Available online at www.sciencedirect.com

ScienceDirect

Discrete Applied Mathematics 155 (2007) 566–571

DISCRETE APPLIED MATHEMATICS

www.elsevier.com/locate/dam

Note

A competitive algorithm in searching for many edges in a hypergraph

Ting Chen[a,*], Frank K. Hwang[b]

[a]*College of Statistics and Mathematics, Zhejiang Gongshang University, Hangzhou, PR China*
[b]*Department of Applied Mathematics, National Chiaotung University, Hsinchu, Taiwan, ROC [1]*

Received 3 October 2004; received in revised form 23 March 2006; accepted 11 July 2006
Available online 10 October 2006

Abstract

We give a competitive algorithm to identify all d defective edges in a hypergraph with d unknown. Damaschke did the $d = 1$ case for 2-graphs, Triesch extended the $d = 1$ case to r-graphs, and Johann did the general d case for 2-graphs. So ours is the first attempt to solve the searching for defective edges problem in its full generality. Further, all the above three papers assumed d known. We give a competitive algorithm where d is unknown.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Competitive algorithm; Hypergraph; Edge-searching problem; Group testing; Graph searching

1. Introduction

A hypergraph is of *rank r* if the maximum number of vertices in an edge is r. In particular, if every edge has r vertices, then it is known as an *r-graph*. We will represent a hypergraph G by its edge-set E, while $V(E)$ denotes the set of vertices in E. We also let $G(S)$ denote the subgraph of G induced by the set S of vertices.

In our problem, we want to identify a subset $D \subseteq E$ of *defective edges* with a minimum number of *edge tests*, where an edge test takes an arbitrary subset S of $V(E)$ and asks whether the subgraph $G(S)$ contains a defective edge. Note that the edge test is an extension of the group test which has a much longer history [4]. In a group-testing problem, a vertex is either defective or good. A group test takes an arbitrary subset S of V and asks whether S contain a defective vertex. The group-testing problem can be interpreted as an edge-testing problem on a 1-graph where a 1-edge is just a vertex.

For a brief history of edge testing, Chang and Hwang [2] first cast a special group-testing problem as searching for a single edge in a complete bipartite graph. Since a group test is convertible to an edge test in the single-edge case, this problem can be interpreted as the first formulation of an edge-testing problem. However, Aigner [1] was the first who consciously proposed the edge-testing problem for general 2-graphs, thus bringing the "graph" into focus. Let

* Corresponding author.
E-mail addresses: steafting@sina.com (T. Chen), fhwang@math.nctu.edu.tw (F.K. Hwang).
[1] This research was done when the second author was visiting Center of Mathematical Sciences, Zhejiang University, Hangzhou, Zhejiang, PR China. The first author was supported by National Natural Science Foundation of China (10271110).

0166-218X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2006.07.008

$M(E, d)$ denote the minimum number of edge tests guaranteeing to find the d defective edges in E. He conjectured for 2-graphs

$$M(E, 1) \leqslant \lceil \log_2 |E| \rceil + c \quad \text{where } c \text{ is a constant.}$$

Damaschke [3] proved the conjecture with $c = 1$. Triesch [8] generalized to hypergraphs (of rank r) with $c = r - 1$.

Du and Hwang [4] conjectured for 2-graphs

$$M(E, d) \leqslant d\left(\left\lceil \log_2 \frac{|E|}{d} \right\rceil + c\right) \quad \text{where } c \text{ is a constant.}$$

Recently, Johann [6] made a breakthrough into the $d > 1$ problem by proving for 2-graphs

$$M(E, d) \leqslant d\left(\left\lceil \log_2 \frac{|E|}{d} \right\rceil + 7\right).$$

All above results assumed that d is known. For the case that d is unknown, Hwang [5] gave a competitive algorithm requiring $d(\lceil \log_2 |E| \rceil + 4)$ tests.

