J Comb Optim (2011) 22:270–281
DOI 10.1007/s10878-010-9291-0

Reconstruction of hidden graphs and threshold group testing

Huilan Chang · Hong-Bin Chen · Hung-Lin Fu · Chie-Huai Shi

Published online: 30 January 2010

Abstract Classical group testing is a search paradigm where the goal is the identification of individual positive elements in a large collection of elements by asking queries of the form "Does a set of elements contain a positive one?". A graph reconstruction problem that generalizes the classical group testing problem is to reconstruct a hidden graph from a given family of graphs by asking queries of the form "Whether a set of vertices induces an edge". Reconstruction problems on families of Hamiltonian cycles, matchings, stars and cliques on n vertices have been studied where algorithms of using at most $2n\lg n$, $(1+o(1))(n\lg n)$, $2n$ and $2n$ queries were proposed, respectively. In this paper we improve them to $(1+o(1))(n\lg n)$, $(1+o(1))(\frac{n\lg n}{2})$, $n+2\lg n$ and $n+\lg n$, respectively. Threshold group testing is another generalization of group testing which is to identify the individual positive elements in a collection of elements under a more general setting, in which there are two fixed thresholds ℓ and u, with $\ell < u$, and the response to a query is positive if the tested subset of elements contains at least u positive elements, negative if it contains at most ℓ positive elements, and it is arbitrarily given otherwise. For the threshold group testing problem with $\ell = u - 1$, we show that p positive elements among n given elements can be determined by using $O(p\lg n)$ queries, with a matching lower bound.

Keywords Graph search · Threshold group testing · Pooling design · Adaptive algorithms

This research is partially supported by NSC 97-2115-M-009-011-MY3.

H. Chang (✉) · H.-B. Chen · H.-L. Fu · C.-H. Shi
Department of Applied Mathematics, National Chiao Tung University, Hsinchu 30010, Taiwan
e-mail: huilan0102@gmail.com

1 Introduction

Graph reconstruction Combinatorial search problems on graphs in the literature (Aigner 1988) consist of identifying an unknown edge or vertex in a given graph, verifying a property of a hidden graph, reconstructing a hidden graph of a given class, and some others. In this paper we consider the *graph reconstruction problem* as follows. Let $\mathcal{H}$ be a family of labeled graphs on the set $V = \{1, 2, \dots, n\}$. The goal is to reconstruct a hidden graph $H \in \mathcal{H}$ by asking queries as few as possible, where a query $Q(S)$ is of the form "Does S contain at least one edge of H?" for $S \subseteq V$, and is answered 1 (positive) or 0 (negative), indicating whether the subgraph of H induced by S contains an edge or not.

Different settings on input data according to the prior knowledge of the hidden graph produce various graph reconstruction problems. For example, the hidden graph of bounded degree was studied in (Bouvel et al. 2005; Grebinski and Kucherov 2000), while the general hidden graph was considered in (Angluin and Chen 2008; Bouvel et al. 2005); moreover, in group testing literature (Du and Hwang 2006), the usual assumption is that the number of edges of the hidden graph is bounded by a parameter d. In this paper we study the problem of reconstructing a hidden graph with the assumption that the topology of the hidden graph is known.

There are various families of hidden graphs for which the problem has been studied. Many recent studies focus on two cases: Hamiltonian cycles and matchings (Alon et al. 2004; Beigel et al. 2001; Grebinski and Kucherov 1998) which have specific application to the *genome sequencing problem*. In the genome sequencing, some clones can overlap forming longer continuous fragments, called *contigs*, that cover the genome with possible gaps. The task is to determine the relative placement of contigs on the genome. A tool for doing this is an experiment called multiplex PCR (*Polymerase Chain Reaction*, see Sorokin et al. 1996). In multiplex PCR, the input to an experiment is a set of *primers*, which are short nucleotide sequences that characterize the ends of the contigs, and whenever the input set contains two primers corresponding to adjacent ends of neighboring contigs, the experiment outputs a reaction bringing a PCR product; hence, the relative placement of contigs can be represented by the reaction graph which is a graph with primers as its vertices and pairs of vertices with a reaction as its edges. In particular, for a circular genome, a reaction graph can be characterized as either a Hamiltonian cycle if the two primers of each contig are mixed together and are considered as a vertex or a matching if primers are treated independently, i.e., each primer corresponds to a vertex. For more general settings, the problem can be considered as to find the pairs that react with each other among the given set of molecules (Alon and Asodi 2005; Torney 1999).

