JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 18, Number 6, 2011

Pp. 851–864
DOI: 10.1089/cmb.2010.0030

An Almost Optimal Algorithm for Generalized Threshold Group Testing with Inhibitors

HONG-BIN CHEN [1] and ANNALISA DE BONIS [2]

ABSTRACT

Group testing is a search paradigm where one is given a population $\mathcal{S}$ of n elements and an unknown subset $\mathcal{P} \subseteq \mathcal{S}$ of defective elements and the goal is to determine $\mathcal{P}$ by performing tests on subsets of $\mathcal{S}$. In classical group testing a test on a subset $Q \subseteq \mathcal{S}$ receives a YES response if $|Q \cap \mathcal{P}| \geq 1$, and a NO response otherwise. In group testing with inhibitors (GTI), identifying the defective items is more difficult due to the presence of elements called inhibitors that interfere with the queries so that the answer to a query is YES if and only if the queried group contains at least one defective item and no inhibitor. In the present article, we consider a new generalization of the GTI model in which there are two unknown thresholds h and g and the response to a test is YES both in the case when the queried subset contains at least one defective item and less than h inhibitors, and in the case when the queried subset contains at least g defective items. Moreover, our search model assumes that no knowledge on the number $|\mathcal{P}|$ of defective items is given. We derive lower bounds on the minimum number of tests required to determine the defective items under this model and present an algorithm that uses an almost optimal number of tests.

Key words: algorithms, group testing, inhibitors, superimposed codes.

1. INTRODUCTION

GROUP TESTING IS A SEARCH PARADIGM where one is given a population $\mathcal{S}$ of n elements and an unknown subset $\mathcal{P} \subseteq \mathcal{S}$ of defective elements and the goal is to determine $\mathcal{P}$ by performing tests on subsets of $\mathcal{S}$. In classical group testing, a test on a subset $Q \subseteq \mathcal{S}$ receives a YES (positive) response if $|Q \cap \mathcal{P}| \geq 1$, and a NO (negative) response otherwise. Group testing origins track back to World War II when it was used to detect among millions of draftees' blood samples those infected with syphilis (Dorfman, 1943). Since its origins, group testing has been fruitfully applied to several areas such as data compression (Hong and Ladner, 2002), efficient access to storage systems (Kautz and Singleton, 1964), conflict resolution algorithms for multiple-access systems (Berger et al., 1984; Wolf, 1985), quality control in product testing (Sobel and Groll, 1959), data gathering in sensor networks (Hong and Scaglione, 2004), and sequential screening of experimental variables (Li, 1962). Nowadays, some of the most interesting applications of group testing concern the

[1]Department of Applied Mathematics, National Chiao Tung University, Hsinchu, Taiwan.
[2]Dipartimento di Informatica ed Applicazioni, Università di Salerno, Fisciano, Italy.

field of Computational Molecular Biology where it is primarily employed for screening library of clones with hybridization probes (Barillot et al., 1991; Bruno et al., 1995), and sequencing by hybridization (Margaritis and Skiena, 1995; Pevzner and Lipshutz, 1994). We refer readers interested in these issues to the references (Balding et al., 1996; Du and Hwang, 2000, 2006; Farach et al., 1997; Ngo and Du, 2000).

The classical paradigm of group testing is well studied, and there exist algorithms for this search model that use a number of tests very close to the information theoretic lower bound $\lceil \log \binom{n}{|\mathcal{P}|} \rceil$. The many different contexts in which group testing finds application often involve complex querying models that call for extensions of the classical paradigm. Motivated by molecular biology applications, Farach et al. (1997) introduced a variation of group testing known under the name of Group Testing with Inhibitors (GTI). In this model, the input population consists not only of regular samples and defective samples, but also of a third category of samples called inhibitors. Inhibitors correspond to spoiled samples capable to interfere with the tests so as to make their outcomes meaningless. In the GTI model a subset tests positive if and only if it contains one or more defective items and no inhibitor. Many results concerning the GTI model have been studied (Chang et al., 2010; De Bonis, 2008; De Bonis et al., 2005; De Bonis and Vaccaro, 1998; Du and Hwang, 2000, 2006; D'yachkov et al., 2001; Farach et al., 1997; Hwang and Liu, 2003).

Group testing with inhibitors also finds application in the discovery of new drugs. The pharmaceutical companies use group testing for screening collections of hundreds of thousands of chemical compounds, with the purpose of selecting highly active compounds that might lead to the developments of new drugs. This job is made more complicated by the presence of chemical compounds that block the detection of active compounds when appear in the tested pool. An analogous situation occurs in blood testing when different samples of blood are pooled together. Indeed, there might be blood samples that neutralize positive blood samples thus causing false negative responses. For an account on issues concerning these applications, see previous studies (Langfeldt et al., 1997; Phatarfod and Sudbury, 1994; Xie et al., 2001). The common simplified assumption is that a single blocking element is able to mask the presence of any number of positive elements in the tested pools. In reality, however, more complex situations may arise, where the blocking action is determined by the combined action of certain samples. Moreover, the blocking action of these samples can be contrasted by the additive effect of two or more positive samples. In the present article, we propose a generalization of the GTI model that better accommodates this kind of applications.

In our generalization of the GTI model, the interaction between defective items and inhibitors is parameterized by two thresholds $g \geq 1$ and $h \geq 1$. In this parameterized version of GTI, at least h inhibitors are needed to neutralize the defective items in the tested subset. Moreover, if the number of defective items in a given subset is at least g, then the response of the test is YES no matter how many inhibitors are in the subset. Of particular note is that we assume that the two thresholds g and h, as well as the number $|\mathcal{P}|$ of defective items, be unknown. We will refer to this search model as *threshold*-GTI. This model is based on the assumption that in the reality a single inhibitory element might have no effect on the response of the tests. It is reasonable to assume that the interference of inhibitory elements might be more or less effective depending on the number of spoiled samples related to the number of positive samples contained in the tested pool. Therefore, we assume that a test is able to detect the eventual presence of positive elements in the pool if and only if the number of inhibitors in the tested pool is below a certain threshold. Moreover, no matter how many inhibitors are in the tested subset, such an interference might be completely neutralized by a sufficiently large number of positive elements. Our assumption that the two thresholds g and h be unknown is consistent with the fact that in practice very little might be known on how powerful inhibitors are and how their action is contrasted by positive samples.

In the literature, other parameterized versions of classical group testing have been considered. The authors (Chen and Fu, 2009; Damaschke, 2005) studied a generalization of group testing in which there are two fixed thresholds ℓ and u, with $\ell < u$, and the response to a test is positive if the tested subset contains at least u defective samples, negative if it contains at most ℓ defective samples, and the response is arbitrarily given otherwise. De Bonis and Vaccaro (2003) introduced a parameterized version of the GTI problem. In that model, a test is positive if and only if it contains at least one defective item and less than d inhibitors. This model coincides with ours in the case when $|\mathcal{P}| < g$ and $d = h$. It is remarkable that the algorithm given in De Bonis and Vaccaro (2003) works under the assumption of the threshold d being known, and consequently it does not work for our model.

2. DESCRIPTION OF THE MODEL AND SUMMARY OF RESULTS

Recall that our group testing scenario consists of a set $\mathcal{S}=\{1, \ldots, n\}$ that contains an unknown subset $\mathcal{P}$ of defective (positive) items and an unknown subset $\mathcal{I}$ of inhibitors. A test on a subset $Q \subseteq \mathcal{S}$ is

$$\text{positive,} \quad \text{if} \begin{cases} |Q \cap \mathcal{P}| \geq g, \\ |Q \cap \mathcal{P}| \geq 1 \text{ and } |Q \cap \mathcal{I}| < h; \end{cases}$$

$$\text{negative,} \quad \text{if} \begin{cases} |Q \cap \mathcal{P}| = 0, \\ 1 \leq |Q \cap \mathcal{P}| < g \text{ and } |Q \cap \mathcal{I}| \geq h. \end{cases}$$

We assume that the two thresholds g and h are *unknown* positive integers and that no information on the size of $\mathcal{P}$ is given, whereas it is known that $|\mathcal{I}| \leq r$ where r is a positive integer.

