2014-04

A Zig-Zag Approach for Competitive Group Testing

UTD AUTHOR(S): Dingzhu Du

INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

A Zig-Zag Approach for Competitive Group Testing

Yongxi Cheng, Ding-Zhu Du, Yinfeng Xu

To cite this article:
Yongxi Cheng, Ding-Zhu Du, Yinfeng Xu (2014) A Zig-Zag Approach for Competitive Group Testing. INFORMS Journal on Computing 26(4):677-689. http://dx.doi.org/10.1287/ijoc.2014.0591

INFORMS Journal on Computing
Vol. 26, No. 4, Fall 2014, pp. 677–689
ISSN 1091-9856 (print) | ISSN 1526-5528 (online)

http://dx.doi.org/10.1287/ijoc.2014.0591

A Zig-Zag Approach for Competitive Group Testing

Yongxi Cheng
School of Management, Xi'an Jiaotong University, China; and
State Key Lab for Manufacturing Systems Engineering, Xi'an 710049, China, chengyx@mail.xjtu.edu.cn

Ding-Zhu Du
Department of Computer Science, University of Texas at Dallas, Richardson, Texas 75080, dzdu@utdallas.edu

Yinfeng Xu
School of Management, Xi'an Jiaotong University, China; and
State Key Lab for Manufacturing Systems Engineering, Xi'an 710049, China, yfxu@mail.xjtu.edu.cn

In many fault-detection problems, we want to identify defective items from a set of n items using the minimum number of tests. Group testing is a scenario in which each test is on a subset of items and determines whether the subset contains at least one defective item. In practice, the number d of defective items is often unknown in advance. In this paper, we present a new algorithm for the above group testing problem and prove that it has very good performance guarantee. More specifically, the number of tests used by the new algorithm is bounded from above by $d\log(n/d)+3d+O(\log^2 d)$. The new algorithm is designed based on a zig-zag approach that has not been studied before and is intuitive and easy to implement. When $0<d<\rho_0 n$ where $\rho_0=1-1/e^2=0.45\ldots$, which holds for most practical applications, our new algorithm has better performance guarantee than any previous best result. Computational results show that the new algorithm has very good practical performances.

Keywords: fault detection; group testing; algorithms
History: Accepted by Karen Aardal, Area Editor for Design and Analysis of Algorithms; received August 2011; revised August 2012, February 2013; accepted December 2013. Published online in *Articles in Advance* April 23, 2014.

1. Introduction

Suppose we are given a set S of n items in which some items are *defective* and others are *good*. Our task is to identify the defective items using a sequence of tests. Each test is on a subset of items in S and tells us whether the subset contains at least one defective item. The subset is said to be *contaminated* if it contains at least one defective item; otherwise, the subset is said to be *pure*.

This scenario is the *group testing* problem, which was first introduced by Dorfman (1943) for efficient mass blood testing. Because of its basic nature, the problem later found various applications, including quality control in industrial product testing (Sobel and Groll 1959), sequential screening of experimental variables (Li 1962), searching files in storage systems (Kautz and Singleton 1964), multiple access communication (Wolf 1985, Goodrich and Hirschberg 2008) and data gathering in sensor networks (Hong and Scaglione 2004), screening of clone libraries (Barillot et al. 1991, Bruno et al. 1995), blood screening for HIV tests (Wein and Zenios 1996, Zenios and Wein 1998), data compression (Hong and Ladner 2002), streaming algorithms (Cormode and Muthukrishnan 2005), and testing and diagnosis for digital logic systems (Kahng and Reda 2006). Recently, Abolnikov and Dukhovny (2003), Bar-Lev et al. (2007), and Claeys et al. (2010) studied the scenario where there is a group testing center and items to be tested arrive at the center at random time points. For a thorough treatment of this subject, we refer the reader to the monograph by Du and Hwang (2000).

Let $M(d,n)$ denote the minimum number of tests required in the worst case, if the number of defectives d is known in advance. There are many studies on the bounds on $M(d,n)$ in the literature. Hwang (1972) proposed a *generalized binary-splitting* algorithm that uses at most $\lceil\log\binom{n}{d}\rceil+d-1$ tests in the worst case. As to lower bounds on $M(d,n)$, in general the information theoretic lower bound $M(d,n)\geq\lceil\log\binom{n}{d}\rceil$ is currently best known. The inequality in Equation (1) implies another lower bound, i.e., the right-hand side of the inequality, on $M(d,n)$. The inequality is proved as Lemma 4.1.1 in Du and Hwang (2000, Chap. 4): For $0<\rho<1$, if $0<d<\rho n$, then

$$\log\binom{n}{d}\geq d\left(\log\frac{n}{d}+\log(e\sqrt{1-\rho})\right)-0.5\log d-0.5\log(1-\rho)-1.567. \quad (1)$$

In many applications, the assumption that d is known a priori is not realistic. Probably we only know

the existence of defectives but nothing else. In such a situation, we still want an algorithm whose number of tests used in the worst case will not be too much more than $M(d, n)$, no matter what the number of defectives is. Motivated by the study of online algorithms (Sleator and Tarjan 1985, Manasse et al. 1988), Du and Hwang (1993) proposed a concept of a *competitive ratio* to measure the quality of algorithms for solving the group testing problem with unknown number of defectives, as the following.

Let A be an algorithm solving the group testing problem when the number d of defectives is unknown, and let $M_A(d \mid n)$ denote the number of tests performed by A in the worst case if there are n items and d of them are defective. Then A is called a c-competitive algorithm if there exists a constant a such that for $0 \le d < n$, $M_A(d \mid n) \le cM(d, n) + a$. The case $d = n$ is excluded in the definition because $M(n, n) = 0$, and $M_A(n \mid n) \ge n$ for any Algorithm A. If c is a constant, a c-competitive Algorithm A is simply called a *competitive* algorithm and c is called the *competitive ratio* of A. Du and Hwang (1993) proposed a 2.75-competitive algorithm. The competitive ratio was improved to 2 by Bar-Noy et al. (1994) and then to 1.65 by Du et al. (1994). Schlaghoff and Triesch (2005) developed an Algorithm A_ε satisfying $M_{A_\varepsilon}(d \mid n) \le (1.5 + \varepsilon)M(d, n) + \lceil 0.4/\varepsilon \rceil + 6$ for any $0 < \varepsilon < 0.01$ and $0 \le d < n$ and thus improved the competitive ratio to $1.5 + \varepsilon$ for all positive $\varepsilon < 0.01$.

Du and Park (1994) introduced another notion of competitiveness. For an Algorithm A solving the group testing problem with unknown number of defectives, define

$$n(d, k) = \max\{n \mid M(d, n) \le k\},$$
$$n_A(d \mid k) = \max\{n \mid M_A(d \mid n) \le k\}.$$

Algorithm A is called a *strongly c-competitive* algorithm if there exists a constant a such that for every $d \ge 1$ and $k \ge 1$, $n(d, k) \le cn_A(d \mid k) + a$. If c is a constant, then a strongly c-competitive algorithm is also called a *strongly competitive* algorithm. Du and Park (1994) proved that every strongly competitive Algorithm A is also competitive in the original sense, as defined in Du and Hwang (1993), and satisfies the following desirable property: $\lim_{n\to\infty}(M_A(d \mid n)/M(d, n)) = 1$. Moreover, they proposed an algorithm, which we will refer to as Algorithm *DP*, with $M_{DP}(d \mid n) \le d \log(n/d) + 4d$ for $1 \le d \le n$. They showed that Algorithm *DP* is strongly competitive and wondered whether new algorithms exist such that the coefficient 4 for term d in the upper bound can be further reduced. Schlaghoff and Triesch (2005) proved that there exists an Algorith B satisfying that for $1 \le d \le n$, $M_B(d \mid n) \le \lceil \log \binom{n}{d} \rceil + 2d$; however, the Algorithm B implied by their constructive proof is rather complicated.

If randomized group testing algorithms are considered, very recent results in Damaschke and Sheikh Muhammad (2012) show that, asymptotically, one can achieve nearly $d \log n$ tests, even with highly nonadaptive testing strategies, to correctly solve (with any prescribed constant success probability) the above group testing problem with an unknown number of defectives. However, deterministic testing algorithms (which do not need to use randomization and can correctly identify all the defectives with certainty) using fewer tests in the worst case are still interesting for further investigation.

The main contributions of this paper, presented next, are for the *combinatorial group testing model*. For the *probabilistic group testing model*, in which each item in S is defective with some probability p, for the case of small defect probability $p \ll 1$ and large number n of items, asymptotically optimal two-stage testing strategies were obtained recently in Mezard and Toninelli (2011).

In this paper, we present a new deterministic, strongly competitive Algorithm Z with $M_Z(d \mid n) \le d \log(n/d) + \beta d + O(\log^2 d)$, for the (combinatorial) group testing problem with unknown number of defectives. Algorithm Z is designed based on a zig-zag approach that has not been studied before. Compared to the result of Schlaghoff and Triesch (2005), our algorithm is much simpler and, by Equation (1), has an asymptotically smaller upper bound on the number of tests performed when $d < \rho n$ for any constant $\rho < 1 - 4/e^2 = 0.45\ldots$. Considering that in practice we usually have $d \ll n$, in such cases our new algorithm has a better performance guarantee. Computational results also indicate that our new algorithm has very good practical performances.

The rest of the paper is organized as follows. We describe our new Algorithm Z in §2 and present the analysis of it in §3. In §4, we show results of computational tests to compare the practical performances of Algorithm Z and Algorithm *DP* as proposed in Du and Park (1994). We conclude our paper in §5. Throughout the paper, log is of base 2 if no base is specified.

