LEARNING A HIDDEN UNIFORM HYPERGRAPH

HUILAN CHANG, HUNG-LIN FU, AND CHIH-HUAI SHIH

Abstract

Motivated by applications in genome sequencing, Grebinski and Kucheerov (1997) studied graph learning problem which is to identify a hidden graph drawn from a given class of graphs with vertex set $\{1, 2, \cdots, n\}$ by edge-detecting queries. Each query tells whether a set of vertices induces an edge of the hidden graph or not. For the class of all hypergraphs whose edges have size at most r, Chodoriwsky and Moura (2015) provided an adaptive algorithm that learns the class in $O(m^r \log n)$ queries if the hidden graph has m edges. In this paper, we provide an adaptive algorithm that learns the class of all r-uniform hypergraphs in $mr \log n + (6e)^r m^{\frac{r+1}{2}}$ queries if the hidden graph has m edges.

1. Introduction

A *graph learning problem* is to identify a hidden graph G drawn from a class $\mathcal{G}$ of labeled graphs by using *edge-detecting* queries of the following form

$$Q(S)\text{: does a set } S \text{ of vertices induce any edge of } G?$$

If S induces some edge of G, then the query $Q(S)$ is answered yes, denoted by $Q(S) = 1$; otherwise, $Q(S)$ is answered no, denoted by $Q(S) = 0$. Graph learning problem originated from applications in DNA physical mapping [10]. Such an application is based on a strategy proposed by Sorokin et al. [14]: In physical mapping, the whole DNA molecule is covered by some replicate cloned fragments with some gaps and some of the cloned fragments are assembled to form longer continuous fragments (contigs). However, the information about the mutual placement of the contigs on the DNA sequence is lost. One can identify the relative replacements of the contigs by using multiplex polymerase chain reaction (*multiplex PCR*) which is an experiment telling whether there are two neighbouring contigs in a set of contigs. For example, if the structure of the DNA molecule is circular, then the order of contigs along the circular molecule can be determined by identifying every pair of neighbouring contigs. This identification work can be modeled by a graph learning problem on the class of Hamiltonian cycles and each Multiplex PCR experiment plays the edge-detecting role. Many conditions on $\mathcal{G}$ have been considered in the literature [2, 3, 5, 6, 7, 10, 11]. The problem of learning a general graph is to deal with labeled graphs that have no restricted topology; specifically, $\mathcal{G}$ collects all simple graphs on the vertex set $[n] := \{1, 2, \cdots, n\}$. Chang *et al.* [8] provided an adaptive algorithm that learns the class of all simple graphs on $[n]$ with at most $m \log n + 10m + 3n$ edge-detecting queries when the hidden graph has m edges.

2010 *Mathematics Subject Classification.* 68W40, 05C90.

Key words and phrases. graph learning, group testing, complex model, adaptive algorithm.

Partially supported by MOST Taiwan under the grant MOST 104-2115-M-009-009 (H.-L. Fu) and MOST 104-2115-M-390-005-MY2 (H. Chang).

Learning a hidden general graph can be described as a group testing problem which is a well-known field of combinatorial search [1]. Given a set of items, each of which is either positive or negative, the main task of *group testing problem* is to identify all positive items by group tests. A *group test* is applied to a subset of items and tells whether the subset contains any positive. An extension of the classical group testing is *the complex model*, where the property of being positive is not defined on single items but on subsets of items, called *complexes*. The corresponding group test used to identify positive complexes tells whether a subset of basic items contains any positive complex. Torney [15] first introduced the concept of the complex model and took substances in eukaryotic DNA transcription and RNA translation as examples of complexes. Group testing on complexes is widely applied in modern molecular and cellular biology. A prominent example is its application in the identification of protein-to-protein interactions (Lappe and Holm [12]; Li et al. [13]). In signal transduction process, the protein-to-protein interactions of the signaling molecules can convey signals from the exterior of a cell to the inside of that cell. This conveying process plays a fundamental role in living cells. Information about the interactions between proteins improves our understanding of diseases and then provides the basis for new therapeutic approaches. The development of some laboratory approaches (Lappe and Holm [12]) enables the application of group testing to this problem. Li et al. [13] formulated this identification problem as a group testing problem in bipartite graphs which can be regarded as a special case of group testing on complexes. Learning a hidden graph is identical to a case of the complex model where every pair of items form a complex; a positive complex corresponds to an edge of the hidden graph and a negative complex corresponds to a pair of vertices which do not form an edge of the hidden graph.

