asks $O(m\log n)$ expected number of queries.

Proof If $m \ge \log^2 n$ then $O(m\log n + \sqrt{m}\log^2 n) = O(m\log n)$. Otherwise, $m < \log^2 n$ and then $O(m\log n + m^{1.5}) = O(m\log n)$.

Now we describe the algorithm. In the first round we run the algorithm in Figure 1 with $p = 1/(16\sqrt{m})$. By the facts in Section 4.2, for $r = 8\sqrt{m}$ we have

1. All the edges $\{u,v\} \in E(H)\backslash E$ satisfy $\deg_G(u)+\deg_G(v) > 8\sqrt{m}+1$.
2. $E_0=\emptyset$, $|E_2| \le m^{1.5}+1$, $E_1 \subset E_W$, $|W| \le \sqrt{m}$ and $|E(H)\backslash E_W| \le m^{1.5}+m+1$.

In the first round we ask $O(1/p^2)(\log n + \log(1/\delta)) = O(m\log n)$ queries. Now for each $w \in W$ we need to find the neighbors $\Gamma_G(w) \cap I_w$ of w in G. If we do that using the previous algorithm we get $O(m^{1.5}\log n)$ queries. Instead, we will add another round that estimates the number of neighbours of each $w \in W$ in G (and in I_w). We then learn the neighbour with $O(deg_G(w)\log n)$. This is possible because I_w is ab independent set. The estimation can be done by doubling and estimating with Chernoff bound. See (Falahatgar et al., 2016). The estimation can be done with $O(\log m \log n)$ queries for each $w \in W$ and success probability $1-1/poly(n)$ and therefore with $O(|W|\log m\log n) = O(\sqrt{m}\log m\log n)$ queries. Then since $|E(H)\backslash E_W| \le 2m^{1.5}+m+1$ finding the other edges in the graph can be done in (round 2 or 3) with $O(m^{1.5})$ queries. The total number of queries is $O(m\log n + m^{1.5})$. ■

4.4. Two-Round Learning for Large n

In this section we prove

Theorem 9 *Let $w > m$. There is a two-round randomized Monte Carlo learning algorithm with $1/w^{O(1)}$ error probability for m-Graph that asks $O(m^2\log w + m\log n)$ queries.*

In particular, when $w = n^{c/m}$ for any constant c (and therefore $m < \log n$) the algorithm asks $O(m\log n)$ queries.

Proof We first partition the set of vertices into $u = poly(w)$ sets $V_1,\ldots,V_u$. The probability that each set contains at most one vertex of degree not equal zero is at least

$$\left(1-\frac{1}{u}\right)\left(1-\frac{2}{u}\right)\cdots\left(1-\frac{2m-1}{u}\right) \ge 1-\frac{m(2m-1)}{u} \ge 1-\frac{1}{w^{O(1)}}. \qquad (1)$$

Assuming the vertices that have degree greater than zero are in different sets, we learn the graph over the u sets in one round with probability at least $1-1/w^{O(1)}$ and $O(m^2\log u) = O(m^2\log w)$ queries using Theorem 3. That is, we assume that each set V_j is one vertex and we learn the graph over the sets $V_1,\ldots,V_u$. Then, for each query $Q \subseteq [u]$ the algorithm asks the query $\cup_{w\in Q}V_w$. When the algorithm discover an edge $\{V_i,V_j\}$ then it knows that there is an edge between one of the vertices in V_i with one of the vertices in V_j. We will call the edge $\{V_i,V_j\}$ a *set edge* and V_i a *set vertex*.

Now, suppose there is a set of edges e with the set of endpoints vertices V_i and V_j. We can learn the endpoints vertices of e deterministically with $O(\log n)$ queries. To learn the endpoint in V_j the algorithm considers V_i as one vertex and runs the algorithm that learn 1-Loop in V_j. That is, for each query $Q \subseteq V_j$ the algorithm asks the query $V_i \cup Q$. Therefore, in the second round, we can deterministically learn the endpoints of all the edges in $O(m\log n)$ queries. ■

We now convert the above algorithm to a three-round Las Vegas algorithm.

We first give one definition and a lemma. For a query Q and a vertex u we define $[u \in Q] = 1$ if $u \in Q$ and $[u \in Q] = 0$ if $u \notin Q$. We now prove

Lemma 10 *There is a non-adaptive deterministic algorithm that asks $O(\log n)$ queries and if the target is 0-Loop or 1-Loop then it learns the target and if it is k-Loop, $k > 1$, then it returns an "ERROR".*

Proof We define the set of $2t$ queries $\{Q_1, \ldots, Q_{2t}\}$ as follows. Each vertex i appears in exactly t queries and no two vertices appears in the same set of queries. We must have $\binom{2t}{t} \geq n$ and therefore it is enough to take $2t = \log n + 2 \log \log n = O(\log n)$.

Now if the target is 0-Loop then the vector of answers is $([i \in Q_1], \ldots, [i \in Q_{2t}])$ is the zero vector. If the target is 1-Loop, $\{i\}$, then the vector of answers is $([i \in Q_1], \ldots, [i \in Q_{2t}])$ which uniquely determines i. This vector contains t ones and t zeros. When the target is k-Loop, $L = \{\{i_1\}, \ldots, \{i_k\}\}$, $k > 1$, then the vector of answers is $\vee_{j=1}^{k}([i_j \in Q_1], \ldots, [i_j \in Q_{2t}])$ (bitwise or) which contains at least $t + 1$ ones. This indicates that the target is a k-Loop for some $k > 1$. ■

Theorem 11 *There is a three-round randomized Las Vegas learning algorithm for m-Graph that asks $O(m^2 \log m + m \log n)$ expected number of queries.*

In particular, when $n \geq m^m$, the algorithm asks $O(m \log n)$ queries.

Proof We run the algorithm in the proof of Theorem 9 but in the second round we use the algorithm in Lemma 10 for learning the endpoints vertices. If the algorithm fails at some round we run the deterministic algorithm that asks $O(m^3 \log n)$ queries in the third round. We now give more details.

We first partition the set of vertices into $u = 2m^4(2m-1)$ sets $V_1, \ldots, V_u$. By (1), the probability that each set contains at most one vertex of degree not equal zero is at least $1 - 1/m^3$. Assuming success, by the proof of Theorem 3, the first round finds all the edges and an expected of $1/poly(u)$ more edges that are not in the target. The edges that are not eliminated are found by the deterministic algorithm in the second round.

Suppose the first round fails to distribute the vertices of degree not equal zero in different sets. We show how to discover that in the second round. We distinguish between two cases: The first case is when there is an edge $\{u, v\}$ where $\{u, v\} \subseteq V_i$ for some i. The second case is when there is no edge between two vertices in the same set but there is at least two nodes u and v of degrees not equal to zero in the same set V_i. One of those two cases happens with probability less than $1/m^3$.

If the first case happens, the algorithm in the first round will not be able to eliminate any of the set edges $\{V_i, V_j\}$ for all V_j. This is because when the set vertex V_i is in the query the answer is always "YES". Therefore, at the end of the first stage, there will be at least $u > m$ set edges that are not eliminated. That is, all the set edges $\{V_i, V_j\}$ for all V_j. Then the algorithm knows that the first case happens and it runs the deterministic algorithm that asks $m^3 \log n$ queries in the second round.