2. The New Algorithm

In this section, we present a new Algorithm Z for the group testing problem when the number d of defective items is unknown. Algorithm Z uses at most $d \log(n/d) + \beta d + O(\log^2 d)$ tests to identify all the defective items in the worst case. The new algorithm is intuitive and easy to implement.

First we describe a binary splitting procedure DIG to identify one defective item (and an unspecified

number of good items) from a contaminated set X. Procedure DIG has been used in a number of previous algorithms in the literature.

Procedure DIG

- Input: a contaminated set X.
- Procedure:
  - —While $|X| > 1$
    1. Select a subset Y of size $\lceil |X|/2 \rceil$ from X. Test Y.
    2. If Y is contaminated then assign $X = Y$, else identify all items in Y as good items and assign $X = X - Y$.
  - —Identify the single item in X as defective.

We will use the following folklore lemma about Procedure DIG for later analysis. The lemma is easy and we state it without giving a proof.

Lemma 2.1. *For a contaminated set X with $2^{v-1} < |X| \le 2^v$, where $v \ge 1$ is an integer, the total number of tests performed by procedure DIG on X is v.*

The main idea of Algorithm Z is as follows. Each time we choose a subset to identify a defective item from it or to determine that the subset is pure. The size of the current subset depends on the size and test result of the previous subset. Roughly speaking, each time the size doubles if the previous subset is tested to be pure, and the size is reduced by half if the previous subset is tested to be contaminated. At the beginning we choose the whole set S as the subset, and the process continues until every item in S is successfully identified. The following is the detailed description of Algorithm Z.

Algorithm Z

- Input: a set S of n items in which an unknown number $d \ge 1$ of them are defectives.
- Procedure:
  1. Initially, set $k = \lceil \log n \rceil$.
  2. While S is not empty
     (a) Select a subset S' of S with $|S'| = 2^k$ (if $|S| < 2^k$ then select the whole S as S'). Test S'.
     (b) If S' is pure, then identify all items in S' as good items and remove them from S; increase k by 1.
     (c) If S' is contaminated, then there are the following two possible cases:
     —If $k > 0$, then apply DIG(S') to identify one defective item (and an unspecified number of good items) in S' and remove these items from S; decrease k by 1.
     —If $k = 0$, then identify the single item in S' as defective and remove it from S (note that in this case $k = 0$ is unchanged).

3. Analysis of Algorithm Z

Though our new algorithm is fairly simple to describe, it is not an easy task to analyze it. In this section, we show an upper bound of $d \log(n/d) + 3d + 0.5 \log^2 d + 1.5 \log d + 2$ on the number of tests used by Algorithm Z in the worst case. First we introduce some notations and definitions before presenting the proof.

Algorithm Z uses two types of tests: the first consists of tests performed in step (a), and the second consists of tests performed in a DIG procedure in step (c). We use $\mathcal{T}$ to denote the set of all the tests of the first type (i.e., all the tests performed in step (a) by Algorithm Z). Let $q = |\mathcal{T}|$, and we use $T_1, T_2, \ldots, T_q$ to denote the tests in $\mathcal{T}$, in the ordering that they are performed by Algorithm Z.

Definition 3.1 (Incurred Tests). For a test $T \in \mathcal{T}$, let S' be the subset that test T is applied to in step (a). Define $I(T)$ to be the set of tests *incurred* by T as follows: If the test result of T is pure, then $I(T)$ contains only one test T; i.e., $I(T) = \{T\}$. If the test result of T is contaminated, then set $I(T)$ contains test T together with all tests performed in the subsequent DIG procedure on S' in step (c).

By the above definition, the set of all tests performed by Algorithm Z can be written as $\bigcup_{T \in \mathcal{T}} I(T)$, and the total number of tests used by Algorithm Z is

$$t_Z = \sum_{T \in \mathcal{T}} |I(T)| \ge \sum_{i=1}^{q} |I(T_i)|. \tag{2}$$

Next, we define the *rank* of a test in $\mathcal{T}$ and a *zig-zag pair* of two tests in $\mathcal{T}$; these are key ingredients of our later analysis.

Definition 3.2 (Rank). During the execution of Algorithm Z, for a test T in $\mathcal{T}$ (i.e., performed in step (a)), if the current value of k is v, we say that T is of rank v.

From Algorithm Z, a test of rank v is applied to a subset of size at most 2^v.

Definition 3.3 (Zig-Zag Pair). We say that two tests T_i and T_j in $\mathcal{T}$ form a zig-zag pair $P = (T_i, T_j)$ if for some integer $v \ge 1$, T_i is of rank v and test result contaminated, and T_j is of rank $v - 1$ and test result pure. Moreover T_j is required to be applied to a subset of size exactly 2^{v-1}. We call v the *rank* of the zig-zag pair $P = (T_i, T_j)$.

Notice that in the definition of a zig-zag pair $P = (T_i, T_j)$, T_i and T_j are not required to be consecutive in $\mathcal{T}$, and there is no requirement on the ordering of T_i and T_j; that is, both $i < j$ and $i > j$ are allowed.

3.1. Partitioning $\mathcal{T}$ Into Three Classes

We partition $\mathcal{T} = \{T_1, T_2, \ldots, T_q\}$—the set of all the tests of the first type (i.e., all the tests performed in step (a))—into three classes: C_1, C_2, and C_3. First we have the following observation on the ranks of tests in $\mathcal{T}$.

Observation 3.4. For any test $T \in \mathcal{T}$, let $r(T)$ be the rank of T. Then $0 \le r(T) \le k$, where $k = \lceil \log n \rceil$.

Proof. First, from Algorithm Z the rank of any test $T \in \mathcal{T}$ is clearly nonnegative. Next we show that the rank of any test $T \in \mathcal{T}$ is at most $k = \lceil \log n \rceil$. For the sake of contradiction, assume that there exists a test T of rank greater than k performed in step (a). Then, from Algorithm Z there must exist a test T' performed in step (a) before T, such that T' is of rank k and test result pure; then T' must be applied to the whole current set S (since $2^k \ge n$) and identifies S as a pure set (i.e., all items in S are good) and terminates the algorithm, which contradicts that test T is performed after T'! □

Next, we define the three classes: C_1, C_2, and C_3. The first class C_1 contains every test in $\mathcal{T}$ that is of rank 0 and test result contaminated, together with T_q, the last test performed in step (a) by Algorithm Z.

By Observation 3.4, for the rank r_q of T_q, we have $0 \le r_q \le k$. By Algorithm Z, starting from test T_1 of rank k performed in step (a), to have a test T_q of rank r_q performed in step (a) there must exist a sequence of tests $T^{(k)}, T^{(k-1)}, \ldots, T^{(r_q+1)}$ of ranks $k, k-1, \ldots, r_q+1$ and test results "contaminated" performed in step (a) (they are not necessarily consecutive tests performed in step (a)). We choose $T^{(k)}, T^{(k-1)}, \ldots, T^{(r_q+1)}$ to be the first such tests (any such tests will work; we choose the first such tests to make the presentation more clear). The second class C_2 is formed by the above sequence of tests $T^{(k)}, T^{(k-1)}, \ldots, T^{(r_q+1)}$ (if $k = r_q$, then the second class is empty).

Since the rank of any test in the second class is higher than r_q and the rank of any test in the first class is at most r_q (in fact, is either 0 or r_q), we have that the first two classes, C_1 and C_2, are disjoint.

The third class, C_3, is formed by all tests in $\mathcal{T}$ that are not in the first two classes; i.e., $C_3 = \mathcal{T} \setminus (C_1 \cup C_2)$.

Lemma 3.5. *The number of tests in C_3 is even. Moreover, tests in C_3 can be paired up to form $|C_3|/2$ zig-zag pairs, such that each test in C_3 appears in exactly one pair.*

Proof. From the definition of C_1, after class C_1 is removed from $\mathcal{T}$, the ranks of any two consecutive tests in $\mathcal{T} \setminus C_1$ (with their original ordering in $\mathcal{T}$) differ by 1. Consider a vertical line segment L (which is part of the y axis) with lower endpoint $y = 0$ and upper endpoint $y = k$. We map each test $T \in \mathcal{T} \setminus C_1$ to a *move* between two consecutive integer points in $\{y = 0, y = 1, \ldots, y = k\}$ of L as follows. Let $r(T)$ be the rank of T. The mapping depends on the test result of T. If the test result of T is pure, because $T \in \mathcal{T} \setminus C_1$ is not the last test T_q performed in step (a), we have $0 \le r(T) \le k-1$ (since otherwise if $r(T) \ge k$, then the test immediately following T performed in step (a) will be of rank greater than k, contradicting Observation 3.4); we map T to a move on L from point $y = r(T)$ to point $y = r(T)+1$. If the test result of T is contaminated, then since $T \notin C_1$, we have $1 \le r(T) \le k$, and we map T to a move on L from point $y = r(T)$ to point $y = r(T)-1$. We use $(p_1 \to p_2)$ to denote the move from point $y = p_1$ to point $y = p_2$ on L, where $0 \le p_1, p_2 \le k$ are integers and $p_2 = p_1 + 1$ or $p_2 = p_1 - 1$.