The remainder of this article is organized as follows. In the next section, we introduce some definitions and auxiliary results. In Section 4 we first show that, if no upper bound on the number of inhibitors is given and the number of defective items $|\mathcal{P}|$ is less than g, then in the worst case n tests are necessary to establish whether the set of the defective items is empty or not. Hence, in this case, the best search strategy would consist in individually testing all items in $\mathcal{S}$. Under the hypothesis of r being known, we derive a lower bound on the number of tests required to determine the set $\mathcal{P}$ when $|\mathcal{P}| < g$ and $|\mathcal{I}| \geq h$. This lower bound holds even in the case when the exact number of inhibitors $|\mathcal{I}|$ and the exact values of the two thresholds g and h are given. If the two thresholds are not given then we show that the minimum number of tests required to determine whether $\mathcal{P}=\emptyset$ is asymptotically at least as large as that required under the GTI model. In Section 5, we provide an algorithm for the threshold-GTI problem that proceeds along two different paths according to whether the given set $\mathcal{S}$ tests positive or negative. In Section 6, we analyze the complexity of our algorithm for the threshold-GTI problem and summarize our main results. Notice that $\mathcal{S}$ tests positive both in the cases when $|\mathcal{P}| \geq g$ and when $|\mathcal{P}| \geq 1$ and $|\mathcal{I}| < h$, whereas $\mathcal{S}$ tests negative when $\mathcal{P}=\emptyset$ or when $|\mathcal{P}| < g$ and $|\mathcal{I}| \geq h$. We show that, if $|\mathcal{P}| \geq g > 2$ and $|\mathcal{I}| \geq h$, our algorithm attains the information theoretic lower bound of $\Omega(\log \binom{n}{|\mathcal{P}|})$, and therefore is asymptotically optimal. In this case that $|\mathcal{P}| \geq g > 2$ and $|\mathcal{I}| \geq h$, optimality is achieved even if no upper bound r on the number of inhibitors is known. In the remaining cases for which $\mathcal{S}$ tests positive, the cost of our algorithm differs from the information theoretic lower bound by an $O(\log r \log n)$ additive term. However, in these cases, if we assume that the exact number $|\mathcal{I}|$ of inhibitors is given, then it is possible to modify the algorithm so as to attain the information theoretic lower bound. As far as it concerns the case when $\mathcal{S}$ tests negative, the cost of our algorithm exceeds the lower bound of Section 4 by an $O(\log r)$ factor.

3. SOME DEFINITIONS AND AUXILIARY RESULTS

To derive our results, we exploit the well known correspondence between binary codes and group testing strategies. Denote the items of $\mathcal{S}$ with the integers from 1 through n. A binary code $\mathcal{C}$ can be represented by an $N \times n$ binary matrix $\mathcal{M}=\|c_j(i)\|$, $i=1, \ldots, N$ and $j=1, \ldots, n$, with codewords as columns. Let $R_1, \ldots, R_N$ denote the rows of $\mathcal{M}$. For each $j \in \{1, \ldots, n\}$, we associate the j-th column c_j of $\mathcal{M}$ with item j and, for each $i \in 1, \ldots, N$, define the subset T_{R_i} associated with row R_i of $\mathcal{M}$ as $T_{R_i}=\{j \in \{1, \ldots, n\} : c_j(i)=1\}$. The group testing algorithm that tests the sets $T_{R_1}, \ldots, T_{R_N}$ is the group testing algorithm defined by $\mathcal{M}$, or equivalently by $\mathcal{C}$.

Superimposed codes play a crucial role in the design of group testing strategies. In the following we introduce some definitions and preliminaries concerning superimposed codes. Given $m > 1$ codewords (columns) $c_{\ell_1}, \ldots, c_{\ell_m}$, we denote with $(c_{\ell_1} \vee \ldots \vee c_{\ell_m})$ the boolean sum (OR) of $c_{\ell_1}, \ldots, c_{\ell_m}$. A column c_h is said to be *covered* by a column c_j if for any index $i \in \{1, \ldots, N\}$ one has that $c_h(i)=1$ implies $c_j(i)=1$.

Definition 1 *(De Bonis and Vaccaro, 2003). Let p, q and d be positive integers with $d \leq q$. A binary code $\mathcal{C}=\{c_1, \ldots, c_n\}$, with $n \geq p+q$, is called (p, q, d)-superimposed if for any distinct $p+q$ indices $h_1, \ldots, h_p, \ell_1, \ldots, \ell_q$ there exist $q-d+1$* distinct *indices $j_1, \ldots, j_{q-d+1} \in \{\ell_1, \ldots, \ell_q\}$ such that $(c_{h_1} \vee \ldots \vee c_{h_p})$ is not covered by $(c_{j_1} \vee \ldots \vee c_{j_{q-d+1}})$. The minimal length of a (p, q, d)-superimposed code of size n is denoted by $N(p, q, d, n)$.*

For $d = 1$ the codes in the above definition have the property that the union of any p columns is not covered by the union of any other q columns. Codes satisfying this property have been introduced in (D'yachkov and Rykov, 1983) and are known under the name of (p, q)-superimposed codes. The minimal length of a (p, q)-superimposed code of size n will be denoted by $N(p, q, n)$.

We will resort to the following upper bound on the minimal length of (1, q, d)-superimposed codes to estimate the cost of our algorithm.

Theorem 3.1 *(De Bonis and Vaccaro, 2003). Let n, q and d be positive integers with $d \leq q \leq n$. Then $N(1, q, d, n) < 24(\frac{q}{d})^2(\log_2 n + 2)$.*

De Bonis and Vaccaro (2006) proved an upper bound on the maximum size of (1, q, d)-superimposed code in terms of the maximum size of (1, $\lfloor q/d \rfloor$)-superimposed codes of the same length. The following lower bound on the minimum length of (1, q, d)-superimposed code is an immediate consequence of that result together with the well known $\Omega(\frac{m^2}{\log m}\log n)$ lower bound on the minimum length $N(1, m, n)$ of (1, m)-superimposed codes of size n (D'yachkov and Rykov, 1982; Füredi, 1996; Ruszinkó, 1994).

Theorem 3.2 *(De Bonis and Vaccaro, 2006). Let n, q and d be positive integers with $d \leq q \leq n$. Then $N(1, q, d, n) = \Omega(\frac{q^2}{d^2 \log q}\log n)$.*

Theorem 3.2 will be exploited in the next section to derive a lower bound on the number of tests required to determine the defective items when it holds that $|\mathcal{P}| < g$ and $|\mathcal{I}| \geq h$.

4. SOME NEGATIVE RESULTS

Solving the threshold-GTI problem is significantly harder when the given set $\mathcal{S}$ tests negative. In this case, it might be either that $\mathcal{P} = \emptyset$ or that $0 < |\mathcal{P}| < g$ and the number $|\mathcal{I}|$ of inhibitors is at least h. We show that if $\mathcal{S}$ tests negative and no upper bound r on $|\mathcal{I}|$ is given, then at least n tests are necessary in the worst case to determine whether the set $\mathcal{P}$ of the defective items is empty or not. To this aim we prove the following lemma.

Lemma 4.1. *Suppose that $\mathcal{S}$ tests negative and that the unknown set $\mathcal{I}$ of inhibitors has size at most r. Then, in the worst case, $g + r - 1$ tests are necessary to establish whether $\mathcal{P} = \emptyset$ or not.*

Proof. Given an arbitrary algorithm $\mathcal{A}$, let us associate to each test $Q \subseteq \mathcal{S}$ performed by $\mathcal{A}$ the equation $\sum_{i \in Q} x_i = 0$. If $\mathcal{A}$ performs less than $r + g - 1$ tests, then the number of equations in the above system is less than $r + g - 1$ and the system admits a non trivial solution in which at most $r + g - 1$ variables x_i have non-zero value. Let $V^+ = \{i \in \mathcal{S} : x_i > 0\}$ and $V^- = \{i \in \mathcal{S} : x_i < 0\}$. For each test Q performed by $\mathcal{A}$ it must hold either that $|Q \cap (V^+ \cup V^-)| = 0$ or that $|Q \cap V^+| \geq 1$ and $|Q \cap V^-| \geq 1$. Since g and h are unknown, a malicious adversary might set $|\mathcal{P}| = |V^+| \leq g - 1$, $|\mathcal{I}| = |V^-| \leq r$ and $h = min_Q\{|Q \cap V^-| : Q \cap V^- \neq \emptyset\}$ so that all responses are negative. As a consequence, the algorithm would not distinguish between the cases $\mathcal{P} = \emptyset$ and $\mathcal{P} \neq \emptyset$. ■

The following corollary is a consequence of Lemma 4.1.

Corollary 4.2. *Suppose that $\mathcal{S}$ tests negative and that no upper bound on the cardinality of the unknown set $\mathcal{I}$ of inhibitors is given. Then, in the worst case, at least n tests are needed to establish whether $\mathcal{P} = \emptyset$ or not.*

Corollary 4.2 *implies that if $\mathcal{S}$ tests negative and we are not given an upper bound on the number of inhibitors then the best strategy consists in individually testing all items in $\mathcal{S}$. Therefore, in this paper we assume known an upper bound r on the number of inhibitors. Under this assumption, we derive the following lower bound on the number of tests required to determine all defective items in $\mathcal{S}$ in the case when $|\mathcal{P}| < g$ and $|\mathcal{I}| \geq h$.*

Theorem 4.3. *Suppose that $|\mathcal{P}| < g$ and $h \le |\mathcal{I}| \le r$, with r being a known positive integer. Any algorithm that identifies all defective items in $\mathcal{S}$ uses at least $N(1, r, h, n)$ tests.*

Proof. Let $\mathcal{A}$ be any algorithm that solves the threshold-GTI problem under the hypothesis that $\mathcal{S}$ contains at most $g-1$ defective items. Let $T_1, \ldots, T_m$ be the sets tested by the algorithm and let $\mathcal{M} = \|c_j(i)\|$ be the $m \times n$ matrix whose rows are the characteristic vectors of $T_1, \ldots, T_m$, that is $c_j(i) = 1$ if and only if $j \in T_i$. In other words $\mathcal{M}$ represents the binary code associated with the group testing strategy that tests $T_1, \ldots, T_m$.