Threshold group testing "In a set of n elements, p elements are *positive* and the other $n - p$ elements are *negative*" is assumed by group testing problem, where p is much smaller than n. A *group test* takes a set S of elements as input. Classically, the test yields a positive outcome if S contains at least one positive element; otherwise, the outcome is negative. The goal is to identify positive elements with as few tests as possible. We suggest readers refer to the book (Du and Hwang 2006) for further study.

With recent advances in molecular biology, group testing has been widely used in various aspects of DNA sequencing projects. While some projects involve quantitative measurements, others consist in applying a basic yes-or-no test to a large collection of objects. However, the sensitivity of tests can be reduced because of the fact that biological assays can be somewhat noisy. Outcomes of experiments could hardly be detected precisely. One might think of the model that being detected correctly or not is decided by the concentration of a substance; specifically there is an upper threshold for sure detection and a lower threshold under which detection is impossible, and between the thresholds, the outcome is arbitrary. This is so-called *threshold group testing* introduced by Damaschke (2006), and a precise definition is as follows.

> Let l and u be nonnegative integers with $l < u$. A test gives a positive (negative) outcome if it contains at least u (at most l) positive elements, and an arbitrary outcome if the number of positive elements is between these fixed thresholds l and u.

In general, there is a gap between the thresholds; thus the positive set could hardly be identified exactly. Instead, Damaschke suggested to find an alternative set close to the positive set with up to a constant number of misclassifications, bounded by the gap. In the present paper, we focus on the case *without gap*, i.e., a test returns positive if the input set contains at least u positives and negative otherwise.

Overview of the paper In this paper we only consider *adaptive algorithms* where queries are performed one by one and the feedbacks of all previous queries can be used to set up the later one. In Sect. 2, we study some families of hidden graphs including Hamiltonian cycles, matchings, stars, and cliques. Grebinski and Kucherov (1998) gave an adaptive algorithm to reconstruct a Hamiltonian cycle in $2n \lg n$ queries, which achieves the information lower bound for the number of queries needed. We improve their result by a factor of $1/2$. Bouvel et al. (2005) provided an adaptive algorithm to reconstruct a matching in $(1 + o(1))(n \lg n)$ queries while $(1 + o(1))(\frac{n}{2} \lg n)$ is the best lower bound known so far and an algorithm to reconstruct a star in $2n$ queries while the information lower bound is $(1 + o(1))n$. We provide algorithms to close up the gaps between lower and upper bounds for the number of queries required to reconstruct a star. Bouvel et al. also proved that a clique can be reconstructed in $2n$ queries. We slightly improve this result by giving an algorithm with at most $n + \lg n$ queries. In Sect. 3, we study the graph reconstruction problem of its hypergraph version where each edge can consist of more than two vertices and notice that this study can greatly contribute to solving the threshold group testing problem. Using the strategy used to reconstruct a hidden hypergraph, we obtain an asymptotically optimal $O(p \lg n)$ solution to the threshold group testing problem without gap for which Damaschke (2006) gave an algorithm using $O((p + u^2) \lg n)$ queries.

Notation Subsequently, lg is the base 2 logarithm.

2 Reconstructing simple graphs

Reconstructions of some families of hidden graphs such as Hamiltonian cycles, matchings, stars, and cliques are considered in this section. We start by presenting some useful algorithms that will be subroutines in our main algorithms.

First, we adapt an algorithm in (Angluin and Chen 2006) to find a vertex contained in at least one edge of a hidden graph on n vertices using at most $\lg n$ queries. The algorithm is described as follows.