By the above mapping, the tests in $\mathcal{T} \setminus C_1$ (with their original ordering in $\mathcal{T}$) map to a walk W^*, which is a set consisting of a sequence of moves; each move may either go upward or go downward on L from point $y = k$ to point $y = r_q$. Since $k \ge r_q$ and the walk W^* goes within L from $y = k$ to $y = r_q$ (note that the upper endpoint of L is $y = k$ and the lower endpoint of L is $y = 0$), it follows that for any $p \in \{k, k-1, \ldots, r_q+1\}$ the number of moves $(p \to p-1)$ in W^* is one more than the number of moves $(p-1 \to p)$ in W^*, and for any $p \in \{r_q, r_q-1, \ldots, 1\}$ the number of moves $(p \to p-1)$ in W^* is equal to the number of moves $(p-1 \to p)$ in W^*.

From the definition of C_2, the tests in C_2 map to $k - r_q$ moves $(k \to k-1), (k-1 \to k-2), \ldots, (r_q+1 \to r_q)$. We use P^* to denote the set of the above $k - r_q$ moves (then P^* can be viewed as a simple directed path on the y axis from point $y = k$ to point $y = r_q$). Since $C_3 = \mathcal{T} \setminus (C_1 \cup C_2) = (\mathcal{T} \setminus C_1) \setminus C_2$, the above mapping gives a one-to-one correspondence between the tests in C_3 and the moves in $W^* \setminus P^*$. From the above analysis, for any $1 \le p \le k$, the number of moves $(p \to p-1)$ in $W^* \setminus P^*$ is equal to the number of moves $(p-1 \to p)$ in $W^* \setminus P^*$. Thus, the total number of moves in $W^* \setminus P^*$ is even. The number of tests in C_3, which is equal to the number of moves in $W^* \setminus P^*$, is also even. By the definition of a zig-zag pair (Definition 3.3), for any $1 \le p \le k$, two tests mapping to the two moves $(p \to p-1)$ and $(p-1 \to p)$ form a zig-zag pair, which establishes the lemma (see Figure 1). □

3.2. Estimating the Total Number of Tests Used by Algorithm Z

In §3.1, we have partitioned $\mathcal{T}$ into three disjoint subsets: C_1, C_2, and C_3. Thus, by Equation (2), the total number of tests used by Algorithm Z is

$$t_Z = \sum_{T \in \mathcal{T}} |I(T)| \le \sum_{T \in C_1} |I(T)| + \sum_{T \in C_2} |I(T)| + \sum_{T \in C_3} |I(T)|. \quad (3)$$

For formula (3), we first estimate the sum of the first term and the third term $\sum_{T \in C_1 \cup C_3} |I(T)|$; then we incorporate the second term $\sum_{T \in C_2} |I(T)|$ to reach an upper bound on t_Z.

Definition 3.6 (Identified Items). For a test $T \in \mathcal{T}$, define $D(T)$ to be the set of items identified by tests in $I(T)$, and define $n(T) = |D(T)|$.

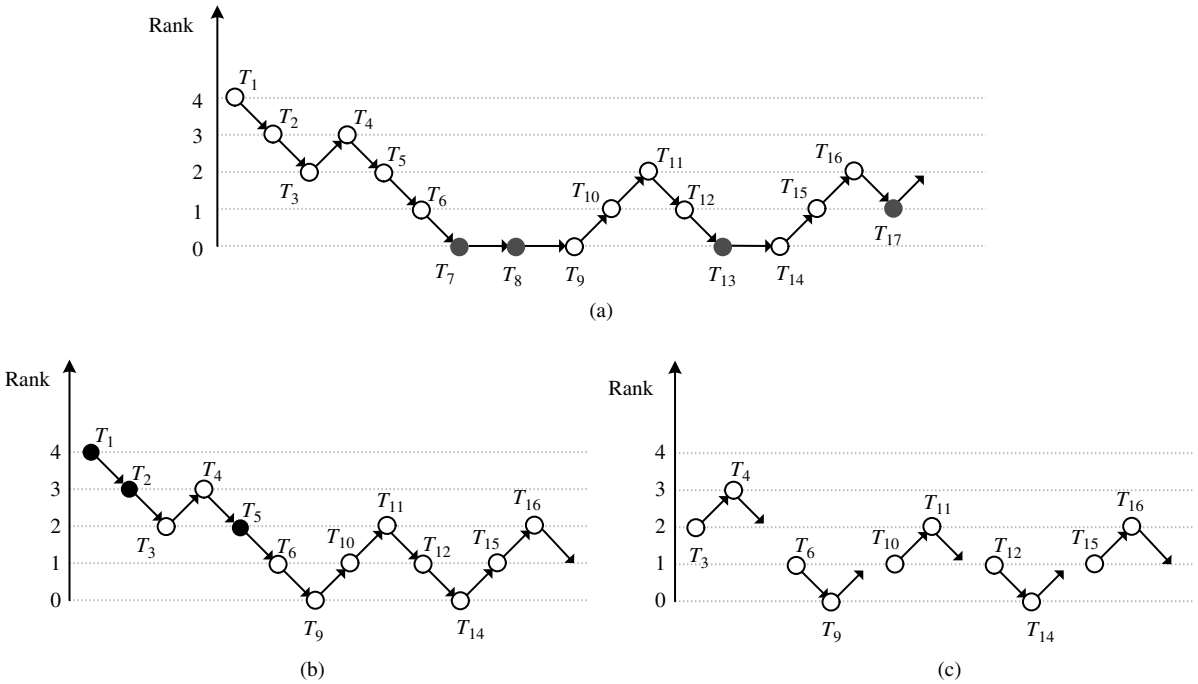

Figure 1 An Example to Illustrate the Partition of $\mathcal{T}$ Into Three Classes and the Partition of C_3 into $|C_3|/2$ Zig-Zag Pairs

Notes. (a) The circles represent tests performed in step (a) (i.e., tests in $\mathcal{T}$), in the order that they are performed by Algorithm Z. In this example, we have $q = |\mathcal{T}| = 17$. An arrow pointing to the bottom right from a circle (test) T indicates that the test result of T is contaminated, and the rank of the test (performed in step (a)) immediately following T is decreased by 1; an arrow pointing to the top right from a test T indicates that the test result of T is pure, and the rank of the test (performed in step (a)) immediately following T is increased by 1. Notice that it is possible that a test T of rank 0 has test result contaminated. In this case we draw a horizontal arrow from T pointing to the right, indicating that the rank of the test immediately following T in step (a) is also 0. For this example, $C_1 = \{T_7, T_8, T_{13}, T_{17}\}$. (b) Tests in $\mathcal{T} \setminus C_1$. They map to a walk W^* with start point $y = k\ (= 4)$ and end point $y = r_q\ (= 1)$. The second class $C_2 = \{T_1, T_2, T_5\}$. (c) Tests in $C_3 = \mathcal{T} \setminus (C_1 \cup C_2)$. In this example, five zig-zag pairs are formed. One possibility for the pairing could be (T_4, T_3), (T_6, T_9), (T_{11}, T_{10}), (T_{12}, T_{14}), and (T_{16}, T_{15}), as illustrated. The pairing is not unique; e.g., we can also pair T_6 with T_{14} and pair T_{12} with T_9, as zig-zag pairs, etc.

From Definitions 3.1 and 3.6, for a test $T \in \mathcal{T}$, $D(T)$ is the set of items identified by test T together with the DIG procedure incurred by T in step (c) (if the test result of T is contaminated, otherwise the DIG procedure is not called). That is, let S' be the subset to which test T is applied in step (a); if the test result of T is contaminated, then T and its incurred DIG procedure identify one defective item $e \in S'$ and a subset $S'' \subseteq S'$ of good items (note that S'' can be empty); thus, $D(T) = \{e\} \cup S''$. If the test result of T is pure, then $I(T) = \{T\}$, and T identifies all items in S' as good items; thus $D(T) = S'$.

Lemma 3.7. *For each test $T \in C_1$, $|I(T)| \leq 2 + \log n(T)$.*

Proof. We first consider tests in C_1 except T_q, where T_q is the last test performed in step (a). By the definition of class C_1, each such test T ($T \in C_1$, and $T \neq T_q$) is of rank 0 and is contaminated, so T is applied to a subset of size one and identifies the only one item in the subset as defective; thus, $n(T) = 1$. Test T incurs only one test (i.e., itself), since a subsequent DIG procedure is not necessary. Hence, $I(T) = \{T\}$ and $|I(T)| \neq 1$. Therefore, we have $|I(T)| \leq 2 + \log n(T)$.

For test T_q, let S' be the subset to which T_q is applied in step (a). There are the following two possible cases.

Case 1. The test result of T_q is pure. Then all items in S' are identified as good items, so $n(T) = |S'| \geq 1$. In this case, a subsequent DIG procedure applied to S' is not required, so $I(T_q) = \{T_q\}$ and $|I(T_q)| \neq 1$. Thus, $|I(T_q)| \leq 2 + \log n(T_q)$ holds.

Case 2. The test result of T_q is contaminated. Since T_q is the last test performed in step (a), it must be that S' is equal to the current set S, and all items in S' are successfully identified by the DIG procedure incurred by T_q. Thus, in this case we have $n(T_q) = |S'|$. By Lemma 2.1, the number of tests used by Procedure DIG on subset S' is $\lceil \log |S'| \rceil$, the total number of tests incurred by T_q (i.e., test T_q together with all the tests used by Procedure DIG on S') is $|I(T_q)| \neq 1 + \lceil \log |S'| \rceil \neq 1 + \lceil \log n(T_q) \rceil \leq 2 + \log n(T_q)$.

