Sequential Group Testing with Graph Constraints

Amin Karbasi
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland

Morteza Zadimoghaddam
Computer Science and Artificial Intelligence Laboratory (CSAIL)
Massachusetts Institute of Technology (MIT)
Cambridge-MA-USA

***Abstract*—In conventional group testing, the goal is to detect a small subset of defecting items $\mathcal{D}$ in a large population $\mathcal{N}$ by grouping *arbitrary* subset of $\mathcal{N}$ into different pools. The result of each group test $\mathcal{T}$ is a binary output depending on whether the group contains a defective item or not. The main challenge is to minimize the number of pools required to identify the set $\mathcal{D}$.**

Motivated by applications in network monitoring and infection propagation, we consider the problem of group testing with graph constraints. As opposed to conventional group testing where *any* subset of items can be pooled, here a test is admissible if it induces a connected subgraph $H \subset G$. In contrast to the non-adaptive pooling process used in previous work, we first show that by exploiting an adaptive strategy, one can dramatically reduce the number of tests. More specifically, for *any* graph G, we devise a 2-approximation algorithm (and hence order optimal) that locates the set of defective items $\mathcal{D}$. To obtain a good compromise between adaptive and non-adaptive strategies, we then devise a multi-stage algorithm. In particular, we show that if the set of defective items are uniformly distributed, then an l-stage pooling strategy can identify the defective set in $O(l \cdot |D| \cdot |\mathcal{N}|^{1/l})$ tests, on the average. In particular, for $l = \log(|\mathcal{N}|)$ stages, the number of tests reduces to $4|\mathcal{D}| \log(|\mathcal{N}|)$, which in turn is order optimum.

I. Introduction

The aim of group testing is to identify a subset of defective items $\mathcal{D}$ (with size at most d) out of a much larger set of items $\mathcal{N}$ (with size n). The problem is particularly interesting in scenarios where multiple items in a group can be simultaneously tested. As a result, instead of testing each item individually, we can hope that by pooling a subset of items together and test them collectively, we can reduce the number of tests[1]. In such scenarios then the fundamental challenge is to identify the set $\mathcal{D}$ by means of the fewest possible number of tests (or queries) in the following form:

"Does the test $\mathcal{T}$, where $\mathcal{T}$ is a subset of $\mathcal{N}$, contains at least one defective item?"

The problem was originally considered during World War II for syphilis screening [1]. It has also found numerous application in a variety of situations such as DNA library screening [2], [3], [4], [5], [6], product testing and quality assurance [7], pattern matching [8], streaming algorithms [9]. We refer the interested reader to [10] for more details on the major developments in this area.

[1]It is customary in the group testing literature to think of d as a parameter that is noticeably smaller than n, i.e., $d = o(n)$. Indeed, if d becomes comparable to n, there would be little point in using a group testing scheme and trivial tests on the individuals are more favorable.

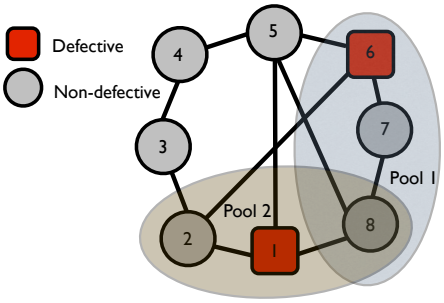

Fig. 1. The pools $\mathcal{T}_1 = \{1, 2, 8\}$ and $\mathcal{T}_2 = \{6, 7, 8\}$ are admissible. On the other hand, the test $\mathcal{T}_3 = \{3, 4, 6\}$ is not admissible, since its induced graph is not connected.