Algorithm 1 FIND-ONE-VERTEX

```
1:  $S \leftarrow V$
2:  if $Q(S) = 0$ then
3:      Return $\emptyset$.
4:  end if
5:  $A \leftarrow V$.
6:  while $|A| > 1$ do
7:      Arbitrarily partition $A$ into roughly equal-sized $A_0$ and $A_1$.
8:      if $Q(S \setminus A_0) = 1$ then
9:          $S \leftarrow S \setminus A_0$, $A \leftarrow A_1$.
10:     else
11:         $A \leftarrow A_0$.
12:     end if
13: end while
14: Return the element in $A$.
```

Observe that the algorithm preserves the invariance that $Q(S) = 1$ and $Q(S \backslash A) = 0$ if the input hidden graph contains at least one edge. This shows A contains a vertex on an edge of the hidden graph; indeed, A is monotonically decreasing and the halving of A's cardinality in each iteration results in $\lg n$ queries. Once the algorithm terminates, $A = \{v\}$ for some vertex v and $S \setminus \{v\}$ contains a vertex adjacent to v; hence, we can find a neighbor of v by using binary splitting algorithm on $S \backslash \{v\}$ with v added to each test. Note that it is the procedure we use to find one edge in the rest of the paper if we do not stress what strategy being used and it is accomplished in $2 \lg n$ queries.

Second, we want to find a maximal matching of the hidden graph before reconstructing the whole graph where a matching is maximal if it is not contained in a matching of larger size. A reconstructed maximal matching of a graph reveals a partial structure of the graph and provides us a direction of reconstructing the remaining graph. Finding a maximal matching M can be accomplished by Algorithm 2 with at most $2m' \lg n + 1$ queries, where m' is the size of the maximal matching. We use $G[S]$ to denote the induced subgraph of graph G with vertex set S.

Algorithm 2 reconstructs edges one by one and removes from S the two vertices in an edge as soon as it is reconstructed; therefore, the returned set M is a matching since the reconstructed edges share no vertex. Indeed, M is a maximal matching because searching an edge in S continues until it induces no edge which implies that

Algorithm 2 FIND-MAXIMAL-MATCHING

```
1:  $M \leftarrow \emptyset$, $S \leftarrow V(G)$, $U \leftarrow \emptyset$, $U' \leftarrow \emptyset$.
2:  while $Q(S) = 1$ do
3:      Reconstruct an edge in $G[S]$, say $\{u, f(u)\}$.
        $M \leftarrow M \cup \{\{u, f(u)\}\}$, $S \leftarrow S \backslash \{u, f(u)\}$, $U \leftarrow U \cup \{u\}$, $U' \leftarrow U' \cup \{f(u)\}$.
4:  end while
5:  Return $(M, U, U', f)$.
```

no larger matching contains M. In addition, the algorithm returns two sets U and U' to collect the vertices in the reconstructed edges and also returns a function f that pairs the vertices between U and U' to record the edges in M. We call $U \cup U'$ the *saturating set* of M for U, U' and M returned by the algorithm.

Third, we propose an algorithm (Algorithm 3) to reconstruct any graph on n vertices that contains only nontrivial paths, that are paths with at least one edge, in $2m \lg n + m + 5$ queries where m is the number of edges of the hidden graph. Figure 1 demonstrates an example of Algorithm 3.