Now suppose the second case happens. Suppose V_j contains two vertices v_1 and v_2 of non-zero degree (with no edge between them). Let $\{u_1, v_1\}$ and $\{u_2, v_2\}$ be two edges in G. We here again distinguish between two subcases. The first subcase is when $u_1, u_2 \in V_k$, $k \neq j$. The second subcase is when $u_1 \in V_{k_1}$ and $u_2 \in V_{k_2}$, and k_1, k_2, j are distinct. In the first subcase, when the algorithm considers the set V_k as one vertex and runs the algorithm in Lemma 10, the algorithm output "ERROR" and then it knows that this subcase happens. In the second subcase, when the algorithm consider the set V_{k_1} as one vertex and runs the algorithm in Lemma 10, it learns v_1 and when the algorithm consider the set V_{k_2} as one vertex it learns $v_2 \neq v_1$. Then the algorithm knows that the second subcase happens. When the second case happens the algorithm runs the deterministic algorithm that asks $m^3 \log n$ queries in the third round.

The expected number of queries is

$$\left(1 - \frac{1}{m^3}\right)(m^2 \log u + m \log n) + \frac{1}{m^3} m^3 \log n = O(m^2 \log m + m \log n).$$

■

4.5. Lower Bound for Las Vegas Randomized Algorithm

In this section we prove

Theorem 12 *Let* $\log n < m = o(\sqrt{n})$. *Any two-round Las Vegas learning algorithm for* m*-Graph must ask at least*

$$\Omega\left(\frac{m^{4/3} \log^{1/3} n}{\log^{1/3} m}\right)$$

queries on average.

Proof Let A be a two-round Las Vegas learning algorithm for m-Graph. Notice that since the algorithm succeeds with probability 1, the second round must be deterministic.

We define a distribution D over the targets as follows: we first choose $r = m/2$ (fixed) distinct vertices $V' = \{v_1, \ldots, v_r\}$. Then randomly and uniformly choose $1 \le t \le r$. Then randomly uniformly choose $s = m/2 - d$ distinct vertices $U = \{u_1, \cdots, u_s\}$ where $U \cap V' = \emptyset$ and $d = (m^{2/3} \log^{1/3} m)/(2^{10} \log^{1/3} n)$. Then randomly uniformly choose d (not necessarily distinct) vertices $W = \{w_1, \ldots, w_d\} \subseteq V \backslash (V' \cup U)$. Then define the target

$$T = \{\{v_t, v_j\} \mid j \ne t; j = 1, \ldots, r\} \cup \{\{v_t, u_j\}, \{v_t, w_k\} \mid j = 1, \ldots, s;\ k = 1, \ldots, d\}.$$

Now let $X_A(y, \mathcal{O}_I)$ be a random variable that is the number of queries asked by the algorithm A with a seed y and target I. If for any deterministic two-round algorithm B we have $\mathbf{E}_{I \in D}[X_B(\mathcal{O}_I)] \ge q$ where $X_B(\mathcal{O}_I)$ is the number of queries asked by B then $\mathbf{E}_{I \in D}[X_A(y, \mathcal{O}_I)|y] \ge q$ and then the query complexity of A is

$$\max_I \mathbf{E}_y[X_A(y, \mathcal{O}_I)] \ge \mathbf{E}_{I \in D}\mathbf{E}_y[X_A(y, \mathcal{O}_I)] = \mathbf{E}_y\mathbf{E}_{I \in D}[X_A(y, \mathcal{O}_I)|y] \ge \mathbf{E}_y[q] = q.$$

Therefore, what remains to prove is that for any two round deterministic algorithm B we have $\mathbf{E}_{I \in D}[X_B(\mathcal{O}_I)] \ge q$.

Consider the first round of B with $q = (m^{4/3} \log^{1/3} n)/\log^{1/3} m$ queries $Q_1, \ldots, Q_q$. Consider the set of queries $S_i = \{Q_i \mid Q_i \cap V' = \{v_i\}\}$ for $i = 1, \ldots, r$. Then $r\mathbf{E}_i[|S_i|] \le q$ and therefore at least $7/8$ of the $i \in [s]$ satisfies $|S_i| \le 8q/r$. Therefore, with probability at least $7/8$ we have $|S_t| \le 8q/r$. Suppose we have chosen such t and after the first round we provide the algorithm v_t. Therefore, the algorithm only needs to learn U and W. We will show next that after the first round even if we provide the learner U there will still be many vertices about which no information is known, with high probability, W.

For the ease of notation we write $Q(I) = 1$ if $Q \cap I \ne \emptyset$ and 0 otherwise. Now every query Q_i that satisfies $Q_i \cap V' \ne \{v_t\}$ will give no information about u_i or w_i. Therefore, the queries that are relevant to learning are only the queries in S_t. Let $S_t = \{Q'_1, \ldots, Q'_\ell\}$. Since $\ell = |S_t| \le 8q/r$, the algorithm can get at most $2^\ell \le 2^{8q/r}$ possible answers. For each vector of ℓ possible answers $a = (a_1, \ldots, a_\ell) \in \{0,1\}^\ell$ to the queries in S_t we define $\mathcal{I}_a = \{I \subseteq V\backslash V' \ : \ |I| = s, (Q'_1(I), \ldots, Q'_\ell(I)) = a\}$. Let $U \in \mathcal{I}_{a'}$, i.e., $(Q'_1(U), \ldots, Q'_\ell(U)) = a'$. Since

$$T := \sum_{a\in\{0,1\}^\ell} |\mathcal{I}_a| = \binom{n-r}{s}, \sum_{a,|\mathcal{I}_a|\le T/2^{\ell+3}} |\mathcal{I}_a| \le \frac{\binom{n-r}{s}}{8},$$

with probability at least $7/8$, $|\mathcal{I}_{a'}| \ge T/2^{\ell+3}$. Suppose the latter statement is true with probability 1. Let $Z_{a'} = \cup_{I\in\mathcal{I}_{a'}} I$. Notice that for every $w \in Z_{a'}$ there is $I \in \mathcal{I}_{a'}$ such that $w \in I$ and therefore (bitwise) $(Q'_1(w), \ldots, Q'_\ell(w)) \le (Q'_1(I), \ldots, Q'_\ell(I)) = a'$ which implies that if $(Q'_1(U\cup W), \ldots, Q'_\ell(U\cup W)) = a'$ no information is known about the vertices in $Z_{a'}$ after the first round. If this happen then there are $|Z_{a'}|$ vertices where no information is known about them. We next prove that with high probability $W \subseteq Z_{a'}$ and therefore $(Q'_1(U\cup W), \ldots, Q'_\ell(U\cup W)) = a'$. Then in the next round we must run a deterministic algorithm that learns the d vertices W in $Z_{a'}$. This, by Lemma 17, requires at least

$$\frac{d^2 \log |Z_{a'}|}{\log d} \tag{2}$$

queries.

Now we estimate $|Z_{a'}|$ and prove that with high probability we have $W \subseteq Z_{a'}$. We have

$$|\mathcal{I}_{a'}| \ge \frac{T}{2^{\ell+3}} \ge \frac{\binom{n-r}{s}}{2^{\ell+3}} \ge \frac{(n-r)^s}{2^{\ell+3}s!}\left(1 - \frac{s(s-1)}{2(n-r)}\right).$$

On the other hand

$$|\mathcal{I}_{a'}| \le \binom{|Z_{a'}|}{s} \le \frac{|Z_{a'}|^s}{s!}.$$

Therefore,

$$|Z_{a'}| \ge \frac{n-r}{2^{(\ell+3)/s}}\left(1 - \frac{s(s-1)}{2(n-r)}\right)^{1/s} = \frac{n}{2^{(\ell+3)/s}}(1-o(n)).$$

The probability that $|W| = d$ and $W \subseteq Z_{a'}$ is at least

$$\left(1 - \frac{d(d-1)}{n-r-s}\right)\left(\frac{|Z_{a'}| - s}{n-s-r}\right)^d \ge \frac{1}{2^{(\ell+3)d/s}}(1-o(1)) \ge \frac{7}{8}.$$

Therefore, if we provide the algorithm U after the first round, with probability at least $7/8$ no information is known about W. All the above is true with probability at least $1/2$. By (2), in the second round the algorithm needs to ask at least

$$\frac{d^2 \log |Z_{a'}|}{\log d} = \Omega\left(\frac{m^{4/3}\log^{1/3} n}{\log^{1/3} m}\right)$$

queries. ■

5. Deterministic Algorithms when m is Known

In this section we study deterministic algorithms for learning m-graph when m is known. It is known that any non-adaptive deterministic algorithm must ask at least $\Omega((m^3\log n)/\log m)$ queries, (D'yachkov and Rykov, 1983), and there is a polynomial time non-adaptive algorithm that asks $O(m^3\log n)$ queries, (Bshouty, 2015). In this section we prove (non-constructively) that there is a two-round algorithm that asks $O(m^2\log n)$ queries and then give the lower bound $\Omega((m^2\log n)/\log m)$ for the number of queries. Finally, we give a four-round deterministic algorithm that asks $O(m^{2+\epsilon}\log n)$ queries for any constant ϵ.

