MODIFICATIONS OF COMPETITIVE GROUP TESTING*

D.-Z. DU†, G.-L. XUE†, S.-Z. SUN† AND S.-W. CHENG†

Abstract. Many fault-detection problems fall into the following model: There is a set of n items, some of which are defective. The goal is to identify the defective items by using the minimum number of tests. Each test is on a subset of items and tells whether the subset contains a defective item or not. Let \( M_\alpha(d, n) \) (\( M_\alpha(d | n) \)) denote the maximum number of tests for an algorithm \( \alpha \) to identify \( d \) defectives from a set of \( n \) items provided that \( d \), the number of defective items, is known (unknown) before the testing. Let \( M(d, n) = \min_\alpha M_\alpha(d, n) \). An algorithm \( \alpha \) is called a competitive algorithm if there exist constants \( c \) and \( a \) such that for all \( n > d > 0 \), \( M_\alpha(d, n) \leq cM(d, n) + a \). This paper confirms a recent conjecture that there exists a bisecting algorithm \( A \) such that \( M_A(d | n) \leq 2M(d, n) + 1 \). Also, an algorithm \( B \) such that \( M_B(d | n) \leq 1.65M(d, n) + 10 \) is presented.

Key words. group testing, competitive algorithm

AMS subject classification. 68P10

1. Introduction. A feature of on-line problems [12], [15] is that information about input is not completely given at the beginning but is collected during the process of seeking a solution. This feature makes an optimal solution very hard to attain; an option is to consider competitive algorithms, which take responsibility for producing a reasonable solution. A similar situation occurs in some searching problems.

Consider a set of \( n \) items. Some items are defective, and others are good. The problem is to identify the defective items by a sequence of tests. Each test is on a subset of items and tells us whether the subset contains a defective item or not. In the former case the subset is said to be contaminated, and in the latter case the subset is said to be pure. The problem has applications in high-speed computer networks [4], string pattern recognition [11], medical examination [5], and quantity searching [3]. It also occurs in statistics [16], information theory [8], and related areas [1], [2], [10]. In the literature, the problem has been named group testing. It has two categories based on whether the tests have errors or are error free [3], [14]. In this paper we study error-free tests.

A classic model for such a searching problem is to assume that the number of defective items is known. This assumption is somewhat artificial since in practice this number is usually unknown a priori and it can be known only after testing. If the number of defective items is unknown at the beginning, how do we design and analyze algorithms? Motivated by the study of on-line problems [12], [15], Du and Hwang [7] proposed the concept of competitive algorithms for the group-testing problem.

Let \( N_\alpha(s \mid d, n)(N_\alpha(s \mid n)) \) be the number of tests that an algorithm \( \alpha \) spends on a sample \( s \) of \( n \) items under the condition that \( d \), the number of defective items, is known (unknown). Denote

\[
M_\alpha(d, n) = \max_{s \in A(n,d)} N_\alpha(s \mid d, n),
\]

\[
M(d, n) = \min_\alpha M_\alpha(d, n),
\]

\[
M_\alpha(d \mid n) = \max_{s \in A(n,d)} N_\alpha(s \mid n),
\]

---

* Received by the editors March 16, 1991; accepted for publication (in revised form) August 4, 1992. Supported in part by NSF grant CCR-9208913.

† Department of Computer Science, University of Minnesota, Minneapolis, Minnesota 55455.

‡ Also at Institute of Applied Mathematics, Chinese Academy of Sciences, Beijing, China.
when $\mathcal{A}(n, d)$ is the set of samples of $n$ items containing $d$ defective items. An algorithm $\alpha$ is called a $c$-competitive algorithm if there exists a constant $a$ such that for $0 < d < n$, $M_{\alpha}(d \mid n) \leq c \cdot M(d, n) + a$. Note that in the definition we exclude two cases, $d = 0$ and $d = n$, because $M(0, n) = M(n, n) = 0$. A $c$-competitive algorithm for a constant $c$ is simply called a competitive algorithm, and $c$ is called the competitive ratio of the algorithm.

Du and Hwang [7] proposed a bisecting algorithm with competitive ratio 2.75 and conjectured that there exists a bisecting algorithm $A$ such that $M_A(d \mid n) \leq 2M(d, n) + 1$ for $1 \leq d \leq n - 1$. Soon thereafter, Bar-Noy, Hwang, Kessler, and Kutten [4] discovered a doubling algorithm $D$ such that $M_D(d \mid n) \leq 2M(d, n) + 5$ for $1 \leq d \leq n - 1$. In this paper we confirm Du and Hwang’s conjecture by presenting a bisecting algorithm $A$ such that $M_A(d \mid n) \leq 2M(d, n) + 1$. Also, we present an algorithm $B$ such that $M_B(d \mid n) < 1.65M(d, n) + 10$.

2. Preliminaries. The analysis of competitive group testing involves both lower-bound and upper-bound problems. In this section we list some results about the lower bound for $M(d, n)$ that will be used in other sections.

There are $\binom{n}{d}$ samples for $n$ items containing $d$ defectives. Each test divides those samples into two sets. Therefore, we have the following information lower bound.

**Lemma 2.1.** For $n > d > 0$

$$M(d, n) \geq \left\lfloor \log_2 \left( \frac{n}{d} \right) \right\rfloor \geq d \log_2 \frac{n}{d}.$$  

A more useful bound derived from the aforementioned information lower bound is stated in the next lemma.

