Searching for and Avoiding Hidden Sets Using Queries with Local Feedback

Tomasz Jurdzinski[1], **Dariusz R. Kowalski**[2]

[1]University of Wroclaw, Poland
[2]Augusta University, USA
tju@cs.uni.wroc.pl, dkowalski@augusta.edu

Abstract

Discovering elements of a hidden set, also known as Group Testing (GT), is a well-established area in which one party tries to discover elements hidden by the other party by asking queries and analyzing feedback. The feedback is a function of the intersection of the query with the hidden set - in our case, it is a classical double-threshold function, which returns i if the intersection is a singleton i and "null" otherwise (i.e., when the intersection is empty or of size at least 2). In this work, we enhance GT by two features. First, we introduce a local feedback framework to this problem: each hidden element is an "autonomous" element and can analyze feedback itself, but only for the queries to which it belongs. The goal is to design a deterministic non-adaptive sequence of queries that enables each non-hidden element to learn about all other hidden elements. We show that, surprisingly, this task requires substantially more queries than the classic group testing – by proving a super-cubic (in terms of the number of hidden elements) lower bound and by constructing a specific query sequence of slightly longer length. Such a query system is also an extension of a well-known superimposed code, in a way that the decoding can be done only by the owners of the codewords. Second, we extend the results to the model where elements may belong to certain clusters and retrieving them could be done only via queries avoiding elements from "interfering" clusters. The main challenge is in not knowing which interfering clusters are non-empty (and thus, need to be avoided) and how to speed up the retrieval process by asking queries across many clusters. Our algorithms can be generalized to other feedback functions, to adversarial/stochastic fault-prone scenarios, implemented in a distributed setting and applied to the information theory and codes.

1 Introduction

In the Group Testing (GT) research field, introduced by Dorfman in 1943 (Dorfman 1943), the goal is to retrieve all elements of an unknown set K, containing k elements of some large universe of size n, by asking queries and analyzing answers (so called, feedback vector). Queries are subsets of the universe, for instance, containing elements satisfying some pre-defined search criteria, e.g., a query may ask for elements having a specific subset of features. Originally GT was applied for identifying infected individuals in large populations using pooled tests (Dorfman 1943), and it has also been very vibrant recently during and after the COVID-19 pandemic (Augenblick et al. 2020; Mallapaty et al. 2020; Sinnott-Armstrong, Klein, and Hickey 2020). Although group testing is an area containing relatively simply-formulated combinatorial problems, it has many applications in other fields. Especially in problems where there is an unknown sparse sub-structure that could improve the performance of ML/AI algorithms, e.g., a small subset of features (model parameters) that determines decisions with high certainty. One could discover such subset by reverse-engineering the decision vector, in a similar way that the hidden set is discovered by analyzing feedback vector to queries. Specific applications include: simplifying multi-label classifiers (Ubaru et al. 2020), approximating the nearest neighbor (Engels, Coleman, and Shrivastava 2021), and accelerating forward pass of a deep neural network (Liang and Zou 2021). The related superimposed codes are used for dimensional reduction and error correction in ML and (online) decision making, cf. (Hajiaghayi et al. 2024).

In adversarial search, information retrieval and databases, GT has been applied to extraction and maintenance of the most frequent elements in data streaming (Cormode et al. 2003; Cormode and Muthukrishnan 2005; Cormode and Hadjieleftheriou 2008; Yu et al. 2004; Kowalski and Pajak 2022a), construction of signature files (Indyk 1997), (distributed) reconstruction of noisy pooled data (Hahn-Klimroth and Kaaser 2022), supporting top-k join queries in relational databases (Ilyas, Aref, and Elmagarmid 2004), indexing (Tomasic and Garcia-Molina 1993), and dimensionality reduction (Aggarwal 2001; Ravi Kanth, Agrawal, and Singh 1998; Shen, Zhou, and Zhou 2007). Group testing is also closely related with coding and information theory (Kautz and Singleton 1964; Porat and Rothschild 2011a; Cheraghchi and Ribeiro 2019). More information, applications and links could be found in the seminal book (Du, Hwang, and Hwang 2000) and recent literature (Klonowski, Kowalski, and Pajak 2022; Kowalski and Pajak 2022b).