Therefore, in either case we have $|I(T_q)| \leq 2 + \log n(T_q)$, the lemma is established. □

By Lemma 3.5, C_3 can be partitioned into $|C_3|/2$ zig-zag pairs, such that each test in C_3 appears in exactly one pair. We fix one such partition and let ZP denote the set of $|C_3|/2$ zig-zag pairs obtained from this partition.

For any zig-zag pair $P = (T_i, T_j)$ in ZP, where $T_i, T_j \in C_3$, based on Definitions 3.1 and 3.6, let $I(P)$

denote the set of all tests incurred by T_i and T_j; i.e., $I(P) = I(T_i) \cup I(T_j)$. Let $D(P)$ denote the set of all items identified by tests in $I(P)$—i.e., $D(P) = D(T_i) \cup D(T_j)$—and let $n(P) = |D(P)|$.

From Algorithm Z, for any two different tests $T_i, T_j \in \mathcal{T}$, clearly, $I(T_i)$ is disjoint to $I(T_j)$, and $D(T_i)$ is disjoint to $D(T_j)$. Thus, for any zig-zag pair $P = (T_i, T_j)$, we have

$$\begin{aligned} |I(P)| &= |I(T_i) \cup I(T_j)| = |I(T_i)| + |I(T_j)|, \\ n(P) &= |D(P)| = |D(T_i) \cup D(T_j)| \\ &= |D(T_i)| + |D(T_j)| = n(T_i) + n(T_j), \end{aligned}$$

and the third term in Equation (3) can be written as

$$\sum_{T \in C_3} |I(T)| = \sum_{P \in ZP} |I(P)|.$$

Lemma 3.8. *For each zig-zag pair $P = (T_i, T_j)$, $|I(P)| \le 3 + \log n(P)$.*

Proof. Let v be the rank of the pair $P = (T_i, T_j)$. By Definition 3.3, T_i has test result contaminated, and an incurred DIG procedure together with test T_i are applied to a subset S_1 of size at most 2^v; T_j has test result pure and is applied to another subset S_2 of size *exactly* 2^{v-1}. Thus, one defective item and an unspecified number of good items from S_1 are identified, and all items in S_2 are identified as good items, it follows that the total number of identified items satisfies $n(P) \ge 1 + |S_2| = 1 + 2^{v-1}$. In contrast, the total number of tests incurred by T_i and T_j, $|I(P)|$, is two plus the number of tests used by DIG(S_1). By Lemma 2.1, DIG(S_1) uses at most v tests, hence $|I(P)| \le 2 + v < 3 + \log n(P)$. □

Based on Lemmas 3.7 and 3.8, we have the following upper bound on the sum of the first and the third term in Equation (3).

Lemma 3.9. *Let x $(0 \le x \le d)$ denote the total number of defective items identified by tests in $\bigcup_{T \in C_1 \cup C_3} I(T)$. If $x = 0$, then we have $\sum_{T \in C_1 \cup C_3} |I(T)| = 1$; if $1 \le x \le d$, then we have $\sum_{T \in C_1 \cup C_3} |I(T)| \le 1 + 3x + x \log(n/x)$.*

Proof. From the definition of ZP, we have

$$\begin{aligned} \sum_{T \in C_1 \cup C_3} |I(T)| &= \sum_{T \in C_1} |I(T)| + \sum_{T \in C_3} |I(T)| \\ &= \sum_{T \in C_1} |I(T)| + \sum_{P \in ZP} |I(P)|. \end{aligned}$$

From the definitions of class C_1 and zig-zag pairs, for each test T (except T_q) in C_1, we have $I(T) = \{T\}$ and test T identifies exactly one defective item. Tests in $I(T_q)$ identify no defective item if the test result of T_q is pure and identify one defective item if the test result is contaminated. For each zig-zag pair $P \in ZP$, tests in $I(P)$ identify exactly one defective item.

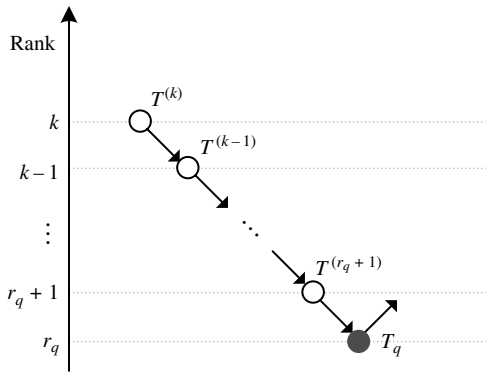

Figure 2 Tests Performed in step (a) by Algorithm Z when $x = 0$ Holds, Where x Is the Total Number of Defective Items Identified by Tests in $\bigcup_{T \in C_1 \cup C_3} I(T)$

Note. In this case (i.e., when $x = 0$), we have $C_1 = \{T_q\}$, where T_q is the last test performed in step (a), and the test result of T_q is pure, $C_2 = \{T^{(k)}, T^{(k-1)}, \ldots, T^{(r_q+1)}\}$, and C_3 is empty.

Thus, if $x = 0$, it must be that C_1 only contains one test T_q, and the test result of T_q is pure. Moreover, ZP (and so class C_3) is empty (see Figure 2). Therefore, if $x = 0$ we have

$$\sum_{T \in C_1 \cup C_3} |I(T)| = \sum_{T \in C_1} |I(T)| + \sum_{T \in C_3} |I(T)| = |I(T_q)| = 1,$$

where the last equality is because the test result of T_q is pure, so the set of all tests incurred by T_q is simply $I(T_q) = \{T_q\}$; hence $|I(T_q)| = 1$. Next we prove the lemma for $1 \le x \le d$. There are the following two possible cases.

Case 1. The test result of T_q is pure. Then $I(T_q) = \{T_q\}$, since a subsequent DIG procedure is not called. Hence $|I(T_q)| = 1$, and tests in $I(T_q)$ identify no defective item. Thus, in this case the total number of defective items identified by tests in $\bigcup_{T \in C_1 \cup C_3} I(T)$—i.e., tests in $(\bigcup_{T \in C_1} I(T)) \cup (\bigcup_{P \in ZP} I(P))$—is $(|C_1| - 1) + |ZP|$. Hence, $(|C_1| - 1) + |ZP| \ge x$.

We have

$$\begin{aligned} &\sum_{T \in C_1} |I(T)| + \sum_{P \in ZP} |I(P)| \\ &= 1 + \sum_{T \in C_1 \setminus \{T_q\}} |I(T)| + \sum_{P \in ZP} |I(P)| && (4) \\ &< 1 + \sum_{T \in C_1 \setminus \{T_q\}} (3 + \log n(T)) + \sum_{P \in ZP} (3 + \log n(P)) && (5) \\ &= 1 + 3(|C_1| + |ZP| - 1) + \sum_{T \in C_1 \setminus \{T_q\}} \log n(T) \\ &\quad + \sum_{P \in ZP} \log n(P) \\ &= 1 + 3x + \sum_{T \in C_1 \setminus \{T_q\}} \log n(T) + \sum_{P \in ZP} \log n(P) \\ &\le 1 + 3x + (|C_1| + |ZP| - 1) \end{aligned}$$

$$\cdot \log \frac{\sum_{T\in C_1\setminus\{T_q\}} n(T) + \sum_{P\in ZP} n(P)}{(|C_1| + |ZP| - 1)} \tag{6}$$

$$= 1 + 3x + x \log \frac{\sum_{T\in C_1\setminus\{T_q\}} n(T) + \sum_{P\in ZP} n(P)}{x}$$

$$\le 1 + 3x + x \log \frac{n}{x}, \tag{7}$$

where Equality (4) holds because $|I(T_q)|=1$; Inequality (5) holds by Lemmas 3.7 and 3.8; Inequality (6) holds because $\log(\cdot)$ is concave and $|C_1| + |ZP| - 1 = x$, and $x \ge 1$ as assumed in the lemma; and Inequality (7) holds because for any two different tests T_i, $T_j \in \mathcal{T}$, $D(T_i)$ is disjoint to $D(T_j)$, and since C_1 is disjoint to C_3 (i.e., ZP), we have that each item in S is counted at most once in the summation $\sum_{T\in C_1\setminus\{T_q\}} n(T) + \sum_{P\in ZP} n(P)$, and so $\sum_{T\in C_1\setminus\{T_q\}} n(T) + \sum_{P\in ZP} n(P) \le n$.

Case 2. The test result of T_q is contaminated. Then tests in $I(T_q)$ identify one defective item. Thus, in this case the total number of defective items identified by tests in $\bigcup_{T\in C_1\cup C_3} I(T)$ is $|C_1| + |ZP|$. Hence $|C_1| + |ZP| \ne x$. As in Case 1, we have

$$\sum_{T\in C_1} |I(T)| + \sum_{P\in ZP} |I(P)|$$

$$< \sum_{T\in C_1} (3 + \log n(T)) + \sum_{P\in ZP} (3 + \log n(P)) \tag{8}$$

$$= 3(|C_1| + |ZP|) + \sum_{T\in C_1} \log n(T) + \sum_{P\in ZP} \log n(P)$$

$$\le 3(|C_1| + |ZP|) + (|C_1| + |ZP|)$$

$$\cdot \log \frac{\sum_{T\in C_1} n(T) + \sum_{P\in ZP} n(P)}{(|C_1| + |ZP|)} \tag{9}$$

$$= 3x + x \log \frac{\sum_{T\in C_1} n(T) + \sum_{P\in ZP} n(P)}{x}$$

$$\le 3x + x \log \frac{n}{x}, \tag{10}$$

where Inequalities (8)–(10) hold for the same reasons as Inequalities (5)–(7) in Case 1.