**Lemma 2.2.** For $0 < d < \rho n$, $\rho \in (0, 1)$,

$$M(d, n) \geq d \left( \log_2 \frac{n}{d} + \log_2 (e \sqrt{1 - \rho}) \right) - 0.5 \log_2 d - 0.5 \log_2 (1 - \rho) + 1.568.$$  

**Proof.** Note that the information lower bound is $\log_2 \binom{n}{d}$ for $M(d, n)$. Since $n/d < (n - i)/(d - i)$ for $0 < i < d$, we have $M(d, n) \geq d \log_2 (n/d)$. Now, we use Stirling’s formula, $n! = \sqrt{2\pi n}(n/e)^n e^{1/(12n)}(0 < e < 1)$ [13], to obtain the following estimation.

$$\binom{n}{d} \geq \sqrt{\frac{n}{2\pi d(n - d)}} \left( \frac{n}{d} \right)^d \left( \frac{n}{n - d} \right)^{n - d} e^{-1/(12d)} e^{-1/(12(n - d))}$$

$$> \frac{1}{\sqrt{d}} \left( \frac{n}{d} \right)^d \left[ 1 + \frac{d}{n - d} \right]^{(n - d)/d + 0.5} d^{(n - d)/d + 0.5} \frac{1}{\sqrt{2\pi}} \exp \left( -\frac{1}{6} \right)$$

$$> \frac{1}{\sqrt{d}} \left( \frac{n}{d} \right)^d (e \sqrt{1 - \rho})^d (1 - \rho)^{-0.5} e^{-1.568}.$$  

Thus

$$M(d, n) > d \left( \log_2 \frac{n}{d} + \log_2 (e \sqrt{1 - \rho}) \right) - 0.5 \log_2 d - 0.5 \log_2 (1 - \rho) + 1.568.$$  

Applying Lemma 2.2 to the case of $\rho = 8.21$, we have the following.

**Corollary 2.3.** For $0 < d < (8/21)n$

$$M(d, n) \geq d \left( \log_2 \frac{n}{d} + 1.096 \right) - 0.5 \log_2 d - 1.222.$$
Sometimes the information lower bound is too rough. Then the bound in the following lemma may be useful. This lemma can be found in [9].

**Lemma 2.4.** For $1 < d < n - 1$

$$M(d, n) \geq \min \left\{ n - 1, \max_{0 \leq k < d} \left[ \log \left( \frac{n - k}{d - k} \right) + 2k \right] \right\}.$$


**Lemma 2.5.** If $n > d \geq (8/21)n$, then $M(d, n) = n - 1$.

### 3. Tight bound for bisecting

Let $S$ be a set of $n$ items. The principle of the bisecting algorithm is that at each step, if a contaminated subset $X$ of $S$ is discovered, then $X$ is bisected and the resulting two subsets $X'$ and $X''$ are tested. The method of bisecting will affect the competitive ratio. Here, we choose $X'$ to contain $2^{\lceil \log_2 |X| \rceil - 1}$ items and $X'' = X \setminus X'$, where $|X|$ denotes the number of elements in $X$.

**Algorithm A:**

- **input** $S$;
- $G := \emptyset$; {a container of good items}
- $D := \emptyset$; {a container of defective items}
- TEST($S$);
- if $S$ is pure then $G := S$ and $Q := \emptyset$
- else $Q := \{S\}$;
- repeat
  - choose the frontier element $X$ of queue $Q$;
  - bisect $X$ into $X'$ and $X''$;
  - TEST($X'$);
  - if $X'$ is contaminated then TEST($X''$);
  - {If $X'$ is pure, then it is known that $X''$ is contaminated.}
  - for $Y \subseteq X'$ and $X''$ do begin
    - if $Y$ is pure then $G := G \cup Y$;
    - if $Y$ is a contaminated singleton then $D := D \cup Y$;
    - if $Y$ is contaminated but not a singleton
      - then put $Y$ into the rear of queue $Q$
  - end-for;
- until $Q = \emptyset$
- end-algorithm.

We will show the following.

**Theorem 3.1.** For $1 < d < n - 1$

(1) \[ M_A(d \mid n) \leq 2M(d, n) + 1. \]

A binary tree is a rooted tree with the property that each internal node has exactly two sons. A node is said to be the $k$th level of the tree if the path from the root to the node has length $k - 1$. So the root is on the first level. Let $i$ be the number of nodes in a binary tree, and let $j$ be the number of internal nodes in the tree. It is well known that $i = 2j + 1$.

Let $T$ denote a binary tree with nodes denoted by $X$'s in Algorithm $A$ such that two nodes $X'$ and $X''$ are sons of $X$ if and only if they are obtained by bisecting $X$. Note that each leaf of $T$ must identify at least one item. So $T$ has at most $n$ leaves. It follows that $T$ has at most $2n - 1$ nodes. Therefore, $M_A(d \mid n) \leq 2n - 1$. By Lemma 2.5 if $d/n \geq 8/21$, then
Before proving Theorem 3.1 for $d/n < 8/21$, let us first show some lemmas.

For convenience we assume that the value of function $d \log_2 \frac{n}{d}$ at $d = 0$ is 0 because $\lim_{d \to 0} d \log_2 \frac{n}{d} = 0$. The following lemma is an important tool for our analysis.