We will show that in order for the algorithm to identify the set of defective items by m tests, $\mathcal{M}$ must be a $(1, |\mathcal{I}|, h)$-superimposed code. Suppose to the contrary that there exist $|\mathcal{I}|+1$ columns $c_{\ell_1}, c_{\ell_2}, \ldots, c_{\ell_{|\mathcal{I}|+1}}$ such that c_{ℓ_1} is covered by the boolean sum $c_{k_1} \vee \ldots \vee c_{k_{|\mathcal{I}|-h+1}}$, for any choice of $|\mathcal{I}|-h+1$ columns $c_{k_1}, \ldots, c_{k_{|\mathcal{I}|-h+1}}$ in $\{c_{\ell_2}, \ldots, c_{\ell_{|\mathcal{I}|+1}}\}$. It follows that for any row index $i \in [m]$ such that $c_{\ell_1}(i) = 1$, one has that at least h columns among $c_{\ell_2}, \ldots, c_{\ell_{|\mathcal{I}|+1}}$ have the i-th entry equal to 1. As a consequence, ℓ_1 occurs in some subset T only if at least h of $\ell_2, \ldots, \ell_{|\mathcal{I}|+1}$ belongs to T. A malicious adversary could make ℓ_1 the unique defective item and $\ell_2, \ldots, \ell_{|\mathcal{I}|+1}$ the inhibitory items so that the responses to the m tests are negative. Hence, the algorithm cannot distinguish between the case when $\mathcal{S}$ contains a defective item, namely item ℓ_1, and the case when $\mathcal{S}$ does not contain any defective item. Since $|\mathcal{I}|$ can be as large as r, it follows that m tests suffices to determine the defective items only if $\mathcal{M}$ form a $(1, r, h)$-superimposed code and consequently $m \ge N(1, r, h, n)$. ∎

Theorems 3.2 and 4.3, along with the information theoretic lower bound, imply the following lower bound.

Corollary 4.4. *Suppose that $|\mathcal{P}| < g$ and $h \le |\mathcal{I}| \le r$, with r being a known positive integer. In the worst case any algorithm that identifies all defective items in $\mathcal{S}$ uses at least $N(1, r, h, n) + \Omega(|\mathcal{P}| \log \frac{n}{|\mathcal{P}|}) = \Omega(\frac{r^2}{h^2 \log r} \log n + |\mathcal{P}| \log \frac{n}{|\mathcal{P}|})$ tests.*

We remark that the above bounds hold even if the exact values of the two thresholds g and h are given and it is known that $\mathcal{S}$ contains exactly r inhibitors. If the two thresholds are not given then the following theorem states that $N(1, r, n)$ tests are needed to discover that $\mathcal{P} = \emptyset$. This is the same number of tests required to determine whether $\mathcal{P} = \emptyset$ in the GTI model (De Bonis, 2008).

Theorem 4.5. *Suppose that $\mathcal{S}$ tests negative and that the unknown set $\mathcal{I}$ of inhibitors has cardinality $|\mathcal{I}| \le r$, with r being a known positive integer. Then, in the worst case, at least $N(1, r, n) = \Omega((\frac{r^2}{\log r} \log n)$ tests are needed to establish whether $\mathcal{P} = \emptyset$.*

Proof. Observe that any algorithm that establishes whether $\mathcal{P} = \emptyset$, for some *fixed* unknown value of the threshold h, is able to establish whether $\mathcal{P} = \emptyset$ for any other value of h. Indeed, negative tests provide no information on the value of h; consequently, a malicious adversary might change the value of h at any stage of the algorithm, even at the end. The lower bound in the statement of the theorem follows from setting $h = 1$ in the lower bound of Theorem 4.3. ∎

5. AN ALMOST OPTIMAL ALGORITHM

The algorithm proceeds along two different paths, namely algorithm A or algorithm B, according to whether the given set $\mathcal{S}$ tests positive or negative, respectively. Recall that $\mathcal{S}$ tests positive both in the cases when $|\mathcal{P}| \ge g$ and when $\mathcal{P}$ is non-empty and $|\mathcal{I}| < h$, whereas $\mathcal{S}$ tests negative if $\mathcal{P} = \emptyset$ or if $|\mathcal{P}| < g$ and $|\mathcal{I}| \ge h$.

Before describing our algorithm, we introduce a procedure that plays a very important role in algorithm A. Given a set $A \subset \mathcal{S}$, we will denote by $Half_0(A)$ and $Half_1(A)$ the sets obtained by partitioning A into two sets of size $\lfloor |A|/2 \rfloor$ and $\lceil |A|/2 \rceil$, respectively. Let V and V' be two disjoint subsets of $\mathcal{S}$ such that V' tests negative, whereas $V \cup V'$ tests positive. The following procedure determines a defective item of V by performing binary search in such a way to cope with the possible presence of h or more inhibitors in V.

Algorithm *MBS(V, V′)*

1. $X \leftarrow V \cup V', A \leftarrow V$
2. **while** $|A| > 1$
3. **do if** $X \setminus Half_0(A)$ tests positive
4. **then** $X \leftarrow X \setminus Half_0(A), A \leftarrow Half_1(A)$
5. **else** $A \leftarrow Half_0(A)$
6. **return** A

Theorem 5.1. *If V and V' are two disjoint subsets of $\mathcal{S}$ such that V' tests negative and $V \cup V'$ tests positive, then the singleton $A \subset V$ returned by MBS(V, V') consists of a defective item.*

Proof. In order to show that the singleton $A \subseteq V$ returned at line 6 consists of a defective item, we will prove that the algorithm preserves the invariant that X tests positive and $X \setminus A$ tests negative, thus implying that A contains at least one item of $V \cap \mathcal{P}$ through the entire execution of the algorithm.

For $i = 0, \ldots, \lceil \log |A| \rceil$, let X^i and A^i denote the sets X and A, respectively, as they appear at the beginning of the i-th iteration of the *while* loop. To prove the theorem, it suffices to show that X^i tests positive and $X^i \setminus A^i$ tests negative for all i, which implies that A^i contains at least one defective item in $V \cap \mathcal{P}$. We show by induction on i that X^i tests positive and $X^i \setminus A^i$ tests negative. It is immediate to verify that $X^0 = V \cup V'$ and $A^0 = V$ satisfy the above condition because of the hypothesis of the theorem. Let us assume by induction hypothesis that those relations hold for X^i and A^i. If the test on $X^i \setminus Half_0(A^i)$ yields a YES response then $X^{i+1} = X^i \setminus Half_0(A^i)$ and $A^{i+1} = Half_1(A^i)$. In this case, the subset X^{i+1} obviously tests positive. To see that $X^{i+1} \setminus A^{i+1}$ tests negative, observe that $X^{i+1} \setminus A^{i+1} = X^i \setminus A^i$ and that, by induction hypothesis, $X^i \setminus A^i$ tests negative. If the test on $X^i \setminus Half_0(A^i)$ yields a NO response, then in this case we have $X^{i+1} = X^i$ and $A^{i+1} = Half_0(A^i)$. By induction hypothesis, $X^{i+1} = X^i$ tests positive. Since $X^{i+1} \setminus A^{i+1} = X^i \setminus Half_0(A^i)$, it is obvious that $X^{i+1} \setminus A^{i+1}$ tests negative. ■

If we have two subsets V and V' that satisfy the hypothesis of Theorem 5.1, then *MBS*(V, V') might be repeatedly applied to search for several defective items of V. To this aim, *MBS*(V, V) is iteratively called to search for a defective item in V and such a defective item is removed from V and eventually added to V' to preserve the condition that $V \cup V'$ should test positive. This process continues as long as the condition that $V \cup V'$ should test positive can be preserved without violating the condition that V' should test negative. The following procedure exploits the above idea.

Algorithm *RMBS(V, V′)*

1. $B \leftarrow \emptyset$
2. **while** $V' \cup V$ tests positive
3. **do** $A \leftarrow MBS(V, V'), B \leftarrow B \cup A, V \leftarrow V \setminus A$
4. **if** $V' \cup A$ tests negative
5. **then** $V' \leftarrow V' \cup A$
6. **return** B

Theorem 5.2. *Let V and V' be two disjoint subsets of $\mathcal{S}$ such that V' tests negative and $V \cup V'$ tests positive, and let B be the subset of V returned by RMBS(V, V'). If it holds that $g > 1, |V' \cap \mathcal{I}| < h$ and $|(V \cup V') \cap \mathcal{I}| \geq h$, then B consists of all but $g - 1$ defective items of V. In the remaining cases, B is the set of all defective items of V.*

Proof. By hypothesis, $V \cup V'$ tests positive and V' tests negative. As a consequence, algorithm *RMBS*(V, V') enters the *while* loop and it is possible to see that, at each iteration of the *while* loop, V and V' satisfy the hypothesis of Theorem 5.1. Indeed, $V \cup V'$ tests positive because of the condition of *while* loop at line 2, whereas V' tests negative because of the condition of the *if* statement at line 4. Therefore, at each iteration the item in the singleton A returned by the call to *MBS*(V, V') at line 3 is a defective item of V. This item is removed from V and eventually added to V' as long as it preserves the condition that V' tests negative.