A group testing problem can be either *combinatorial* or *probabilistic*. In combinatorial group testing the subset of defective items $\mathcal{D}$, can be any element of a predetermined family of subsets of $\mathcal{N}$. The task is thus to devise an algorithm which requires the minimal number of tests to identify $\mathcal{D}$ in the worst case. In the probabilistic setting, a probability distribution μ on the subset of defective items is given where the goal is to minimize the expected number of tests (with respect to μ) to identify D. For both combinatorial and probabilistic scenarios there is an additional classification according to the number of stages, i.e., the pool design can be composed of one or more stages of parallel queries. In one-stage or *non adaptive* algorithms all tests are specified in advance: the choice of the pools does not depend on the outcome of the tests. The other extreme case is *adaptive* algorithms: tests are performed one at a time and the results from previous tests are allowed to be used to guide future tests. While non-adaptive algorithms are easier to implement, the number of tests required by them can be potentially much larger than adaptive ones. A good compromise between adaptive and non-adaptive algorithms is the *multi-stage* algorithms where at each stage a set of predetermined pools is designed and their outcomes is used for the subsequent stage (see for example [11]). In this paper we consider a generalization of the group testing problem where the test should conform to the restrictions imposed by a graph. More precisely, the set $\mathcal{N}$ is furnished with an undirected graph $G(V, E)$. A test $\mathcal{T} \subset \mathcal{N}$ is *admissible* if and only if the subgraph induced by $\mathcal{T}$ (i.e., containing all edges of G between vertices in $\mathcal{T}$) is connected (see Figure 1). Like in standard group testing, the main task is to identify the set of

defective items with the fewest possible number of tests in the following form:

"Does the admissible test $\mathcal{T}$, where $\mathcal{T}$ induces a connected subgraph of G, contains at least one defective item?"

Inspired by applications in network tomography [12], the non-adaptive graph-constrained group testing was first introduced in [13] and was further investigated in [14]. A few recent works also consider graph constraints in compressed sensing [15], optical networks [16], sensor failure detection [17] and interactive search [18].

Our aim in this paper is to provide provably efficient algorithms for graph-constrained group testing. To this end, we first look at adaptive or sequential algorithms and provide constructions based on which any subset of defective items can be identified. To close the gap between adaptive and non-adaptive algorithms, we then look at multi-stage strategies. Unfortunately, one can provide simple examples (by carefully choosing the set of defective items on the graph) in which there is a huge gap between adaptive and multi-stage algorithms. However, if the set of defective items are chosen uniformly at random (and not adversarially), then we can again provide an order-optimal algorithm. In brief, our contributions are as follows:

- For any constraint graph G, we derive a lower bound on the number of admissible tests needed to identify the set of defective items.
- We devise an adaptive algorithm that can find the set of defective items efficiently. We also show that our algorithm test at most twice as often as the optimum algorithm, in the worst case.
- Finally, for the uniformly distributed set of defective items, we provide a multi-stage algorithm whose expected number of tests is order optimum.

The remainder of this paper is organized as follows. We briefly review the related work in Section II. In Section III we introduce the notation and our corresponding model. We then describe our main results in Section IV.

II. Related Work

Any non-adaptive group testing problem can be reduced to constructing a $0-1$ *test matrix* M, where each row corresponds to a group test. The design of such a matrix has been extensively studied in literature (see for example, [19], [20], [21], [22], [23]). In particular, the best known non-adaptive design requires a test matrix whose number of rows scale as $O(d^2 \log n)$ [19] which almost matches the lower bound $\Omega(d^2 \log n / \log d)$ [24]. The closest works to ours are [13], [14] where authors consider non-adaptive group testing under graph constraints. While the main contribution of [13] is for the case of only one defective item ($d = 1$), the authors in [14] could generalize their approach for $d \geq 1$. Under the assumption that the constraint graph G is richly connected and has the mixing time $T(n)$, they showed that with $O(d^2 T^2(n) \log(n/d))$ non-adaptive tests, one could identify the defective set $\mathcal{D}$. In contrast to [14], we aim to devise an efficient strategy that works under *any* constraint graph G. To this end, we consider adaptive algorithms.

Since in group testing, each test returns one of the two values (either the subset contains no defectives or it contains at least one defective) a simple information theoretic lower bound on the number of tests scales as $\log \binom{n}{d} \approx d \log(n/d)$. Hwang's adaptive algorithm [25] exceeds the above information theoretic lower bound by at most d, a result that was further improved in [26], [27]. In particular, authors in [27] provide a 1.5-approximation algorithm. Our approach in this work is similar to [27] where we provide a 2-approximation algorithm when the tests should conform to the constraints imposed by a graph.