5.1. Nonconstructive Upper Bound for Two-Round Deterministic Algorithm

We first prove the following three results.

Lemma 13 *Let $E=\{e_1,\ldots,e_t\}$ be a set of edges. Let $Q\subseteq V$ be a random query where for each vertex $i\in V$, i is included in Q independently with probability p. Then*

$$\Pr[(\exists i\in[t])e_i\subseteq Q]\ge p\cdot(1-(1-p)^t).$$

Proof Define the event $A_{k,j}=[(\exists i\in[k,j])e_i\subseteq Q]$. We prove the result by induction on t. For $t=1$ we have one edge e and $\Pr[e\subseteq Q]=p^2\ge p\cdot(1-(1-p)^1)$. Let $u\in e_1$. Assume w.l.o.g that $u\in e_i$ for $i\in[\ell]$ and $u\not\in e_i$ for $i\in[\ell+1,t]$. Define the event $B=[(\exists i\in[\ell])(e_i\backslash\{u\})\subseteq Q]$. Then, by the induction hypothesis,

$$\begin{aligned}
\Pr[A_{1,t}] &= \Pr[u\in Q]\Pr[A_{1,t}|u\in Q]+\Pr[u\not\in Q]\Pr[A_{1,t}|u\not\in Q]\\
&= p\Pr[B\vee A_{\ell+1,t}]+(1-p)\Pr[A_{\ell+1,t}]\\
&\ge p\Pr[B]+(1-p)p(1-(1-p)^{t-\ell})\\
&= p(1-(1-p)^{\ell})+(1-p)p(1-(1-p)^{t-\ell})\\
&= p(1+(1-p)(1-(1-p)^{\ell-1})-(1-p)^{t-\ell+1})\\
&\ge p(1+(1-p)^{t-\ell+1}(1-(1-p)^{\ell-1})-(1-p)^{t-\ell+1})\\
&= p\cdot(1-(1-p)^t).
\end{aligned}$$

■

Lemma 14 *Let $E=\{e_1,\ldots,e_r\}$ and $E'=\{e'_1,\ldots,e'_t\}$ be two disjoint sets of edges. Let $Q\subseteq V$ be a random query where for each vertex $i\in V$, i is included in Q independently with probability p. Then*

$$\Pr[(\forall i\in[r])\ e_i\not\subseteq Q\mid(\exists j\in[t])e'_j\subseteq Q]\ge(1-p)^r.$$

Proof The proof is by induction on the number of edges in E. Define the events $B=[(\forall i\in[r])\ e_i\not\subseteq Q]$ and $A=[(\exists j\in[t])e'_i\subseteq Q]$. Assume w.l.o.g $u\in e_i$ for $i\in[\ell]$, $u\not\in e_i$ for $i\in[\ell+1,r]$, $u\in e'_i$ for $i\in[\ell']$ and $u\not\in e'_i$ for $i\in[\ell'+1,t]$. Here, $\ell\ge 1$ and $\ell'\ge 0$. Then $\Pr[B|A]\ge\Pr[u\not\in Q]\cdot\Pr[B|A \text{ and } u\not\in Q]=(1-p)\cdot\Pr[B|A\text{ and }u\not\in Q]$. If $u\not\in Q$ then $e_i\not\subseteq Q$ for all $i\in[\ell]$ and then $B=B':=[(\forall j\in[\ell+1,r])e_j\not\subseteq Q]$. Also, $e'_i\not\subseteq Q$ for all $i\in[\ell']$ and the event $[A\text{ and }u\not\in Q]$ is equivalent to $[(A'\text{ and }u\not\in Q)]$ where $A':=[\exists i\in[\ell'+1,t])\ e'_i\subseteq Q]$. Therefore $\Pr[B|A]=\Pr[B'|A']$. By the induction hypothesis $\Pr[B'|A']\ge(1-p)^{r-\ell}$ and therefore $\Pr[B|A]\ge(1-p)^{r-\ell+1}\ge(1-p)^r$. ■

Lemma 15 *Let $G = (V, E)$ be the target graph with n vertices and m edges. Let $Q \subseteq V$ be a random query where for each vertex $i \in V$, i is included in Q independently with probability p. Let $e'_1, \ldots, e'_t \notin E$. Suppose the probability that there is $i \in [t]$ such that $e'_i \subseteq Q$ is at least q. Then*

$$\Pr[(\exists i \in [t])\ e'_i \subseteq Q \textit{ and } \mathcal{O}_G(Q) = \textit{``NO''}] \ge q(1-p)^m.$$

In particular,

$$\Pr[(\exists i \in [t])\ e'_i \subseteq Q \textit{ and } \mathcal{O}_G(Q) = \textit{``NO''}] \ge p \cdot (1-(1-p)^t)(1-p)^m.$$

Proof Let $A := [(\exists i \in [t])\ e'_i \subseteq Q]$. Let $E = \{e_1, \ldots, e_m\}$. Then the event $\mathcal{O}_G(Q) =$ "NO" is equivalent to the even $B := [(\forall j \in [m]) e_j \not\subseteq Q]$. Now by Lemma 13 and 14 we have

$$\Pr[A \text{ and } B] \;=\; \Pr[A] \cdot \Pr[B|A] \ge q(1-p)^m \ge p \cdot (1-(1-p)^t)(1-p)^m.$$

■

We are now ready to prove our main result

Theorem 16
There is a two-round deterministic learning algorithm for m-Graph that asks $t = O(m^2 \log n)$ queries.

Proof We will first show that there is a set of t queries $Q_1, \ldots, Q_t$ that satisfies: For every graph $G = ([n], E = \{e_1, \ldots, e_m\})$ with m edges and for every set of m edges $E' = \{e'_1, \ldots, e'_m\}$ not in G, there is a query Q_j such that $\mathcal{O}_G(Q_j) =$ "NO" and $(\exists i \in [m])\ e'_i \subseteq Q_j$.

Choose $Q_1, \ldots, Q_t$ where each $Q_j \subseteq V$ is a random query where for each vertex $i \in [n]$, i is included in Q independently with probability $1/m$. By Lemma 15, probability that above event is not true for some graph G is at most

$$\binom{\binom{n}{2}}{m}^2 \left(1 - \frac{1}{m}\left(1 - \left(1 - \frac{1}{m}\right)^m\right)\left(1 - \frac{1}{m}\right)^m\right)^t.$$

This is less than 1 for $t = O(m^2 \log n)$. Therefore there are such queries.

In the first round, the algorithm defines $E(H) = \{\{u, v\} | u, v \in V, u \neq v\}$ and asks all the queries $Q_1, \ldots, Q_t$. For each query Q_j with answer "NO" it eliminates from $E(H)$ all the pairs $\{u', v'\}$ where $u', v' \in Q_j$.