Let us consider the case when $g > 1, |V' \cap \mathcal{I}| < h$ and $|(V \cup V') \cap \mathcal{I}| \geq h$. If $|V' \cap \mathcal{I}| < h$, then V' tests negative if and only if $V' \cap \mathcal{P} = \emptyset$, and consequently no defective item of V is ever added to V' at line 5. In this case $(V \cup V') \cap \mathcal{P} = V \cap \mathcal{P}$ through all iterations of the *while* loop. Therefore, if in addition we assume that $g > 1$ and $|(V \cup V') \cap \mathcal{I}| \geq h$, then the *while* loop terminates as soon as $|V \cap \mathcal{P}| < g$, that is, as soon as all but $g - 1$ defective items of V have been detected and added to B.

Now let us prove that in the remaining cases *RMBS*(V, V') determines all defective items of V. If $g = 1$ or $|(V \cup V') \cap \mathcal{I}| < h$ then one has that $V' \cap \mathcal{P} = \emptyset$ through all iterations of the *while* loop and that $V \cup V'$ tests positive if and only if $V \cap \mathcal{P} \neq \emptyset$. Consequently, the *while* loop terminates after all defective items of V have been detected and added to B. If $|V' \cap \mathcal{I}| \geq h$ then V', as well as $V \cup V'$, tests positive if and only if it contains at least g defective items. Therefore, the algorithm stops adding defective items to V' at line 5 as soon as V' contains exactly $g - 1$ defective items. From that point on, $V \cup V'$ tests positive if and only if V contains at least one defective item; therefore the *while* loop terminates when $V \cap \mathcal{P} = \emptyset$, that is, after all defective items of V have been detected and collected into B. ■

Now we are ready to describe our algorithm. As said at the beginning of this section, our algorithm proceeds along two different paths, namely algorithm A or algorithm B, based on the result of a preliminary test on $\mathcal{S}$. If the response of this test is YES then our algorithm runs algorithm A, otherwise it runs algorithm B. We present algorithm A and algorithm B in the following two separate sections. In order to keep the description of algorithm A as clear and short as possible, we first illustrate its behavior and then prove its correctness. The correctness of algorithm B follows directly from its description.

5.1. Algorithm A

Step 1. This step makes a call to *RMBS*(V, V') with $V = \mathcal{S}$ and $V' = \emptyset$. If $|\mathcal{I}| < h$ or $g = 1$, then *RMBS*$(\mathcal{S}, \emptyset)$ finds all defective items in $\mathcal{S}$. If $g > 1$ and $|\mathcal{I}| \geq h$, then *RMBS*$(\mathcal{S}, \emptyset)$ finds exactly $|\mathcal{P}| - g + 1$ defective items.

Let $\mathcal{S}^*$ be the subset consisting of the items of $\mathcal{S}$ that have not been classified as defective yet. Notice that $\mathcal{S}^*$ tests negative since it either contains no defective item or it contains exactly $g - 1$ defective items and at least h inhibitors.

Step 2. The goal of this step is to find a partition $(T, \mathcal{S}^* \setminus T)$ of $\mathcal{S}^*$ such that at least one of the subsets T and $\mathcal{S}^* \setminus T$ can be further searched either by exploiting algorithm *RMBS*(V, V') or by a competitive strategy for classical group testing. We recall that a group testing algorithm is said competitive if it has no knowledge on the number d of defective items and is asymptotically as efficient as the best procedures that work under the hypothesis that d is known in advance. See Du and Hwang (1992, 2000) for an account on the design of these procedures. If it is possible to determine the defective items in both subsets T and $\mathcal{S}^* \setminus T$ then the algorithm terminates, otherwise it goes to Step 3. If such a partition $(T, \mathcal{S}^* \setminus T)$ does not exist then the algorithm infers that either $g = 1$ or $|\mathcal{I}| < h$, and consequently all defective items have been determined in the previous step.

Step 2 makes two attempts to find the above said partition. The first attempt succeeds if $g > 2$ and $|\mathcal{I}| \geq h$, whereas the second one succeeds if $g = 2$ and $|\mathcal{I}| \geq h$. Both attempts use a testing strategy based on superimposed codes. In the following we describe these two attempts and illustrate, for each of them, how the algorithm behaves in case of success.

Attempt 1: Let $\mathcal{M}$ denote a (1, 1)-superimposed code of size $|\mathcal{S}^*| \leq n$, and let z be one of the defective items determined in Step 1. For each row R of $\mathcal{M}$, the algorithm tests both T_R and its complement $\mathcal{S}^* \setminus T_R$ and if both subsets test negative then it performs an additional pair of tests on the subsets $T_R \cup \{z\}$ and $(\mathcal{S}^* \setminus T_R) \cup \{z\}$. This testing strategy continues until it finds a row R such that one of the following two cases occurs: (i) at least one of T_R and $\mathcal{S}^* \setminus T_R$ tests positive; (ii) T_R and $\mathcal{S}^* \setminus T_R$, as well as $T_R \cup \{z\}$ and $(\mathcal{S}^* \setminus T_R) \cup \{z\}$, test negative.

If case (i) occurs then the algorithm determines the defective items in one or both of the subsets T_R and $\mathcal{S}^* \setminus T_R$ by a competitive strategy for classical group testing. Indeed, a subset of $\mathcal{S}^*$ is positive only if it contains less than h inhibitors. If both T_R and $\mathcal{S}^* \setminus T_R$ are positive, then the algorithm terminates. If one of the subsets T_R and $\mathcal{S}^* \setminus T_R$ tests negative then the algorithm goes to Step 3 and search for the defective items in this subset. Notice that this subset contains at most $g - 2$ defective items.

If case (ii) occurs then the algorithm invokes algorithm *RMBS*(V, V') once with $V = T_R$ and $V' = (\mathcal{S}^* \setminus T_R) \cup \{z\}$, and once with $V = \mathcal{S}^* \setminus T_R$ and $V' = T_R \cup \{z\}$.

Attempt 2: Let $\hat{\mathcal{M}}_i$, $i = 1, 2 \ldots$, denote a $(1, 2^{i+1}, 2^{i-1})$-superimposed code of size $|\mathcal{S}^*| \leq n$. For each row $\tilde{\boldsymbol{R}}$ of codes $\hat{\mathcal{M}}_1, \hat{\mathcal{M}}_2, \ldots$, the algorithm tests both $T_{\tilde{\boldsymbol{R}}}$ and $\mathcal{S}^* \setminus T_{\tilde{\boldsymbol{R}}}$ and if both of them test negative then it performs an additional pair of tests on the subsets $T_{\tilde{\boldsymbol{R}}} \cup \{z\}$ and $(\mathcal{S}^* \setminus T_{\tilde{\boldsymbol{R}}}) \cup \{z\}$. This testing strategy terminates as soon as it discovers a row $\tilde{\boldsymbol{R}}$ such that one of the two following cases occurs: (i′) exactly one of $T_{\tilde{\boldsymbol{R}}}$ and $\mathcal{S}^* \setminus T_{\tilde{\boldsymbol{R}}}$ tests positive; (ii′) both $T_{\tilde{\boldsymbol{R}}}$ and $\mathcal{S}^* \setminus T_{\tilde{\boldsymbol{R}}}$ test negative and exactly one of $T_{\tilde{\boldsymbol{R}}} \cup \{z\}$ and $(\mathcal{S}^* \setminus T_{\tilde{\boldsymbol{R}}}) \cup \{z\}$ tests positive. Notice that we are assuming that $g \leq 2$ and $|\mathcal{I}| \geq h$, and consequently case (i′) or case (ii′) occurs only if $g = 2$ and $|\mathcal{I}| \geq h$.

If case (i′) occurs, then the defective item is obviously contained in the one of the two subsets $T_{\tilde{\boldsymbol{R}}}$ and $\mathcal{S}^* \setminus T_{\tilde{\boldsymbol{R}}}$ that tests positive and can be found by binary search.

If case (ii′) occurs then the algorithm invokes *MBS*(*V, V′*) either with $V = T_{\tilde{\boldsymbol{R}}}$ and $V' = (\mathcal{S}^* \setminus T_{\tilde{\boldsymbol{R}}}) \cup \{z\}$, or with $V = \mathcal{S}^* \setminus T_{\tilde{\boldsymbol{R}}}$ and $V' = T_{\tilde{\boldsymbol{R}}} \cup \{z\}$, according to which one between $T_{\tilde{\boldsymbol{R}}} \cup \{z\}$ and $(\mathcal{S}^* \setminus T_{\tilde{\boldsymbol{R}}}) \cup \{z\}$ tests positive, respectively.

Step 3. The algorithm performs this step if and only if case (i) occurs in Step 2 with exactly one of T and $\mathcal{S}^* \setminus T$ being positive. Let $\mathcal{G}$ denote the one of T and $\mathcal{S}^* \setminus T$ that tests negative. Recall that this subset contains at most $g - 2$ defective items. Step 3 searches for the defective items eventually contained in $\mathcal{G}$ by the following search algorithm *MGT*($\mathcal{G}$) that, at each iteration of the *while* loop, partitions the search space into two halves and determines the defective items in at least one of those halves by a competitive algorithm for classical group testing, here denoted by *CGT*. If there are undetected defective items left after exiting the *while* loop then these will be discovered by exploiting algorithm *RMBS*(*V, V′*). In the following, $\mathcal{R}$ denotes the subset of $\mathcal{P}$ consisting of all defective items of $\mathcal{S}^*$ determined by Step 2. Again, we denote by z one of the defective items determined in Step 1.