By combining the above two cases, the lemma is proved. □

For the second term $\sum_{T\in C_2} |I(T)|$ in Equation (3), we have the following lemma.

Lemma 3.10. *Let x $(0 \le x \le d)$ denote the total number of defective items identified by tests in $\bigcup_{T\in C_1\cup C_3} I(T)$. Then $\sum_{T\in C_2} |I(T)| \le [(2k+3) - (d-x)](d-x))/2$.*

Proof. By the definition of C_2, for each test $T \in \mathcal{C}_2$, tests in $I(T)$ identify exactly one defective item (and an unspecified number of good items). Thus, the total number of defective items identified by tests in $\bigcup_{T\in C_2} I(T)$ is $|C_2| \le k - r_q$, where $k = \lceil \log n \rceil$. Clearly, Algorithm Z successfully identifies all the d defective items in S. Since we assume that the total number of defective items identified by tests in $\bigcup_{T\in C_1\cup C_3} I(T)$ is x, and notice that the set of all tests performed by Algorithm Z is $\bigcup_{T\in C_1\cup C_2\cup C_3} I(T)$, it follows that the total number of defective items identified by tests in $\bigcup_{T\in C_2} I(T)$ is $d - x$. Therefore, $k - r_q = d - x$.

For each test $T^{(j)} \in \mathcal{C}_2$ of rank j $(r_q + 1 \le j \le k)$, $T^{(j)}$ and the DIG procedure incurred by $T^{(j)}$ are applied to a subset of size no more than 2^j. Thus, by Lemma 2.1, $|I(T^{(j)})| \le j + 1$. We have

$$\sum_{T\in C_2} |I(T)| \le \sum_{j=r_q+1}^{k} |I(T^{(j)})|$$

$$\le \sum_{j=r_q+1}^{k} (j+1)$$

$$= \frac{(r_q + k + 3)(k - r_q)}{2}$$

$$= \frac{[(2k+3) - (k - r_q)](k - r_q)}{2}$$

$$= \frac{[(2k+3) - (d-x)](d-x)}{2}. \quad \square$$

In addition, the following lemma is needed to prove a final upper bound on t_Z, the total number of tests used by Algorithm Z.

Lemma 3.11. *Let $f_d(x) = 1 + 3d + x \log(n/x) + ([(2k-3) - (d-x)](d-x))/2$, where d and n are integers, $0 < d \le n$, and $k = \lceil \log n \rceil$. Then, $f_d(x) \le d \log(n/d) + 3d + 0.5 \log^2 d + 1.5 \log d + 2$ for integers $x \in \{1, 2, \ldots, d\}$.*

Proof. The proof is available in Appendix A. □

Now we are ready to prove the main result in this paper.

Theorem 3.12. *For $1 \le d \le n$, $M_Z(d \mid n) \le d \log(n/d) + 3d + 0.5 \log^2 d + 1.5 \log d + 2$.*

Proof. As in Lemmas 3.9 and 3.10, let x $(0 \le x \le d)$ be the total number of defective items identified by tests in $\bigcup_{T\in C_1\cup C_3} I(T)$. We divide the proof into the following two possible cases: (1) $1 \le x \le d$ and (2) $x = 0$.

Case 1. $1 \le x \le d$. In this case, by Lemmas 3.9 and 3.10,

$$t_Z = \sum_{T\in C_1\cup C_3} |I(T)| + \sum_{T\in C_2} |I(T)|$$

$$\le 1 + 3x + x \log \frac{n}{x} + \frac{[(2k+3) - (d-x)](d-x)}{2}$$

$$= 1 + 3d + x \log \frac{n}{x} + \frac{[(2k-3) - (d-x)](d-x)}{2}.$$

The last expression in the above derivation is $f_d(x)$ in Lemma 3.11. By Lemma 3.11, for integer $1 \le$

$x \le d$ we have $f_d(x) \le d\log(n/d) + 3d + 0.5\log^2 d + 1.5\log d + 2$; thus, for this case the theorem is proved.

Case 2. $x = 0$. In this case, by Lemmas 3.9 and 3.10,

$$\begin{aligned} t_Z &= \sum_{T\in C_1\cup C_3} |I(T)| + \sum_{T\in C_2} |I(T)| \\ &\le 1 + \frac{[(2k+3) - d)]d}{2} \\ &< 1 + d\log n - 0.5d^2 + 2.5d, \end{aligned}$$

where the last inequality is because $k = \lceil \log n \rceil < \log n + 1$. Let

$$\begin{aligned} \Delta_d &= d\log\frac{n}{d} + 3d + 0.5\log^2 d + 1.5\log d + 2 \\ &\quad - (1 + d\log n - 0.5d^2 + 2.5d) \\ &= 0.5d^2 + 0.5d + 0.5\log^2 d + 1.5\log d + 1 - d\log d. \end{aligned}$$

For $d \ge 4$, we have $\log d \le 0.5d$, and so

$$\begin{aligned} \Delta_d &\ge 0.5d^2 + 0.5d + 0.5\log^2 d + 1.5\log d + 1 - 0.5d^2 \\ &= 0.5d + 0.5\log^2 d + 1.5\log d + 1 \\ &> 0. \end{aligned}$$

For $d = 1, 2, 3$, it can be verified directly that $\Delta_1 = 2$, $\Delta_2 = 4$, and $\Delta_3 = 5.87\ldots$. Thus, $\Delta_d > 0$ also holds for $d = 1, 2, 3$. Hence, $\Delta_d > 0$ holds for every integer $d \ge 1$, and so the theorem is proved for the case $x = 0$. □

Though some relaxations are used to obtain the upper bound on $M_Z(d \mid n)$ in Theorem 3.12, we comment that when d is small compared to n (more accurately, when $8 \le d \le \sqrt{n}$), the above analysis for Algorithm Z is almost tight, up to the lower ordered terms $O(\log d)$, as illustrated by the example described in Appendix B.

4. Computational Tests

We have implemented Algorithm Z and Algorithm DP (Du and Park 1994) with Matlab. In this section, we present computational results on the practical performances of these two algorithms and comparisons between them.

Roughly speaking, Algorithm DP works in the following way. For a subset X of items (initially X is the input set S of all items), test X. If X is pure, then identify all items in X as good items; otherwise, use Procedure DIG to identify and remove one defective item (and an unspecified number of good items) from X. Then bisect the current set X (if $|X| > 1$) into two subsets, X' and X'', and repeat the above process for both. The algorithm stops when all items in S are successfully identified. The rationale behind using Procedure DIG to remove one defective item from X (which is also the key ingredient of the algorithm) is that, if both X and X' are contaminated sets and $X' \subset X$, then the test result "X' is contaminated" renders the test result "X is contaminated" redundant; i.e., the test result on X provides no extra information, given the test result on X'. To resolve this issue, in Algorithm DP once a contaminated set X is found, a defective item is identified and removed from X before bisecting it into two smaller subsets. It is proved in Du and Park (1994) that the number of tests used by Algorithm DP is at most $d\log(n/d) + 4d$, where n is the number of items in S and d ($1 \le d \le n$) is the number of defective items in S.

Table 1 reports the simulation results of Algorithms Z and DP for $n = 1{,}000$. For each $d \in \{100, 200, \ldots, 1{,}000\}$, all the statistics are collected as the average over 1,000 independent random problem instances, each of which is uniformly and randomly selected from the set of all the $\binom{n}{d}$ possible problem instances with parameters n and d. In Table 1, N_Z and N_{DP} denote the average number of tests performed by Algorithms Z and DP, respectively; $\sigma_1 = (N_{DP} - N_Z)/N_{DP}$ is the percent of tests that Algorithm Z saves compared to Algorithm DP on average; $\sigma_2 = (N_Z - M(d,n))/M(d,n)$ denotes on average how much more percent of tests Algorithm Z requires compared to the information theoretic lower bound $M(d,n)$; c_Z and c_{DP} denote the average coefficient c for Algorithms Z and DP, respectively, if the number of tests performed by them is written as $d\log(n/d) + cd$.

From σ_1 in Table 1, we can see that the advantage of Algorithm Z over Algorithm DP becomes more obvious as the value of d grows. For example, when $d = 100$ ($= n/10$), Algorithm Z on average saves about 14.2% tests compared to Algorithm DP; when d grows to 500 ($= n/2$), the percentage grows to 33.1%.