A generalization of learning a hidden graph problem is to consider learning a hidden hypergraph. A hypergraph is *r-uniform* if each edge contains exactly r vertices. Thus a simple graph can be referred as a 2-uniform hypergraph. An *(r_1, r_2)-uniform* hypergraph is a hypergraph whose minimum and maximum edge sizes are r_1 and r_2, respectively. Use m to denote the number of edges of the hidden graph. It is an unknown parameter if the graph class does not collect graphs of a fixed number of edges. Angluin and Chen [4] showed that the class of all (r_1, r_2)-uniform hypergraphs is learnable with $O(f_1(r_1, r_2)m^{1+\frac{r_2-r_1}{2}} poly(\log n)\log(1/\delta))$ queries with probability at least $1-\delta$, where f_1 depends only on r_1 and r_2. Especially, they showed that the class of all r-uniform hypergraphs is learnable with $O(2^{4r}m \cdot poly(r, \log n)\log(1/\delta))$ queries with probability $1-\delta$, where the queries can be made in $O(\min(2^r r^2 \log^2 n, r^3 \log^3 n))$ rounds. For the class of all hypergraphs whose maximum edge size is r, Chodoriwsky and Moura [9] provided an adaptive algorithm to identify a hidden graph G drawn from this class in $O(m^r \log n)$ queries.

In this paper, we focus on learning a hidden r-uniform hypergraph. We provide a deterministic adaptive algorithm to learn the class of all r-uniform hypergraphs within $mr\log n + (6e)^r m^{\frac{r+1}{2}}$ queries.

2. Lower bound

We start off this section by presenting some lower bounds. Angluin and Chen [5] provided an information-theoretic lower bound on learning the graph class that collects all simple graphs of n vertices with a given number of edges. This result can be easily extended to the class of all r-uniform hypergraphs.

Theorem 2.1. *For any $0 < \epsilon < 1$, $\Omega(\epsilon mr \log n)$ edge-detecting queries are required to identify a hidden graph drawn from the class of all r-uniform hypergraphs with n vertices and $m = (\frac{n^{(1-\epsilon)}}{r})^r$ edges.*

Proof. Since there are at least $\binom{\binom{n}{r}}{m} \geq (\frac{(\frac{n}{r})^r}{m})^m$ r-uniform hypergraphs with m edges and each edge-detecting query is answered 1 or 0, we obtain the following information-theoretic lower bound

$$\log(\frac{(\frac{n}{r})^r}{m})^m = m \log(\frac{n^r}{r^r m}) = \epsilon mr \log n.$$

□

Let $\mathcal{G}$ be the class of all $(2, r)$-uniform hypergraphs on the vertex set $[n]$ with m edges. To provide a lower bound on learning the graph class $\mathcal{G}$, Angluin and Chen [5] considered a subclass $\mathcal{G}' \subset \mathcal{G}$, where each hypergraph in $\mathcal{G}'$ consists of exactly one edge of size r and some edges of size two. Adversary argument is a useful method to obtain lower bounds for combinatorial search problems. Angluin and Chen [5] used this technique to obtain a lower bound on learning $\mathcal{G}'$ and thus gave the following lower bound on learning $\mathcal{G}$.

Theorem 2.2. *[Theorem 6.1 [5]] $\Omega((2m/r)^{r/2})$ edge-detecting queries are required to identify a hypergraph drawn from the class of all $(2, r)$-uniform hypergraphs with n vertices and m edges.*

3. Learning an r-uniform hypergraph

In this section, we introduce our adaptive algorithm RECONSTRUCT-r-UNIFORM-HYPERGRAPH that identifies a hidden graph G drawn from the class of all r-uniform hypergraphs on the vertex set $[n]$. Our algorithm first identifies an edge of the hidden graph and the complexity of identifying an edge has been studied by Angluin and Chen [4].