In this paper, we study the problem of searching for all defective edges in an r-graph and give a competitive algorithm (guaranteeing its performance against all d) requiring

$$d\lceil \log_2 |E| \rceil + (r-1)^{\lfloor \frac{r}{2} \rfloor} d^r + o(d^r)$$

tests, where r is assumed to be a small constant. Note that the information-theoretic lower bound with d known is

$$\log_2 \binom{|E|}{d} \sim d(\log_2 |E| - \log_2 d).$$

Then we extend it to general hypergraphs.

To motivate our problem, we cite a biological application [7]. Many biological phenomena occur due to the joint presence of several molecules. Suppose there exists a set D where each member is a set of molecules whose joint presence would cause a particular biological phenomenon. Then we can construct a hypergraph by taking the set of relevant molecules as vertices, all subsets of vertices which are candidates of members of D as edges, and the unknown set D as the set of defective edges to be identified. There exist biological experiments which can recognize the existence of a set of molecules causing the particular phenomenon.

2. The general approach for r-graphs

We follow Johann's approach in general, but also have to resolve some problems unique to r-graphs for $r > 2$ and to competitive algorithms. Three important issues are the following:

(i) After a *positive test* (a test with a positive outcome), how do we identify a defective edge efficiently?
(ii) A defective edge can be removed only by removing some vertices in it, which is undesirable since there may be other defective edges incident to these vertices. But if we do not remove defective edges, how do we avoid repetitively identifying the same defective edge?
(iii) How do we analyze the number of tests?

We deal with these issues one by one.

(i) Johann commented that for $r = 2$, Triesch's algorithm to find the single defective edge in E by $\lceil \log_2 |E| \rceil + 1$ tests works with a little modification even if E contains many defective edges. In a private communication, Johann actually revealed that the modification works for general hypergraphs with rank r.

In the following, we describe Johann's edge-testing version of Triesch's algorithm. Triesch used group testing to describe the testing scheme which is known to be convertible to edge testing if $|D| = 1$. To be more specific, let $C = (v_1, \ldots, v_c)$ be a vertex cover on E. Then each edge contains a vertex of C. We say an edge e is *led* by v_i if e contains no $v_j \in C$ with $j < i$. Triesch's group tests are always on $\{v_1, \ldots, v_j\}$ for some j, which can be converted to edge tests on the complementary sets, i.e., on $V \backslash \{v_1, \ldots, v_j\}$. The testing ends when a leader of a defective edge is identified. Then the problem is reduced to finding a defective $(r-1)$-edge which can be done by induction. We will refer to Johann's edge-testing version as the *TJ-procedure*. We now formally state the TJ-procedure. Let V denote the vertex set of E.

Step 1: Use the greedy algorithm (as specified by Triesch) to construct a vertex cover $C=(v_1,\dots,v_c)$ of E with $d(v_1)\geqslant d(v_2)\geqslant\cdots\geqslant d(v_c)$ where $d(v_i)$ is the degree of vertex i in the graph obtained from E by deleting $v_1,\dots,v_{i-1}$ and their edges. Let T be the rooted binary tree with leaves $v_1,\dots,v_c$ from left to right in that order, and the path of v_i has length $\lceil\log_2\frac{|E|}{d(v_i)}\rceil$, $1\leqslant i\leqslant c$ (the existence of T is guaranteed by Kraft's inequality). Let T_{L} and T_{R} denote the left and right subtree of T, and V_{L} and V_{R} their leave-sets, respectively.

Step 2: Test $V\backslash V_{\mathrm{L}}$. If positive, set $T=T_{\mathrm{R}}$ and $V=V\backslash V_{\mathrm{L}}$; if negative, set $T=T_{\mathrm{L}}$.

Step 3: If T has more than one leave, go back to Step 2.

Step 4: Output the only leave in T as the leader in a defective edge.