From σ_2 in Table 1, we can conclude that when d is not very large (say, less than $n/2$), on average Algorithm Z has very good competitive ratio (in the original sense proposed in Du and Hwang 1993), as compared to the information theoretic lower bound

Table 1 **Computational Results for $n = 1{,}000$**

d	$M(d,n)$	N_Z	N_{DP}	σ_1 (%)	σ_2 (%)	c_Z	c_{DP}
100	465	532.6	621.0	14.2	14.5	2.00	2.89
200	717	829.2	1,018.8	18.6	15.7	1.82	2.77
300	877	1,019.1	1,328.4	23.3	16.2	1.66	2.69
400	966	1,135.2	1,579.2	28.1	17.5	1.52	2.63
500	995	1,198.0	1,790.9	33.1	20.4	1.40	2.58
600	966	1,222.9	1,972.1	38.0	26.6	1.30	2.55
700	877	1,217.5	2,127.1	42.8	38.8	1.22	2.52
800	717	1,188.0	2,261.9	47.5	65.7	1.16	2.51
900	465	1,135.4	2,379.9	52.3	144.2	1.11	2.49
1,000	0	1,055.0	2,483.0	57.5	∞	1.06	2.48

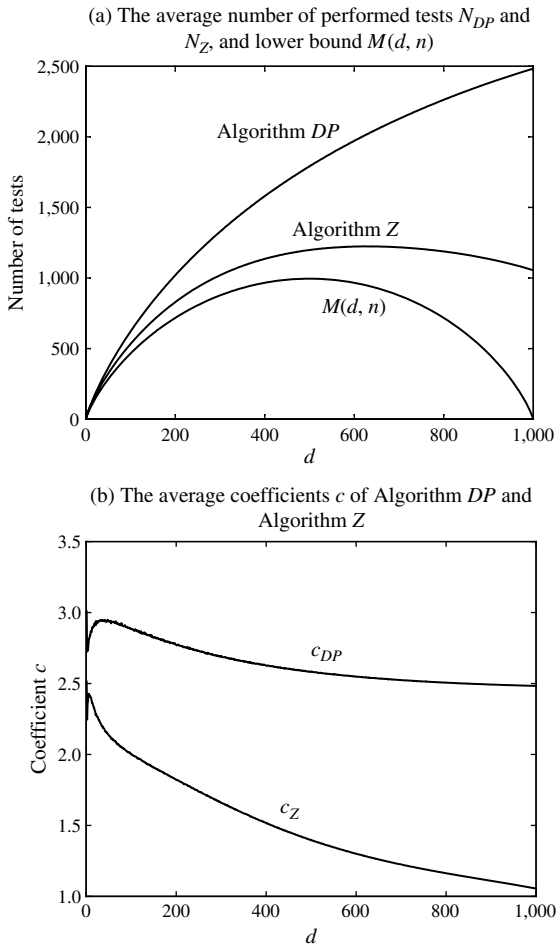

Figure 3 **Average Number of Tests Performed by and the Average Coefficients c of Algorithms DP and Z for $n = 1{,}000$, over 1,000 randomly generated problem instances for each $d = 1, 2, \ldots, 1{,}000$**

$M(d, n)$. For example, when $d = 100$ $(= n/10)$, Algorithm Z on average only requires about 14.5% more tests than $M(d, n)$; as d grows to 500 $(= n/2)$, the percentage only grows to 20.4%. As with σ_1, the general trend of σ_2 also increases as the value of d increases.

Figure 3(a) illustrates the comparison between the average number of tests performed by Algorithms Z and DP and the information theoretic lower bound $M(d, n)$ for $1 \le d \le n$. From Figure 3(b), the value of c_{DP} stays above (or very close to, when d is close to n) 2.5 for the whole range $1 \le d \le n$. In contrast, the general trend of c_Z drops from slightly less than 2.5 to very close to 1 as d increases, which is also consistent with the theoretical finding in Theorem 3.12.

5. Concluding Remarks

Du and Park (1994) gave a sufficient condition for an Algorithm A to be strongly competitive as the following: if there exists a constant c such that for $1 \le d < n$, $M_A(d \mid n) \le d \log(n/d) + cd$, then A is strongly competitive. Therefore, Theorem 3.12 immediately implies the following.

Corollary 5.1. *Algorithm Z is strongly competitive.*

By Equation (1), when $d < \rho n$ for some constant $0 < \rho < 1 - 4/e^2 = 0.45\ldots$, the upper bound $\lceil \log \binom{n}{d} \rceil + 2d$ of Schlaghoff and Triesch (2005) satisfies that

$$\left\lceil \log \binom{n}{d} \right\rceil + 2d \ge d \log \frac{n}{d} + \left(2 + \log\left(e \sqrt{1-\rho}\right)\right) d - 0.5 \log d - 0.5 \log(1-\rho) - 1.567,$$

and so is asymptotically larger than our upper bound $d \log(n/d) + 3d + O(\log^2 d)$, as in the right-hand side of the above inequality the coefficient $2 + \log(e\sqrt{1-\rho})$ for term d is larger than 3. However, it is not hard to verify that when d is close to n, their upper bound is smaller.

Equation (1) also indicates that in general one may not expect to get an upper bound $d \log(n/d) + cd$ with $c < \log e$. Nevertheless, it is interesting to investigate whether our current upper bound in Theorem 3.12 can be further improved. The arguments in Appendix B indicate that we need to come up with new algorithms to achieve this goal for the case $d \le \sqrt{n}$. New upper bounds of form $\log \binom{n}{d} + cd$ with $c < 2$ would also be attractive to investigate.

Whether there exists a nontrivial lower bound other than $\lceil \log \binom{n}{d} \rceil$ for combinatorial group testing with an unknown number of defectives stands as a central open problem in this field. New heuristics with better practical performances than Algorithm Z are also interesting to investigate.

Acknowledgments

The authors thank the referees for their comments and suggestions, which significantly improved the previous versions of this paper in the presentation, organization, and references. This work was partially supported by the National Natural Science Foundation of China [Grants 11101326 and 71071123], and the Program for Changjiang Scholars and Innovative Research Team in University [IRT1173].

Appendix A. Proof of Lemma 3.11

For $d = 1, 2, 3$ and $x \in \{1, 2, \ldots, d\}$, using $k = \lceil \log n \rceil < \log n + 1$ we can enumerate all the six combinations of (x, d) to verify the lemma, as in the following: for $d = 1$, the inequality is $f_1(x) \le \log n + 3$. Since $f_1(1) = \log n + 4$, the lemma holds; for $d = 2$, the inequality is $f_2(x) \le 2 \log n + 8$. Since $f_2(1) = \log n + 3 + k < 2 \log n + 6$, and $f_2(2) = 2 \log n + 3$, the lemma holds. For $d = 3$, the inequality is $f_3(x) \le 3 \log n + 11 + 0.5(\log 3)^2 - 1.5 \log 3 = 3 \log n + 9.87\ldots$, since $f_3(1) = \log n + 3 + 2k < 3 \log n + 7$, $f_3(2) = 2 \log n + k + 6 < 3 \log n + 7$, and $f_3(3) = 3 \log n + 10 - 3 \log 3 = 3 \log n + 5.24\ldots$, the lemma also holds.

Next we prove the lemma for $d \ge 4$. First,

$$f_d'(x) = (d + \log n - k + 1.5 - \log e) - (x + \log x).$$

Since function $h(x) = x + \log x$ is monotonically increasing in x and has value range $(-\infty, +\infty)$, $f_d'(x) = 0$ has a unique solution x_0.

Since $k = \lceil \log n \rceil \geq \log n$, we have

$$\begin{aligned} x_0 + \log x_0 &= d + \log n - k + 1.5 - \log e \\ &\leq d + 1.5 - \log e < d + 2. \end{aligned}$$

Since $d \geq 4$, we have $h(d) \geq d + 2$, by the above inequality and the monotonicity of $h(x) = x + \log x$ we have

$$x_0 < d. \tag{A1}$$

Next we prove a lower bound on x_0. From $x_0 + \log x_0 = d + \log n - k + 1.5 - \log e$, and $k < \log n + 1$, we have

$$x_0 + \log x_0 > d + \log n - (\log n + 1) + 1.5 - \log e = d - (\log e - 0.5).$$

Let $c_0 = \log e - 0.5$. Then, $0 < c_0 < 1$, and the above inequality is

$$h(x_0) = x_0 + \log x_0 > d - c_0.$$

Since $h(x) = x + \log x$ is monotonically increasing in x and

$$\begin{aligned} h(d - \log d - c_0) &= (d - \log d - c_0) + \log(d - \log d - c_0) \\ &< (d - \log d - c_0) + \log d = d - c_0, \end{aligned}$$

together with $h(x_0) > d - c_0$, it follows that

$$x_0 > d - \log d - c_0. \tag{A2}$$

Let $k_0 = \log n + 1.5 - \log e$ (here the choice of k_0 is technical, which is to make the analysis easier and will be explained in Claim A.1 below). Define

$$g_d(x) = 1 + 3d + x \log \frac{n}{x} + \frac{[(2k_0 - 3) - (d - x)](d - x)}{2},$$

which replaces k by k_0 in $f_d(x)$. Then $f_d(x_0) = g_d(x_0) + (k - k_0)(d - x_0)$. Since

$$k - k_0 < (\log n + 1) - (\log n + 1.5 - \log e) = c_0,$$

and by the inequalities in (A1) and (A2), we have $0 < d - x_0 < \log d + c_0$. Thus,

$$f_d(x_0) < g_d(x_0) + c_0(\log d + c_0) < \max_{0<x<d} g_d(x) + c_0 \log d + 1, \tag{A3}$$

where in the above the second inequality holds because $0 < c_0 < 1$. By Claim A.1, given below, when $d \geq 4$ we have

$$\begin{aligned} \max_{0<x<d} g_d(x) &\leq 1 + 3d + d \log \frac{n}{d} + 0.5 \log^2 d \\ &\quad + (2 - \log e) \log d. \end{aligned} \tag{A4}$$

Since $c_0 = \log e - 0.5$, the two inequalities in (A3) and (A4) imply

$$\begin{aligned} f_d(x_0) &< \max_{0<x<d} g_d(x) + c_0 \log d + 1 \\ &\leq d \log \frac{n}{d} + 3d + 0.5 \log^2 d + 1.5 \log d + 2, \end{aligned}$$

which establishes Lemma 3.11. □

Claim A.1. *Assume that $d \geq 4$. Let $g_d(x) = 1 + 3d + x \log(n/x) + ([(2k_0 - 3) - (d - x)](d - x))/2$ for $0 < x < d$, where $k_0 = \log n + 1.5 - \log e$ (here k_0 is chosen to make $g_d'(x) = d - x - \log x$, which makes the analysis easier, as to be seen in the proof of this claim given below). Then $g_d'(x) = 0$ has a unique solution x_1, and $d - \log d < x_1 < d - \log d + (2 \log d)/d$. Moreover,*

$$\begin{aligned} \max_{0<x<d} g_d(x) = g_d(x_1) &\leq 1 + 3d + d \log \frac{n}{d} + \frac{\log^2 d}{2} \\ &\quad + (2 - \log e) \log d. \end{aligned}$$

Before proving Claim A.1, first we need the following observation.