Lemma 3.1. *[Lemma 3, [4]] For any hidden hypergraph G on $[n]$ with each edge of size at most r, there is an algorithm that identifies an edge of G in $r\lceil \log n \rceil$ edge-detecting queries.*

To deal with the problem of learning a general graph, the following idea has been used by Chang et al. [8]: partition the vertex set $[n]$ into parts $S_1, S_2, \cdots, S_l$ such that each edge of G has at most one vertex in S_i, $i = 1, 2, \cdots, l$ and then identify the hidden subgraph induced by each pair of $S_1, S_2, \cdots, S_l$. In our algorithm, we adopt this idea as a beginning strategy. Note that our algorithm is applicable even though the number of edges in G is unknown.

We first outline the steps of our algorithm (Algorithm 1). For a set S of vertices, we use $G[S]$ to denote the subgraph of G induced by S. In step 1, we first identify an edge, say e, and separate all vertices in $[n]$ into r parts such that e intersects each part in at most one vertex. We continue to identify edges induced by the union of every $r - 1$ parts and rearrange parts (new parts are created if necessary) such that each identified edge intersects each part in at most one vertex. Suppose that at the end of Step 1 there are l parts, say $S_1, S_2, \cdots, S_l$. Then the union of any $r - 1$ parts of $S_1, S_2, \cdots, S_l$ induces no more edge, that is, all vertices of any unidentified edge belong to distinct parts. Our next step (Step 2) is to identify every edge that contains a vertex belonging to no identified edge one by one. Therefore, the remaining work is to identify every edge each of whose vertices belongs to some identified edge (see Step 3).

Algorithm 1 RECONSTRUCT-r-UNIFORM-HYPERGRAPH

Step 1:

Find an edge $e = \{v_1, \cdots, v_r\}$.
Create r parts: $S_1 = [n] \setminus \{v_2, \cdots, v_r\}$, $S_i = \{v_i\}$ for $2 \le i \le r$. Let $E \leftarrow \{e\}$.
for every group of $r-1$ parts **do**
  Let I be the index set of the chosen $r-1$ parts.
  while $Q(\cup_{i\in I} S_i) = 1$ **do**
    Find an edge e in $\cup_{i\in I} S_i$ and let $E \leftarrow E \cup \{e\}$. If more than one vertex of e are in the same part, then remove one of them, say v, to the first available part such that v and any vertex in the part do not belong to the same identified edge (if there is no such a part, then create a new one and remove v to it); repeat the process until all vertices of e belong to different parts.
  end while
  If a part S_i is expanded in the above **while-loop**, then every group of $r-1$ parts including S_i is reconsidered in this **for-loop**.
end for

Step 2:

for the union of every r parts $S_I = \bigcup_{i\in I} S_i$ where $I = \{i_1, \cdots, i_r\}$ **do**
  Let $S_i^*(I) = \bigcup(e \cap S_i)$, where the union is running through all identified edges e in $G[S_I]$.
  while $S_k \setminus S_k^*(I) \neq \emptyset$ and $Q(S_I \setminus S_k^*(I))$=1 for some $k \in I$ **do**
    Find an edge e in $S_I \setminus S_k^*(I)$. For $j = 1, \cdots, r$, if e contains a vertex in $S_{i_j} \setminus S_{i_j}^*(I)$, say v, then include v into $S_{i_j}^*(I)$.
  end while
end for

Step 3:

for the union of every r subsets $S_I^* = \bigcup_{i\in I} S_i^*(I)$ where $I = \{i_1, \cdots, i_r\}$ **do**
  Let $s_I = \max_{i\in I} |S_i^*(I)|$. Separate vertices in $S_{i_j}^*(I)$ into $x_I = \lceil s_I^{\frac{r+1}{2r}} \rceil$ groups $G_{i_j,1}, G_{i_j,2}, \cdots, G_{i_j,x_I}$ of size either $\lceil \frac{|S_{i_j}^*(I)|}{x_I} \rceil$ or $\lfloor \frac{|S_{i_j}^*(I)|}{x_I} \rfloor$.
  for each r-tuple of groups $(G_{i_1,k_1}, G_{i_2,k_2}, \cdots, G_{i_r,k_r})$ **do**
    if $Q(\bigcup_{j=1}^r G_{i_j,k_j}) = 1$ **then**
      Test all r-subsets of $\bigcup_{j=1}^r G_{i_j,k_j}$ whose vertices are from different groups and do not form an identified edge.
    end if
  end for
end for