Algorithm *MGT*($\mathcal{G}$)

```
1.  if G ∪ {z} tests positive then return ∅
2.  X ← G, C ← ∅
3.  while X ≠ ∅ and Half_i(X) ∪ {z} tests positive for some i ∈ {0, 1}
4.    do for i ← 0, 1
5.          do if Half_i(X) ∪ {z} tests positive
6.               then C ← C ∪ CGT(Half_i(X)), X ← X \ Half_i(X)
7.  for i ← 0, 1
8.       do if Half_i(X) ∪ R ∪ C ∪ {z} tests negative
9.          then C ← C ∪ RMBS(Half_{1−i}(X), Half_i(X) ∪ R ∪ C ∪ {z})
10. return C
```

5.1.1. Correctness of Algorithm A.

Step 1. The correctness of Step 1 immediately follows from Theorem 5.2.

Step 2. We show that if $g \geq 2$ and $|\mathcal{I}| \geq h$ then either Attempt 1 or Attempt 2 succeeds. Therefore, if none of Attempt 1 and Attempt 2 succeeds then it holds that either $g = 1$ or $|\mathcal{I}| < h$ and the algorithm infers that all defective items have been detected by Step 1. Moreover, if $g \geq 2$ and $|\mathcal{I}| \geq h$ then the algorithm determines all defective items of $\mathcal{S}^*$ with the exception of those determined in Step 3.

First we show that if $g > 2$ and $|\mathcal{I}| \geq h$ then Attempt 1 succeeds. Indeed, by definition of (1, 1)-superimposed codes, one has that for any choice of two defective items $x, y \in \mathcal{S}^* \cap \mathcal{P}$, there is a row R in $\mathcal{M}$ such that exactly one of the two entries associated with x and y is equal to 1. Therefore, if $g > 2$ and $|\mathcal{I}| \geq h$, there exists a row R of $\mathcal{M}$ such that both T_R and $\mathcal{S}^* \setminus T_R$ contain a number of defective items between 1 and $g - 2$. If at least one of T_R and $\mathcal{S}^* \setminus T_R$ contains less than h inhibitors then case (i) occurs. If both T_R and $\mathcal{S}^* \setminus T_R$ contain at least h inhibitors then case (ii) occurs.

Notice that if $g = 2$ and $|\mathcal{I}| \geq h$, there might still be a row R of $\mathcal{M}$ such that one of the two subsets T_R and $\mathcal{S}^* \setminus T_R$ tests positive but we are not guaranteed that such a row exists. Accordingly, if Attempt 1 fails to find a partition of $\mathcal{S}^*$ such that either case (i) or case (ii) occurs, then the algorithm infers that either $g \leq 2$ or $|\mathcal{I}| < h$ and performs Attempt 2.

Now we show that if $g = 2$ and $|\mathcal{I}| \geq h$ both hold then Attempt 2 succeeds. To this aim we show that the algorithm is guaranteed to find a row $\tilde{\boldsymbol{R}}$ such that one of cases (i′) and (ii′) occurs by the time it has examined the pairs of subsets corresponding to the rows of $\hat{\mathcal{M}}_1, \ldots, \hat{\mathcal{M}}_{\lfloor \log |\mathcal{I}| \rfloor}$. Observe that $\hat{\mathcal{M}}_{\lfloor \log |\mathcal{I}| \rfloor}$ is a

$(1, |\mathcal{I}|, \lfloor |\mathcal{I}|/2 \rfloor)$-superimposed code and by Definition 1, for any choice of $|\mathcal{I}|+1$ columns $c_1, \ldots, c_{|\mathcal{I}|+1}$, there exists a row $\tilde{R}$ such that $\tilde{R}$ has a 1 in correspondence of column c_1 and at most $\lfloor |\mathcal{I}|/2 \rfloor - 1$ entries equal to 1 among those corresponding to columns $c_2, \ldots, c_{|\mathcal{I}|+1}$. It follows that the subset $T_{\tilde{R}}$ contains the unique defective item of $\mathcal{S}^*$ and less than $|\mathcal{I}|/2$ inhibitors, whereas $S^* \setminus T_{\tilde{R}}$ contains no defective item and more than $|\mathcal{I}|/2$ inhibitors. If $h \geq |\mathcal{I}|/2$ then $T_{\tilde{R}}$ contains less than h inhibitors and consequently tests positive so that case (i′) occurs. If $h < |\mathcal{I}|/2$ then $\mathcal{S}^* \setminus T_{\tilde{R}}$ contains at least h inhibitors and consequently one has either that $T_{\tilde{R}}$ tests positive, or that both $T_{\tilde{R}}$ and $\mathcal{S}^* \setminus T_{\tilde{R}}$ test negative. In the former case one has that case (i′) occurs, whereas in the latter case one has that case (ii′) occurs since $T_{\tilde{R}} \cup \{z\}$ tests positive and $(\mathcal{S}^* \setminus T_{\tilde{R}}) \cup \{z\}$ tests negative. Indeed, it holds that $g = 2$ and that $T_{\tilde{R}} \cup \{z\}$ contains two defective items, whereas $(\mathcal{S}^* \setminus T_{\tilde{R}}) \cup \{z\}$ contains one defective item and at least h inhibitors.

To conclude the proof of the correctness of Step 2, we need to show that if Attempt 1 succeeds then the algorithm determines all defective items of $\mathcal{S}^*$ with the exception of those determined in Step 3, whereas if Attempt 2 succeeds then the algorithm determines the unique defective item contained in $\mathcal{S}^*$. This is quite obvious if case (i) or case (i′) occurs. Indeed, in both cases the search space is restricted to a subset containing less than h inhibitors that is searched by exploiting a competitive algorithm for classical group testing in case (i), and by binary search in case (i′). We have to show that if case (ii) of Attempt 1 occurs then the two calls to algorithm *RMBS*(V, V') determine all defective items in $\mathcal{S}^*$, whereas if case (ii′) of Attempt 2 occurs then the call to *MBS*(V, V') determines the unique defective item of $\mathcal{S}^*$.

If case (ii) of Attempt 1 occurs, then the algorithm invokes *RMBS*(V, V') once with $V = T_R$ and $V' = (\mathcal{S}^* \setminus T_R) \cup \{z\}$, and once with $V = \mathcal{S}^* \setminus T_R$ and $V' = T_R \cup \{z\}$. Notice that both pairs of subsets V and V' satisfy the hypothesis of Theorem 5.2, in view of the fact that V' test negative, whereas the union $V \cup V' = \mathcal{S}^* \cup \{z\}$ contains exactly g defective items and consequently tests positive. Moreover, both $T_R \cup \{z\}$ and $(\mathcal{S}^* \setminus T_R) \cup \{z\}$ test negative and therefore contain at least h inhibitors. Hence, Theorem 5.2 implies that the calls to *RMBS*$(T_R, (\mathcal{S}^* \setminus T_R) \cup \{z\})$ and to *RMBS*$(\mathcal{S}^* \setminus T_R, T_R \cup \{z\})$ determine all defective items in T_R and $\mathcal{S}^* \setminus T_R$, respectively.

If case (ii′) of Attempt 2 occurs then the algorithm makes a call to *MBS*(V, V') either with $V = T_{\tilde{R}}$ and $V' = (\mathcal{S}^* \setminus T_{\tilde{R}}) \cup \{z\}$, or with $V = \mathcal{S}^* \setminus T_{\tilde{R}}$ and $V' = T_{\tilde{R}} \cup \{z\}$, according to which one between $T_{\tilde{R}} \cup \{z\}$ and $(\mathcal{S}^* \setminus T_{\tilde{R}}) \cup \{z\}$ tests positive. Indeed, the defective item of $\mathcal{S}^*$ is contained in the one of $T_{\tilde{R}} \cup \{z\}$ and $(\mathcal{S}^* \setminus T_{\tilde{R}}) \cup \{z\}$ that tests positive, in view of the fact that $g = 2$ and consequently the one of the two sets that tests negative contains no defective item in addition to z. The sets V and V' satisfy the hypothesis of Theorem 5.1, since V' tests negative and $V \cup V' = \mathcal{S}^* \cup \{z\}$ contains exactly two defective items and consequently tests positive. Therefore, Theorem 5.1 implies that *MBS*(V, V') determines the defective item in V.

Step 3. We need to prove that algorithm *MGT*($\mathcal{G}$) determines all defective items of $\mathcal{G}$. First we show that the algorithm terminates at line 1 if and only if $\mathcal{G}$ contains no defective item. Recall that $\mathcal{G}$ tests negative and as a consequence $|\mathcal{G} \cap \mathcal{I}| < h$ implies that $\mathcal{G}$ contains no defective item. To see whether this is the case, the algorithm tests $\mathcal{G} \cup \{z\}$. Since $\mathcal{G} \cup \{z\}$ contains at most $g-1$ defective items then it tests positive if and only if it contains less than h inhibitors. Therefore, if $\mathcal{G} \cup \{z\}$ tests positive then $\mathcal{G}$ contains no defective item and the algorithm returns the empty set.