Multi-stage algorithms are at the mid-point between adaptive and non-adaptive algorithms. Much of the existing work in the literature focuses on the probabilistic settings and two stage scenarios (e.g., [28], [29]). In this regard, our work can be seen as an extension to [28], [29] where we consider multi-stage strategies with graph constraints.

III. Setting and Notation

Consider a set of items $\mathcal{N}$ with size n which are enumerated from 1 to n. Among this set of objects we assume that at most d of them are defective. Let $\mathcal{D}$ denote the set of defective items. In our problem, the set $\mathcal{N}$ is furnished with a given graph $G = (V, E)$ where V is the set of nodes (and in our case $V = \mathcal{N}$) and E is the set of edges. An admissible group test $\mathcal{T}$ corresponds to a walk (a sequence of adjacent edges, allowing repetitions) on the corresponding graph (see Figure 1). The goal is to identify the set $\mathcal{D}$ with minimum number of admissible group tests, where each test determines whether the set of vertices observed along the walk has an intersection with the defective set or not. Throughout this paper we denote by $G|_{\mathcal{T}}$ a subgraph of G induced by the subset $\mathcal{T} \subset \mathcal{N}$.

To have a concrete example in mind, we can mention the network monitoring application where the network can be modeled as a graph G. The sets V and E correspond to the routers/hosts and communication links, respectively. Among the routers and hosts, we assume that only a few of them are defective , i.e., drop the received packets. An admissible test is simply performed by sending packets through the network for which the path (routers and end hosts) are predetermined. A central server collects all the results - whether each packet has reached its destination or not - and based on them tries to identify the subset of defective routers/hosts.

IV. Main Results

In this section, we first derive a lower bound and then describe our adaptive and multi-stage algorithms. To this end, we need the following definition.

Definition IV.1. *For any graph G, and an integer $l \geq 0$, we define $CC(G, l)$ to be the maximum number of connected components of graph $G \setminus S$ where S could be any subset of vertices with size at most l.*

Algorithm 1 AdaptiveFinder: Finds all targets in a subgraph H.

Input: Connected sub-graph $H \subset G$.

$v \leftarrow$ SingleFinder(H)

Return v as one of the defectives and remove it from H.

Test which one of the connected components of $H \setminus \{v\}$ contains a defective item.

Recurse on the ones that contain a defective item.

Equipped with this definition, we can state our lower bound.

Lemma IV.2. *The number of admissible tests required by any strategy to identify the set of d defective items on a graph G is lower bounded by* $\max\{CC(G, d-1) - 1, d\log(n/d)\}$.

Proof: Suppose $CC(G, d-1)$ is realized by a subset $S \subset V$ with $|S| \leq d-1$, i.e. removing S leaves G with $CC(G, d-1)$ connected components. Note that any strategy, including the optimum strategy, should identify the set of defectives $\mathcal{D}$ no matter where they are located on the graph. In particular, the optimum strategy has to identify D when all nodes of S are defective and this side information is given. In this case, the optimum strategy gains no information by testing a pool that has a node in S (the result is positive anyway). Hence, the optimum strategy tests only connected subgraphs that have no intersection with S. Since $G \setminus S$ has $CC(G, d-1)$ connected components, the optimum algorithm, therefore, makes at least $CC(G, d-1) - 1$ tests. Otherwise, there will be at least two connected components T_1 and T_2 of $G \setminus S$ with which none of the tests of the optimum strategy intersects. As a result, even the optimum strategy cannot distinguish between the defective and non-defective elements of $T_1 \cup T_2$ (Note that in the worst case, it is always possible to have some, and not all, of the nodes in $T_1 \cup T_2$ defective). The second term in the lower bound is simply the information theoretic lower bound stated earlier. ∎

A. Adaptive Algorithm

Our adaptive algorithm, named AdaptiveFinder, is shown in Algorithm 1. Its task is to identify all of the defective items in a connected subgraph $H \subset G$. To do so, it uses Algorithm 2, named SingleFinder, as a subroutine. The task of Algorithm 2 is to identify a single defective item of H. Once this single item is identified, it is removed from H, which then decomposes $H \setminus \{v\}$ into (potentially) a number of connected components. Algorithm 1 tests which one contains a defective item and finds them by recursively calling Algorithm 2. To start, we call Algorithm 1 with G as its input.