We now show that $E(H)$ contains at most $2m$ edges. Assume for the contrary that $E(H)$ contains $2m + 1$ edges. Take $E' \subseteq E(H) \backslash E$ of size m. There is a query Q_j such that $\mathcal{O}_G(Q_j) =$ "NO" and $(\exists e \in E')\ e \subseteq Q_j$. This gives a contradiction. Now in the second round the algorithm asks a query for each edge in $E(H)$. ■

5.2. Lower Bound for Two-Round Deterministic Algorithm

In this subsection we give a lower bound. We first give two known facts from (D'yachkov and Rykov, 1983; Füredi, 1996).

Lemma 17 *Any deterministic nonadaptive learning algorithm for m-Loop must ask at least $\Omega((m^2/\log m)\log n)$ queries.*

Lemma 18 *Let $\mathcal{Q}$ be a set of queries that satisfies: for every set $S \subset [n]$ of size m and every $i \in S$ there is a query $Q \in \mathcal{Q}$ such that $S \cap Q = \{i\}$. Then $|\mathcal{Q}| = \Omega((m^2/\log m)\log n)$.*

We now prove

Theorem 19 *Any two-round deterministic learning algorithm for m-Graph must ask at least $\Omega((m^2/\log m)\log n)$ queries.*

Proof Suppose there is a two-round deterministic learning algorithm for m-Graph that asks $t = o((m^2/\log m)\log n)$ queries. Let $Q_1, \ldots, Q_t$ be the queries in the first round. Since $t = o((m^2/\log m)\log n)$, by Lemma 18, there must be a set of $\lfloor m/2 \rfloor$ elements $S \subseteq [n]$ and $i \in S$ such that no query Q_j satisfies $S \cap Q_j = \{i\}$. The adversary defines a set of $\lfloor m/2 \rfloor$ edges $E' = \{\{i,j\} \mid j \in S\backslash\{i\}\}$. The other $\lceil m/2 \rceil$ edges $E'' = \{\{i,j\} \mid j \in S', S' \subseteq [n]\backslash S\}$ will be determined in the second round. The answers of the queries in the first round are determined only by the edges E'. That is, if $|Q_j \cap S| > 1$ and $i \in Q_j$ then the answer is "YES", and if $i \notin Q_j$ then the answer is "NO". After the first round no information is known about S'.

In the second round we need to learn S' from queries Q that contain i. Otherwise, the answer is "NO"and no information is gained about S'. When $i \in Q$ then the problem is equivalent to learning $\lceil m/2 \rceil$-Loop, and by Lemma 17 we must ask at least $\Omega((m^2/\log m)\log n)$ queries. ■

5.3. Deterministic Five-Round Algorithm

In this section we give a deterministic five-round algorithm that asks $O(m^{2+\epsilon}\log n)$ queries.

In the first round the algorithm finds a partition of $[n]$ to $w = O(m)$ sets $S_1, \ldots, S_w$ where no edge of the target has both endpoint vertices in the same set. In the second round it learns the pairs of sets $\{S_i, S_j\}$ for which there is an edge with one endpoint in S_i and the other in S_j. In the third and fourth rounds it finds the vertices in each set that are endpoints of some edge. Then in the fifth round it learns the edges.

For the first round we use the following result from (Bshouty, 2015).

Lemma 20 *There is a linear time algorithm that for every n and m constructs a $t \times n$-matrix, $t = O(n\log m)$, M with entries in $[w]$, $w = O(m)$, with the following property: Every two columns in M are equal in at most $t/(2m)$ entries.*

We now prove

Lemma 21 *Let M be the matrix in Lemma 20. There is a row vector $u \in [w]^n$ in M such that in the partition $\{S_{u,1}, \ldots, S_{u,w}\}$ of $[n]$ where $S_{u,i} = \{j|u_j = i\}$ no edge of the target has both endpoint vertices in the same set.*

Proof Let $e_i = \{u_i, w_i\}$, $i = 1, \dots, m$ be the edges of the target. By Lemma 20, there is at most $t/(2m)$ entries in columns u_i and w_i in the matrix that are equal. Therefore, there is at least one entry (actually, at least $t/2$ entries) such that for all $i = 1, \dots, m$, columns u_i and w_i in the matrix are not equal in that entry. This implies the result. ■

To know this row, for each row u in M and for each $j \in [w]$, the algorithm asks the query $\mathcal{O}_G(S_{u,j})$. Obviously, if no edge of the target has both endpoint vertices in the same set then the answers are zeros for all $j \in [w]$. The number of queries in this round is $wt = O(m^2 \log n)$.

After the first round we have a partition of $[n]$ to $w = O(m)$ sets $S_1, \dots, S_w$ where no edge of the target has both endpoint vertices in the same set. In the second round we ask the query $\mathcal{O}_G(S_i \cup S_j)$ for all $i \neq j$. If $\mathcal{O}_G(S_i \cup S_j) = 1$ then the algorithm knows that there is an edge with one endpoint in S_i and the other in S_j. Since $w = O(m)$, this takes $O(m^2)$ queries. If $\mathcal{O}_G(S_i \cup S_j) = 1$ then we call $\{S_i, S_j\}$ a *set edge*. Obviously, there are at most m set edges.

In the third and fourth rounds, the algorithm runs Cheraghchi's two-round algorithm, (Cheraghchi, 2013), to find the vertices in each S_i that are endpoints of some edge. This algorithm is a two-round algorithm for m-Loop. It asks $O(m^{1+\epsilon} \log n)$ queries for any constant ϵ. We use Cheraghchi's algorithm as follows: if $\{S_i, S_j\}$ is a set edge then each query $S_i \cup (Q \cap S_j)$ for the graph is the query $Q \cap S_j$ for the m-Loop that contains the vertices in S_j that are endpoints of the edges. Therefore, running Cheraghchi's two-round algorithm for each set edge $\{S_i, S_j\}$ gives all the vertices that are endpoints of some edge in the cut $(S_i; S_j)$. Since the number of edge sets is at most m this takes $O(m^{2+\epsilon} \log m)$ queries.

In the fifth stage the algorithm exhaustively asks a query about each possible pair. That is, if for the edge set $\{S_i, S_j\}$ we have learned that $V_1 \subseteq S_i$ and $V_2 \subseteq S_j$ are the endpoints of some edge in the cut $(S_i; S_j)$, then we ask the queries $\mathcal{O}_G(\{v_1, v_2\})$ for each $v_1 \in V_1$ and $v_2 \in V_2$. This takes at most $m^2/2 = O(m^2)$ queries.

The above algorithm implies

Theorem 22 *There is a five-round deterministic learning algorithm for m-Graph that asks $O(m^2 \log n)$ queries.*

6. Unknown m - Upper Bounds

In this section we prove two results when m is not known to the learner.

Here $\log^{[0]} n = n$ and $\log^{[k]} n = \log \log^{[k-1]} n$. When we say w.h.p (with high probability) we mean with probability at least $1 - 1/\text{poly}(n)$.