1.1 The setting of this work

We consider one of the classical query's feedback models: if the intersection of the query with the hidden set is a single element, the id of this element is returned as the query's feedback (we say that this element is selected, or revealed);

otherwise an arbitrary null value is returned. We focus on deterministic non-adaptive solutions, i.e., when a sequence of queries is determined by GT algorithm prior the hidden set is selected by some adversary. The goal of the algorithm, often called a *selector*, is to reveal every element of the hidden set, using as small number of queries as possible. It is already known that GT can be solved in this feedback model by using $O(k^2 \log(n/k))$ queries, see e.g., (De Bonis, Gasieniec, and Vaccaro 2003), and an explicit polynomial-time construction of length $O(k^2 \log n)$ exists (Porat and Rothschild 2011b). The best known lower bound on the number of queries is $\Omega(\min\{k^2 \log n/\log k, n\})$ (Clementi, Monti, and Silvestri 2001). [1]

This work, inspired by recent papers by (Jurdzinski et al. 2018, 2020) on applications in collision-avoiding communication, studies non-adaptive GT enhanced by two properties.

First, we would like to assure *local learning*, that is, each element in the near universe (but not necessarily in the hidden set) needs to reveal the whole hidden set. However, it can do so only by being included in queries that simultaneously select hidden elements. Note that these two goals, selection and locality, are not necessarily aligned: fast selection would rather require smaller queries, to avoid intersections of size bigger than 1 with the hidden set; quick local learning, on the other hand, would like to place as many near elements to each query as possible – so that many of them could learn, in case the intersection with the hidden set is a singleton. Thus, efficient construction of selector with locality is a challenge.

Second, we go further to two-dimensional space[2] and ask the question of efficient local learning within any single one-dimensional sub-space, simultaneously *avoiding* any elements from a union of any other ℓ one-dimensional spaces (here, called clusters). Intuitively, the presence of such elements in queries may cause negative inference to the learning process, e.g., in testing biological/chemical samples, in shared-medium communication (learning "free channels"), or in sampling from dependent distributions. For reasons similar to the local learning, stated above, avoiding any subset of other clusters may not be aligned with simultaneous local learning in the chosen cluster (the choice of which cluster/sub-space to learn from and which to avoid could be made arbitrarily by an adversary); thus, its efficient construction is challenging.

Apart from providing efficient polynomial-time constructions and almost matching lower bounds on the number of queries in the two considered extended types of group testing, we consider several extensions, remarks and open directions – see details in Sections 2.2, 2.4, 5 and 6.

Paper overview. Section 2 formalizes the model, the problem and describes our contribution. Sections 3 and 4 provide technical details regarding polynomial constructions and lower bounds for local and avoiding selectors. Interesting extensions and corresponding applications are given in Section 5. Section 6 discusses the results and potential future work. Some discussion on the model features and motivation, related work, and some proofs are deferred to the full version of the paper due to space limitation.

[1] In this paper we use a common asymptotic notation $O(f), \Omega(f), \Theta(f)$ to denote that a considered formula is, respectively, asymptotically upper bounded, lower bounded or equivalent to the function f used inside this notation, up to a constant factor.

[2] The first coordinate could be viewed as individual id of an element, while the other – as its cluster id.

2 Model, Problems, Results, Preliminaries

2.1 Model and selectors' construction problems

Consider the universe of all elements – set $\mathcal{N} = [n] = \{0, \ldots, n-1\}$. Throughout the paper, we will associate an element with its identifier. Let K, with $|K| \leq k$, denote a hidden subset of $\mathcal{N}$, chosen arbitrarily by an adversary. It is typically assumed in the literature that k is substantially smaller than n, denoted $k \ll n$. Let $\mathcal{Q} = \langle Q_1, \ldots, Q_m \rangle$ be a fixed sequence of m queries generated by a given non-adaptive algorithm, where each query corresponds to a subset of the universe $\mathcal{N}$. A non-adaptive algorithm is also colloquially called in the literature a (n,k)-*selector* or a (n,k)-*strong selector* (as the resulting sequence of queries is in fact a fixed mathematical structure), while m is called the *length* or the *size* of the selector.