**Lemma 3.2.** Let $d = d' + d''$ and $n = n' + n''$, where $d' \geq 0$, $d'' \geq 0$, $n' > 0$, and $n'' > 0$. Then

$$d' \log_2 \frac{n'}{d'} + d'' \log_2 \frac{n''}{d''} \leq d \log_2 \frac{n}{d}.$$

**Proof.** Note that $(d^2/dx^2)(-x \log_2 x) = -\frac{1}{x \ln^2 2} < 0$ for $x > 0$. So $-x \log_2 x$ is a concave function. Thus

$$d' \log_2 \frac{n'}{d'} + d'' \log_2 \frac{n''}{d''} = n \left( \frac{n'}{n} \log_2 \frac{n'}{d'} + \frac{n''}{n} \log_2 \frac{n''}{d''} \right) \leq n \left( \frac{d}{n} \log_2 \frac{n}{d} \right) = d \log_2 \frac{n}{d}. \quad \square$$

Clearly, when $n$ is a power of 2, the analysis is relatively easy. So we first study this case.

**Lemma 3.3.** Let $n$ be a power of 2. Then for $0 < d < n$

$$M_A(d | n) \leq 2d \left( \log_2 \frac{n}{d} + 1 \right) - 1.$$

**Proof.** Consider the binary tree $T$ defined before Lemma 3.2. Clearly, every internal node must be contaminated and there exist exactly $d$ contaminated leaves. Next, we count how many contaminated nodes we can have. Denote $u = \lfloor \log_2 n \rfloor$, $v = \lfloor \log_2 d \rfloor$, and $v' = v - \log_2 d$. Then the tree $T$ has $u + 1$ levels, and the $i$th level contains $2^{i-1}$ nodes. Note that each level has at most $d$ contaminated nodes and the $(v + 1)$st level is the first level that has at least $d$ nodes. Thus the number of contaminated nodes is at most

$$\sum_{i=1}^{v} 2^{i-1} + (u - v + 1)d = 2^v - 1 + d \left( \log_2 \frac{n}{d} + 1 - v' \right) = -1 + d \left( \log_2 \frac{n}{d} + 1 - v' + 2^v \right) \leq -1 + d \left( \log_2 \frac{n}{d} + 2 \right).$$

The last inequality sign holds since $f(v') = -v' + 2^v$ is a convex function of $v'$ and $v'$ is between 0 and 1. Thus $T$ has at most $-1 + d \left( \log_2 \frac{n}{d} + 1 \right) - 1$ internal nodes and hence at most $2d \left( \log_2 \frac{n}{d} + 1 \right) - 1$ nodes. $\square$

According to the way the bisection was done, each level of $T$ contains at most one node that is a set whose size is not a power of 2. This property plays an important role in the following.

**Lemma 3.4.** For $0 \leq d \leq n$

$$M_A(d | n) \leq 2d \left( \log_2 \frac{n}{d} + 1 \right) + 1.$$
Proof. We prove the lemma by induction on $n$. For $n = 1$ the proof is trivial. For $n > 1$ let $S$ be the set of $n$ items. If $d = 0$, then one test is sufficient so that the lemma holds obviously. If $d > 0$, then we consider two cases corresponding to $S'$ and $S''$ that are obtained by bisecting $S$.

Case 1. $S'$ is contaminated. Since the number of items in $S'$ is a power of 2, Algorithm $A$ spends at most $2d' \left( \log_2 \left( \frac{|S'|}{d'} \right) + 1 \right) - 1$ tests on $S'$, where $d'$ is the number of defective items in $S'$. Let $d''$ be the number of defective items in $S''$. By the induction hypothesis, Algorithm $A$ spends at most $2d'' \left( \log_2 \left( \frac{|S''|}{d''} \right) + 1 \right) + 1$ tests. Adding one for testing $S$, we obtain that the total number of tests is at most

$$2d' \left( \log_2 \frac{|S'|}{d'} + 1 \right) + 2d'' \left( \log_2 \frac{|S''|}{d''} + 1 \right) + 1 \leq 2d \left( \log_2 \frac{n}{d} + 1 \right) + 1.$$

Case 2. $S'$ is pure. In this case Algorithm $A$ spends a test on $S'$ and at most $2d \left( \log_2 \frac{|S'|}{d} + 1 \right) + 1$ tests on $S''$. So, adding one more test for $S$, we obtain that the total number of tests is at most

$$2 + 2d \left( \log_2 \frac{|S'|}{d} + 1 \right) + 1 = 2d \left( \log_2 \frac{2|S'|}{d} + 1 \right) + 1$$

$$\leq 2d \left( \log_2 \frac{n}{d} + 1 \right) + 1. \quad \Box$$

To prove Theorem 3.1, by the remark that we made after the proof of Theorem 3.1, we need only to consider the case for which $d/n < 8/21$. In this case, by Corollary 2.3 we have

$$M(d, n) > d \left( \log_2 \frac{n}{d} + 1.096 \right) - 0.5 \log_2 d - 1.222.$$

Thus by Lemma 3.4

$$M_A(d \mid n) \leq 2M(d, n) + 2(0.5 \log_2 d + 1.222 - 0.096d) + 1.$$

Let us look at the function $h(d) = 0.5 \log_2 d - 0.096d$. $h'(d) = \frac{0.5}{d \cdot \log_2} - 0.096$. So $h(d)$ is decreasing for $d \geq 8$. We want $h(d) \leq -1.222$. This yields $d \geq 41$. Therefore, for $d \geq 41,$

$$M_A(d \mid n) \leq 2M(d, n) + 1.$$

Next, we use a more accurate analysis to deal with the case of $1 \leq d \leq 41$.