Notice that at each step of algorithm *MGT*($\mathcal{G}$), the set $C \subset \mathcal{P}$ consists of the defective items determined so far by *MGT*($\mathcal{G}$), whereas $X \subseteq \mathcal{G}$ denotes the current search space. Notice also that $X \cup \mathcal{R} \cup C \cup \{z\}$ contains exactly g defective items.

Each iteration of the *while* loop in algorithm *MGT*($\mathcal{G}$) either cuts the search space X by a half or determines all defective items in X. Indeed, $Half_i(X) \cup \{z\}$, $i \in \{0, 1\}$, contains at most $g-1$ defective items and consequently tests positive if and only if $Half_i(X)$ contains less than h inhibitors. Therefore, a call to *CGT*($Half_i(X)$) determines all defective items in $Half_i(X)$. The *while* loop terminates either in the case when $X = \emptyset$ or in the case when both $Half_0(X) \cup \{z\}$ and $Half_1(X) \cup \{z\}$ test negative. In the former case, it is obvious that all defective items of $\mathcal{G}$ have been detected. In the latter case, the algorithm executes the *for* loop at lines 7-9. It is possible to see that if $X \cap \mathcal{P} \neq \emptyset$ then at least one of $Half_0(X) \cup \mathcal{R} \cup C \cup \{z\}$ and $Half_1(X) \cup \mathcal{R} \cup C \cup \{z\}$ tests negative. To see this, observe that if for some $i \in \{0, 1\}$ tests positive, then $Half_i(X)$ contains all defective items of X. Indeed, for $i = 0, 1$, the subset $Half_i(X) \cup \mathcal{R} \cup C \cup \{z\} \subset X \cup \mathcal{R} \cup C \cup \{z\}$ contains at most g defective items and at least h inhibitors, and consequently tests positive only if it contains exactly g defective items, that is, only if $Half_i(X)$ contains all defective items of X. Therefore, it is not possible that both $Half_0(X) \cup \mathcal{R} \cup C \cup \{z\}$ and $Half_1(X) \cup \mathcal{R} \cup C \cup \{z\}$ test positive unless $X \cap \mathcal{P} = \emptyset$. If for some $i \in \{0, 1\}$, $Half_i(X) \cup \mathcal{R} \cup C \cup \{z\}$ tests positive and

$Half_{1-i}(X) \cup \mathcal{R} \cup C \cup \{z\}$ tests negative, then all defective items of X are contained in $Half_i(X)$ and are determined by a call to $RMBS(Half_i(X), Half_{1-i}(X) \cup \mathcal{R} \cup C \cup \{z\})$. Indeed, $V = Half_i(X)$ and $V' = Half_{1-i}(X) \cup \mathcal{R} \cup C \cup \{z\}$ satisfy the hypothesis of Theorem 5.2, given that $V \cup V' = X \cup \mathcal{R} \cup C \cup \{z\}$ contains g defective items and consequently tests positive, whereas V' test negative. Moreover, V' contains at least h inhibitors and consequently Theorem 5.2 implies that $RMBS(V, V')$ determines all defective items of $Half_i(X)$. If both $Half_0(X) \cup \mathcal{R} \cup C \cup \{z\}$ and $Half_1(X) \cup \mathcal{R} \cup C \cup \{z\}$ tests negative, the algorithm determines all defective items in X by a call to $RMBS(Half_0(X), Half_1(X) \cup \mathcal{R} \cup C \cup \{z\})$ and a call to $RMBS(Half_1(X), Half_0(X) \cup \mathcal{R} \cup C \cup \{z\})$. By the same argument as above, we conclude that those two calls determine all defective items in $Half_0(X)$ and $Half_1(X)$, respectively.

5.2. Algorithm B

Step 1. The goal of this step is to determine a subset $Q \subset \mathcal{S}$ that tests positive. If such a subset cannot be found then the algorithm infers that $\mathcal{S}$ contains no defective item.

Let $\tilde{\boldsymbol{M}}_\ell$, for $\ell = 0, 1, \ldots$, denote a $(1, r, \lfloor r/2^\ell \rfloor)$-superimposed code of size n. Step 1 tests the subsets corresponding to the rows of $\tilde{\boldsymbol{M}}_0, \tilde{\boldsymbol{M}}_1, \ldots,$ until it finds a subset Q that tests positive. If $|\mathcal{P}| \geq 1$ then we are guaranteed to find such a subset by the time we have tested the subsets corresponding to the rows of $\tilde{\boldsymbol{M}}_0, \tilde{\boldsymbol{M}}_1, \ldots, \tilde{\boldsymbol{M}}_{\lceil \log(r/h) \rceil}$. Indeed, by Definition 1, there exists at least one row R of $\tilde{\boldsymbol{M}}_{\lceil \log(r/h) \rceil}$ whose associated subset T_R contains at least one defective item and no more than $h-1$ inhibitors. If $\mathcal{P} = \emptyset$ then the algorithm does not get any positive response and becomes aware that $\mathcal{P} = \emptyset$ only after testing all subsets associated with the rows of $\tilde{\boldsymbol{M}}_1, \ldots, \tilde{\boldsymbol{M}}_{\lfloor \log r \rfloor}$. If $\mathcal{P} = \emptyset$ then the algorithm terminates, otherwise it goes to the next step.

Step 2. The algorithm performs this step if and only if $\mathcal{P} \neq \emptyset$ and $|\mathcal{I}| \geq h$. The goal of this step is to partition $\mathcal{S}$ into a certain number of subsets, and each of which contains at most $h-1$ inhibitors so that the defective items in each subset can be detected by a competitive algorithm for classical group testing.

Let W and W' be two subsets of $\mathcal{S}$ such that W' contains at least one defective item and no more than $h-1$ inhibitors, and such that $W \cup W'$ contains at least h inhibitors. Notice that $W \cup W'$ is a subset of $\mathcal{S}$ and therefore contains no more than $g-1$ defective items. The following algorithm, which is a slightly modified version of that given in De Bonis and Vaccaro (2003), is used to determine a subset of $W \cup W'$ with exactly $h-1$ inhibitors.

Algorithm *Find(W, W′)*

```
1.  C ← W, D ← W′
2.  while |C| > 1
3.    do T ← D ∪ Half_0(C)
4.      if T tests negative
5.        then C ← Half_0(C)
6.        else D ← T, C ← Half_1(C)
7.  return D
```

If W and W' are as described above, then $Find(W, W')$ returns a subset D of $W \cup W'$ such that $W' \subseteq D$ and $|D \cap \mathcal{I}| = h-1$. Indeed, $Find(W, W')$ preserves the invariant that D tests positive, implying that it contains at most $h-1$ inhibitors, and that $D \cup C$ tests negative, indicating that it contains at least h inhibitors. Since the algorithm terminates as soon as $|C|$ becomes equal to 1, then the set D returned by the procedure contains exactly $h-1$ inhibitors.

Let Q be the positive subset determined in Step 1. Since $|Q \cap \mathcal{I}| < h$, it is possible to determine all defective items in Q by a competitive algorithm for classical group testing. Let x be one of the defective items in $Q \cap \mathcal{P}$. If $Find(W, W')$ is invoked with W being any subset of $\mathcal{S} \setminus x$ such that $|W \cap \mathcal{I}| \geq h$ and with $W' = \{x\}$, then it returns a subset of $W \cup \{x\}$ with exactly $h-1$ inhibitors. The following algorithm finds the desired partition of $\mathcal{S}$ by iteratively invoking $Find(W, \{x\})$ with the sets W's being pairwise disjoint subsets of $\mathcal{S} \setminus x$ with at least h inhibitors.

Algorithm *RFind*

1. $A \leftarrow \mathcal{S} \setminus Q$
2. $\mathcal{F} \leftarrow \{Q\}$
3. **while** $A \cup \{x\}$ tests negative
4. **do** $B \leftarrow Find(A, \{x\})$
5. $A \leftarrow A \setminus B, \mathcal{F} \leftarrow \mathcal{F} \cup \{B \setminus \{x\}\}$
6. **return** $\mathcal{F} \cup \{A\}$

The subsets in the collection $\mathcal{F}$ returned by *RFind* form the desired partition of $\mathcal{S}$ since each of them contains at most $h-1$ inhibitors.

6. ANALYSIS OF THE ALGORITHM

To analyze the cost of algorithm A, we need to estimate the number of tests performed by algorithm *RMBS*(V, V'). Observe that the *while* loop of *RMBS*(V, V') is iterated $|B|$ times, where B is the set of defective items returned by the algorithm. Each iteration performs two tests in addition to those performed by the call to *MBS*(V, V') (actually *RMBS*(V, V') can be implemented in such a way that the test at line 4 is no longer performed after the first YES response). Since each call to *MBS*(V, V') performs $\lceil \log |V| \rceil = O(\log n)$ tests, then the overall cost of *RMBS*(V, V') is $O(|B| \log n)$.

6.1. Analysis of Algorithm A

Step 1. This step makes a call to *RMBS*($\mathcal{S}^*, \emptyset$). If $g > 1$ and $|\mathcal{I}| \geq h$ then the set of defective items returned by *RMBS*($\mathcal{S}^*, \emptyset$) has size $|B| = |\mathcal{P}| - g + 1$, and consequently the cost of Step 1 is $O((|\mathcal{P}| - g + 1) \log n)$. If $g = 1$ or $|\mathcal{I}| < h$ then $|B| = |\mathcal{P}|$ and the cost of Step 1 is $O(|\mathcal{P}| \log n)$.