A feedback function $\mathcal{F}$ is a function from subsets of $\mathcal{N}$ into an arbitrary domain. A function is applied to $K \cap Q_i$ and the result is called a feedback for query Q_i. In our work, we assume a classic feedback $\mathcal{F}$ that returns $null$ in $|K \cap Q_i| \neq 1$ and returns x if $K \cap Q_i = \{x\}$. In the latter case, we say that the query *selects* element x from the hidden set. The goal of the selector is to have every element of the hidden set selected by some query, for any possible hidden set.

In what follows, we present an extension of the classic selectors, defined above, by two additional properties that we require from selectors: locality and avoidance. They were introduced recently by (Jurdzinski et al. 2018) in the context of collision-avoiding schedules in network communication.[3]

Local Selectors. A sequence $\mathcal{Q} = (Q_1, \ldots, Q_m)$ of sets over $[n]$ satisfies *Local Selection* property (or LocS property, for short) for a set $K \subseteq [n]$, if for any $x \in K$ and any $y \notin K$, there is a set $Q_i \in \mathcal{Q}$ such that $K \cap Q_i = \{x\}$ and $y \in Q_i$. One may interpret the above definition as a non-hidden element y being a "witness" of a selection of a hidden element x, or alternatively, a non-hidden element y learning that x is in the hidden set K but only if y itself is in the current query Q_i. (Note that although $y \in Q_i$, y is not in $Q_i \cap K$ as it is not a hidden element.)

A sequence $\mathcal{Q} = (Q_1, \ldots, Q_m)$ is an (n,k)-*Local-Selector* (or (n,k)-*LocS*, for short) of length m if, for every subset $K \subseteq [n]$ of size k, the family $\mathcal{Q}$ satisfies the LocS property for K. The following *non-constructive upper bound* was proved using a probabilistic argument.

Lemma 1 (non-constructive (Jurdzinski et al. 2018))**.** *For each positive integers n and $k \leq n$, there exists an (n,k)-LocS of length $O(k^3 \log n)$.*

[3] (Jurdzinski et al. 2018) called these extended selectors (n,k)-witnessed strong selector (or (n,k)-wss) for our (n,k)-LocS, and (n,k)-witnessed clusters aware strong selector (or (n,k,ℓ)-wcass) for our (n,k,ℓ)-LocAS. In our work, we propose a unified system of names refined to the GT area: locality and avoidance.

One can generalize the notion of (n, k, ℓ)-LocS even further – to the situation that LocS property must hold only in sub-spaces (with fixed second coordinate, and called clusters) of a two-dimensional space. Even more, that this property holds using only queries that avoid a given set of ℓ other clusters. Intuitively, having elements from other clusters in the query may negatively influence, or even clash, the learning process within a given cluster – hence, the goal is to do local learning of k hidden elements within the cluster and simultaneously avoiding the other ℓ "bad" clusters. More formal definitions follow.

Local Avoiding Selectors. We say that a set $Q \subseteq [n]^2$ is *free* of $\phi \in [n]$ if for all $(x, \phi') \in Q$ we have $\phi' \neq \phi$. A set Q is free of a given set $C \subseteq [n]$ if Q is free of each element $\phi \in C$. We call a set of pairs $K \times \{\phi\}$ a *slice*, set $[n] \times \{\phi\}$ a *cluster*, and ϕ a *cluster number*. Let $K \subseteq [n] \times \{\phi\}$ be a set of elements that we would like to select locally in the cluster ϕ, and $L \subseteq [n] \setminus \{\phi\}$ be a set of cluster numbers in conflict with the cluster ϕ, i.e., elements of these clusters we want to avoid when locally selecting elements in K. Then, a sequence $\mathcal{Q} = (Q_1, ..., Q_m)$ of subsets of $[n]^2$ satisfies *Local Avoiding Selection property* (LocAS property, for short) for K with respect to L if for each $x \in K$ and each $y \notin K$ from cluster ϕ (i.e., $y \in [n] \times \{\phi\}$) there is a set Q_i such that $Q_i \cap K = \{x\}$, $y \in Q_i$ and $Q_i \cap ([n] \times L) = \emptyset$ (i.e., Q_i is free of clusters from the set L of cluster names). In less formal words, LocAS property requires that for each $x \in K$ and each $y \notin K$ such that $y \in [n] \times \{\phi\}$ is in the same cluster as x: x is selected by some Q_i, y learns about $x \in K$ (i.e., $y \in Q_i$), and Q_i is free of the clusters from L (i.e., elements from clusters L do not interfere learning by y about x).