Algorithm 3 FIND-ALL-PATHS

Let G be a graph on a set V of n vertices and contain only nontrivial paths.

```
1:  $E \leftarrow \emptyset$.
2:  if $Q(V) = 0$ then
3:      Return $\emptyset$.
4:  else
5:      Apply FIND-MAXIMAL-MATCHING on $G$. Assume $(M, U, U', f)$ is returned.
6:  end if
7:  $E \leftarrow E \cup M$, $I \leftarrow V \backslash (U \cup U')$.
8:  Apply FIND-MAXIMAL-MATCHING on $G[U]$ and $G[U']$. Assume $(M_1, A, A', f_1)$
    and $(M_2, B, B', f_2)$ are returned, respectively.
9:  $E \leftarrow E \cup M_1 \cup M_2$, $I_1 \leftarrow U \setminus (A \cup A')$, $I_2 \leftarrow U' \setminus (B \cup B')$.
10: for $u \in I_1$ do
11:     Apply a binary splitting algorithm on $I_2 \backslash \{f(u)\}$ with $u$ added to each test. Assume $v$
        (if any) is obtained from the search.
        $E \leftarrow E \cup \{\{u, v\}\}$, $I_1 \leftarrow I_1 \backslash \{u\}$, $I_2 \leftarrow I_2 \backslash \{v\}$.
12: end for
13: while $Q(I_1 \cup I) = 1$ do
14:     Reconstruct an edge in $G[I_1 \cup I]$, say $\{u, i\}$ where $u \in I_1$ and $i \in I$.
        $E \leftarrow E \cup \{\{u, i\}\}$, $I_1 \leftarrow I_1 \backslash \{u\}$.
15: end while
16: Reconstruct edges between $I_2$ and $I$ by the same way as lines 13–15.
17: Return $E$.
```

Lemma 2.1 *Algorithm 3 reconstructs any graph G on n vertices containing only nontrivial paths in $2m \lg n + m + 5$ queries where m is the number of edges of G.*

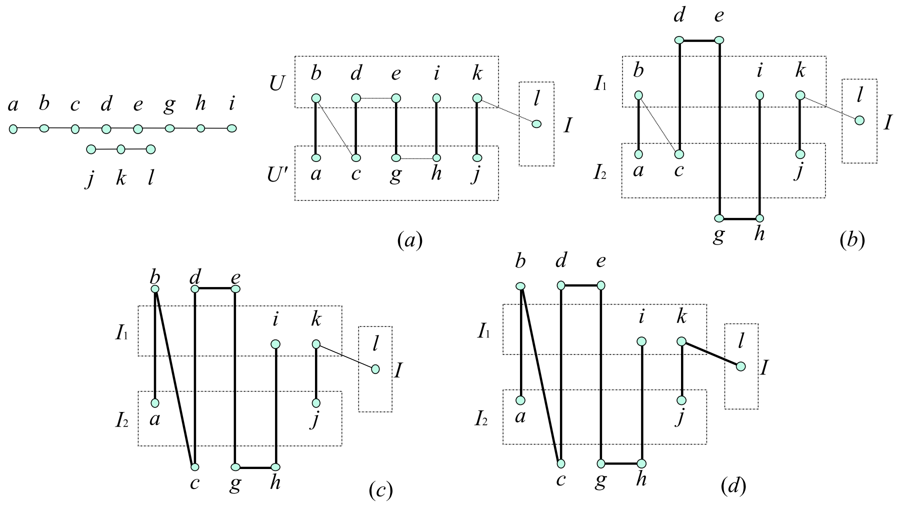


Fig. 1 (**a**) (line 5) The bold edges form a maximal matching and an independent set I is produced. (**b**) (lines 8–9) Reconstruct edges in $G[U]$ and $G[U']$. Then finally two independents sets $I_1 = \{b, i, k\}$ and $I_2 = \{a, c, j\}$ are obtained. (**c**) (lines 10–12) Reconstruct edges between I_1 and I_2. By applying a binary splitting algorithm to $I_2 \backslash \{f(b)\} = \{c, j\}$ with b added to each test, edge $\{b, c\}$ is reconstructed. Finally, $I_1 = \{i, k\}$ and $I_2 = \{a, j\}$. (**d**) Reconstruct edges between I and $I_1 \cup I_2$

Proof We first take at most $2|M|\lg n + 1$ queries to find a maximal matching M of G. Since M is a maximal matching of G and $U \cup U'$ is the saturating set of M, $E[U]$ and $E[U']$ are both matchings of G and I is an independent set. Thus FIND-MAXIMAL-MATCHING reconstructs $G[U]$ and $G[U']$ in at most $2(|M_1| + |M_2|)\lg n + 2$ queries (see line 8). Then the incident edges of all vertices in $A \cup A' \cup B \cup B'$ are reconstructed so it remains to reconstruct edges between three independent sets I, $I_1 = U \setminus (A \cup A')$, and $I_2 = U' \setminus (B \cup B')$.