Step 2. If $g > 2$ and $|\mathcal{I}| \geq h$ then Attempt 1 succeeds, and thus the number of tests performed to determine a suitable partition $(T_R, \mathcal{S}^* \setminus T_R)$ of $\mathcal{S}^*$ is equal to at most four times the minimal length of a (1, 1)-superimposed code of size $|\mathcal{S}^*|$. Notice that (1, 1)-superimposed codes correspond to Sperner families and it is a very well known result (Spener 1928) that the largest size of such a family of subsets of $\{1, \ldots, t\}$ is $\binom{t}{\lfloor t/2 \rfloor}$. Hence, if $g > 2$ and $|\mathcal{I}| \geq h$, then $O(\log |\mathcal{S}^*|) = O(\log n)$ tests are sufficient to determine a suitable partition $(T_R, \mathcal{S}^* \setminus T_R)$ of $\mathcal{S}^*$. Let $b_1 \geq 0$ and $b_2 \geq 0$ be the number of defective items detected in T_R and in $\mathcal{S}^* \setminus T_R$, respectively, and let $b = b_1 + b_2$. The number of tests performed to find b_i defective items is $O(b_i \log |\mathcal{S}^*|) = O(b_i \log n)$. Indeed, these items are discovered either by a competitive strategy for classical group testing, or by exploiting algorithm *RMBS*(V, V'). Therefore, if $g > 2$ and $|\mathcal{I}| \geq h$, the overall cost of this step is $O(b_1 \log n) + O(b_2 \log n) \pm O(\log n) = O(b \log n)$.

Let us estimate the cost of Step 2 in the case $g \leq 2$ or $|I| < h$. To find a suitable partition $(T_R, \mathcal{S}^* \setminus T_R)$, Attempt 2 performs at most $4 \cdot \sum_{i=1}^{\lfloor \log r \rfloor} N(1, 2^{i+1}, 2^{i-1}, |\mathcal{S}^*|)$ tests, in addition to the $O(\log n)$ tests performed by Attempt 1. The upper bound of Theorem 3.1 implies that $\sum_{i=1}^{\lfloor \log r \rfloor} N(1, 2^{i+1}, 2^{i-1}, |\mathcal{S}^*|) = \sum_{i=1}^{\lfloor \log r \rfloor} O(\log |\mathcal{S}^*|) = O(\log r \log |\mathcal{S}^*|)$.. Attempt 2 succeeds if and only if $g = 2$ and $|\mathcal{I}| \geq h$, and in this case $O(\log |\mathcal{S}^*|)$ tests are used to search for the unique defective item of $\mathcal{S}^*$, either by binary search or by algorithm *MBS*(V, V'). Therefore, if $g \leq 2$ or $|\mathcal{I}| < h$ then the overall cost of Step 2 is $O(\log r \log |\mathcal{S}^*|) = O(\log r \log n)$.

Step 3. The *while* loop of algorithm *MGT*($\mathcal{G}$) is iterated at most $\lceil \log |\mathcal{G}| \rceil$ times. Each iteration performs at most two tests in addition to those performed to eventually search for the defective items either in one of *Half*$_0(\mathcal{G})$ and *Half*$_1(\mathcal{G})$, or in both of them. Therefore, the total number of tests performed by the *while* loop is $O(c_1 \log |\mathcal{G}|) + O(\log |\mathcal{G}|) = O(c_1 \log n)$, where c_1 denotes the number of defective items detected by the *while* loop. If the algorithm exits the *while* loop with $X \neq \emptyset$ then the algorithm executes the *for* loop at lines 7–9 and determines the defective items in X by exploiting algorithm *RMBS*(V, V'). The total number of tests needed to search for the defective items in X is therefore $O(c_2 \log |X|)$, where c_2 denotes the number of defective item in X. It follows that the total number of tests performed by Step 3 is $O(c \log |\mathcal{G}|) + O(\log |\mathcal{G}|) = O(c \log n)$, where c denotes the number of defective items in $\mathcal{G}$.

The following theorem summarizes the conclusions of the above analysis.

Theorem 6.1. *If the given set of items $\mathcal{S}$ tests positive then our algorithm successfully identifies all defective items in $\mathcal{S}$ by $O(|\mathcal{P}|\log n)$ tests when $g>2$ and $|\mathcal{I}|\geq h$, and by $O(|\mathcal{P}|\log n+\log r\log n)$ tests, otherwise. Moreover, if $g>2$ and $|\mathcal{I}|\geq h$, then our algorithm attains the same complexity as above even if no upper bound r on the number of inhibitors is given.*

If $g>2$ and $|\mathcal{I}|\geq h$ then the upper bound of Theorem 6.1 asymptotically matches the information theoretic lower bound thus implying that the algorithm is asymptotically optimal. If $g\leq 2$ or $|\mathcal{I}|<h$, the cost of algorithm A exceeds the information theoretic lower bound by an $O(\log r\log n)$ additive term. For $\log\log r=O(|\mathcal{P}|)$ this upper bound asymptotically matches the information theoretic lower bound and consequently is asymptotically optimal. Intuitively, the $O(\log r\log n)$ additional term corresponds to the cost of determining the exact number $|\mathcal{I}|$ of inhibitors, given that $0\leq|\mathcal{I}|\leq r$. In the analysis of algorithm A, this additional term represents the cost incurred by Step 2 when testing the subsets associated with the rows of $\tilde{M}_1,\ldots,\tilde{M}_{\lfloor\log r\rfloor}$. If we assume known the exact number $|\mathcal{I}|$ of inhibitors, then in Step 2 it suffices to test the subsets corresponding to the rows of a $(1,|\mathcal{I}|,\lfloor|\mathcal{I}|/2\rfloor)$-superimposed code, instead of those associated with the rows of $\tilde{M}_1,\ldots,\tilde{M}_{\lfloor\log r\rfloor}$. Hence, we have that the following theorem holds.

Theorem 6.2. *If the given set of items $\mathcal{S}$ tests positive and the exact number $|\mathcal{I}|$ of inhibitors is known, then there exists an asymptotically optimal algorithm that successfully identifies all defective items in $\mathcal{S}$ by $O(|\mathcal{P}|\log n)$ tests.*

Proof. We will show that if the exact number $|\mathcal{I}|$ of inhibitors is known, then in Attempt 2 of Step 2 it suffices to consider the subsets corresponding to the rows of a $(1,|\mathcal{I}|,\lfloor|\mathcal{I}|/2\rfloor)$-superimposed code, instead of those associated with the rows of $\tilde{M}_1,\ldots,\tilde{M}_{\lfloor\log r\rfloor}$. Recall that Attempt 2 is performed only if $g\leq 2$ or $|\mathcal{I}|<h$.

Let z denote one of the defective item determined in Step 1. For each row R of the $(1,|\mathcal{I}|,\lfloor|\mathcal{I}|/2\rfloor)$-superimposed code, the algorithm tests both T_R and $\mathcal{S}^*\setminus T_R$ and if both of them test negative then it performs an additional pair of tests on $T_R\cup\{z\}$ and $(\mathcal{S}^*\setminus T_R)\cup\{z\}$. We will show that if $g=2$ and $|\mathcal{I}|\geq h$ both hold, then one of cases (i′) and (ii′) of Attempt 2 occurs, and consequently Step 2 is able to detect the unique defective item of $\mathcal{S}^*$ either by binary search or by algorithm *MBS*(*V, V′*) as explained in Step 2. To this aim, we observe that by Definition 1 there exists a row R of the $(1,|\mathcal{I}|,\lfloor|\mathcal{I}|/2\rfloor)$-superimposed code such that T_R contains the unique defective item of $\mathcal{S}^*$ and less than $\lfloor|\mathcal{I}|/2\rfloor$ inhibitors, whereas $\mathcal{S}^*\setminus T_R$ contains no defective item and more than $\lceil|\mathcal{I}|/2\rceil$ inhibitors. Therefore, if $h\geq\lfloor|\mathcal{I}|/2\rfloor$ then T_R contains at most $h-1$ inhibitors and consequently tests positive, whereas if $h<\lfloor|\mathcal{I}|/2\rfloor$ then $\mathcal{S}^*\setminus T_R$ contains at least h inhibitors. and consequently tests negative even after adding the defective item z to it. Notice that $T_R\cup\{z\}$ contains two defective items and consequently tests positive even if it contains h or more inhibitors. If T_R tests positive then case (i′) occurs, whereas if T_R tests negative then by the above argument we have that $h<\lfloor|\mathcal{I}|/2\rfloor$ and consequently $(\mathcal{S}^*\setminus T_R)\cup\{z\}$ tests negative and we have that case (ii′) occurs.

If none of cases (i′) and (ii′) occurs, then the algorithm concludes that $g=1$ or $|\mathcal{I}|<h$ and consequently all defective items have been determined in Step 1.