Define

$$f(n, d) = \frac{n \binom{n}{d}}{\binom{n}{d} \cdot 2^d}.$$  

If $f(n, d) > 1$, then from Lemma 3.4 and the information lower bound for $M(d, n)$ it is easy to see that

$$M_A(d \mid n) < 2M(d, n) + 2.$$

Since both sides of the inequality are integers, we have

$$M_A(d \mid n) \leq 2M(d, n) + 1.$$

Next, we study the case of $f(n, d) \leq 1$. 

Consider the following ratio:

\[
\frac{f(n, d + 1)}{f(n, d)} = \frac{n - d}{2n} \left(1 + \frac{1}{d}\right)^d.
\]

It is easy to see that

\[
\frac{n - d}{2n} \cdot e > \frac{f(n, d + 1)}{f(n, d)} > \frac{n - d}{2n} \left(\frac{d}{d + 1}\right)^{1/2} \cdot e.
\]

Thus we have the following lemma.

**Lemma 3.5.** For \(\frac{d}{n} \geq 1 - \frac{2}{e} \sqrt{d/(d + 1)}\), \(f(n, d)\) is decreasing with respect to \(d\). For \(\frac{d}{n} \leq 1 - \frac{2}{e} \sqrt{d/(d + 1)}\), \(f(n, d)\) is increasing with respect to \(d\).

This lemma tells us the behavior of function \(f(n, d)\) with respect to \(d\). Next, we study its behavior with respect to \(n\); Consider

\[
g(n, d) = \frac{f(n + 1, d)}{f(n, d)} = \frac{n + 1}{n - d + 1} \left(\frac{n}{n + 1}\right)^d.
\]

Note that

\[
g(n, d + 1) = \frac{n(n - d + 1)}{(n + 1)(n - d)} \geq 1
\]

because

\[
n(n - d + 1) - (n + 1)(n - d) = d.
\]

Moreover, \(g(n, 1) = 1\). Thus for \(d \geq 1\), \(g(n, d) \geq 1\). Therefore, we have the following lemma.

**Lemma 3.6.** For \(d \geq 1\), \(f(n, d)\) is increasing in \(n\).

From Lemmas 3.5 and 3.6 we see that if \(f(n^*, d^*) > 1\), then for every \(n \geq n^*\) and \((1 - \frac{2}{e} \sqrt{d/(d - 1)})n + 1 \geq d^*\), \(f(n, d) > 1\). Note that \(f(157, 5) > 1\) and \((1 - \frac{2}{e} \sqrt{41/40}) \cdot 157 + 1 > 41\). It follows that for \(n \geq 157\) and \(d \geq 5\), \(f(n, d) > 1\) (see Fig. 1). Unfortunately, the preceding argument does not help in the case of \(d \leq 4\). In fact, it is easy to prove that for \(1 \leq d \leq 4\), \(f(n, d) < 1\). Actually, we need a more accurate upper bound for \(M_A(d \mid n)\) for \(1 \leq d \leq 4\). The details can be found in Appendix 1. Now, the remainder is a finite set of pairs \((n, d)\)'s located in the polygon \(oxyz\) as shown in Fig. 1.
For such finitely many pairs we compute $M_A(d \mid n)$ by the following formula:

$$M_A(d \mid n) = \max_{0 \leq d' \leq d} [1 + M_A(d' \mid n') + M_A(d'' \mid n'')]$$

where $n' = 2^{[\log n - 1]}$, $n'' = n - n'$, and $d'' = d - d'$. We also compute a lower bound of $M(d, n)$ by the following formula:

$$\ell(d, n) = \min \left\{ n - 1, \max_{0 \leq k < d} \left[ \log \left( \frac{n - k}{d - k} \right) + 2k \right] \right\}.$$

Comparing two computational results, we find that $M_A(d \mid n) \leq 2\ell(d, n) + 1$. (We can save some computation by a careful analysis; see Appendix 2 for details). This completes the proof of Theorem 3.1.

4. 1.65-Competitive algorithm. Bar-Noy, Hwang, Kessler, and Kutten [4] discovered another way to design a competitive group-testing algorithm. Their basic idea is as follows. Because $d$, the number of defective items, is unknown, the algorithm tries to estimate the value of $d$. If $d$ is small, the algorithm would like to find large pure sets, whereas if $d$ is large, the algorithm would like to find small contaminated sets. To produce this behavior, the algorithm uses a doubling strategy. It tests a disjoint set of size $1, 2, \ldots, 2^i$ until a contaminated set is found. Namely, the first $i$ sets are pure and the last set is contaminated. So the algorithm finds $1 + 2 + \cdots + 2^{i-1} = 2^i - 1$ good items and a contaminated set of size $2^i$ by using $i + 1$ tests. Next, the algorithm identifies a defective item from the contaminated item by a binary search with $i$ tests as follows.

Procedure DIG($X$);

repeat
  $X' := \lfloor |Y|/2 \rfloor$ items from $X$;
  TEST($X'$);
  if $X'$ is contaminated then $X := X'$
  else $X := X \setminus X'$;
  $S := S \setminus X'$;
  $G := G \cup X'$;
until $X$ is a singleton;
$S := S \setminus X$;
$D := D \cup X$;
end-procedure;

Thus, the algorithm used a total of $2i + 1$ tests and identified $2^i$ items.