The reconstructions of edges between I_1 and I_2 and edges between I and $I_1 \cup I_2$ can be done separately. Here, we deal with them sequentially. For the hidden edges between I_1 and I_2, since each $u \in I_1$ has at most one neighbor in $I_2 \backslash \{f(u)\}$, we can reconstruct them by applying a binary splitting algorithm to $I_2 \backslash \{f(u)\}$ with u included in each query for each $u \in I_1$. Actually, in this part, once an edge $\{u, v\}$ is reconstructed, we can remove u from I_1 and v from I_2 since their incident edges are reconstructed, making them become redundant in the reconstruction of edges between I and $I_1 \cup I_2$. Since $|I_1| \leq |M|$, there are at most $m_1 \lg n + |M|$ queries spent in this portion where m_1 is the number of edges reconstructed here.

Finally, we turn to reconstruct hidden edges between I and $I_1 \cup I_2$. Notice that they are all unreconstructed so far. By the symmetry of I_1 and I_2, we only discuss the reconstruction of edges between I and I_1. As shown in lines 13–14, we recursively reconstruct an edge in $G[I_1 \cup I]$, say $\{u, i\}$ where $u \in I_1$ and $i \in I$ and remove u from I_1 until $I \cup I_1$ induces no edge where u can be removed because both its incident edges are reconstructed after the reconstruction of $\{u, i\}$ and indeed the remove of u is to make sure that edges in $I \cup I_1$ are unreconstructed before each iteration. The number of queries spent here is at most $2m_2 \lg n + 2$ where m_2 is the number of edges between I and $I_1 \cup I_2$.

It is easily observed that each edge is reconstructed once and hence the overall cost of this algorithm is upper bounded by $2m \lg n + m + 5$. Therefore, the result follows. □

In the following, we introduce our main results on graph reconstruction problems including asymptotically optimal adaptive algorithms for reconstructing Hamiltonian cycles, matchings, stars, and cliques.

Assume that the hidden graph we want to reconstruct is a Hamiltonian cycle. Since there are $\frac{(n-1)!}{2}$ Hamiltonian cycles on n vertices, the theoretic information lower bound is $\lg \frac{(n-1)!}{2} \leq n \lg n$. Grebinski and Kucherov (1998) gave an adaptive algorithm to reconstruct a Hamiltonian cycle with $2n \lg n$ queries. We improve their result by providing an algorithm using at most $(1 + o(1))(n \lg n)$ queries.

An affine plane of order p is a balanced incomplete block design with p^2 elements and $p^2 + p$ blocks of size p such that each pair of elements appear together in exactly one block. It is well-known that an affine plane of order p exists whenever p is a prime power (see Anderson 1990). Using the so-called affine plane method together with the algorithm FIND-ALL-PATHS, the following theorem slightly improves the result in (Grebinski and Kucherov 1998).

Theorem 2.2 *For any hidden Hamiltonian cycle H of order n, H can be reconstructed in $n \lg n + cn$ queries for larger n and some small constant c.*

Proof Let p be the smallest prime power such that $p^2 \geq n$. Add $p^2 - n$ dummy vertices to obtain an affine plane and take them away when testing the blocks. A block is said to be positive if its testing result is positive. It is obvious that each positive block induces a graph containing only nontrivial paths; hence, a Hamiltonian cycle can be reconstructed by applying FIND-ALL-PATHS to these blocks (see an example in Table 1). Since there are $p^2 + p$ blocks and every edge exactly appears in a block, there are totally at most $(p^2 + p) + 2m \lg p + m + 5(p^2 + p)$ queries where $m = n$. Nagura (1952) proved that $p \leq 1.2q$ for two consecutive prime numbers $29 \leq q < p$ so there always exists a prime number p such that $n \leq p^2 \leq 1.44n < 2n$ for $n \geq 29^2$. Hence, a Hamiltonian cycle can be reconstructed within $12n + 6\sqrt{2n} + n \lg 2n + n < n \lg n + cn$ queries for larger n and some small constant c. □