Theorem 23 *For every constant $k > 1$ there is an $O(1)$-round Las Vegas randomized learning algorithm for m-Graph that asks $O(m \log n + \sqrt{m}(\log^{[k]} n) \log n)$ queries.*

Theorem 24 *There is a $(\log^* n)$-round Las Vegas randomized learning algorithm for m-Graph that asks $O(m \log n)$ queries.*

We recall Chernoff Bound

Lemma 25 (Chernoff Bound). *Let $X_1, \dots, X_t$ be independent random variables that takes values in $\{0, 1\}$. Let $X = (X_1 + \cdots + X_t)/t$ and $\mu = \mathbf{E}[X]$. Then for any $0 < \delta < 1$,*

$$\Pr[|X - \mu| \geq \delta\mu] \leq 2e^{-\delta^2 \mu t/3},$$

and for any $\delta \geq 1$

$$\Pr[X \geq (1+\delta)\mu] \leq 2e^{-\delta\mu t/3}.$$

Let $Q \subseteq V$ be a p-random query. Let $N_G(p)$ be the probability that $\mathcal{O}_G(Q) = 0$. It is easy to see that (see (Angluin and Chen, 2008))

$$1 - mp^2 \leq N_G(p) \leq 1 - p^2. \tag{3}$$

Consider p_i-random queries Q_i for $i = 1, 2$. Then $Q_1 \cup Q_2$ is a $(p_1 + p_2 - p_1 p_2)$-random query. Now

$$\begin{aligned} N_G(p_1 + p_2 - p_1 p_2) &= \Pr[\mathcal{O}_G(Q_1 \cup Q_2) = 0] \\ &\leq \Pr[\mathcal{O}_G(Q_1) = 0 \text{ and } \mathcal{O}_G(Q_2) = 0] \\ &= N_G(p_1) \cdot N_G(p_2). \end{aligned}$$

It is shown in (Angluin and Chen, 2008) that for any G the function $N_G(x)$ is continuous monotonic decreasing function. Therefore

Lemma 26 *For any* $0 \leq q_1 < q_2 \leq 1$ *we have* $N_G(q_1) > N_G(q_2)$ *and for any* $0 \leq p_1 + p_2 \leq 1$ *we have*

$$N_G(p_1 + p_2) < N_G(p_1 + p_2 - p_1 p_2) \leq N_G(p_1) \cdot N_G(p_2).$$

In particular, for any integer $k \geq 2$

$$N_G(kp) < N_G(p)^k.$$

Let p_* be such that $N_G(p_*) = 1/2$. By (3) we have

$$\frac{1}{\sqrt{2m}} \leq p_* \leq \frac{1}{\sqrt{2}}. \tag{4}$$

Our first goal is to estimate p_*. Consider the following procedure

```
Estimate($\mathcal{O}_G, M$)
For each $p_i = 1/2^i$, $i = 0, 1, 2, \cdots$, such that $2^i \leq 2^{2.5}\sqrt{M}$
    For $t = \Theta(\log n)$ independent $p_i$-queries $Q_{i,1}, \ldots, Q_{i,t}$ do:
        $q_i = (\mathcal{O}_G(Q_{i,1}) + \cdots + \mathcal{O}_G(Q_{i,t}))/t$.
Choose the first $p' := p_{i_0}/2$ such that $1 - q_{i_0} > 1/2$.
If no such $i_0$ exists then output("$m > M$").
Otherwise output($p'$) \$\star$ $p_* \geq p' \geq p_*/8$ $\star$\.
```

We now show

Lemma 27 *Let* $M \in [n]$ *be any integer. The procedure* **Estimate**($\mathcal{O}_G, M$)*, w.h.p, asks* $\Theta((\log M)(\log n))$ *queries and either outputs* p' *such that* $p_* \geq p' \geq p_*/8$ *or proclaims that* $m > M$.

Proof Let k be such that $p_*/2 < p_k \le p_*$. Then $p_{k-2} > 2p_*$ and therefore, by Lemma 26, for all $i \le k-2$, $N_G(p_i) \le N_G(p_{k-2}) < N_G(2p_*) < N_G(p_*)^2 \le 1/4$. Therefore, by Chernoff bound ($X_j = 1 - \mathcal{O}_G(Q_{i,j})$, $t = \Theta(\log n)$, $\mu = N_G(p_i) \le 1/4$ and $\delta\mu = 1/4$)

$$\Pr[i_0 \le k-2] \le \Pr[(\exists i \le k-2)\ 1 - q_i > 1/2] \le 1/\mathrm{poly}(n).$$

Therefore, w.h.p $i_0 \ge k-1$. Also $p_{k+1} = p_k/2 \le p_*/2$ and therefore, by Lemma 26, $N_G(p_{k+1}) > N_G(p_*/2) > N_G(p_*)^{1/2} > 0.7$. Therefore, by Chernoff bound, ($X_j = 1 - \mathcal{O}_G(Q_{k+1,j})$, $t = \Theta(\log n)$, $\mu = N_G(p_{k+1}) \ge 0.7$ and $\delta\mu = 0.2$)

$$\Pr[i_0 \ge k+2] \le \Pr[1 - q_{k+1} < 1/2] \le 1/\mathrm{poly}(n).$$

Therefore, w.h.p $k-1 \le i_0 \le k+1$ and then $2p_* \ge p_{k-1} \ge p_{i_0} \ge p_{k+1} \ge p_*/4$. Therefore w.h.p

$$p_* \ge p' = p_{i_0}/2 \ge p_*/8.$$

Let i' be such that $2^{i'} \le 2^{2.5}\sqrt{M} < 2^{i'+1}$. If i_0 does not exist then w.h.p $i' \le k$ and therefore by (4),

$$2^{2.5}\sqrt{M} < 2^{i'+1} \le 2^{k+1} = \frac{1}{p_{k+1}} \le \frac{4}{p_*} = 4\sqrt{2m}$$

and then $m > M$. ■

The following procedure estimates p_* in k rounds

```
    k-Estimate($\mathcal{O}_G$)
1.  For $j = k-1$ downto 0
2.      $M_j \leftarrow (\log^{[j]} n)^2$.
3.      Estimate($\mathcal{O}_G, M_j$)
4.      If the output is $p'$ then Goto 6.
5.  EndFor.
6.  Output($p'$) $\backslash\star\ p_* \ge p' \ge p_*/8\ \star\backslash$.
```

We now show

Lemma 28 *The procedure k-**Estimate**($\mathcal{O}_G$) runs in k rounds and w.h.p asks*

$$O((\log^{[k]} n)(\log n) + m\log n)$$

queries and outputs p' such that $p_* \ge p' \ge p_*/8$.

Proof For $j = 0$ we have $M_0 = (\log^{[0]} n)^2 = n^2$ and since $m < M_0$ the algorithm must stops and output such p'. It remains to prove the query complexity.

If **Estimate**($\mathcal{O}_G, M_{k-1}$) returns p' then, by Lemma 27, the algorithm w.h.p asks

$$\Theta((\log M_{k-1})(\log n)) = \Theta((\log^{[k]} n)(\log n))$$

queries. Otherwise, $m > M_{k-1}$. Let j be such that $M_{j+1} < m \le M_j$. Then p' is returned by some **Estimate**($\mathcal{O}_G, M_{j'}$) where $j' \ge j$. Therefore, by Lemma 27, w.h.p the number of queries is O of

$$\sum_{i=j}^{k} (\log M_i)(\log n) \le 2(\log M_j)(\log n) = 4\sqrt{M_{j+1}} \log n \le 4m \log n.$$

■

In particular,

Corollary 29 *The procedure* $\log^* n$-**Estimate**($\mathcal{O}_G$) *runs in* $\log^* n$ *rounds, w.h.p asks* $O(m \log n)$ *queries and outputs* p' *such that* $p_* \ge p' \ge p_*/8$.