Here, we introduce a new idea as follows: Instead of testing disjoint sets of size $1, \ldots, 2^i$, the algorithm tests disjoint sets of size $1 + 2, 4 + 8, \ldots, 2^i + 2^{i+1}$ for even $i$ until a contaminated set is found. In this way the algorithm detects $2^i - 1$ good items by using $i/2$ tests instead of $i$ tests. However, it found a contaminated set of size $3 \cdot 2^i$ instead of $2^i$, which requires only one more test on a subset of size $2^i$ in order to reduce the contaminated set to either size $2^i$ or size $2^{i+1}$ with $2^i$ more good items. This idea is an extension of an idea given in [4] for combining the first two tests to further combining.
Let us first describe a procedure for three items, which is given in [4]. The input for this procedure is a contaminated set of three items. With two tests the procedure identifies either two defective items or at least one good item and one defective item.

**Procedure 3-TEST({x, y, z});**

    TEST(x);
    TEST(y);
    if x is defective then D := D ∪ {x}
    else G := G ∪ {x};
    if y is defective then D := D ∪ {y}
    else G := G ∪ {y};
    if x and y both are good
    then S := S\{x, y, z}
        D := D ∪ {z}
    else S := S\{x, y};

    end-procedure;

An extension of Procedure 3-TEST is as follows. The input is a contaminated set of 3 · 2^k items (k > 0). The procedure first finds either a contaminated set of size 2^k or a pure set of size 2^k and a contaminated set of size 2^{k+1} and then digs out a defective item from the resultant contaminated set.

**Procedure 3-SET-TEST(X);**

    X' := min(2^k, |X|) items from X;
    TEST(X');
    if X' is contaminated
    then X := X'
    else X := X\X'
    G := G ∪ X'
    S := S\X';

    DIG(X);

    end-procedure

Now, we describe the main body of the algorithm.

**Algorithm B:**

    input S;
    D := Ø;
    G := Ø;
    while |S| ≥ 3 do
        k := 0;
        repeat [jumping process]
            X := min(2^k + 2^{k+1}, |S|) items from S;
            TEST(X);
            if X is pure then G := G ∪ X
                S := S\X
                k := k + 2;
            if k = 10 then TEST(S)
                if S is pure then G := G ∪ S
                    S := Ø;
        end-repeat
    end-while
until $X$ is contaminated;
if $k = 0$ then 3-TEST($X$);
if $k > 0$ then 3-SET-TEST($X$);
end-while;
while $S ≠ \emptyset$ do
    $x :=$ an item from $S$;
    TEST($x$);
    if $x$ is good then $G := G \cup \{x\}$;
    if $x$ is defective then $D := D \cup \{x\}$;
    $S := S\setminus\{x\}$;
end-while;
end-algorithm

Next, we analyze Algorithm B in a way similar to that in §3.

**Lemma 4.1.** $M_B(d | n) ≤ 1.65d(\log_2 \frac{n}{d} + 1.031) + 5$.

**Proof.** We prove the lemma by induction on $d$. For $d = 0$, since the algorithm will test $S$ when $k = 10$, it takes at most five tests to find that $S$ is pure. Thus $M_B(0 | n) ≤ 5$.

For $d > 0$ suppose that the first time that the computation goes out from the jumping process is with $k = i$. So a contaminated set $X$ of size at most $2^i + 2^{i+1}$ ($i$ is even) and $2^i - 1$ good items are found with $i/2 + 1$ tests. Next, consider three cases.

**Case 1.** $i = 0$. Procedure 3-TEST identifies either two defective items or at least one good item and one defective item by two tests. Applying the induction hypothesis to the remaining $n - 2$ items, we obtain that in the former subcase the total number of tests is at most

$$3 + 1.65(d - 2) \left(\log_2 \frac{n - 2}{d - 2} + 1.031\right) + 5$$

$$= 1.5 \cdot 2 \left(\log_2 \frac{2}{2} + 1\right) + 1.65(d - 2) \left(\log_2 \frac{n - 2}{d - 2} + 1.031\right) + 5$$

$$≤ 1.65d \left(\log_2 \frac{n}{d} + 1.031\right) + 5;$$

in the latter subcase the total number of tests is at most

$$3 + 1.65(d - 1) \left(\log_2 \frac{n - 2}{d - 1} + 1.031\right) + 5$$

$$= 1.5(\log_2 2 + 1) + 1.65(d - 1) \left(\log_2 \frac{n - 2}{d - 1} + 1.031\right) + 5$$

$$≤ 1.65d \left(\log_2 \frac{n}{d} + 1.031\right) + 5.\]

**Case 2.** $2 ≤ i ≤ 8$. Procedure 3-SET-TEST finds either one defective item by using at most $i + 1$ tests or one defective item and $2^i$ good items by using at most $i + 2$ tests. In the former subcase the total number of identified items is $2^i$ and the total number of tests for detecting them is at most

$$(i/2 + 1) + (i + 1) ≤ 1.65(\log_2 2^i + 1.031);$$
in the latter case the total number of identified items is $2^{i+1}$ and the total number of tests for identifying them is at most

$$(i/2 + 1) + (i + 2) \leq 1.50(\log_2 2^{i+1} + 1).$$

Applying the induction hypothesis to the remaining unidentified items and using Lemma 3.2, we can obtain the upper bound $1.65d/(\log_2 2 + 1.031) + 5$ for the total number of tests.