A matching on n vertices is a graph with each vertex of degree 1 or 0 while a perfect matching on n vertices is a matching on n vertices which matches all vertices. The reconstruction of matchings on n vertices has been studied in (Alon and Asodi 2005; Bouvel et al. 2005). The number of perfect matchings on n vertices is $\frac{n!}{2^{\frac{n}{2}} \frac{n}{2}!}$ providing an information lower bound $\lg \frac{n!}{2^{\frac{n}{2}} \frac{n}{2}!} = (1 + o(1))(\frac{n}{2} \lg n)$ on the reconstruction of matchings. Bouvel et al. (2005) gave adaptive algorithms to reconstruct an unknown size matching and a perfect matching (a graph with each vertex of degree 1) on n vertices with $(1 + o(1))(n \lg n)$ and $(1 + o(1))(\frac{n}{2} \lg n)$ queries, respectively. We now exploit the affine plane method to reconstruct a matching using at most $(1 + o(1))(\frac{n}{2} \lg n)$ queries.

Theorem 2.3 *Reconstructing a matching on n vertices can be done in* $m \lg n + 3n$ *queries, where* $m \le \frac{n}{2}$ *is the number of edges of the matching.*

Proof As in the proof of Theorem 2.2, we first use the affine plane method. Since each block induces a graph containing just a matching, we simply apply FIND-MAXIMAL-MATCHING to each positive block (see an example in Table 1). Similarly, for n large, it takes at most $(p^2 + p) + 2m \lg p < 2n + \sqrt{2n} + m \lg 2n$ queries to reconstruct a matching on n vertices, where $m \le n/2$. □

We now give examples of small order to illustrate Theorem 2.2 and Theorem 2.3. Let $n = 7$. Then $p = 3$ is the smallest prime power such that its square is at least 7. $\{\{1,2,3\}$, $\{4,5,6\}$, $\{7,8,9\}$, $\{1,4,7\}$, $\{2,5,8\}$, $\{3,6,9\}$, $\{1,5,9\}$, $\{2,6,7\}$, $\{3,4,8\}$, $\{3,5,7\}$, $\{1,6,8\}$, $\{2,4,9\}\}$ is an affine plane of order 3. In Table 1, the hidden graph G_1 is a Hamiltonian cycle and the hidden graph G_2 is a matching. For G_1, $\{1,2,3\}$, $\{4,5,6\}$, $\{1,4,7\}$, $\{2,6,7\}$ and $\{3,4,8\}$ are positive blocks and we apply FIND-ALL-PATHS to each of them to reconstruct edges induced by any of them. For G_2, $\{1,2,3\}$, $\{4,5,6\}$ and $\{3,5,7\}$ are positive blocks and we apply FIND-MAXIMAL-MATCHING to each of them (see the corresponding cell in Table 1). Notice that the dummy vertex 8 is removed when testing the blocks. Based on the property of affine plane, the edge set of the hidden graph is decomposed into the edge sets of graphs induced by positive blocks, and therefore the whole graph is reconstructed by collecting edges induced by positive blocks.

A star is a graph where all its edges have a common incident vertex called center. A star of k edges can be defined by choosing a vertex as the center and other k vertices that are adjacent to it so the number of all stars on n vertices is upper bounded by $\sum_{k=2}^{n-1} n\binom{n-1}{k} + \frac{n(n-1)}{2} + 1 = n(2^{n-1} - 1) - \frac{n(n-1)}{2} + 1$. Accordingly, we obtain the information lower bound $(1+o(1))n$ for the number of queries required to reconstruct a hidden star. Bouvel et al. (2005) gave an adaptive algorithm using queries achieving the lower bound $\Omega(n)$. In fact, their algorithm requires $2n$ queries in the worst case. We now prove that the lower bound $(1 + o(1))n$ can be achieved by an adaptive algorithm.