The following is Lemma 4.1 in (Angluin and Chen, 2008)

Lemma 30 *Suppose* I *is an independent set in* G *and let* $\Gamma(I)$ *be the set of neighbors of the vertices in* I. *For a* p*-random query* Q *we have*

$$\Pr[\mathcal{O}_G(Q) = 0 \mid I \subseteq Q] \ge (1-p)^{|\Gamma(I)|} \cdot N_G(p) \ge (1-p)^{|\Gamma(I)|} \cdot (1 - mp^2).$$

The following is Lemma 5.2 in (Angluin and Chen, 2008). We give the proof for completeness

Lemma 31 *Let* $p_* \ge p' \ge p_*/8$. *For any constant* c' *there is a constant* c *such that: If* $\{u, v\} \notin E(G)$ *and* $\deg_G(u) + \deg_G(v) \le c'/p'$ *then for* p'*-random query* Q

$$\Pr[\mathcal{O}_G(Q) = 0 \text{ and } \{u, v\} \subseteq Q] \ge cp'^2.$$

Proof First notice that since $p' \le p_* \le 1/\sqrt{2}$ we have $(1-p')^{1/p'} \ge (1-p_*)^{1/p_*} \ge 0.17$. By Lemma 30, the probability that $\mathcal{O}_G(Q) = 0$ and $\{u, v\} \subseteq Q$ is equal to

$$\begin{aligned} \Pr[\{u,v\} \subseteq Q] \cdot \Pr[\mathcal{O}_G(Q) = 0 \mid \{u,v\} \subseteq Q] &\ge p'^2 (1-p')^{c'/p'} N_G(p') \\ &\ge p'^2 (0.17)^{c'} N_G(p_*) = cp'^2. \end{aligned}$$

■

We use Lemma 31 for the following

```
Split($\mathcal{O}_G$)
$E(H) \leftarrow \{\{u,v\} \mid u, v \in V, u \ne v\}$.
Choose $t = \Theta((1/p')^2 \log n)$ $p'$-random queries $Q_1, \ldots, Q_t$
For each query $Q_i$.
    If $\mathcal{O}_G(Q_i) = 0$ then
        For every $u, v \in Q_i$ do $E(H) \leftarrow E(H) \backslash \{\{u,v\}\}$
EndFor.
$V_1(H) \leftarrow \{u \in V \mid \deg_H(u) \ge 3/p'\}$, $V_2(H) = V \backslash V_1$
```

Lemma 32 *Let* $p_* \ge p' \ge p_*/8$. *The procedure* **Split**($\mathcal{O}_G$) *asks at most* $O(m \log n)$ *queries and w.h.p the following hold:*

1. *For all $u, v \in V_2(H)$, $\{u, v\} \in E(H)$ if and only if $\{u, v\} \in E(G)$.*

   In particular,

2. *If $\deg_G(u) \le 1/p'$ and $\{u, v\} \in E(H)\backslash E(G)$ then $\deg_G(v) > 1/p'$.*

Proof By (4), the number of queries is $O((1/p'^2)\log n) = O((1/p_*^2)\log n) = O(m \log n)$.

If $\mathcal{O}_G(Q_i) = 0$ and $u, v \in Q_i$ then $\{u, v\}$ is not an edge in G. Therefore, procedure **Split**($\mathcal{O}_G$) only removes edges in $E(H)$ that are not in $E(G)$. Therefore, if $\{u, v\} \in E(G)$ then $\{u, v\} \in E(H)$. Suppose $u, v \in V_2(H)$ and $\{u, v\} \notin E(G)$. Since $\deg_H(u), \deg_H(v) < 3/p'$ we also have $\deg_G(u), \deg_G(v) < 3/p'$. Therefore, by Lemma 31,

$$\begin{aligned} \Pr[\{u, v\} \in E(H)] &= \Pr[(\forall i)\ \mathcal{O}_G(Q_i) = 1 \text{ or } \{u, v\} \not\subseteq Q] \\ &\le \left(1 - cp'^2\right)^t = \frac{1}{\text{poly}(n)}. \end{aligned}$$

■

The following is Lemma 5.4 in (Angluin and Chen, 2008). We give the proof for completeness

Lemma 33 *Let $p_* \ge p' \ge p_*/8$. There are at most $2/p' \le 16/p_* \le 16\sqrt{2m}$ vertices in G that have degree more than $1/p'$.*

Proof Suppose h vertices, $v_1, \ldots, v_h$, have degree more than $1/p'$. Let Q be a p'-random query. Then

$$\begin{aligned} \frac{1}{2} &= N_G(p_*) \le N_G(p') = \Pr[\mathcal{O}_G(Q) = 0] \\ &= \Pr[\mathcal{O}_G(Q) = 0 \mid (\exists i) v_i \in Q] \cdot \Pr[(\exists i) v_i \in Q] \\ &\quad + \Pr[\mathcal{O}_G(Q) = 0 \mid (\forall i) v_i \notin Q] \cdot \Pr[(\forall i) v_i \notin Q] \\ &\le (1 - p')^{1/p'}(1 - (1 - p')^h) + (1 - p')^h \\ &\le e^{-1} + (1 - e^{-1})(1 - p')^h \le e^{-1} + (1 - e^{-1})e^{-p'h}. \end{aligned}$$

Therefore $h \le 2/p'$. ■

Lemma 34 *Let $V_3 = \{v \in G \mid \deg_G(v) \ge 1/p'\}$. Then w.h.p $V_1(H) \subseteq V_3$.*

In particular, w.h.p $|V_1(H)| \le |V_3| \le 1/p' \le 8\sqrt{2m}$.

Proof If $v \notin V_3$ then $\deg_G(v) < 1/p'$. Now if $\{u, v\}$ is an edge in H but not in G then by Lemma 32, $\deg(u) \ge 1/p'$. Since by Lemma 33 the number of vertices of degree more than $1/p'$ is less than $2/p'$ we have $\deg_H(v) < \deg_G(v) + 2/p' \le 3/p'$. Therefore $v \notin V_1(H)$. ■

Therefore, it remains to learn the edges of the vertices in $V_1(H)$. Since $V_1(H) \subseteq V_3$ we have that

Lemma 35 *For every $v \in V_1(H)$, $d_v := \deg_G(v) \ge 1/p'$.*

This is one of the main properties that we will use in the sequel.

Let Q be a p-random query and $u \in V$ be a vertex of degree $d_u \geq 1/p'$. Let $N_{u,G}(p)$ be the probability that $\mathcal{O}_G(Q \cup \{u\}) = 0$. As before, $N_{u,G}(p)$ is monotonically decreasing and $N_{u,G}(kp) \leq N_{u,G}(p)^k$. Let p_u be the probability such that

$$N_{u,G}(p_u) = e^{-1}. \tag{5}$$

Next, we show that estimating p_u implies estimating d_u. Since

$$e^{-1} = N_{u,G}(p_u) \leq (1 - p_u)^{d_u} \leq e^{-p_u d_u}$$

we have $p_u \leq 1/d_u \leq p'$. Notice that $N_G(p_u) \geq N_G(p') \geq N_G(p_*) = 1/2$. Now by Lemma 30,

$$e^{-1} = N_{u,G}(p_u) \geq (1 - p_u)^{d_u} N_G(p_u) \geq \frac{1}{2}(1 - d_u p_u).$$

Therefore

$$1 - \frac{2}{e} \leq p_u d_u \leq 1. \tag{6}$$

Lemma 36 *Let $M \in [0, n]$. The procedure* **EstimateDegree(**$\mathcal{O}_G, u, M$**)** *asks $\Theta((\log M)(\log n))$ queries and either output $p'_u \geq 1/(2M)$ such that $p_u \geq p'_u \geq p_u/8$ and then $d_u \leq 32M$ or proclaims that $d_u > M$.*

```
EstimateDegree($\mathcal{O}_G, u, M$)
For each $p_{u,i} = 1/2^i$ such that $2^i \leq 16M$
    for $t = \Theta(\log n)$ independent $p_{u,i}$-queries $Q_{i,1}, \ldots, Q_{i,t}$ do:
        $q_{u,i} = (\mathcal{O}_G(Q_{i,1} \cup \{u\}) + \cdots + \mathcal{O}_G(Q_{i,t} \cup \{u\}))/t$.
Choose the first $p'_u := p_{i_{u,0}}/2$ such that $1 - q_{u,i_{u,0}} > 1/e$.
If no such $i_{u,0}$ exists then output("$d_u > M$").
Otherwise output($p'_u$) \ $\star$ $p_u \geq p'_u \geq p_u/8$ $\star$ \.
```