Case 3. $i \geq 10$. This case is similar to Case 2. The difference is that the algorithm uses one more test for testing on $S$ when $i = 10$. So, corresponding to the two subcases in Case 2, we have that in the former subcase the total number of tests for identifying $2^i - 1$ good items and one defective item is at most

$$(i/2 + 1) + (i + 2) \leq 1.65(\log_2 2^i + 1.031);$$

in the latter subcase the total number of tests for identifying $2^{i+1}$ good items and one defective item is at most

$$(i/2 + 1) + (i + 3) \leq 1.65(\log_2 2^{i+1} + 1.031).$$

The proof is completed by applying the induction hypothesis to the remaining unidentified items and using Lemma 3.2.

**Lemma 4.2.** For $0 \leq d \leq n$, $M_B(d \mid n) \leq 1.5n$.

**Proof.** We prove the lemma by induction on $d$. For $d = 0$ the algorithm needs one test when $n \leq 3$, two tests when $4 \leq n \leq 15$, and at most five tests when $n > 16$, so that $M_B(0 \mid n) \leq 1.5n$. For $d > 0$ suppose that the first time that the computation goes out from the jumping process is with $k = i$. So a contaminated set $X$ of size at most $2^i + 2^{i+1}$ $(i$ is even) and $2^i - 1$ good items are found with $i/2 + 1$ tests. Next, we follow the trace of the proof of the last lemma to verify that in each case the number of tests is at most $1.5$ times the number of identified items.

Case 1. $i = 0$. Two items were identified by using three tests.

Case 2. $2 \leq i \leq 8$. Either $2^i$ items were identified by using $1.5i + 2$ tests or $2^{i+1}$ items were identified by using $1.5i + 3$ tests. Since $i \geq 2$, $1.5i + 2 \leq 1.5 \cdot 2^i$ and $1.5i + 3 \leq 1.5 \cdot 2^{i+1}$. Case 3. $i \geq 10$. Either $2^i$ items were identified by using $1.5i + 3$ tests or $2^{i+1}$ items were identified by using $1.5i + 4$ tests. Since $i \geq 10$, $1.5i + 3 \leq 1.5 \cdot 2^i$ and $1.5i + 4 \leq 1.5 \cdot 2^{i+1}$.

The proof is completed by applying the induction hypothesis to the remaining unidentified items and adding the bound to the inequality in each of the cases or subcases.

**Theorem 4.3.** For $1 \leq d \leq n - 1$, $M_B(d \mid n) \leq 1.65M(d, n) + 10$.

**Proof.** If $d/n \geq 8/21$, then $M(d, n) = n - 1$. The theorem then follows from Lemma 4.2. If $d/n < 8/21$, then by Lemma 4.1 and Corollary 2.3 we have

$$M_B(d \mid n) \leq 1.65M(d, n) + 5 + 1.65(0.5 \log_2 d - 0.065d + 1.222).$$

Denote $h(d) = 0.5 \log_2 d - 0.065d$. Then $h(d)$ increases for $d \leq 11$ and decreases for $d \geq 12$. Thus $h(d) \leq \max(h(11), h(12)) \leq 1.015$. Hence $M_B(d \mid n) \leq 1.65M(d, n) + 9.3$.

By modifying Algorithm B, the competitive ratio could be further improved to approach $1.5$. The modification can be done through studying the competitive group testing for a small number of items. For example, if instead of Procedure 3-TEST we use a procedure for testing 12 items, the competitive ratio can be decreased to be less than $1.6$. However, how to push the competitive ratio down from $1.5$ is unknown.
Appendix 1: Proof of Theorem 3.1 for $1 \leq d \leq 4$. Let us prove the following lemmas.

**Lemma 3.7.**

- $M_A(1 \mid 2^u) = 2u + 1$ for $u \geq 0$,
- $M_A(2 \mid 2^u) = 4u - 1$ for $u \geq 1$,
- $M_A(3 \mid 2^u) = 6u - 5$ for $u \geq 2$,
- $M_A(4 \mid 2^u) = 8u - 9$ for $u \geq 2$.

*Proof.* We prove the lemma by induction on $u$. It is easy to check each initiation. For induction we employ formulae (3), (4), and (5) to yield the following:

- $M_A(1 \mid 2^u) = 1 + M_A(1 \mid 2^{u-1}) + M_A(0 \mid 2^{u-1})$
  
  $= 1 + 2(u - 1) + 1 + 1$
  
  $= 2u + 1$,

- $M_A(2 \mid 2^u) = \max(1 + M_A(2 \mid 2^{u-1}), 1 + 2M_A(1 \mid 2^{u-1}))$
  
  $= \max(4u - 4, 4u - 1)$
  
  $= 4u - 1$,

- $M_A(3 \mid 2^u) = \max(1 + M_A(3 \mid 2^{u-1}), 1 + M_A(1 \mid 2^{u-1}) + M_A(2 \mid 2^{u-1}))$
  
  $= \max(6u - 10, 6u - 5)$
  
  $= 6u - 5$,

- $M_A(4 \mid 2^u) = \max_{0 \leq d' \leq 4} (M_A(d' \mid 2^{u-1}) + M_A(4 - d' \mid 2^{u-1}))$
  
  $= \max(8u - 17, 8u - 11, 8u - 9)$
  
  $= 8u - 9$. \(\Box\)

**Lemma 3.8.** Let $u + 1 = \lceil \log n \rceil$, $v = \lceil \log(n - 2^u) \rceil$, and $w = \lceil \log(n - 2^u - 2^{v-1}) \rceil$. Then

- $M_A(1 \mid n) \leq 2(u + 1) + 1$,
- $M_A(2 \mid n) \leq \max(4u + 1, 2(u + v + 1) + 1)$,
- $M_A(3 \mid n) \leq \max(6u - 3, 4u + 2v + 1)$,
- $M_A(4 \mid n) \leq \max(8u - 7, 6u + 2v - 3, 4u + 4v - 3, 4u + 2v + 2w + 1)$.