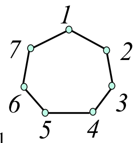
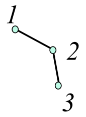
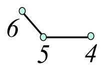
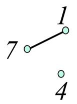
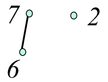
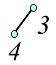
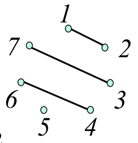
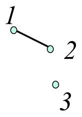
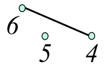
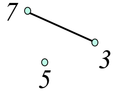
Theorem 2.4 *A star on n vertices can be reconstructed in* $n + 2\lg n$ *queries.*

Proof The fist step is to find the center of the star, and then to find all its neighbors by querying each vertex with the center. We first reconstruct an edge of the star in $2\lg n$ queries. It is easy to see that one of the two vertices in the edge is the center. Simply testing one of these two vertices together with all other vertices, we can determine which one is the center. Clearly, the process takes at most $2\lg n + n$ queries. □

Assume that the hidden graph we consider here is a clique. It is easily seen that the information lower bound is $\lg 2^n = n$. Bouvel et al. (2005) provided an adaptive algorithm to reconstruct a clique in $2n$ queries. We slightly improve their result by giving an algorithm with at most $n + \lg n$ queries in the worst case.

Theorem 2.5 *A clique on n vertices can be reconstructed in* $n + \lg n$ *queries.*

Table 1 A cell is empty means the corresponding block is not positive, i.e., the graph induced by it contains no edge

hidden graph\block	{1, 2, 3}	{4, 5, 6}	{1, 4, 7}	{2, 6, 7}	{3, 4, 8}	{3, 5, 7}	others
 G_1							
 G_2							

Proof By applying FIND-ONE-VERTEX, we can find a vertex x on the clique in $\lg n$ queries and then the clique can be reconstructed by querying each vertex with x. The whole process takes at most $n + \lg n$ queries. □

3 Reconstructing hypergraphs and threshold group testing

Given a finite set X, a hypergraph $H = (X, \mathcal{F})$ is a family $\mathcal{F} = \{E_1, E_2, \ldots, E_m\}$ of subsets of X. The elements of X are called vertices, the subsets E_i's are the edges of the hypergraph H, and $|X|$ is the order of H. In this section we study the graph reconstruction problem where the hidden hypergraph is a k*-hyperclique*, that is, every k-subset of its vertices forms an edge. Note that the information lower bound of reconstructing a k-hyperclique is $\lg 2^n = \Omega(n)$.

It is not difficult to see that the algorithm FIND-ONE-VERTEX also returns a vertex in some edge of the hidden graph under the hypergraph version. By extending the idea used to reconstruct a simple graph, we have the following result.

Theorem 3.1 *A k-hyperclique on n vertices can be reconstructed using at most $k \lg n + n$ queries.*

Proof Initially, we can find a vertex in the k-hyperclique in $\lg n$ queries. Suppose that we have found B, a set of vertices in the hypergraph. Apply FIND-ONE-VERTEX with $S \leftarrow V \setminus B$ and $A \leftarrow V \setminus B$ as its initial condition and include the vertices in B for each query. Then we can find a further vertex in the hypergraph if there remains one. So, we can find $k - 1$ vertices in the hidden k-hyperclique in $k \lg n$ queries. Furthermore, by testing those $k - 1$ vertices together with each of the other vertices separately, we can reconstruct the whole graph. This completes the proof. □

In case of that the order of the hidden k-hyperclique is at most p, the following lemma shows that there is a big reduction in the query complexity if $p \ll n$.

Lemma 3.2 *A k-hyperclique of order at most p on n vertices can be reconstructed in $O((p + k) \lg n)$ queries.*

Proof Similar to Theorem 3.1, we can find $k - 1$ vertices in this hidden k-hyperclique in $k \lg n$ queries. Then we can introduce the ordinary group testing strategy here to find all other vertices (positive elements) by treating these $k - 1$ vertices as positive elements and using them as an auxiliary pool. Since $O(p \lg n)$ is a query bound for group testing (Du and Hwang 2006), the result follows. □

Now, we turn to the threshold group testing problem without gap, i.e., $u = l + 1$. Let p denote the number of positive elements in the given n elements. Then, the information lower bound for this problem is $\lg \binom{n}{p} = O(p \lg n)$. We now prove that the bound $O(p \lg n)$ can be achieved in the following theorem, which improves Demeschke's result $O((p + u^2) \lg n)$.