Observation A.2. $-\log(1 - \log d/d) \leq (2 \log d)/d$ for $d \geq 4$.

Proof of Observation A.2. Since

$$\frac{-\log(1-x)}{x}' = \frac{\log e}{x^2}\left(\frac{x}{1-x} - \ln\left(1 + \frac{x}{1-x}\right)\right) > 0, \quad \text{for } 0 < x < 1,$$

where in the above the inequality holds because $y > \ln(1 + y)$ for $y > 0$, it follows that $-\log(1 - x)/x$ is monotonically increasing when $0 < x < 1$. Also, it is easy to prove that $0 < \log d/d \leq \frac{1}{2}$ for $d \geq 4$. Thus, for $d \geq 4$ we have $-\log(1 - \log d/d)/\log d/d \leq -\log(1 - 1/2)/(1/2) = 2$. That is, $-\log(1 - \log d/d) \leq (2 \log d)/d$ for $d \geq 4$. □

Proof of Claim A.1. First,

$$\begin{aligned} g_d'(x) &= x \log n - x \log x - \frac{(2k_0 - 3)(x - d)}{2} - \frac{(x - d)^2}{2}' \\ &= \log n - (\log x + \log e) - (k_0 - 1.5) - (x - d) \\ &= d - x - \log x, \end{aligned}$$

where the last equality holds because $k_0 = \log n + 1.5 - \log e$. Thus, $g_d''(x) = -1 - (\log e)/x < 0$ for $x > 0$, and so $g_d(x)$ is concave for $x > 0$. Also, since $d \geq 4$, we have $g_d'(0^+) > 0$ (that is, when x approaches 0 from the above, $g_d'(x) > 0$), and $g_d'(d) = -\log d < 0$. Thus, for x arbitrarily close to but greater than 0, $g_d(x)$ is increasing in x; for x arbitrarily close to but smaller than d, $g_d(x)$ is decreasing in x. Therefore, $\max_{0<x<d} g_d(x)$ is achieved at the unique solution of $g_d'(x) = 0$ (the uniqueness follows from the monotonicity of $x + \log x$). Let x_1 denote the unique solution of $g_d'(x) = 0$; then we have $x_1 + \log x_1 = d$.

We first show the lower and upper bounds on x_1 in the claim. Since $d \geq 4$, we have $x_1 > 1$, so $x_1 = d - \log x_1 < d$, which implies the lower bound $x_1 = d - \log x_1 > d - \log d$, and the upper bound

$$\begin{aligned} x_1 &= d - \log x_1 < d - \log(d - \log d) \\ &= d - \log d - \log\left(1 - \frac{\log d}{d}\right) \leq d - \log d + \frac{2 \log d}{d}, \end{aligned}$$

where the last inequality is by Observation A.2. Next we estimate $g_d(x_1)$, based on the above lower and upper bounds on x_1.

$$\begin{aligned} g_d(x_1) &= 1 + 3d + x_1 \log \frac{n}{x_1} + \frac{[(2k_0 - 3) - (d - x_1)](d - x_1)}{2} \\ &= 1 + 3d + x_1 \log n - x_1 \log x_1 \\ &\quad + \frac{[(2 \log n - 2 \log e) - (d - x_1)](d - x_1)}{2} \end{aligned}$$

$$= 1 + 3d + x_1 \log n - x_1 \log x_1 + (d - x_1) \log n - (d - x_1) \log e - \frac{(d - x_1)^2}{2}$$

$$= 1 + 3d + d \log n - x_1 \log x_1 - \frac{(d - x_1)^2}{2} - (d - x_1) \log e.$$

Because $x_1 + \log x_1 = d$, we have

$$-x_1 \log x_1 - \frac{(d - x_1)^2}{2} - (d - x_1) \log e = -x_1(d - x_1) - \frac{(d - x_1)^2}{2} - (d - x_1) \log e = \frac{x_1^2 - d^2}{2} + (x_1 - d) \log e.$$

Therefore,

$$g_d(x_1) = 1 + 3d + d \log n + \frac{x_1^2 - d^2}{2} + (x_1 - d) \log e. \quad \text{(A5)}$$

From $d \geq 4$ and $d - \log d < x_1 < d - \log d + (2 \log d)/d$, we have

$$\frac{x_1^2 - d^2}{2} + (x_1 - d) \log e \leq \frac{1}{2}\left(d - \log d + \frac{2 \log d}{d}\right)^2 - \frac{d^2}{2} + \left(-\log d + \frac{2 \log d}{d}\right) \log e$$

$$= \frac{\log^2 d}{2} + \frac{2 \log^2 d}{d^2} - d \log d + 2 \log d - \frac{2 \log^2 d}{d} + \left(-\log d + \frac{2 \log d}{d}\right) \log e$$

$$= \frac{\log^2 d}{2} - d \log d + (2 - \log e) \log d + \frac{2 \log^2 d}{d^2}\left(1 - d + \frac{d \log e}{\log d}\right).$$

Since $\log e < 1.5$, for $d \geq 4$ we have $(d \log e)/\log d < (1.5d)/\log d \leq (1.5d)/\log 4 = (3d)/4$, and so $1 - d + (d \log e)/\log d < 1 - d/4 \leq 0$. Hence, the above inequality implies that

$$\frac{x_1^2 - d^2}{2} + (x_1 - d) \log e \leq \frac{\log^2 d}{2} - d \log d + (2 - \log e) \log d.$$

By combining this inequality with Equation (A5), we obtain

$$g_d(x_1) \leq 1 + 3d + d \log n + \frac{\log^2 d}{2} - d \log d + (2 - \log e) \log d = 1 + 3d + d \log \frac{n}{d} + \frac{\log^2 d}{2} + (2 - \log e) \log d,$$

establishing Claim A.1. □

Appendix B. An Asymptotically Tight Example for $8 \leq d \leq \sqrt{n}$

Without loss of generality, we consider the input set S (and X) to Algorithm Z (and Procedure DIG) as an ordered set, and we assume that during the execution of Algorithm Z (and Procedure DIG), the ordering of all the unidentified items in S (and X) always remains the same, as their original ordering in S (and X). More specifically, assume that in step 1 of Procedure DIG, we always choose a prefix of the ordered set X to form subset Y; in step (a) of Algorithm Z, we always choose a prefix of the ordered set S to form subset S'; and each time after step (c) of Algorithm Z is executed, all unidentified items in S' (with their original ordering unchanged) are put back to set S as the first part of S, so that the ordering of all the unidentified items in S always stays the same as their original ordering in S.

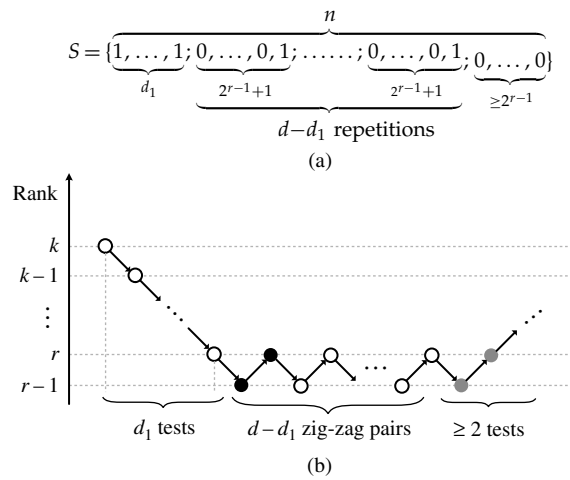

Figure B.1 (a) The Input Set S Described in Appendix B, Where "0" Stands for a Good Item, "1" Stands for a Defective Item, $|S| \neq n = 2^k$, the Number of Defective Items in S is $d = 2^{d_1}$, and $r = k - d_1 + 1$; (b) Tests Performed in step (a) of Algorithm Z on Set S

Note. The tests performed in step (a) start with d_1 tests of results contaminated, and of ranks $k, k-1, \ldots, r$; followed by $2(d - d_1)$ tests of results alternating between "pure" and "contaminated," and of ranks alternating between $r-1$ and r, where each two consecutive tests with respective ranks $r-1$ and r (e.g., the two black nodes) form a zig zag pair; then followed by a sequence of (at least two) tests and they are all of test results pure (the gray nodes).

Consider the scenario where $n = 2^k$, $d = 2^{d_1}$ for integers $k > d_1$; let $r = k - d_1 + 1$, and consider the following input set S of n items in which d of them are defectives. The ordered items in S can be divided into three parts: the first part is a sequence of d_1 defective items, the second part is a sequence of $(d - d_1)$ repetitions such that each is formed by 2^{r-1} good items followed by one defective item, and the third (last) part is a sequence of good items of length at least $2^r - 1$ (see Figure B.1(a)). That is, we assume that the following inequality holds:

$$n - d_1 - (d - d_1)(2^{r-1} + 1) \geq 2^r - 1, \quad \text{(B1)}$$

which, when $d > 4$, is equivalent to $n \geq (d(d_1 - 1))/(\log d - 2)$, and so is clearly satisfied when $8 \leq d \leq \sqrt{n}$. Next, we calculate the total number of tests Algorithm Z performs on S. According to the three parts of S described above, the execution of Algorithm Z on S can be divided into three phases (see Figure B.1(b)).