*Proof.* Use the recursive formula (3), and note that $M_A(d \mid n - 2^u) \leq M_A(d \mid 2^u)$. \(\Box\)

Now, we prove Theorem 3.1 in the case of $1 \leq d \leq 4$ as follows. Note that $u$ and $v$ are defined the same as in Lemma 3.8.
For $d = 1$, $M_A(1 \mid n) \leq 2(u + 1) + 1 = 2M(1, n) + 1$.

For $d = 2$, if $v < u$, then $M_A(2 \mid n) \leq 4u + 1$ and

$$M(2, n) \geq \left\lfloor \log \left( \frac{n}{2} \right) \right\rfloor \geq \left\lfloor \log \left( \frac{(2^u + 1)2^u}{2} \right) \right\rfloor = 2u.$$

So $M_A(2 \mid n) \leq 2M(2, n)$. If $u = v$, then $M_A(2 \mid n) = 2(2u + 1) + 1$ and

$$M(2, n) \geq \left\lfloor \log \left( \frac{2^u + 2^u - 1}{2} \right) \right\rfloor \geq \left\lfloor \log(2^{2u} + 2^{2u-3}) \right\rfloor = 2u + 1.$$

Thus (1) holds.

For $d = 3$ verification is trivial for $u = 1$. Next, we consider $u \geq 2$. If $v < u$, then $M_A(3 \mid n) \leq 2(3u - 2) + 1$ and

$$M(3, n) \geq \left\lfloor \log \left( \frac{n}{3} \right) \right\rfloor \geq \left\lfloor \log \left( \frac{(2^u + 1)2^u(2^u - 1)}{6} \right) \right\rfloor \geq \left\lfloor \log((2^u + 1)2^{2u-3}) \right\rfloor \geq 3u - 2.$$

Thus (1) holds. If $u = v$, then $M_A(3 \mid n) \leq 6u + 1$ and

$$M(3, n) \geq \left\lfloor \log \left( \frac{n}{3} \right) \right\rfloor \geq \left\lfloor \log \left( \frac{(2^u + 2^u - 1)(2^u + 2^u - 1)(2^u + 2u - 1)}{6} \right) \right\rfloor \geq \left\lfloor \log((2^{2u+1} + 2^{2u-2} - 1)2^{u-2}) \right\rfloor \geq 3u.$$

For $d = 4$ it is trivial to verify (1) for $u = 1$ and 2. Next, consider $u \geq 3$. If $v < u$ and $w < u - 1$, then $M_A(4 \mid n) \leq 2(4u - 3) + 1$ and

$$\left\lfloor \log \left( \frac{n}{4} \right) \right\rfloor \geq \left\lfloor \log \left( \frac{(2^u + 1)2^u(2^u - 1)(2^u - 2)}{8 \cdot 3} \right) \right\rfloor = \left\lfloor \log \left( \frac{(2^{3u} - 2^{2u} - 2^{u-1} + 1)2^{u-2}}{3} \right) \right\rfloor \geq 4u - 3.$$
Thus (1) holds. If \( u = v \) and \( w < u - 1 \), then \( M_A(4 | n) \leq 2(4u - 2) + 1 \) and

\[
\log \left( \frac{n}{4} \right) \geq \log \left( \frac{(2^u + 2^{u-1} + 1)(2^u + 2^{u-1})(2^u + 2^{u-1} - 1)(2^u + 2^{u-1} - 2)}{8 \cdot 3} \right)
\]

\[
> \left[ \log \left( \frac{(2^u + 2^{u-1})^2 - 1)((2^u + 2^{u-1} - 1)^2 - 1)}{3 \cdot 8} \right) \right]
\]

\[
> \left[ \log \left( \frac{2^{2u+1} + 2^{2u-2} - 1}{3 \cdot 8} \cdot \frac{2^{2u+1} + 2^{2u-2} - 2^u - 2^{u-1}}{8} \right) \right]
\]

\[
= \left[ \log((2^{2u-1} + 1)2^{2u-2}) \right]
\]

\[
\geq 4u - 2.
\]

So (1) holds. If \( w = u - 1 \), then we must have \( u = v \) and \( M_A(4 | n) \leq 2(4u - 1) + 1 \). Note that \( n \geq 2^u + 2^{u-1} + 2^{u-2} + 1 \). Thus

\[
\log \left( \frac{n}{4} \right) \geq \log \left( \frac{(2^u + 2^{u-1} + 2^{u-2})^2 - 1)((2^u + 2^{u-1} + 2^{u-2} - 1)^2 - 1)}{3 \cdot 8} \right)
\]

\[
> \left[ \log \left( \frac{2^{2u+1} + 2^{2u-4} - 1}{3 \cdot 8} \cdot \frac{2^{2u+1} + 2^{2u-4} - 2^u - 2^{u-1} - 2^{u-2}}{8} \right) \right]
\]

\[
= \left[ \log((2^{2u} + 1)2^{2u-2}) \right]
\]

\[
\geq 4u - 1.
\]

Therefore, (1) holds.