Theorem 3.3 *There is an asymptotically optimal algorithm with $O(p \lg n)$ tests for the threshold group testing problem without gap.*

Proof The problem can be reduced to reconstructing a hidden u-hyperclique of order at most p on n vertices. Then by applying Lemma 3.2, the theorem follows. □

4 Concluding remarks

Threshold group testing is a generalization of group testing and the setting of thresholds seem to be natural in some chemical scenarios, and thus the threshold group testing problem is worth studying. In particular, one cannot simply test individual items and then identify all positives using a trivial strategy; instead, one has to test subsets of items. In view of this, it is intuitive to consider the connection between the two seemingly unrelated problems: threshold group testing and graph search. Chen and Fu (2009) first observed the relation between threshold group testing and graph search. Consequently, they showed that generalized disjunct matrices, a standard tool for graph search, can also be useful in constructing nonadaptive algorithms for the threshold group testing. In this paper, we propose an optimal algorithm for the case without gap by using the strategy of getting smaller and smaller sets that contain a positive. In doing this, we can identify each positive in $O(\lg n)$ tests. However, the strategy we use here does not work for the case when there is a gap between the thresholds. The reason for this is that in the duration a positive outcome indicates that $l+1$ or more positives are in that test, while a negative one cannot immediately help us to recognize a smaller set containing at least a positive and then continue this strategy recursively to narrow down the candidate set.

Acknowledgement The authors would like to express their gratitude to the referees for their helpful comments.

References

Aigner M (1988) Combinatorial search. Wiley, New York
Alon N, Asodi V (2005) Learning a hidden subgraph. SIAM J Discrete Math 18:697–712
Alon N, Beigel R Kasif S, Rudich S, Sudakov B (2004) Learning a hidden matching. SIAM J Comput 33:487–501
Anderson I (1990) Combinatorial designs: construction methods. Ellis Horwood, Chichester
Angluin D, Chen J (2006) Learning a hidden hypergraph. J Mach Learn Res 7:2215–2236
Angluin D, Chen J (2008) Learning a hidden graph using $O(\log n)$ queries per edge. J Comput Syst Sci 74:546–556
Beigel R, Alon N, Apaydin MS, Fortnow L, Kasif S (2001) An optimal procedure for gap closing in whole genome shotgun sequencing. In: Proceedings 2001 RECOMB. ACM, New York, pp 22–30
Bouvel M, Grebinski V, Kucherov G (2005) Combinatorial search on graphs motivated by bioinformatics applications: a brief survey. Lect Notes Comput Sci 3787:16–27
Chen HB, Fu HL (2009) Nonadaptive algorithms for threshold group testing. Discrete Appl Math 157:1581–1585
Damaschke P (2006) Threshold group testing. Lect Notes Comput Sci 4123:707–718
Du DZ, Hwang FK (2006) Pooling designs and nonadaptive group testing—important tools for DNA sequencing. World Scientific, Singapore

Grebinski V, Kucherov G (1998) Reconstructing a Hamiltonian cycle by querying the graph: Application to DNA physical mapping. Discrete Appl Math 88:147–165

Grebinski V, Kucherov G (2000) Optimal reconstruction of graphs under the additive model. Algorithmica 28:104–124

Nagura J (1952) On the interval containing at least one prime number. Proc Jpn Acad 28:177–181

Sorokin A, Lapidus A, Capuano V, Galleron N, Pujic P, Ehrlich SD (1996) A new approach using multiplex long accurate PCR and yeast artificial chromosomes for bacterial chromosome mapping and sequencing. Genome Res 6:448–453

Torney DC (1999) Sets pooling designs. Ann Comb 3:95–101