Note that Triesch's procedure always identifies the defective edge with the minimum leader while Johann's edge-testing version identifies the one with the maximum leader. When $d=1$, the minimum leader is the maximum leader, so both procedures are reduced to the same $(r-1)$-graph. But for $d>1$, they are reduced to different $(r-1)$-graphs. So not only the number of tests in finding the minimum and maximum leader can be different, the subsequent reductions can also lead to different number of tests. However, these differences disappear in our worst-case analysis. By the above explanation and since Triesch's algorithm requires at most $\lceil\log_2|E|\rceil+r$ tests, we have:

Lemma 1 (*TJ-procedure*). *Let E be a hypergraph of rank r, i.e., $|e|\leqslant r$ for all edges $e\in E$. There exists an algorithm which finds one of several defective edges in E with at most $\lceil\log_2|E|\rceil+r$ edge-tests.*

(ii) The working of the TJ-procedure is based on the premise that E does not contain a defective edge e already identified. Because if it did, then the procedure might identify e repeatedly. To prevent this from happening, as soon as a defective edge is identified, we must partition its vertices into different subsets such that a future test on vertices of a given subset will not encounter an identified defective edge. This is the purpose of the partition stage in Johann's procedure.

However, we still need to identify defective edges whose vertices spread into different subsets. But testing vertices from different subsets may bring back identified defective edges. For $r=2$, each defective edge involves only two vertices. This problem was cleverly handled by Johann by mixing vertices from two subsets only when one of them contains a single vertex. Say, the two subsets are V_i and V_j, and v is the single vertex from V_j. Then we can avoid any identified defective edges (v,u), $u\in V_i$ by deleting (temporarily) u from V_i. For $r>2$, the problem is much more complicated.

Instead of taking one vertex from outside (of V_i) to test with vertices in V_i, we have to take a set K of k vertices from outside to test with vertices in V_i since such a combination of vertices may contain a defective edge. We do this by fixing a K and then searching for all defective edges containing this K. This is achieved by requiring every test to contain K. Let $\mathrm{CH}(r)$ denote our proposed algorithm for r-graphs. Suppose we test $K\cup S$, where S is a subset of the vertex set of $V\backslash K$. Then we can call the subroutine $\mathrm{CH}(r-k)$ to search for all induced defective edges e' (with rank $r-k$) such that $e'\cup K$ is a defective edge.

We have to avoid identifying a defective edge contained in $K\cup S$ which is already identified before the induction stage $\mathrm{CH}(r-k)$ is reached. This means that all such identified defective edges must be broken into at least two subsets which are not to be mixed in tests. On the other hand, these subsets may contain an induced defective edge not identified yet, thus we need to mix them in tests. This dilemma is solved by recursively partitioning V_i in a sequence of subroutines until the induced defective edge is of rank 1, namely, $r-1$ vertices have been specified to attach to each test in the subroutine. Then any vertex in V_i which together with the $r-1$ specified vertices constitute an identified defective edge can be deleted from V_i. Since the breaking up of an identified defective edge for general r has more implications than the $r=2$ case, we have to replace the more efficient but complicated multi-subset scheme in [6] by a 2-subset scheme.

For $r'<r$, since $\mathrm{CH}(r')$ may be called as a subroutine, we need to solve $\mathrm{CH}(r')$ in a more general context. Suppose $\mathrm{CH}(r')$ has vertex-set V'. Then a set K not in V' is imposed so that all defective edges in $\mathrm{CH}(r')$ are induced by K, i.e., the set of induced defective edges in $\mathrm{CH}(r')$ is $\{X\backslash K\colon$ where X is a defective edge containing $K\}$. If $\mathrm{CH}(r')$ is not used as a subroutine, i.e., $r'=r$, then $K=\emptyset$.

(iii) We assume that r is small and can be treated as a constant. Thus the number of tests will be represented as a function of $|V|$, $|E|$ and d. As in Johann's algorithm, we will bound the number of tests in identifying each defective edge, including *negative tests* (tests with a negative outcome) occurred during the process. Since a positive test always initiates the TJ-procedure and is counted as a test of the procedure, all positive tests are counted as tests in identifying defective edges. Further, some negative tests also occur in that procedure and are counted. So the analysis is reduced to counting negative tests occurred elsewhere. Unlike Johann's algorithm, we do not attempt to optimize the size of negative tests (we cannot since we do not know d), and we do not associate each negative test with the identification of a defective edge. The consequence is a very simple algorithm and analysis. The price we pay is that each defective edge consumes $\lceil \log_2 |E| \rceil$ tests instead of $\lceil \log_2 \frac{|E|}{d} \rceil$ tests as Johann obtained (we suspect that a competitive algorithm cannot achieve the latter).