We demonstrate how our algorithm works with an example. Let $G = (V, E)$ be a 3-uniform hypergraph where the vertex set $V = \{1, 2, 3, 4, 5, 7, 8, 9\}$ and the edge set $E = \{\{1, 2, 3\}, \{1, 2, 4\}, \{2, 4, 5\}, \{2, 4, 6\}, \{2, 5, 7\}, \{2, 3, 8\}, \{2, 5, 8\}\}$. TABLE 1 demonstrates how our algorithm is applied to this graph.

Algorithm	Query	Identified edge	Partition of $[n]$: $S_1/S_2/S_3/\dots$
Step 1: Line 1-2	$Q(V) = 1$. Identify an edge in $G[V]$.	$e_1 = \{1,2,3\}$	$\{1,4,5,6,7,8,9\}/\{2\}/\{3\}$
Step 1: Line 4-7 with $I = \{1,2\}$.	$Q(S_1 \cup S_2) = 1$. Identify an edge in $G[S_1 \cup S_2]$.	$e_2 = \{2,4,5\}$	$4,5 \in e_3 \cap S_1$. Since $2,5 \in e_2$, remove 5 to S_3. The partition becomes $\{1,4,6,7,8,9\}/\{2\}/\{3,5\}$
	$Q(S_1 \cup S_2) = 1$. Identify an edge in $G[S_1 \cup S_2]$.	$e_3 = \{2,4,6\}$	$4,6 \in e_3 \cap S_1$. Since $2,6 \in e_3$, remove 6 to S_3. The partition becomes $\{1,4,7,8,9\}/\{2\}/\{3,5,6\}$.
	$Q(S_1 \cup S_2) = 1$. Identify an edge in $G[S_1 \cup S_2]$.	$e_4 = \{1,2,4\}$	$1,4 \in e_4 \cap S_1$. Since $2,4 \in e_4$ and $4,6 \in e_3$, remove 4 from S_1 and create a new part $S_4 = \{4\}$. The partition becomes $\{1,7,8,9\}/\{2\}/\{3,5,6\}/\{4\}$.
	$Q(S_1 \cup S_2) = 0$		
Step 1: Line 4-7 with $I = \{1,3\}$, $I = \{1,4\}$, $I = \{2,3\}$, $I = \{2,4\}$, and $I = \{3,4\}$, respectively.	$Q(S_1 \cup S_3) = 0$, $Q(S_1 \cup S_4) = 0$, $Q(S_2 \cup S_3) = 0$, $Q(S_2 \cup S_4) = 0$, and $Q(S_3 \cup S_4) = 0$.		
After Step 1, the partition becomes $\{1,7,8,9\}/\{2\}/\{3,5,6\}/\{4\}$. In the following, we take Step 2 with $I = \{1,2,3\}$ as an example. $S_I = \{1,7,8,9,2,3,5,6\}$ and e_1 is the only identified edge induced by S_I. We got a subpartition $\{\underline{1},7,8,9\}/\{\underline{2}\}/\{\underline{3},5,6\}$, where the numbers in each $S^*_{i_j}(I)$ are underlined.			
Step 2: Line 3-5	$Q(\{7,8,9,2,3,5,6\}) = 1$. Identify an edge in $G[\{7,8,9,2,3,5,6\}]$.	$e_5 = \{2,5,7\}$	The subpartition becomes $\{\underline{1},\underline{7},8,9\}/\{\underline{2}\}/\{\underline{3},\underline{5},6\}$.
	$Q(\{8,9,2,3,5,6\}) = 1$. Identify an edge in $G[\{8,9,2,3,5,6\}]$.	$e_6 = \{2,3,8\}$	The subpartition becomes $\{\underline{1},\underline{7},\underline{8},9\}/\{\underline{2}\}/\{\underline{3},\underline{5},6\}$.
	$Q(\{9,2,3,5,6\}) = 0$.		
	$Q(\{1,7,8,9,2,6\}) = 0$.		
In the following, we take Step 3 with $I = \{1,2,3\}$ as an example. For such an I, the subpartition is $\{\underline{1},\underline{7},\underline{8},9\}/\{\underline{2}\}/\{\underline{3},\underline{5},6\}$, $s_I = 3$ and $x_I = 3$. Then $S^*_1(I) = \{1,7,8\}$ is partitioned into $G_{1,1} = \{1\}$, $G_{1,2} = \{7\}$ and $G_{1,3} = \{8\}$; $S^*_2(I) = \{2\}$ is partitioned into $G_{2,1} = \{2\}$; $S^*_3(I) = \{3,5\}$ is partitioned into $G_{3,1} = \{3\}$ and $G_{3,2} = \{5\}$.			
Step 2: Line 3-7	$Q(\{1,2,5\}) = 0$, $Q(\{7,2,3\}) = 0$, $Q(\{8,2,5\}) = 1$. Note that $\{1,2,3\}$, $\{7,2,5\}$ and $\{8,2,3\}$ are identified edges.	$e_7 = \{8,2,5\}$	