A sequence $\mathcal{Q} = (Q_1, ..., Q_m)$ of subsets of $[n]^2$ is an (n, k, ℓ)-*Local-Avoiding-Selector* (or (n, k, ℓ)-LocAS, for short) if for any set $L \subseteq [n]$ of size ℓ, any $\phi \notin L$ and any set $K \subseteq [n] \times \{\phi\}$ of size k, selector $\mathcal{Q}$ satisfies LocAS property for K with respect to L. The following *non-constructive upper bound* was proved using a probabilistic argument.

Lemma 2 (non-constructive (Jurdzinski et al. 2018))**.** *For each natural n, and $k, \ell \leq n$, there exists an (n, k, ℓ)-LocAS of length $O((k + \ell)\ell k^2 \log n)$.*

2.2 Technical contribution

We present the first *polynomial-time constructions* of efficient (n, k)-LocS of length

$$O(k^3 \log^2 n(\log_k n + (\log \log n / \log k)^2)) ,$$

see Section 3 and Theorem 1, and (n, k, ℓ)-LocAS of length

$$O((k + \ell)\ell k^2 \text{ polylog}(n))$$

for some polylogarithmic function $\text{polylog}(n)$, see Section 4 and Theorem 3 for detail formula.[4] Both our results give almost the same formulas as the best known *existential results* – see the cited Lemmas 1 and 2, respectively, and the overhead is only a small degree polylogarithm. Throughout the paper, by "existential results" we mean that they only proved, by using a probabilistic argument, that selectors of similar length exist, without showing how to construct them efficiently (ie., in polynomial time), see (Jurdzinski et al. 2018, 2020). Ours are the *first efficient constructions* of such selectors.

Although one could obtain (n, k)-LocS by taking a (n, k, ℓ)-LocAS for $\ell = 1$, our LocS construction is more efficient than such transformation and thus beneficial on its own – details are deferred to the full version of the paper.

We complement our constructive results by proving *almost-matching lower bounds*, correspondingly: $\Omega(\min\{k^3 \log_k n, kn\})$ on the length of any (n, k)-LocS, see Theorem 2 in Sec. 3, and $\Omega(\ell \cdot \min\{k^3 \log_k n, kn\})$ on the length of any (n, k, ℓ)-LocAS, see Theorem 4 in Sec. 4.

Extensions of technical results to fault-prone testing environments, different feedback functions (such as classic beeping) and applications to codes and graph testing, are given in Sec. 5 and in the full version of the paper. Simpler but much less efficient constructive approach – by taking a product of classic selectors – is described, as a warm-up, in Sec. 2.4.

Note on practicality of our results Typical applications of GT and related superimposed codes (SC) assume k to be a small parameter or even a constant with respect to the universe size n. Thus, an additional factor k or $k \cdot \ell$, comparing to the classic GT and SC setting, is not very harmful. The constants in our constructions, hidden under asymptotic notation, are small ($c \leq 8$), and the values of exponential formulas used in the constructions are bounded by some polynomial in n, which yields that operations are on numbers of logarithmic representations.

2.3 Preliminaries and notation

A sequence $\mathcal{Q} = (Q_1, ..., Q_m)$ of queries could also be, equivalently, viewed as, and represented by, a 0-1 matrix, in which rows represent elements of the universe, columns represent queries, and an intersection of a row with a column stores value 1 iff the element corresponding to the row belongs to the query corresponding to the column. Such matrix correspond to a *matrix of some specific code*, see Section 5.3 for relation between non-adaptive GT and codes.

We say that $\mathcal{Q}$ is *constructible in polynomial time* if there exists a polynomial-time algorithm, that given parameters n, k, ℓ (in case of LocS – only parameters n, k), outputs an appropriate sequence of queries satisfying the requirements. In our case, the requirements are those defining LocS or LocAS. W.l.o.g., in order to avoid rounding in the presentation, we assume that n and other crucial parameters used in this work are powers of 2.