Note that there are two types of admissible group tests made by our adaptive algorithm:

1) test made on the connected components of $H \setminus \{v\}$ in Algorithm 1.
2) test made on the subtrees of H in Algorithm 2.

The following lemmas bound the total number of tests made for each type.

Algorithm 2 SingleFinder: Finds a target in a subgraph H.

Input: Connected sub-graph $H \subset G$.

Grow a connected subtree $T \subset G$ until T spans $\lceil n/2 \rceil$ vertices.

Test the connected subtree T (size $\lceil n/2 \rceil$).

if T contains a defective item **then**

Recurse on the subgraph of G induced by T.

else

Collape all nodes in T into a single node v_T, which is connected to all $v \in \mathcal{N}$ adjacent to nodes in T in the original G. Create G' by Connecting all neighbours of v_T to one another and removing v_T. Recurse on G'.

end if

Lemma IV.3. *The total number of tests made on the connected components in Algorithm 1 is at most $CC(G, d-1) + d$.*

Proof: The progress of Algorithm 1 can be shown by a tree T_1 in the following way. The nodes of the tree are simply the connected components of G tested by Algorithm 1 during its process. More precisely, for each connected component $H \subset G$, Algorithm 1 first finds a defective item $v \in H$, removing of which decomposes H into a number of connected components. Each of these connected components can be shown as a child of a subtree rooted at H. Algorithm 1 tests all the connected components (i.e., all the children) and only recurses on those components that contain a defective item/node. Hence, the total number of tests made on the connected components in Algorithm 1 equals the total number of nodes of T_1. Therefore, we just need to bound the total number of nodes of T_1, i.e., the total number of leaves plus internal nodes. Note that after finding the d-th defective item, Algorithm 1 terminates. Hence, Algorithm 1 tests the connected components for the first $d-1$ defective items. Therefore, the leaves of T_1 are the connected components of graph G after removing at most $d-1$ defective nodes. This implies that the total number of leaves of T_1 is at most $CC(G, d-1)$. On the other hand, the total number of internal (non-leaf) nodes of T_1 cannot be more than d. This is due to the fact that each internal node is associated with a distinct defective item whose removal results the children of this internal node. ∎

Lemma IV.4. *The total number of tests made in Algorithm 2 does not exceed $d\log(n)$.*

Proof: Each test made by Algorithm 2 halves the number of remaining items. Hence in $\log n$ tests a defective item is identified. Since there are no more than d defective items, the total number of tests performed by Algorithm 2 is upper bounded by $d\log(n)$. ∎

The following lemma provides the approximation factor of our adaptive algorithm.

Lemma IV.5. *For any constant $\epsilon > 0$, there exists some n_0 such that the total number of tests performed by Algorithm 1 is at most $2 + \epsilon$ times the optimum number of tests needed to*

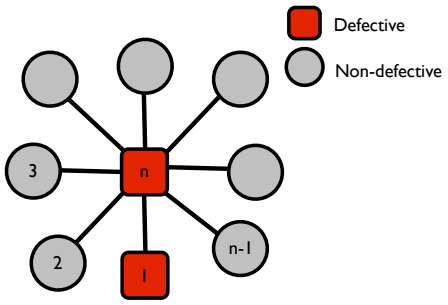

Fig. 2. A star-shape graph with two defective items.

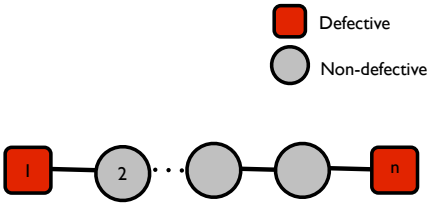

Fig. 3. A line graph with two defective items.

identify defective nodes in a graph with $n > n_0$ nodes.