TABLE 1. An example demonstrating Algorithm 1

Lemma 3.2. *There are at most $r\sqrt{2m}$ parts created in Step 1 of RECONSTRUCT-r-UNIFORM-HYPERGRAPH if the hidden graph has m edges.*

Proof. Suppose that there are l parts. We count in two different ways the number β of ordered pairs (v, e) such that v is a vertex and e is an edge containing v. Since the hidden graph is r-uniform and it contains at most m edges, $\beta \leq rm$. Focusing now on a vertex v, suppose that $v \in S_i$. That v is moved into S_i is due to the reason that each part S_j with $j < i$ has a vertex that shares an edge with v. Thus there are at least $\lceil \frac{i-1}{r-1} \rceil$ edges containing v. Therefore, $rm \geq \beta \geq \sum_{i=1}^{l} |S_i| \lceil \frac{i-1}{r-1} \rceil \geq \sum_{i=1}^{l} \lceil \frac{i-1}{r-1} \rceil \geq \frac{l^2}{2r}$ and the conclusion $l \leq r\sqrt{2m}$ follows. □

Theorem 3.3. *For the class $\mathcal{G}$ of all r-uniform hypergraphs on $[n]$, a hidden graph G drawn from $\mathcal{G}$ can be identified in at most $mr \log n + (6e)^r m^{\frac{r+1}{2}}$ edge-detecting queries where m is the number of edges of G.*

Proof. At the beginning, we find an arbitrary edge and separate its vertices into r different parts $S_1, S_2, \cdots, S_r$. In the for-loop of Step 1, we check whether the union of any $r-1$ parts induces any edge: If the query on the union is answered yes, then we identify an edge in it and separate its vertices into different parts. Then Step 1 terminates when the vertices of each edge (identified or not) of G are separated to r different parts, say $S_1, S_2, \cdots, S_l$ and no edge is identified twice in Step 1.

It remains to identify all hidden edges whose vertices are in different parts of $S_1, S_2, \cdots, S_l$. For the union of fixed r parts $S_I = \bigcup_{i \in I} S_i$ where $I = \{i_1, \cdots, i_r\}$, $S^*_{i_j}(I)$ collects all vertices of S_{i_j} each of which belongs to some identified edge in $G[S_I]$. If $S_k \setminus S^*_k(I) \neq \emptyset$ and $Q(S_I \setminus S^*_k(I))$ is answered yes for some $k \in I$, then $S_I \setminus S^*_k(I)$ induces at least one edge and every edge induced by it is unidentified and contains one vertex in $S_k \setminus S^*_k(I)$. Line 3-5 of Step 2 are to identify such edges and include each of their vertices into associated $S^*_{i_j}(I)$. Therefore, after Step 2, the vertices of each unidentified edge in $G[S_I]$ are in $S^*_{i_1}(I), S^*_{i_2}(I), \ldots, S^*_{i_r}(I)$, respectively. Step 3 is to identify the remaining unidentified edges by first partitioning $S^*_{i_j}(I)$ into groups and then identifying each edge whose vertices are in groups of $S^*_{i_1}(I), S^*_{i_2}(I), \ldots, S^*_{i_r}(I)$, respectively. Therefore, the correctness of RECONSTRUCT-r-UNIFORM-HYPERGRAPH is obtained.