2.4 First (not so efficient) approach – via a product of (classic) selectors

One could be tempted to construct (n, k)-LocS or (n, k, ℓ)-LocAS by taking a product of two classic (n, k)-selectors, or two classic (n, k)-selectors and one (n, ℓ)-selectors, respectively. This way, each of the three properties: selection, locality (witnessing) and avoidance, would be as-

[4] By a polylogarithmic $\text{polylog}(x, \ldots)$ over parameters $x, \ldots$ we mean a function that could be upper bounded by $O((\log x)^c \cdot \ldots)$, for some absolute constant $c > 0$.

sured "independently" by the means of a different selector in such product. Note, however, that such constructions would not be efficient, and would result in linear or even quadratic overhead comparing to our constructions provided later in Sections 3 and 4, due to the super-quadratic lower bound on the length of a classic (n,k)-selector, $\Omega(\min\{k^2 \log n/\log k, n\})$ (Clementi, Monti, and Silvestri 2001). Hence, the product will be of order at least k^4 for LocS and $k^4\ell^2$ for LocAS, which are asymptotically much larger than the corresponding lower bounds proved later.

To convey more details, first we need to formally define the meaning of a product of two selectors. (Here we only focus on the first paradigm, the locality, since as we argued earlier this approach is not very efficient anyways.) It could be seen as a product with "or" operation on pairs of columns. Consider a column Q_i of the first selector and a column Q'_j of the second selector that has length m'. In the product, the column $(i-1)m' + j$ is defined as follows: the value stored in a row $v \in [n]$ is 1 if $Q_i[v] = 1$ or $Q'_j[v] = 1$, otherwise it is 0. Having a product of two (n,k)-selectors, consider any set K of size k, any element $v \in K$ and any $w \in [n] \setminus K$. Consider a column Q_i in the first selector in which v is selected from set K – its existence is guaranteed by the definition of (n,k)-selector. If $w \in Q_i$ we are done – there is a selection of v witnessed by w. Suppose then that $w \notin Q_i$. Consider set $K' = (K \setminus \{v\}) \cup \{w\}$. It has k elements, thus the second (n,k)-selector has a query Q'_j in which w is selected from K'. It means that in the product of the two selectors, in the query/column $(i-1)m' + j$, row v has value 1 (by the fact that $Q_i[v] = 1$ and definition of selectors' product), row w has value 1, while all elements in $K \setminus \{v\} = K' \setminus \{w\}$ have value 0. Thus, this column guarantees selection of v from K with w being a "witness".

3 Local Selectors

The following polynomial-time algorithm produces an (n,k)-LocS of length polylogarithmically close to the existential results from (Jurdzinski et al. 2018). (Recall that existential result means that only existence of such selectors was shown, without any efficient construction.) Later, we also show that it is actually polylogarithmically close to the absolute lower bound on the length of any (n,k)-LocS, c.f., Theorem 2. This construction is parameterized by some (suitable) constant $c \in (2,4]$.

1. Let $d = \lceil \log_k n \rceil$ and let $q = c \cdot k \cdot d$, for some constant $2 < c \leq 4$, be a prime number such that $q^{d+1} = (c \cdot k \cdot d)^{d+1} \geq n$. We argue that such constant c exists. $\lfloor (4kd)^{d+1} \rfloor \geq (4kd)^{d+1} - 1 \geq 4n$ and between two integers $\lfloor \frac{1}{2} \lfloor (4kd)^{d+1} \rfloor \rfloor$ and $\lfloor (4kd)^{d+1} \rfloor$ there is at least one prime number (by the well known distribution property of prime numbers). Observe that $\lfloor (4kd)^{d+1} \rfloor \leq (4kd)^{d+1}$ while $\lfloor \frac{1}{2} \lfloor (4kd)^{d+1} \rfloor \rfloor$ can be represented as $(c'kd)^{d+1}$ for some constant $2 < c' < 4$, since $d + 1 \geq 2$. Hence, the existing prime number is equivalent to $(ckd)^{d+1}$ for some $c \in [c', 4] \subseteq (2,4]$. Let q_1 be the qth prime number.
2. Consider all polynomials P_i of degree d over field $[q]$, for $1 \leq i \leq q^{d+1}$. Notice that there are q^{d+1} of such different polynomials.
3. Create a matrix M' of size $q^{d+1} \times q$. Each row i contains subsequent values $P_i(x)$ of polynomial P_i for arguments $x = 0, 1, \ldots, q-1$, where x is the column number (columns of M' are numbered from 0 to $q-1$). A matrix M'' is created from M' as follows: each value $y = P_i(x) \in [q]$ is represented and padded in q_1^2 consecutive columns of 0s and 1s, where value 1 is on positions $y_1 \cdot z$, for any prime number $z \leq q_1$ and for y_1 defined as the yth prime number; all other positions are filled with value 0. We call these columns an *x-segment*. Notice that each row of M'' has $q \cdot q_1^2$ columns (q_1^2 columns in each segment, where segments correspond to different arguments x), thus M'' has size $q^{d+1} \times q \cdot q_1^2$; we number the columns from 1 to $q \cdot q_1^2$, where the first segment (corresponding to argument $x = 0$) consists of columns $1, \ldots, q_1^2$, the second segment (corresponding to the argument $x = 1$) has columns $q_1^2 + 1, \ldots, 2q_1^2$, and so on.
4. Remove $q^{d+1} - n$ arbitrary rows from matrix M'', creating matrix M with exactly n remaining rows. Recall that we earlier guaranteed that q^{d+1} is at least n.
5. Each column of matrix M corresponds to one qyery set Q_i of an (n,k)-LocS $\{Q_i\}_{i=1}^{q \cdot q_1^2}$ over the set of n elements, where an element corresponds to a row and it belongs to Q_i iff there is value 1 in the intersection of the corresponding row with ith column.