Proof: Using Lemmas IV.4 and IV.3, we have that the total number of tests in our algorithm is at most $d(\log(n)+1)+CC(G,d-1)$. On the other hand, by Lemma IV.2, we know that any algorithm has to make at least $\max\{CC(G,d-1)-1, d\log(n/d)\}$ tests. Hence, the approximation factor of our algorithm is at most $\frac{d(\log(n)+1)+CC(G,d-1)}{\max\{CC(G,d-1)-1,d\log(n/d)\}}$. Since we are working in the regime that $n \gg d$, this ratio approaches 2 from above as n goes to ∞. In the other words, for every $\epsilon > 0$, this ratio is at most $2+\epsilon$ for large enough values of n. ■

In Figure 2 a simple example is shown. Observe that any connected component with at least two vertices contains the defective node n. After removing this node, the resulting graph decomposes into $n-1$ components. To find the second defective node, any algorithm has to test $n-1$ nodes, in the worst case. Hence, even in this simple case (and at the same time extreme case), the optimum algorithm has to perform $\Theta(n)$ number of admissible tests in the worst-case. On the other hand, for a line graph shown in Figure 3 the number of tests perform by our adaptive algorithm is $O(\log(n))$. It was shown by [13] that any non-adaptive algorithm requires $O(n)$ tests to identify even a single defective node.

The vast difference between the performance of an adaptive and non-adaptive algorithm in Figure 3 is due to the fact that the set of defective items can be chosen arbitrarily. If, on the other hand, the set of defectives is chosen randomly, this difference diminishes and we can devise an order optimum algorithm at least for the multi-stage scenario. This is the topic of next section.

B. Multi-Stage Algorithm

In this section, we address the performance of multi-stage algorithms when the set of defectives is chosen uniformly at random. Formally, there are $\Sigma_{i=0}^{d}\binom{n}{i}$ possibilities for the set of at most d defective nodes, and the probability that the defective items form a specific set $\mathcal{D}$ (where $|\mathcal{D}| \le d$) is $1/(\Sigma_{i=0}^{d}\binom{n}{i})$.

Our approach is to reduce the multi-stage scenario in the average case to the problem of identifying defective nodes on a line graph (defined below) in the worst case. To start we need the following definition.

Algorithm 3 PathFinder: a multi-stage algorithm to find all targets in a path

Input: A subpath $P_{i,j}$

if $i = j$ **then**
  Return v_i as a target.
else
  Divide subpath $P_{i,j}$ into $n^{1/l}$ (almost) equal subpaths, and test each of them.
  Recurse on any subpath that contains some defective node.
end if

Definition IV.6. *The line graph, P_n, consists of n vertices $v_1, v_2, \cdots, v_n$ such that v_i is connected to v_{i+1} for each $1 \le i \le n-1$. Subpath $P_{i,j}$ denotes the connected subgraph of P_n consisting of vertices $v_i, v_{i+1}, \cdots, v_j$ for $1 \le i \le j \le n$.*

For any $1 \le l \le \log(n)$, Algorithm 3 presents an l-stage algorithm that finds up to d defective nodes on a line graph P_n. Using our reduction, we solve the problem for any connected graph in the uniformly distributed setting with the same bound on the number of tests. Algorithm 3 is a simple divide-and-conquer algorithm that receives as an input a subpath $P_{i,j}$, and divides it into $n^{1/l}$ subpaths of length $\lfloor (j-i+1)/n^{1/l} \rfloor$ or $\lceil (j-i+1)/n^{1/l} \rceil$. By testing each of them, Algorithm 3 only recurses on the ones that contain a defective node.

Lemma IV.7. *For a line graph P_n with at most d defective nodes, the number of stages and the number of admissible tests required by Algorithm 3 are at most l and $d \cdot l \cdot n^{1/l}$, respectively.*

Proof: At the first stage, Algorithm 3 receives the line graph P_n and divides it into $n^{1/l}$ disjunct subpaths. Then, each of the subpathes are tested and only the defective ones (i.e., the ones containing a defective item) will be probed for the next stage. Since there are at most d defective items/nodes, at most d subpaths will be defective. The same process is applied to each defective subpath, i.e., they will be divided into $n^{1/l}$ equal parts. Hence, Algorithm 3 makes at most $d \cdot n^{1/l}$ tests in each stage, $n^{1/l}$ for each defective path. On the other hand, since each defective path is divided into $n^{1/l}$ equal parts in each stage, there cannot be more than l stages. In total, the number of tests is bounded by $l \cdot d \cdot n^{1/l}$. ■

We are ready to use Algorithm 3 to solve our problem when the defective items are uniformly distributed. We need to define some notions before presenting the reduction.