Next, we analyze the number of queries made in Algorithm 1. Denote the number of edges identified in Step i by m_i. In Step 1, we make one query to check whether the union of every $r-1$ parts induces any edge of the hidden graph or not, and if a part S_i is expanded, then the union of every $r-1$ parts including S_i is reconsidered. According to Lemma 3.2, there are finally at most $r\sqrt{2m}$ parts; furthermore, once an edge is identified, at most $r-1$ parts are expanded and thus at most $(r-1)\binom{r\sqrt{2m}}{r-2}$ groups of $r-1$ parts should be reconsidered. Thus the number of queries made in Line 3-5 of Step 1 is at most

$$\begin{aligned}
&\binom{r\sqrt{2m}}{r-1} + (m_1 - 1)(r-1)\binom{r\sqrt{2m}}{r-2} + (m_1 - 1) \\
&\leq (\frac{r\sqrt{2m}e}{r-1})^{r-1} + (m_1 - 1)(r-1)(\frac{r\sqrt{2m}e}{r-2})^{r-2} + m_1 - 1 \\
&\leq (2\sqrt{2}e)^{r-1} m^{\frac{r-1}{2}} + (3e\sqrt{2})^{r-2}(r-1)m^{\frac{r}{2}} + m_1 - 1,
\end{aligned}$$

where the last term of the sum comes from that if the union of parts with indexes in I (Line 3) induces some edge of the hidden graph when the first time it is considered, then one query is made (in the while-loop of Line 5) to ensure that union of parts with indexes in I induces no more edge of the hidden graph after identifying some edges and rearranging

the parts. Since identifying an edge can be done in $r\lceil \log n \rceil$ ($\leq r(\log n + 1)$) queries (by Lemma 3.1), the overall queries made in Step 1 is at most

$$((2\sqrt{2}e)^{r-1}m^{\frac{r-1}{2}} + (3e\sqrt{2})^{r-2}(r-1)m^{\frac{r}{2}} + m_1 - 1) + m_1 r(\log n + 1).$$

In Step 2, it is easy to check that the while-loop of Line 3 is executed at most $r\binom{r\sqrt{2m}}{r} + m_2$ times and thus the overall queries made in Step 2 is at most

$$(\sqrt{2}e)^r m^{\frac{r}{2}} r + m_2 + m_2 r(\log n + 1).$$

Finally, we analyze the number of queries made in Step 3. Let m_I be the number of edges in $G[S_I^*]$. Since all vertices in $S_{i_j}^*(I)$ belong to distinct identified edges in $G[S_I^*]$, $s_I = \max_{i \in I} |S_i^*(I)|$ (defined in Line 2 of Step 3) is at most m_I. In Line 4 of Step 3, we make at most $\lceil s_I^{\frac{r+1}{2r}} \rceil^r \leq 2^r s_I^{\frac{r+1}{2}} \leq 2^r m_I^{\frac{r+1}{2}}$ queries to check whether the union of each r-tuple of groups induces at least one edge. Then we check all possible r-subsets of the union as stated in Line 5 of Step 3 if the query on the union is answered yes; then this process takes at most $m_I \lceil \frac{s_I}{P_I} \rceil^r \leq m_I (2 s_I^{\frac{r-1}{2r}})^r \leq 2^r m_I^{\frac{r+1}{2}}$ queries. Therefore, there are at most

$$\sum_{I \in \binom{[l]}{r}} 2^{r+1} m_I^{\frac{r+1}{2}} \leq 2^{r+1} m^{\frac{r+1}{2}}$$

queries made in Step 3, where l is the number of parts created in Step 1.