Theorem 1. *There is a suitable constant $2 < c \leq 4$ such that the polynomially constructed family of queries $\{Q_i\}_{i=1}^{q \cdot q_1^2}$ is an (n,k)-LocS of length $q \cdot q_1^2 = O(k^3 \log^2 n \cdot (\log_k n + (\log\log n/\log k)^2))$.*

The next lower bound almost matches Theorem 1.

Theorem 2. *Every (n,k)-LocS has length $\Omega(\min\{k^3 \log_k n, kn\})$.*

Both theorems' proofs are deferred to the full version of the paper.

4 Local Avoiding Selectors

Suppose that we are given a set $X \subseteq [n]^2$ of size at most $k \cdot \ell$ and such that it consists of at most ℓ slices of size at most k each. The (n,k)-LocS from Section 3 guarantees that all other agents learn set X in $O((k\ell)^3 \operatorname{polylog}(n))$ rounds. Is faster learning possible if we require that only other agents in each slice's cluster learn the slice? Even more, if we additionally would like to avoid clashes between in such learning rounds from other slices of X?

The following polynomial-time algorithm produces an (n,k,ℓ)-LocAS of length polylogarithmically close to the existential result – see Figure 1 for an illustration. (Recall that by existential result we mean no construction is known, only a proof of existence.) The construction is parameterized by a (suitable) constant $c \in (4,8]$.

The crucial point of the construction is in item (5). The base for item (5) is already created $n^2 \times q$ matrix M'. The entries in a row (i,ϕ) of this matrix are pairs of values, denoted $(i^\star, \phi^\star) \in [q_k] \times [q_\ell]$, defined in item (4) as values of