3. An algorithm for r-graphs

Let $E_0(K) = \{e | e \cup K \subseteq E,\ e \subseteq V_0\}$. Let K_i denote the subset imposed on $\mathrm{CH}(i)$, i.e., K_i is a part of every test in $\mathrm{CH}(i)$. If the original problem is defined on an r-graph, then $|K_i| = r - i$. Finally, let I denote the set of currently identified defective edges (in the original problem). For a set W of vertices, let $I(W)$ denote the subset of I formed by those edges whose vertices all belong to W.

We give an algorithm $\mathrm{CH}(r)$ recursively. We first define $\mathrm{CH}(1)$.

```
Algorithm CH(1)
Input: $E$, $K_1$, $V_0$, $V_1$, $I$ (If CH(1) is not a subroutine, then $V_0 = V(E)$, $K_1 = V_1 = I = \emptyset$). Attach $K_1$ to every test.
Step 1: Test $V_0$. If positive, use the halving procedure, which we will treat as a special case of the TJ-procedure, to
        identify a defective vertex $u$ ($e = K_1 \cup \{u\}$). Set $I = I \cup \{e\}$, $V_0 = V_0 \backslash \{u\}$ and go back to Step 1.
Step 2: For every vertex $v \in V_1$, test $K_1 \cup \{v\}$. If positive, set $I = I \cup \{K_1 \cup \{v\}\}$.
Step 3: Stop.
  Algorithm CH($r'$) ($r \geqslant r' \geqslant 2$)

The partition stage:
Input: $E$, $K_{r'}$, $V_0$, $V_1$, $I$ (If CH($r'$) is not a subroutine, i.e., $r' = r$, then $V_0 = V(E)$ and $V_1 = K_{r'} = I = \emptyset$). Attach $K_{r'}$
        to every test.
Step 1: Test $V_0$. If positive, use the TJ-procedure to identify a defective edge $e = \{v_1, v_2, \ldots, v_{r'}\} \subseteq V_0$. Add the vertex
        $v_1$ to $V_1$. Set $V_0 = V_0 \backslash \{v_1\}$ and $I = I \cup \{\{e\} \cup K_{r'}\}$. If $|V_0| \geqslant r'$, go back to Step 1.
Step 2: Stop.
  Suppose $V_0$, $V_1$ are nonempty. Go to the search stage.

The search stage:
Step 1: Set $k = 1$.
Step 2: Let $K$ be a $k$-subset of $V_1$. Set $K_{r'-k} = K_{r'} \cup K$. Construct a vertex cover $C$ ($C \cap K_{r'-k} = \emptyset$) on $I(K_{r'-k} \cup$
        $V(E_0(K_{r'-k})))$. Call subroutine CH($r' - k$) with $E = E_0(K_{r'-k})$, $V = V(E_0(K_{r'-k}))$, $V_0 = V(E_0(K_{r'-k}))\backslash C$
        and $V_1 = C$. If for some $v \in V_0$, $K_{r'-k} \cup \{v\} \in I(K_{r'-k} \cup V(E_0(K_{r'-k})))$ (possible only for $k = r' - 1$), delete
        $v$ from $V_0$.
        Do this for all $k$-subsets $K$. Set $k = k + 1$. If $k < r'$, go back to Step 2.
Step 3: Test all $r'$-subsets $S$ (except those such that $S \cup K_{r'} \in I$) of $V_1$. If positive, set $I = I \cup \{S \cup K_{r'}\}$.
---------------------------------------------------------------------------------------------------------------
        We will refer to tests in Step 3 as direct tests.
Step 4: Stop.
```

Theorem 1. *Let E be an arbitrary r-graph which contains d defective edges, where d is not necessarily known. Then the algorithm* $\mathrm{CH}(r)$ *identifies all defective edges of E with at most $d\lceil \log_2 |E| \rceil + (r-1)^{\lfloor \frac{r}{2} \rfloor} d^r + o(d^r)$ tests.*

Proof. Clearly, all edges identified as positive by the algorithm are through either the TJ-procedure or direct tests, both are error-free. Thus it suffices to prove that a defective edge is always identified.