It is easy to observe that each edge is identified once and hence the overall cost of RECONSTRUCT-r-UNIFORM-HYPERGRAPH is at most

$$(m_1 + m_2) r(\log n + 1) + 2^{r+1} m^{\frac{r+1}{2}} + ((\sqrt{2}e)^r r + (3e\sqrt{2})^{r-2}(r-1)) m^{\frac{r}{2}}$$

$$+ (2\sqrt{2}e)^{r-1} m^{\frac{r-1}{2}} + m_1 + m_2$$

$$\leq mr \log n + (6e)^r m^{\frac{r+1}{2}}.$$

Therefore, the result follows. □

4. Concluding Remark

We show that learning the class $\mathcal{G}$ of all r-uniform hypergraphs on $[n]$ can be done in $mr \log n + (6e)^r m^{\frac{r+1}{2}}$ queries when the hidden graph has m edges. If the number of edges in the hidden graph is asymptotically smaller than $\log n$ and r is a constant, then our algorithm taking $O(mr \log n)$ queries to identify the hidden one is optimal. Let $\mathcal{G}'$ be the class of all $(2, r)$-uniform hypergraphs with n vertices and m edges. Comparing to $\mathcal{G}$, the sizes of the edges of the hypergraphs in $\mathcal{G}'$ are relaxed (ranging from 2 to r) but only hypergraphs of exactly m edges are collected in $\mathcal{G}'$. We note that the lower bound $\Omega((2m/r)^{r/2})$ on learning $\mathcal{G}'$ and the second term of our upper bound on learning $\mathcal{G}$ asymptotically differ by a factor of $\sqrt{m}$ if r is a constant, although there is no argument showing the relation between the difficulties of dealing with $\mathcal{G}'$ and $\mathcal{G}$.

References

[1] M. Aigner, Combinatorial Search, John Wiley and Sons (1988).
[2] N. Alon and V. Asod, Learning a hidden subgraph, SIAM J. Discrete Math. 18, 697-712 (2005)
[3] N. Alon, R. Beigel, S. Kasif, S. Rudich, and B. Sudakov, Learning a hidden matching, SIAM J. Comput. 33 (2004) 487-501.
[4] D. Angluin and J. Chen, Learning a hidden hypergraph, J. Mach. Learn. Res. 7 (2006) 2215-2236.
[5] D. Angluin and J. Chen, Learning a hidden graph using $O(\log n)$ queries per edge, J. Comput. Syst. Sci. 74 (2008) 546-556.
[6] R. Beigel, N. Alon, M. S. Apaydin, L. Fortnow, S. Kasif, An optimal procedure for gap closing in whole genome shotgun sequencing, Proc. 2001 RECOMB, ACM Press, 22-30.
[7] M. Bouvel, V. Grebinski, G. Kucherov, Combinatorial search on graphs motivated by bioinformatics applications: a brief survey. WG, LNCS, 3787 (2005) 16-27.
[8] H. Chang, H. L. Fu, and C. H. Shih, Learning a hidden graph, Optim. Lett., 8 (2014) 2341-2348.
[9] J. Chodoriwsky and L. Moura, An adaptive algorithm for group testing for complexes, theor. comp. sci. 592 (2015) 1-8.
[10] V. Grebinski and G. Kucherov, Reconstructing a Hamiltonian cycle by querying the graph: Application to DNA physical mapping, Discrete Appl. Math., 88 (1998) 147-165.
[11] V. Grebinski and G. Kucherov, Optimal reconstruction of graphs under the additive model, Algorithmica, 28 (2000) 104-124.
[12] M. Lappe and L. Holm, Unraveling protein interaction networks with near-optimal efficiency, Nature Biotechnology 22 (2003) 98-103.
[13] Y. Li, M. Thai, Z. Liu, and W. Wu, Protein-to-protein interactions and group testing in bipartite graphs, International J. of Bioinformatics Research and Applications 1 (2005) 414-419.
[14] A. Sorokin, A. Lapidus, V. Capuano, N. Galleron, P. Pujic, S. D. Ehrlich, A new approach using multiplex long accurate PCR and yeast artificial chromosomes for bacterial chromosome mapping and Sequencing, Genome Res., 6 (1996) 448-453.
[15] D. C. Torney, Sets pooling designs, Ann. Comb. 3 (1999) 95-101.

H. Chang: Department of Applied Mathematics, National University of Kaohsiung, Kaohsiung 811, Taiwan, ROC. E-mail: huilan0102@gmail.com

H.-L. Fu: Department of Applied Mathematics, National Chiao Tung University, Hsinchu 30010, Taiwan, ROC. E-mail: hlfu@math.nctu.edu.tw

C.-H. Shih: Department of Applied Mathematics, National Chiao Tung University, Hsinchu 30010, Taiwan, ROC. E-mail: skyking.shih@gmail.com