Proof Let k be such that $p_u/2 < p_{u,k} \leq p_u$. Then $p_{u,k-2} > 2p_u$ and therefore for all $i \leq k-2$, $N_{u,G}(p_{u,i}) \leq N_{u,G}(p_{u,k-2}) < N_{u,G}(2p_u) < N_{u,G}(p_u)^2 \leq 1/e^2$. Therefore, ($\mathbf{E}[1 - q_{u,i}] = N_{u,G}(p_{u,i}) \leq 1/e^2$)

$$\Pr[i_{u,0} \leq k-2] \leq \Pr[(\exists i \leq k-2) 1 - q_{u,i} > 1/e] \leq 1/\mathrm{poly}(n).$$

Therefore, w.h.p, $i_{u,0} \geq k-1$. Now, $p_{u,k+1} = p_{u,k}/2 \leq p_u/2$ and therefore $N_{u,G}(p_{u,k+1}) > N_{u,G}(p_u/2) > N_{u,G}(p_u)^{1/2} > 0.6$. Therefore

$$\Pr[i_{u,0} \geq k+2] \leq \Pr[1 - q_{u,k+1} < 1/e] \leq 1/\mathrm{poly}(n).$$

Therefore, w.h.p $k-1 \leq i_{u,0} \leq k+1$ and then $2p_u \geq p_{u,k-1} \geq p_{u,i_{u,0}} \geq p_{u,k+1} \geq p_u/4$. Therefore w.h.p

$$p_u \geq p'_u = p_{u,i_{u,0}}/2 \geq p_u/8.$$

Now, by (6) and step 1 in the procedure,

$$d_u \le \frac{1}{p_u} \le \frac{1}{p'_u} = \frac{2}{p_{u,i_{u,0}}} \le 32M.$$

Let i' be such that $2^{i'} \le 16M$ and $2^{i'+1} > 16M$. If no such $i_{u,0}$ exists then $i' \le k$ and therefore by (6),

$$16M < 2^{i'+1} \le 2^{k+1} = \frac{1}{p_{u,k+1}} \le \frac{4}{p_u} \le \frac{4d_u}{(1-2/e)}$$

and then $m \ge d_u > M$. ■

```
    k-EstimateDegree($\mathcal{O}_G, u$)
1.  For $j = k-1$ downto 0
2.      $M_j \leftarrow (\log^{[j]} n)^2$.
3.      EstimateDegree($\mathcal{O}_G, u, M_j$)
4.      If the output is $p'_u$ then Goto 6.
5.  EndFor.
6.  Output($p'_u$) \* $p_u \ge p'_u \ge p_u/8$ *\.
```

We now show

Lemma 37 *The procedure k-**EstimateDegree**($\mathcal{O}_G, u$) runs in k rounds, with probability $1 - 1/\mathrm{poly}(n)$, asks*

$$O((\log^{[k]} n)(\log n) + \sqrt{m}\log n)$$

queries and outputs p'_u such that $p_u \ge p'_u \ge p_u/8$.

Proof For $j = 0$ we have $M_0 = (\log^{[0]} n)^2 = n^2$ and since $m < M_0$, by Lemma 36, the algorithm must stop and output such p'_u. It remains to prove the query complexity.

If **EstimateDegree**($\mathcal{O}_G, u, M_{k-1}$) returns p'_u then, by Lemma 36, the algorithm asks

$$\Theta((\log M_{k-1})(\log n)) = \Theta((\log^{[k]} n)(\log n))$$

queries. Otherwise, $m \ge d_u > M_{k-1}$. Let j be such that $M_{j+1} < m \le M_j$. Then p'_u is returned by some **EstimateDegree**($\mathcal{O}_G, u, M_{j'}$) where $j' \ge j$. Therefore the number of queries is O of

$$\sum_{i=j'}^{k} (\log M_i)(\log n) \le 2(\log M_j)(\log n) = 4\sqrt{M_{j+1}}\log n \le 4\sqrt{m}\log n.$$

■

In particular,

Corollary 38 *The procedure $\log^* n$-**EstimateDegree**($\mathcal{O}_G$) runs in $\log^* n$ rounds, with probability $1 - 1/\mathrm{poly}(n)$, asks $O(\sqrt{m}\log n)$ queries and outputs p'_u such that $p_u \ge p'_u \ge p_u/8$.*

In particular

Lemma 39 *The degrees of all the vertices u in $V_1(H)$ can be estimated with probability $1 - 1/\mathrm{poly}(n)$ by $1/p'_u$ that falls in the range $[d_u, 31d_u]$ with*

1. *k-round algorithm that asks at most $O((\log^{[k]} n)\sqrt{m}\log n + m\log n)$ queries.*
2. *$\log^* n$-round algorithm that asks at most $O(m\log n)$ queries.*

Proof By Lemma 34, w.h.p $|V_1(H)| = O(\sqrt{m})$ and by (6) and Lemma 37, for every $u \in V_1(H)$, the value of $1/p'_u$ is bounded by

$$d_u \le \frac{1}{p_u} \le \frac{1}{p'_u} \le \frac{8}{p_u} \le \frac{8d_u}{1 - 2/e} \le 31d_u.$$

■

After estimating the degree the algorithm finds the edges of each $v \in V_1(H)$ in G.

We first have (the proof is the same as of Lemma 30)

Lemma 40 *Suppose I is an independent set in G and let $\Gamma(I)$ be the set of neighbors of the vertices in I. Let u be a vertex in G such that $u \notin \Gamma(I)$. For a p-random query Q we have*

$$\Pr[\mathcal{O}_G(Q \cup \{u\}) = 0 \mid I \subseteq Q] \ge (1-p)^{|\Gamma(I)|} \cdot N_{u,G}(p).$$

We now prove

Lemma 41 *The edges of each $v \in V_1(H)$ can be learned with probability $1 - 1/\mathrm{poly}(n)$ in $O(m\log n)$ queries*

Proof Let $u \in V_1(H)$. Let Q be a p'_u-random query and $\{u, v\} \notin E$. By Lemma 40, Lemma 37, (5)

$$\begin{aligned}
\Pr[\mathcal{O}_G(Q \cup \{u\}) = 0 \text{ and } v \in Q] &= \Pr[v \in Q] \cdot \Pr[\mathcal{O}_G(Q \cup \{u\}) = 0 \mid v \in Q] \\
&= p'_u \cdot (1 - p'_u)^{1/p'_u} N_{u,G}(p'_u) \\
&\ge \frac{p_u}{8}(1 - p'_u)^{1/p'_u} N_{u,G}(p_u) \\
&\ge \frac{1 - 2/e}{8d_u}\frac{1}{4}e^{-1} \\
&\ge \frac{0.003}{d_u}.
\end{aligned}$$

Therefore each p'_u-random query discover that $\{u, v\}$ is not an edge with probability at least $0.003/d_u$. Therefore $O(d_u \log n)$ queries are enough to find all the neighbours of u with probability $1 - 1/\mathrm{poly}(n)$. The total number of queries is O of

$$\sum_{u \in V_1(H)} d_u \log n \le 2m\log n.$$

■

The following is the procedure

```
FindEdges($\mathcal{O}_G$,$V_1(H)$)
For each $u \in V_1(H)$ do.
    Choose $t = \Theta((1/p'_u)\log n)$ $p'_u$-random queries $Q_1, \ldots, Q_t$
    For each query $Q_i$.
        If $\mathcal{O}_G(Q_i \cup \{u\}) = 0$ then
        For every $v \in Q_i$ do $E(H)\backslash\{\{u, v\}\}$
EndFor
Return $(V, E(H))$.
```

7. Open Problems

In this section we give some open problems

1. The problem of whether there is a $O(1)$-round learning algorithm for m-Graph with $O(m \log n)$ queries when m is unknown to the learner is still open.