Definition IV.8. *A trail in a graph is a sequence of vertices*

such that each two consecutive vertices are connected via a single edge (i.e.,a walk in which all the edges are distinct). A trail is closed when its start and end vertices are the same. An Eulerian trail contains each edge of the graph exactly once. A graph is Eulerian if it contains an Eulerian trail. A graph contains a closed Eulerian trail if it is connected and all of its vertices have even degree. The degree of vertex v in a graph H is denoted by $d_H(v)$.

Let T be a spanning tree of graph G. We duplicate each edge of T to obtain a closed Eulerian graph T^2. We then convert this closed Eulerian trail to an open Eulerian trail $W = v_1, v_2, \cdots, v_{2n-2}$ by removing one of its edges arbitrarily. Note that W has $2n-3$ edges. Each vertex $v \in G$ appears in trail W at least once, and at most $d_T(v)$ times. We treat multiple appearances of the same vertex in W as different vertices. This way W can be seen as a line graph P_{2n-2}. Thus, one can run Algorithm 3 on W. Since every node of the original graph G appears at least once in W, each defective node will be identified at least once. The performance of this multi-stage algorithm is given in the following lemma.

Lemma IV.9. *The number of stages and expected number of tests of Algorithm 3 applied on W is at most l and $2d \cdot l \cdot (2n)^{1/l}$, respectively.*

Proof: Using Lemma IV.7, we know that the number of stages is at most l, and it remains the same in our reduction. We also know that the number of tests is at most the number of defective items times $l \cdot (2n-2)^{1/l}$. Although the number of defective items in graph G is at most d, this number can be potentially higher in W due to multiple appearances of nodes. Nevertheless, we can bound the expected number of appearances of defective items in W as follows. The number of appearances of each node in W is at most its degree in the spanning tree T. Hence, the expected number of appearances of defective items in W is at most $\Sigma_{v \in T} d_T(v) \Pr[v \in \mathcal{D}]$. The set of defective nodes $\mathcal{D}$ is also chosen uniformly at random (among all sets of at most d nodes in the graph), so each node is defective with probability at most d/n. As a result, the expected number of defective nodes' appearances in W is at most $(d/n)\Sigma_{v \in T} d_T(v) = (d/n)(2n-2) < 2d$. Therefore, the expected number of tests is at most $2d \cdot l \cdot (2n-2)^{1/l}$. ∎

Note that the number of tests matches (up to a constant) the information theoretic lower bound for $l = \log(n)$ stages.

V. Conclusion

In this paper, we considered the problem of group testing with graph constraints. We devised efficient algorithms for adaptive and multi-stage settings. An interesting future direction is to address general kinds of noise for this line of work.

References

[1] R. Dorfman, "The detection of defective members of large populations," *Annals of Mathematical Statistics*, vol. 14, pp. 436–440, 1943.

[2] H. Ngo and D. Du, "A survey on combinatorial group testing algorithms with applications to DNA library screening," *DIMACS*, vol. 55, pp. 171–182, 2000.

[3] A. Schliep, D. Torney, and S. Rahmann, "Group testing with DNA chips: Generating designs and decoding experiments," in *IEEE Computer Society Bioinformatics Conference*, vol. 2, pp. 84–91, 2003.

[4] A. Macula, "Probabilistic nonadaptive group testing in the presence of errors and DNA library screening," *Annals of Combinatorics*, vol. 3, no. 1, pp. 61–69, 1999.

[5] A. Macula, "Probabilistic nonadaptive and two-stage group testing with relatively small pools and DNA library screening," *Journal of Combinatorial Optimization*, vol. 2, pp. 385–397, 1999.

[6] Y. Cheng and D. Z. Du, "New constructions of one- and two-stage pooling designs," *Journal of Computational Biology*, vol. 15, no. 2, pp. 195–205, 2008.

[7] A. Blass and Y. Gurevich, "Pairwise testing," *Bulletin of the European Association for Theoretical Computer Science*, vol. 78, pp. 100–132, 2002.