- *Phase* 1: From input set S and the assumptions on Algorithm Z (and Procedure DIG) made at the beginning of this section, it can be verified that for the first d_1 tests performed

in step (a) by Algorithm Z on S, each has test result contaminated. These d_1 tests are performed on subsets of sizes $2^k, 2^{k-1}, \ldots, 2^r$, for each such test a subsequent DIG procedure is incurred, and each such incurred DIG procedure identifies only one defective item and zero good items.

For each j ($r \le j \le k$), for a subset S' of size 2^j, by Lemma 2.1 the number of tests used by Procedure DIG on S' is j, together with the test performed on S' in step (a), in total $j+1$ tests are used. Thus, for the first d_1 tests performed in step (a), the total number of tests incurred by them, i.e., the total number of tests performed by Algorithm Z in Phase 1, is $\sum_{j=r}^{k}(j+1) = (k+r+2)d_1)/2$. In contrast, the total number of items identified by Algorithm Z in Phase 1—i.e., identified by these d_1 tests performed in step (a) and their incurred tests—is d_1, and they are all defective items.

- *Phase* 2: From input set S and by Inequality (B1), it can be verified that the next $2(d-d_1)$ tests (which immediately follow the first d_1 tests) performed in step (a) have test results alternating between pure and contaminated, and are applied to subsets of sizes alternating between 2^{r-1} and 2^r. Moreover, each of the subsets of size 2^{r-1} contains 2^{r-1} good items and so has test result pure, and each of the subsets of size 2^r is formed by one defective item followed by 2^r-1 good items, and so has test result contaminated. For each test result contaminated from these $2(d-d_1)$ tests, Procedure DIG is called in step (c). Since each such incurred DIG procedure is applied to a subset S' of size 2^r and the first item of S' is defective, it follows that the DIG procedure identifies only one defective item (i.e., the first item in S') and zero good items from S'. Here S' is considered to be an ordered subset, as mentioned at the beginning of this section.

These $2(d-d_1)$ tests performed in step (a) form $(d-d_1)$ zig-zag pairs, where each two consecutive tests (i.e., the first and second test, the third and fourth test, etc.) form a pair. Since d_1 defective items have been identified in Phase 1, all the remaining $d-d_1$ defective items in S are identified in Phase 2. For each of the above $(d-d_1)$ zig-zag pairs, one test is performed on a pure subset of size 2^{r-1}, and one test and a subsequent DIG procedure with—by Lemma 2.1—r tests are performed on a contaminated subset of size 2^r, thus the number of tests incurred by each zig-zag pair is $2+r$. Therefore, the total number of tests performed by Algorithm Z in Phase 2 is $(d-d_1)(r+2)$. Each zig-zag pair, together with its incurred tests, identifies $2^{r-1}+1$ items, of which one item is defective and the other 2^{r-1} items are good.

- *Phase* 3: After Phase 2, all the d defective items in S are identified; thus Phase 3 is formed by a sequence of tests performed in step (a) with test results pure. In fact, it can be seen that this sequence contains at least two tests, as follows.

After Phase 2, the number of items remained (i.e., unidentified) in S is

$$n - d_1 - (d-d_1)(2^{r-1}+1),$$

which is at least 2^r-1 by Inequality (B1). Since we assume that $k > d_1$, and so $r = k - d_1 + 1 \ge 2$, thus $2^r - 1 > 2^{r-1}$. The last test performed in step (a) in Phase 2 is of rank r and is of test result contaminated, it follows that the first test performed (in step (a)) in Phase 3 is of rank $r-1$, and so the test is performed on a pure subset of size 2^{r-1}. After these 2^{r-1} items are identified as good items and removed from S, the number of remaining items in S is at least $2^r - 1 - 2^{r-1} \ge 1$, where the inequality is because $r \ge 2$, as shown above. Thus, at least one more test performed in step (a) is required. Therefore, the number of tests performed in step (a) in Phase 3 is at least two, and they are all of test results pure.

Combining Phases 1 and 2, the total number of tests performed by Algorithm Z is at least

$$\frac{(k+r+2)d_1}{2} + (d-d_1)(r+2).$$

By substituting in $k = \log n$, $d_1 = \log d$, and $r = k - d_1 + 1 = \log n - \log d + 1$, the above formula can be easily verified being equal to $d\log(n/d) + 3d + 0.5\log^2 d - 1.5\log d$. Compared with the upper bound given in Theorem 3.12, we can see that the input example S constructed above is tight for Algorithm Z when $8 \le d \le \sqrt{n}$, up to the lower ordered terms $O(\log d)$.

The assumptions made at the beginning of this section on Algorithm Z (and Procedure DIG)—that they preserve the ordering of all the unidentified items in S—are not essential. The arguments (on the tightness of the upper bound given in Theorem 3.12 when $8 \le d \le \sqrt{n}$) presented here can be easily transformed into adversary type arguments, which are applicable to the more general frameworks of Algorithm Z (and Procedure DIG) as described in §2, that is, without the order preserving assumptions on the unidentified items in set S.

References

Abolnikov L, Dukhovny A (2003) Optimization in HIV screening problems. *J. Appl. Math. Stochastic Anal.* 16:361–374.

Bar-Lev SK, Parlar M, Perry D, Stadje W, Schouten FAVDD (2007) Applications of bulk queues to group testing models with incomplete identification. *Eur. J. Oper. Res.* 183:226–237.

Bar-Noy A, Hwang FK, Kessler I, Kutten S (1994) A new competitive algorithm for group testing. *Discrete Appl. Math.* 52:29–38.

Barillot E, Lacroix B, Cohen D (1991) Theoretical analysis of library screening using a n-dimensional pooling strategy. *Nucleic Acids Res.* 19:6241–6247.

Bruno WJ, Balding DJ, Knill EH, Bruce D, Whittaker C, Doggett N, Stallings R, Torney DC (1995) Design of efficient pooling experiments. *Genomics* 26:21–30.

Claeys D, Walraevens J, Laevens K, Bruneel H (2010) A queueing model for general group screening policies and dynamic item arrivals. *Eur. J. Oper. Res.* 207:827–835.

Cormode G, Muthukrishnan S (2005) What's hot and what's not: Tracking most frequent items dynamically. *ACM Trans. Database Systems* 30:249–278.

Damaschke P, Sheikh Muhammad A (2012) Randomized group testing both query-optimal and minimal adaptive. *Proc. 38th Internat. Conf. Current Trends in Theory and Practice Comput. Sci. (SOFSEM 2012)*, Lecture Notes in Computer Science, Vol. 7147 (Springer, Berlin), 214–225.

Dorfman R (1943) The detection of defective members of large populations. *Ann. Math. Statist.* 14:436–440.

Du DZ, Hwang FK (1993) Competitive group testing. *Discrete Appl. Math.* 45:221–232.

Du DZ, Hwang FK (2000) *Combinatorial Group Testing and Its Applications*, 2nd ed. (World Scientific, Singapore).

Du DZ, Park H (1994) On competitive group testing. *SIAM J. Comput.* 23:1019–1025.

Du DZ, Xue GL, Sun SZ, Cheng SW (1994) Modifications of competitive group testing. *SIAM J. Comput.* 23:82–96.

Goodrich MT, Hirschberg DS (2008) Improved adaptive group testing algorithms with applications to multiple access channels and dead sensor diagnosis. *J. Combin. Optim.* 15:95–121.

Hong EH, Ladner RE (2002) Group testing for image compression. *IEEE Trans. Image Processing* 11:901–911.

Hong YW, Scaglione A (2004) On multiple access for distributed dependent sources: A content-based group testing approach. *Proc. IEEE Inform. Theory Workshop, San Antonio, Texas*, 298–303.

Hwang FK (1972) A method for detecting all defective members in a population by group testing. *J. Amer. Statist. Assoc.* 67: 605–608.

Kahng AB, Reda S (2006) New and improved BIST diagnosis methods from combinatorial group testing theory. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* 25:533–543.

Kautz WH, Singleton RC (1964) Nonrandom binary superimposed codes. *IEEE Trans. Inform. Theory* 10:363–377.

Li CH (1962) A sequential method for screening experimental variables. *J. Amer. Statist. Assoc.* 57:455–477.

Manasse M, McGeoch LA, Sleator D (1988) Competitive algorithms for on-line problems. *Proc. 20th Annual ACM Sympos. Theory Comput.* (ACM, New York), 322–333.

Mezard M, Toninelli C (2011) Group testing with random pools: Optimal two-stage algorithms. *IEEE Trans. Inform. Theory* 57: 1736–1745.

Schlaghoff J, Triesch E (2005) Improved results for competitive group testing. *Combinatorics, Probab. Comput.* 14:191–202.

Sleator D, Tarjan R (1985) Amortized efficiency of list update and paging rules. *Comm. ACM* 28:202–208.

Sobel M, Groll PA (1959) Group testing to eliminate efficiently all defectives in a binomial sample. *Bell System Technical J.* 38: 1179–1252.

Wein LM, Zenios SA (1996) Pooled testing for HIV screening: Capturing the dilution effect. *Oper. Res.* 44:543–569.

Wolf J (1985) Born again group testing: Multiaccess communications. *IEEE Trans. Inform. Theory* 31:185–191.

Zenios SA, Wein LM (1998) Pooled testing for HIV prevalence estimation: Exploiting the dilution effect. *Statist. Medicine* 17:1447–1467.