2. Find a tight lower bound for two-round Monte-Carlo algorithm.

References

Hasan Abasi, Nader H. Bshouty, and Hanna Mazzawi. On exact learning monotone DNF from membership queries. In Peter Auer, Alexander Clark, Thomas Zeugmann, and Sandra Zilles, editors, *Algorithmic Learning Theory - 25th International Conference, ALT 2014, Bled, Slovenia, October 8-10, 2014. Proceedings*, volume 8776 of *Lecture Notes in Computer Science*, pages 111–124. Springer, 2014. ISBN 978-3-319-11661-7. doi: 10.1007/978-3-319-11662-4_9. URL https://doi.org/10.1007/978-3-319-11662-4_9.

Hasan Abasi, Nader H. Bshouty, and Hanna Mazzawi. Non-adaptive learning of a hidden hypergraph. *Theor. Comput. Sci.*, 716:15–27, 2018. doi: 10.1016/j.tcs.2017.11.019. URL https://doi.org/10.1016/j.tcs.2017.11.019.

Noga Alon and Vera Asodi. Learning a hidden subgraph. *SIAM J. Discrete Math.*, 18(4): 697–712, 2005. doi: 10.1137/S0895480103431071. URL https://doi.org/10.1137/S0895480103431071.

Noga Alon, Richard Beigel, Simon Kasif, Steven Rudich, and Benny Sudakov. Learning a hidden matching. *SIAM J. Comput.*, 33(2):487–501, 2004. doi: 10.1137/S0097539702420139. URL https://doi.org/10.1137/S0097539702420139.

Dana Angluin and Jiang Chen. Learning a hidden hypergraph. *Journal of Machine Learning Research*, 7:2215–2236, 2006. URL http://www.jmlr.org/papers/v7/angluin06a.html.

Dana Angluin and Jiang Chen. Learning a hidden graph using o(logn) queries per edge. *J. Comput. Syst. Sci.*, 74(4):546–556, 2008. doi: 10.1016/j.jcss.2007.06.006. URL https://doi.org/10.1016/j.jcss.2007.06.006.

Dana Angluin, James Aspnes, and Lev Reyzin. Inferring social networks from outbreaks. In Marcus Hutter, Frank Stephan, Vladimir Vovk, and Thomas Zeugmann, editors, *Algorithmic Learning Theory, 21st International Conference, ALT 2010, Canberra, Australia, October 6-8, 2010. Proceedings*, volume 6331 of *Lecture Notes in Computer Science*, pages 104–118. Springer, 2010. ISBN 978-3-642-16107-0. doi: 10.1007/978-3-642-16108-7_12. URL https://doi.org/10.1007/978-3-642-16108-7_12.

Mathilde Bouvel, Vladimir Grebinski, and Gregory Kucherov. Combinatorial search on graphs motivated by bioinformatics applications: A brief survey. In Dieter Kratsch, editor, *Graph-Theoretic Concepts in Computer Science, 31st International Workshop, WG 2005, Metz, France, June 23-25, 2005, Revised Selected Papers*, volume 3787 of *Lecture Notes in Computer Science*, pages 16–27. Springer, 2005. ISBN 3-540-31000-2. doi: 10.1007/11604686_2. URL https://doi.org/10.1007/11604686_2.

Nader H. Bshouty. Linear time constructions of some d -restriction problems. In Vangelis Th. Paschos and Peter Widmayer, editors, *Algorithms and Complexity - 9th International Conference, CIAC 2015, Paris, France, May 20-22, 2015. Proceedings*, volume 9079 of *Lecture Notes in Computer Science*, pages 74–88. Springer, 2015. ISBN 978-3-319-18172-1. doi: 10.1007/978-3-319-18173-8_5. URL https://doi.org/10.1007/978-3-319-18173-8_5.

Mahdi Cheraghchi. Noise-resilient group testing: Limitations and constructions. *Discrete Applied Mathematics*, 161(1-2):81–95, 2013. doi: 10.1016/j.dam.2012.07.022. URL https://doi.org/10.1016/j.dam.2012.07.022.

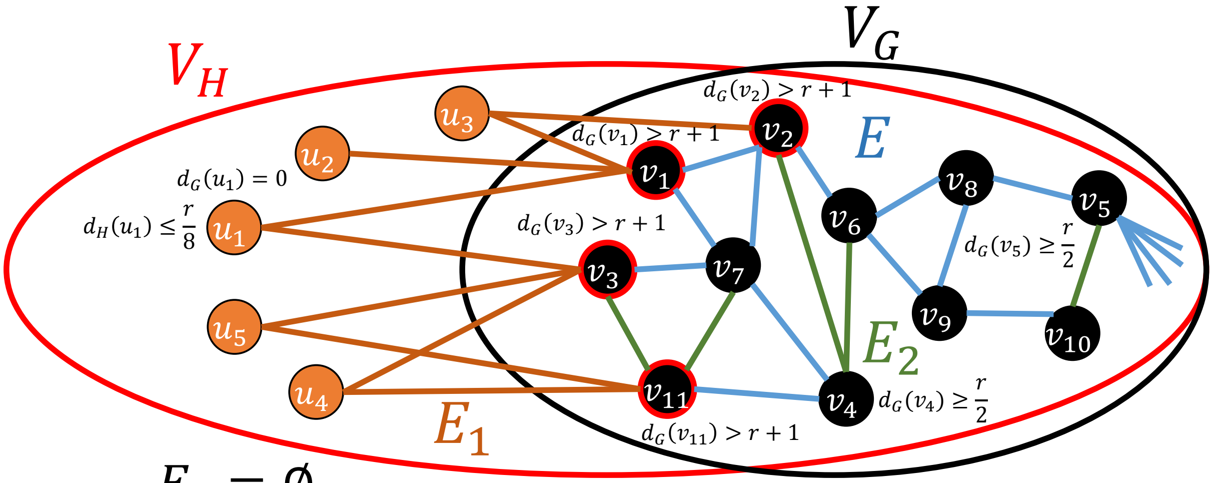
Dingzhu Du, Frank K Hwang, and Frank Hwang. *Combinatorial group testing and its applications*, volume 12. World Scientific, 2000.

AG D'yachkov and VV Rykov. Bounds on the length of disjunctive codes. *PROB. INFO. TRANSMISSION.*, 18(3):166–171, 1983.

Moein Falahatgar, Ashkan Jafarpour, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh. Estimating the number of defectives with group testing. In *IEEE International Symposium on Information Theory, ISIT 2016, Barcelona, Spain, July 10-15, 2016*, pages 1376–1380. IEEE, 2016. ISBN 978-1-5090-1806-2. doi: 10.1109/ISIT.2016.7541524. URL https://doi.org/10.1109/ISIT.2016.7541524.

Zoltán Füredi. On r-cover-free families. *J. Comb. Theory, Ser. A*, 73(1):172–173, 1996. doi: 10.1006/jcta.1996.0012. URL https://doi.org/10.1006/jcta.1996.0012.

Hwang Frank Kwang-ming and Du Ding-zhu. *Pooling designs and nonadaptive group testing: important tools for DNA sequencing*, volume 18. World Scientific, 2006.



$$E_0 = \emptyset$$

$$|E_2| \leq \frac{8m^2}{r} \qquad E(H) = E_1 \cup E_2 \cup E$$

$$W = \{v_1, v_2, v_3, v_4, v_5, v_{11}\}$$

$$I_{v_1} = \{u_1, u_2, u_3, v_7\} \qquad I_{v_2} = \{u_3, v_4\}$$

$$I_{v_3} = \{u_1, u_4, u_5, v_7\} \qquad I_{v_4} = \{v_6, v_7\}$$

Figure 4: An Example