By Theorem 3.1, we obtain an $O(\log n)$ upper bound on the minimum length of $(1,|\mathcal{I}|,\lfloor|\mathcal{I}|/2\rfloor)$-superimposed codes of size n. For each row of the $(1,|\mathcal{I}|,\lfloor|\mathcal{I}|/2\rfloor)$-superimposed code, the above testing strategy performs at most four tests. Moreover $O(\log n)$ tests are used to determine the unique defective item either by binary search or by algorithm *MBS*(*V, V′*). Therefore, the overall number of tests performed by this modified version of Attempt 2 is $O(\log n)$. The upper bound stated in the theorem follows from the analysis of algorithm A by replacing the $O(\log r\log n)$ cost of Attempt 2 with the $O(\log n)$ cost of the above testing strategy. ■

6.2. Analysis of Algorithm B

Step 1. This step performs at most $\sum_{i=1}^{\lfloor\log r\rfloor}N(1,r,\lfloor r/2^i\rfloor,n)$ tests, if $\mathcal{P}=\emptyset$, and at most $\sum_{i=1}^{\lceil\log r/h\rceil}N(1,r,\lfloor r/2^i\rfloor,n)$ tests, otherwise. Theorem 3.1 implies $N(1,\ r,\lfloor\ r/2^i\rfloor,\ n)=O(2^{2i}\log n)$, and consequently the cost of this step is $O(r^2\log n)$, if $\mathcal{P}=\emptyset$, and $O(r^2/h^2\log n)$, otherwise.

Step 2. Algorithm *Find*(W, W') performs $O(\log |W|)$ tests. The *while* loop in algorithm *RFind*($\mathcal{S}, x$) is iterated at most $\left\lceil \frac{|\mathcal{I} \cap Q|}{h-1} \right\rceil$ times and at each iteration *Find*(B, $\{x\}$) performs $O(\log |B|) = O(\log n)$ tests. Therefore, the overall number of tests executed by *RFind*($\mathcal{S}, x$) is $O(|\mathcal{I}|/(h-1)\log n)$. Finding all defective items in Q and in each of the subsets of $\mathcal{F}$ requires $O(|\mathcal{P}| \log n)$ tests. Hence, the following theorem holds.

Theorem 6.3. *If the given set of items $\mathcal{S}$ tests negative then our algorithm successfully identifies all defective items in $\mathcal{S}$ by $O((r/h)^2 \log n + |\mathcal{I}|/(h-1)\log n + |\mathcal{P}| \log n)$ tests when $\mathcal{P} \neq \emptyset$, and by $O(r^2 \log n)$ tests otherwise.*

The upper bounds of Theorem 6.3 exceed by an $O(\log r)$ factor the lower bounds of Corollary 4.4 and Theorem 4.5, respectively. This gap is a consequence of the $\log r$ gap between the known upper and lower bounds on the minimal length of superimposed codes, thus closing this gap would correspond to solving a crucial question in extremal combinatorics.

DISCLOSURE STATEMENT

No competing financial interests exist.

REFERENCES

Balding, D.J., Bruno, W.J., Knill, E., et al. 1996. A comparative survey of non-adaptive pooling design, 133–154. *In* Speed, T.P., and Waterman, M.S., eds. *Genetic Mapping and DNA Sequencing, IMA Vol. Math. Appl.* Springer-Verlag, New York.

Barillot, E., Lacroix, B., and Cohen, D. 1991. Theoretical analysis of library screening using an n-dimensional pooling strategy. *Nucleic Acids Res.* 6241–6247.

Berger, T., Mehravari, N., Towsley, D., et al. 1984. Random multiple-access communication and group testing. *IEEE Trans. Commun.* 32, 769–779.

Bruno, W.J., Balding, D.J., Knill, E., et al. 1995. Design of efficient pooling experiments. *Genomics* 26, 21–30.

Chang, H.L., Chen, H.-B., and Fu, H.L. 2010. Identification and classification problems on pooling designs for inhibitor models. *J. Comput. Biol.* (to appear).

Chen, H.-B., and Fu, H.L. 2008. Nonadaptive algorithms for threshold group testing. *Discrete Appl. Math.* 157, 1581–1585.

Damaschke, P. 2006. Threshold group testing. *Lect. Notes Comput. Sci.* 4123, 707–718.

De Bonis, A. 2008. New combinatorial structures with applications to efficient group testing with inhibitors. *J. Comb. Optim.* 15, 77–94.

De Bonis, A., and Vaccaro, U. 2006. Optimal algorithms for two group testing problems and new bounds on generalized superimposed codes. *IEEE Trans. Inform. Theory* 52, 4673–4680.

De Bonis, A., Gasieniec, L., and Vaccaro, U. 2005. Optimal two-stage algorithms for group testing problems. *SIAM J. Comput.* 34, 1253–1270.

De Bonis, A., and Vaccaro, U. 2003. Constructions of generalized superimposed codes with applications to group testing and conflict resolution in multiple access channels. *Theoret. Comput. Sci.* 306, 223–243.

De Bonis, A., and Vaccaro, U. 1998. Improved algorithms for group testing with inhibitors. *Inform. Process Lett.* 67, 57–64.

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

Du, D.Z., and Hwang, F.K. 2006. *Pooling Designs and Nonadaptive Group Testing—Important Tools for DNA Sequencing*. World Scientific, New York.

Du, D.Z., and Hwang, F.K. 2000. *Combinatorial Group Testing and Its Applications*. World Scientific, New York.

Du, D.Z., and Hwang, F.K. 1992. Competitive group testing, 125–134. *In* McGeoch, L.A., and Sleator, D.D., eds. *On-Line Algorithm, DIMACS Ser. Discrete Math. Theoret. Comput. Sci.* Amer. Math. Soc., Washington, DC.

D'yachkov, A.G., and Rykov, V.V. 1983. A survey of superimposed code theory. *Probl. Control Inform. Theory* 12, 229–242.

D'yachkov, A.G., Macula, A.J., Torney, D.C., et al. 2001. Two models of nonadaptive group testing for designing screening experiments, 63–75. *In* Attkinson, A.C., Hackl, P., and Muller, W.G., eds. *Proc. 6th Int. Workshop in Model Oriented Design and Analysis*. Physica-Verlag, Berlin.

D'yachkov, A.G., and Rykov, V.V. 1982. Bounds on the length of disjunct codes. *Probl. Control Inform. Theory* 12, 7–13.

Farach, M., Kannan, S., Knill, E., et al. 1997. Group testing with sequences in experimental molecular biology. *Proc. Compression Complexity Sequences* 357–367.

Füredi, Z. 1996. On *r*-cover free families. *J. Combin. Theory Ser. A* 73, 172–173.

Hong, E.S., and Ladner, R.E. 2002. Group testing for image compression. *IEEE Trans. Image Process.* 11, 901–911.

Hong, Y.W., and Scaglione, A. 2004. On multiple access for distributed dependent sensors: a content-based group testing approach. *Proc. IEEE Inform. Theory Workshop* 298–303.

Hwang, F.K., and Liu, Y.C. 2003. Error-tolerant pooling designs with inhibitors. *J. Comput. Biol.* 10, 231–236.

Kautz, W.H., and Singleton, R.R. 1964. Nonrandom binary superimposed codes. *IEEE Trans. Inform. Theory* 10, 363–377.

Langfeldt, S.A., Hughes-Oliver, J.M., Ghosh, S., et al. 1997. *Optimal Group Testing in the Presence of Blockers.* Institute of Statistics, North Carolina State University.

Li, C.H. 1962. A sequential method for screening experimental variables. *J. Am. Stat. Assoc.* 57, 455–477.

Margaritis, D., and Skiena, S. 1995. Reconstructing strings from substrings in rounds. *Proc. 37th IEEE Annu. Symp. FOCS 95* 613–620.

Ngo, H.Q., and Du, D.Z. 2000. A survey on combinatorial group testing algorithms with applications to DNA library screening, 171–182. *In: Discrete Mathematical Problems with Medical Applications, DIMACS Ser. Discrete Math. Theoret. Comput. Sci.* 55, Amer. Math. Soc., Washington, DC.

Phatarfod, R.M., and Sudbury, A. 1994. The use of a square array scheme in blood testing. *Stat. Med.* 13, 2337–2343.

Pevzner, P.A., and Lipshutz, R. 1994. Towards DNA sequencing chips. *Lect. Notes Comput. Sci.* 841, 143–158.

Ruszinkó, M. 1994. On the upper bound of the size of the *r*-cover-free families. *J. Combin. Theory Ser. A* 66, 302–310.

Sobel, M., and Groll, P.A. 1959. Group testing to eliminate efficiently all defectives in a binomial sample. *Bell Syst. Tech. J.* 38, 1179–1252.

Sperner, E. 1928. Ein Satz über Untermengen einer endlichen Menge. *Math. Z.* 27, 544–548.

Wolf, J. 1985. Born again group testing: multiaccess communications. *IEEE Trans. Inform. Theory* 31, 185–191.

Xie, M., Tatsuoka, K., Sacks, J., et al. 2001. Group testing with blockers and synergism. *J. Am. Stat. Assoc.* 96, 92–102.

Address correspondence to:
Dr. Annalisa De Bonis
Dipartimento di Informatica ed Applicazioni
Università di Salerno
84084 Fisciano (SA), Italy

E-mail: debonis@dia.unisa.it