**Appendix 2: Proof of Theorem 3.1 for 5 \leq d \leq 40.** To save some computation, let us first compute \( n_d = \min\{n \mid f(n, d) > 1\} \) for 5 \leq d \leq 40, where \( f(n, d) \) is the function defined by (2). The result is shown in the following.

\[
\begin{align*}
n_5 &= 73, \quad n_6 = 44 \quad n_7 = 38, \quad n_8 = 37, \quad n_9 = 37, \quad n_{10} = 37, \\
n_{11} &= 39, \quad n_{12} = 40, \quad n_{13} = 42, \quad n_{14} = 43, \quad n_{15} = 45, \quad n_{16} = 47, \\
n_{17} &= 49, \quad n_{18} = 50, \quad n_{19} = 52, \quad n_{20} = 54, \quad n_{21} = 56, \quad n_{22} = 58, \\
n_{23} &= 60, \quad n_{24} = 62, \quad n_{25} = 64, \quad n_{26} = 66, \quad n_{27} = 68, \quad n_{28} = 70, \\
n_{29} &= 72, \quad n_{30} = 74, \quad n_{31} = 76, \quad n_{32} = 78, \quad n_{33} = 80, \quad n_{34} = 82, \\
n_{35} &= 84, \quad n_{36} = 86, \quad n_{37} = 88, \quad n_{38} = 90, \quad n_{39} = 92, \quad n_{40} = 94.
\end{align*}
\]

From the preceding equations we see that for 21 \leq d \leq 40, \( d/n_d \geq 8/21 \). Therefore, for 21 \leq d \leq 40, (1) holds. Now, for 5 \leq d \leq 20 and \( (21/8)d < n < n_d \) we compute \( M_A(d | n) \) and \( \ell(d, n) \). The results are given in Table 1 and show that \( M_A(d | n) \leq 2\ell(d, n) + 1 \).
TABLE 1

(d, n, l(d, n), MA(d In)) values for 5 ≤ d ≤ 20 and (21/8)d < d < nd.

<table>
<thead>
<tr>
<th>d</th>
<th>l(d, n)</th>
<th>MA(d In)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>14, 23</td>
<td>15, 25</td>
</tr>
<tr>
<td>5</td>
<td>16, 23</td>
<td>17, 27</td>
</tr>
<tr>
<td>5</td>
<td>18, 27</td>
<td>19, 29</td>
</tr>
<tr>
<td>6</td>
<td>17, 29</td>
<td>18, 31</td>
</tr>
<tr>
<td>6</td>
<td>19, 31</td>
<td>20, 33</td>
</tr>
<tr>
<td>6</td>
<td>21, 33</td>
<td>22, 35</td>
</tr>
<tr>
<td>7</td>
<td>22, 35</td>
<td>23, 37</td>
</tr>
<tr>
<td>7</td>
<td>24, 37</td>
<td>25, 39</td>
</tr>
<tr>
<td>8</td>
<td>22, 39</td>
<td>23, 41</td>
</tr>
<tr>
<td>8</td>
<td>24, 41</td>
<td>25, 43</td>
</tr>
<tr>
<td>9</td>
<td>24, 43</td>
<td>25, 45</td>
</tr>
<tr>
<td>9</td>
<td>26, 45</td>
<td>27, 47</td>
</tr>
<tr>
<td>10</td>
<td>27, 47</td>
<td>28, 49</td>
</tr>
<tr>
<td>10</td>
<td>29, 49</td>
<td>30, 51</td>
</tr>
<tr>
<td>11</td>
<td>30, 51</td>
<td>31, 53</td>
</tr>
<tr>
<td>11</td>
<td>32, 53</td>
<td>33, 55</td>
</tr>
<tr>
<td>12</td>
<td>33, 55</td>
<td>34, 57</td>
</tr>
<tr>
<td>12</td>
<td>35, 57</td>
<td>36, 59</td>
</tr>
<tr>
<td>13</td>
<td>36, 59</td>
<td>37, 61</td>
</tr>
<tr>
<td>13</td>
<td>38, 61</td>
<td>39, 63</td>
</tr>
<tr>
<td>14</td>
<td>39, 63</td>
<td>40, 65</td>
</tr>
<tr>
<td>14</td>
<td>41, 65</td>
<td>42, 67</td>
</tr>
<tr>
<td>15</td>
<td>42, 67</td>
<td>43, 69</td>
</tr>
<tr>
<td>15</td>
<td>44, 69</td>
<td>45, 71</td>
</tr>
<tr>
<td>16</td>
<td>45, 71</td>
<td>46, 73</td>
</tr>
<tr>
<td>16</td>
<td>47, 73</td>
<td>48, 75</td>
</tr>
<tr>
<td>17</td>
<td>48, 75</td>
<td>49, 77</td>
</tr>
<tr>
<td>17</td>
<td>49, 77</td>
<td>50, 79</td>
</tr>
<tr>
<td>18</td>
<td>50, 79</td>
<td>51, 81</td>
</tr>
<tr>
<td>18</td>
<td>51, 81</td>
<td>52, 83</td>
</tr>
<tr>
<td>19</td>
<td>52, 83</td>
<td>53, 85</td>
</tr>
<tr>
<td>19</td>
<td>53, 85</td>
<td>54, 87</td>
</tr>
<tr>
<td>20</td>
<td>54, 87</td>
<td>55, 89</td>
</tr>
</tbody>
</table>

REFERENCES