Suppose a defective edge with vertex set X is not identified at the partition stage of $\mathrm{CH}(r)$. Then a nonempty subset $X' \subseteq X$ must lie in V_1. At the search stage, the selection of K runs through all k-subsets of V_1 for $1 \leqslant k \leqslant r$. One such selection is $K = X'$. Suppose $|X'| = k$. Then the problem is reduced to the subroutine CH$(r-k)$ with K imposed. By induction on r, the induced defective edge $X \backslash K$ can be identified in the subroutine, which implies $X\backslash K \cup K = X$ is a defective edge.

It remains to count the number of tests CH(r) uses. By Lemma 1, the TJ-procedure uses at most $\lceil \log_2 |E| \rceil + r$ tests. Since a defective edge is identified by either the TJ-procedure or a direct test, the number of tests consumed in identifying one defective edge is bounded by $\lceil \log_2 |E| \rceil + r$. This bounded number of tests includes the possible positive test initiating the identification process, and all negative tests occurred during the process of identifying the defective edge. Thus, the number of tests identifying d defective edges is at most $d(\lceil \log_2 |E| \rceil + r)$. Further it suffices to count the number $N(r)$ of negative tests occurred elsewhere in CH(r). There are three sources for tests in $N(r)$: one negative test from the partition stage, those from subroutines and the direct tests.

Denote D_K as the set of all defective edges in $K \cup V(E_0(K))$. Let $d_K = |D_K|$. Note that for $K \neq K'$, D_K and $D_{K'}$ may overlap in defective edges containing some vertices in $K \cap K'$ and other vertices in $V(E_0(K))$. Hence we can only bound d_K by d. However, for $|K| = 1$, D_K and $D_{K'}$ are disjoint; hence $\sum_{K:|K|=1} d_K$ is bounded by d. We count the number $N(r)$ of negative tests in CH(r) by induction on r.

For $r = 1$, $N(r)$ is easily verified to be at most $1 + d_K$ since the negative tests counted in $N(r)$ consists of 1 at the end of Step 1 and occur in Step 2; but $|V_1| \leqslant d_K$. So Theorem 1 holds for $r = 1$.

Note that a vertex is in V_1 either because it is in the vertex cover C (which covers the set of identified defective edges), or it is a vertex in a defective edge identified from the updated V_0 at the partition stage. Thus every vertex in V_1 corresponds to a distinct defective edge; hence $|V_1| \leqslant d$.

We prove the general $r \geqslant 2$ case by induction.

$$\begin{aligned}
N(r) &\leqslant 1 + \sum_{k=1}^{r-2} \sum_{K \subseteq V_1:|K|=k} N(r-k) + \sum_{K \subseteq V_1:|K|=r-1} N(1) + \binom{|V_1|}{r} \\
&\leqslant \sum_{k=1}^{r-2} \sum_{K \subseteq V_1:|K|=k} \left((r-k-1)^{\lfloor \frac{r-k}{2} \rfloor} d_K^{r-k} + \mathrm{o}(d_K^{r-k})\right) + \sum_{K \subseteq V_1:|K|=r-1} (1+d_K) + d^r \\
&\leqslant \sum_{K \subseteq V_1:|K|=1} (r-2)^{\lfloor \frac{r-1}{2} \rfloor} d_K^{r-1} + \sum_{k=2}^{r-2} \sum_{K \subseteq V_1:|K|=k} (r-k-1)^{\lfloor \frac{r-k}{2} \rfloor} d_K^{r-k} + d^{r-1}(1+d) + d^r \\
&\quad + \mathrm{o}(d^r) \\
&\leqslant (r-2)^{\lfloor \frac{r-1}{2} \rfloor} \left(\sum_{K \subseteq V_1:|K|=1} d_K \right)^{r-1} + (r-3)^{\lfloor \frac{r}{2} \rfloor - 1} \sum_{k=2}^{r-2} \binom{d}{k} d^{r-k} + 2d^r + \mathrm{o}(d^r) \\
&\leqslant (r-3)^{\lfloor \frac{r}{2} \rfloor - 1} (r-3) d^r + 2d^r + \mathrm{o}(d^r) \\
&= ((r-3)^{\lfloor \frac{r}{2} \rfloor} + 2) d^r + \mathrm{o}(d^r) \\
&\leqslant (r-1)^{\lfloor \frac{r}{2} \rfloor} d^r + \mathrm{o}(d^r).
\end{aligned}$$