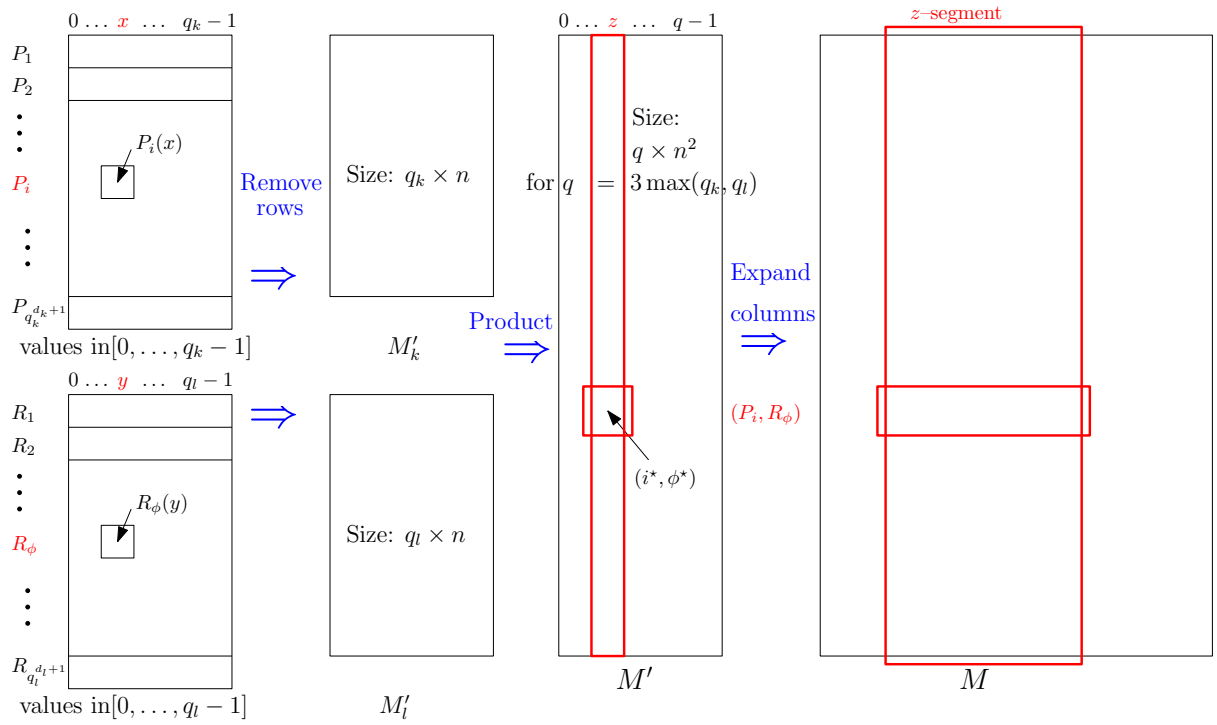

Figure 1: An illustration of consecutive matrices used in the algorithm building (n, k, l)-LocAS. On the figure, $(i^\star, \phi^\star) = (P_i(z \bmod q_k), Q_\phi(z \bmod q_l))$, the matrix M is obtained by the replacement of each element of M' with a binary sequence of the length $(q'_k)^2 q'_l$ as described in the item 5 of the description of the algorithm before the theorem. "Remove rows" between the left and the second left matrices corresponds to point (3) of the construction. "Product" in the middle of the picture corresponds to item 4 of the construction. Finally, "Expand columns" between the second right and the right matrices corresponds to point (5) of the construction.

some polynomials P_i, R_ϕ. In item (5), we replace each such pair of values by a 0-1 vector of length $(q'_k)^2 q'_\ell$ as follows. We fix two prime numbers, $p_{i^\star}, p_{\phi^\star}$, one corresponding to the first value, $i^\star$, and the other – to the second value, $\phi^\star$. Then, for any prime number p' of prime order $2, 4, \dots, 2q_k$, we put value 1 in position $p_{i^\star} \cdot p' \cdot p_{\phi^\star}$ of the vector, and 0s elsewhere. Intuitively: 1s in the multiplicities of $p_{i^\star}$ allow to eventually avoid other rows from the hidden set in the same cluster and thus assure selection; 1s in the multiplicities of $p_{\phi^\star}$ allow to avoid other ℓ clusters; and multiplying by range of p' allows to assure locality (ie., long enough feedback, so the selection with avoidance could eventually occur). The details follow:

1. Let $d_k = \lceil \log_k n \rceil$, and $q_k = c \cdot k d_k$ be a prime number such that $q_k^{d_k+1} = (c \cdot k d_k)^{d_k+1} \geq n$, for some constant $4 < c \leq 8$. Note that such constant c exists, because $(k d_k)^{d_k+1} > n$ and between two integers: $\lceil (4kd_k)^{d_k+1} \rceil$ and its double $2\lceil (4kd_k)^{d_k+1} \rceil \leq \lceil (8kd_k)^{d_k+1} \rceil$, there is at least one prime number. Analogously, we define d_ℓ and q_ℓ.
2. Consider all polynomials P_i of degree d_k over field $[q_k]$, for $1 \leq i \leq q_k^{d_k+1}$. Notice that there are $q_k^{d_k+1}$ of such different polynomials. Analogously, consider all polynomials R_ϕ of degree d_ℓ over field $[q_\ell]$, for $1 \leq \phi \leq q_\ell^{d_\ell+1}$. Notice that there are $q_\ell^{d_\ell+1}$ of such different polynomials.