[8] R. Clifford, K. Efremenko, E. Porat, and A. Rothschild, "k-mismatch with don't cares," in *Proceedings of the 15th European Symposium on Algorithm (ESA)*, vol. 4698, pp. 151–162, 2007.

[9] G. Cormode and S. Muthukrishnan, "What's hot and what's not: tracking most frequent items dynamically," *ACM Transactions on Database Systems*, vol. 30, no. 1, pp. 249–278, 2005.

[10] D. Z. Du and F. Hwang, *Pooling Designs and Nonadaptive Group Testing*. World Scientific Series on Applied Mathematics, 2006.

[11] C. M. Jones and A. A. Zhigljavsky, "Comparison of costs for multi-stage group testing methods in the pharmaceutical industry," *Communications in Statistics - Theory and Methods*, vol. 30, no. 10, pp. 2189–2209, 2001.

[12] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Network tomography: recent developments," *Statistical Science*, vol. 19, pp. 499–517, 2004.

[13] N. J. A. Harvey, M. Patrascu, Y. Wen, S. Yekhanin, and V. W. S. Chan, "Non-adaptive fault diagnosis for all-optical networks via combinatorial group testing on graphs," in *INFOCOM*, pp. 697–705, 2007.

[14] M. Cheraghchi, A. Karbasi, S. Mohajer, and V. Saligrama, "Graph-constrained group testing," *IEEE Transactions on Information Theory*, vol. 58, no. 1, pp. 248–262, 2012.

[15] W. Xu, E. Mallada, and A. Tang, "Compressive sensing over graphs," in *INFOCOM*, pp. 2087–2095, 2011.

[16] J. Tapolcai, P.-H. Ho, B. Wu, and L. Rnyai, "A novel approach for failure localization in all-optical mesh networks," *IEEE/ACM Transactions on Networking*, vol. 19, pp. 275–285, 2011.

[17] T. Tosic, N. Thomos, and P. Frossard, "Distributed sensor failure detection in sensor networks," *CoRR*, 2011.

[18] A. Karbasi and M. Zadimoghaddam, "Compression with Graphical Constraints: An Interactive Browser," in *ISIT*, 2011.

[19] A. Dyachkov, V. Rykov, and A. Rashad, "Bound of the length of disjunct codes," *Problems of Information Transmission*, vol. 18, pp. 7–13, 1982.

[20] A. Dyachkov and V. Rykov, "A survey of superimposed code theory," *Problem of Control and Information Theory*, vol. 12, pp. 1–13, 1983.

[21] P. Erdos, P. Frankl, and Z. Furedi, "Families of finite sets in which no set is covered by the union of two others," *J. Combin. Theory Ser. A*, vol. 33, pp. 158–166, 1982.

[22] P. Erdos, P. Frankl, and Z. Furedi, "Families of finite sets in which no set is covered by the union of r others," *Israel J. Math*, vol. 51, pp. 79–89, 1985.

[23] F. Hwang and V. Sos, "Non-adaptive hypergeometric group testing," *Studia Sci. Math.*, vol. 22, pp. 257–263, 1985.

[24] A. Dyachkov and V. V. Rykov, "Bounds for the length of disjunctive codes," in *Problem of Information Transmission*, pp. 7–13, 1982.

[25] F. K. Hwang, "A method for detecting all defective members in a population by group testing," in *Journal of the American Statistical Association*, pp. 605–608, 1972.

[26] A. D. Bonis, L. Gasieniec, and U. Vaccaro, "Generalized framework for selectors with applications in optimal group testing," in *ICALP*, pp. 81–96, 2003.

[27] J. Schlaghoff and E. Triesch, "Improved results for competitive group testing," *Combinatorics, Probability & Computing*, vol. 14, no. 1-2, pp. 191–202, 2005.

[28] A. D. Bonis, L. Gasieniec, and U. Vaccaro, "Optimal two-stage algorithms for group testing problems," *SIAM J. Comput.*, vol. 34, no. 5, pp. 1253–1270, 2005.

[29] M. Mézard and C. Toninelli, "Group testing with random pools: Optimal two-stage algorithms," *IEEE Transactions on Information Theory*, vol. 57, no. 3, pp. 1736–1745, 2011.