Thus, $N(r) \leqslant (r-1)^{\lfloor \frac{r}{2} \rfloor} d^r + \mathrm{o}(d^r)$ holds for general r.

Let $T(r)$ denote the total number of tests required by CH(r). Since $T(r) \leqslant d(\lceil \log_2 |E| \rceil + r) + N(r)$ and $N(r) \leqslant (r-1)^{\lfloor \frac{r}{2} \rfloor} d^r + \mathrm{o}(d^r)$, we have that $T(r) \leqslant d\lceil \log_2 |E| \rceil + (r-1)^{\lfloor \frac{r}{2} \rfloor} d^r + \mathrm{o}(d^r)$ holds for $r \geqslant 2$. Therefore, algorithm CH(r) needs at most $d\lceil \log_2 |E| \rceil + (r-1)^{\lfloor \frac{r}{2} \rfloor} d^r + \mathrm{o}(d^r)$ tests to identify all d defective edges in E. □

Unfortunately, we are not able to provide more specific functions than the o() function we used in derivation.

4. An algorithm for hypergraphs

Let E be a hypergraph of rank r, i.e., $|e| \leqslant r$ for all edges $e \in E$. It is assumed that no defective edge is contained in another (this is an assumption made in all models dealing with the quoted biological application). To identify the set $D \subseteq E$ in a hypergraph, we follow the general approach in algorithm CH(r) for r-graphs with a slight modification. Let CH*(r) denote the algorithm for hypergraphs of rank r.

The search stage in $\mathrm{CH}^*(r)$ will be a little different from $\mathrm{CH}(r)$. When we choose a k-subset K of V_1 before constructing a vertex cover and then calling $\mathrm{CH}^*(r-k)$, we should test K itself. If the outcome is positive, then $K \in D$. By our assumption, there is no other defective edge containing K, so we do not need to call $\mathrm{CH}^*(r-k)$ further. If the outcome is negative, call $\mathrm{CH}^*(r-k)$ to identify all induced defective edges in $E_0(K)$.

Algorithm $\mathrm{CH}^*(1)$ is same as $\mathrm{CH}(1)$. Now, we give the algorithm $\mathrm{CH}^*(r)$ recursively.

Algorithm $\mathrm{CH}^*(r')$ $(r \geqslant r' \geqslant 2)$
Input: E, $K_{r'}$, V_0, V_1, I (If $\mathrm{CH}^*(r')$ is not a subroutine, then $V_0 = V(E)$ and $V_1 = K_{r'} = I = \emptyset$).

The partition stage:
Step 1: Test V_0. If positive, use the TJ-procedure to identify a defective edge $e = \{v_1, v_2, \ldots, v_s\} \subseteq V_0$ $(s \leqslant r')$. Add the vertex v_1 to V_1. Set $V_0 = V_0 \backslash \{v_1\}$ and $I = I \cup \{\{e\} \cup K_{r'}\}$. If $V_0 \neq \emptyset$, go back to Step 1.
Step 2: Stop.
Suppose V_0, V_1 are nonempty. Go to the search stage.

The search stage:
Step 1: Set $k = 1$.
Step 2: Choose a k-subset K of V_1, where $G(K_{r'} \cup K)$ does not contain any identified defective edge in I. Set $K_{r'-k} = K_{r'} \cup K$.
Test $K_{r'-k}$. If positive, let $I = I \cup \{K_{r'-k}\}$.
Else construct a vertex cover C $(C \cap K_{r'-k} = \emptyset)$ on $I(K_{r'-k} \cup V(E_0(K_{r'-k})))$. Call subroutine $\mathrm{CH}^*(r'-k)$ with $E = E_0(K_{r'-k})$, $V = V(E_0(K_{r'-k}))$, $V_0 = V(E_0(K_{r'-k})) \backslash C$ and $V_1 = C$. If for some $v \in V_0$, $K_{r'-k} \cup \{v\} \in I(K_{r'-k} \cup V(E_0(K_{r'-k})))$ (possible only for $k = r'-1$), delete v from V_0. Attach $K_{r'-k}$ to any test in $\mathrm{CH}^*(r'-k)$.
Do this for all k-subsets K. Set $k = k+1$. If $k < r'$, go back to Step 2.
Step 3: Test all r'-subsets S (except those such that $S \cup K_{r'} \in I$) of V_1. If positive, set $I = I \cup \{S \cup K_{r'}\}$.

We will also refer to tests in Step 3 as *direct tests*.
Step 4: Stop.

Theorem 2. *Let E be a hypergraph of rank r with d defective edges, where d is not necessarily known. Then the algorithm* $\mathrm{CH}^*(r)$ *identifies all defective edges in E with at most* $d\lceil \log_2 |E| \rceil + (r-1)^{\lfloor \frac{r}{2} \rfloor} d^r + o(d^r)$ *tests.*

Proof. Similar to the proof of Theorem 1, we can show that $\mathrm{CH}^*(r)$ identifies all defective edges of the hypergraph E.

To count the number of tests $\mathrm{CH}^*(r)$ uses, let $N^*(r)$ and $T^*(r)$ be the counterparts of $N(r)$ and $T(r)$ in $\mathrm{CH}^*(r)$. The analysis of the test number of $\mathrm{CH}^*(r)$ is also similar to that of $\mathrm{CH}(r)$. The only difference is that the subroutine of $\mathrm{CH}^*(r)$ should need $N^*(r-k)+1$ tests instead of $N(r-k)$ tests in $\mathrm{CH}(r)$. But it does not change the result; so $N^*(r) \leqslant (r-1)^{\lfloor \frac{r}{2} \rfloor} d^r + o(d^r)$. Consequently, $T^*(r) \leqslant d\lceil \log_2 |E| \rceil + (r-1)^{\lfloor \frac{r}{2} \rfloor} d^r + o(d^r)$ holds for $r \geqslant 2$. Therefore, algorithm $\mathrm{CH}^*(r)$ needs at most $d\lceil \log_2 |E| \rceil + (r-1)^{\lfloor \frac{r}{2} \rfloor} d^r + o(d^r)$ tests to identify all d defective edges of E. □

Acknowledgment

We thank Dr. P. Johann for a careful reading of the paper and many helpful suggestions. We also thank the reviewers for their efforts to improve the readability of this paper.

References

[1] M. Aigner, Combinatorial Search, Wiley-Teubner Series in Computer Science, Wiley, New York, 1988.
[2] G.J. Chang, F.K. Hwang, A group testing problem on two disjoint sets, SIAM J. Algebraic Discrete Methods 2 (1981) 35–38.
[3] P. Damaschke, A tight upper bound for group testing in graphs, Discrete Appl. Math. 48 (1994) 101–109.
[4] D.Z. Du, F.K. Hwang, Combinatorial Group Testing and its Applications, World Scientific, Singapore, 1993.
[5] F.K. Hwang, A competitive algorithm to find all defective edges in a graph, Discrete Appl. Math. 148 (2005) 273–277.
[6] P. Johann, A group testing problem for graphs with several defective edges, Discrete Appl. Math. 117 (2002) 99–108.
[7] D.C. Torney, Set pooling designs, Ann. Combin. 3 (1999) 95–101.
[8] E. Triesch, A group testing problem for hypergraphs of bounded rank, Discrete Appl. Math. 66 (1996) 185–188.