   The defined two sets of polynomials are used in next steps. The role of the first set, P_i, is to assure selection of elements in hidden set, while the role of the second set, R_ϕ, is to assure avoidance of elements in different clusters. Parameters with sub-index k correspond to components assuring local selection, while those with sub-index ℓ – to components assuring avoidance of other clusters.
3. Create a matrix M'_k of size $q_k^{d_k+1} \times q_k$. Each row i contains subsequent values $P_i(z)$ of polynomial P_i for arguments $z = 0, 1, \dots, q_k - 1$, where z is the column number (columns of M'_k are numbered from 0 to $q_k - 1$). We trim matrix M'_k to n rows by removing $q_k^{d_k+1} - n$ arbitrary rows.

   Analogously, we create a matrix M'_ℓ of size $q_\ell^{d_\ell+1} \times q_\ell$, in which each row ϕ contains subsequent values $R_\phi(z)$ of polynomial R_ϕ for arguments $z = 0, 1, \dots, q_\ell - 1$, where z is the column number (columns of M'_ℓ are numbered from 0 to $q_\ell - 1$). We trim matrix M'_ℓ to n rows by removing $q_\ell^{d_\ell+1} - n$ arbitrary rows.
4. Matrix M' is created from M'_k, M'_ℓ as follows: there are n^2 rows and $q = 3\max\{q_k, q_\ell\}$ columns. In each row (i, ϕ), corresponding to the pair of polynomials P_i, R_ϕ,

and each column $z \in [q]$, we put in the intersection a pair of values $(P_i(z \mod q_k), R_\phi(z \mod q_\ell))$.

5. Matrix M is created from M' as follows: each pair of values $(i^\star, \phi^\star) = (P_i(z \mod q_k), R_\phi(z \mod q_\ell))$ in column z is represented and padded in $(q'_k)^2 q'_\ell$ consecutive columns of 0s and 1s as follows. Let q'_k, q'_ℓ be the prime numbers of order $2q_k$ and $2q_\ell + 1$, respectively (i.e., the $(2q_k)$-th prime number in the order of all prime numbers, and $(2q_\ell + 1)$-st prime number in the order of all prime numbers). Let $p_{i^\star}$ be the prime number of order $2i^\star$; let p' be any prime number of order $2, 4, 6, \ldots, 2q_k$; let $p_{\phi^\star}$ be the prime number of order $2\phi^\star + 1$. We put values 1 in columns $p_{i^\star} \cdot p' \cdot p_{\phi^\star}$, and values 0 in the remaining columns. We call these columns a *z-segment*. Notice that each row of M has $q \cdot (q'_k)^2 q'_\ell$ columns ($(q'_k)^2 q'_\ell$ columns in each segment, where segments corresponds to different columns z of M'). Thus, M is an $n^2 \times q \cdot (q'_k)^2 q'_\ell$ matrix; we number its columns from 1 to $q \cdot (q'_k)^2 q'_\ell$, where the first segment (corresponding to argument $z = 0$) consists of columns $1, \ldots, (q'_k)^2 q'_\ell$, the second segment (corresponding to the argument $z = 1$) of columns $(q'_k)^2 q'_\ell + 1, \ldots, 2(q'_k)^2 q'_\ell$, and so on.
6. Each column of matrix M corresponds to one query set Q_i of an (n, k, ℓ)-LocAS $\{Q_i\}_{i=1}^{q \cdot (q'_k)^2 q'_\ell}$ over the set of n^2 elements (corresponding to the rows of M).

Theorem 3. *The constructed $\{Q_i\}_{i=1}^{q \cdot (q'_k)^2 q'_\ell}$ is an (n, k, ℓ)-LocAS of length $O\left((k+\ell)k^2\ell \cdot \operatorname{polylog}(n, k, \ell)\right)$, where $\operatorname{polylog}(n, k, \ell) = \frac{\log^4 n}{\log^2 k \log \ell \log(k\ell)}(\log k + \log\log n)^2(\log \ell + \log\log n)$, for some suitable constant $4 < c \leq 8$. The construction itself is polynomial in n.*

The proof is deferred to the full version of the paper. Below we show a lower bound that nearly matches Theorem 3.

Theorem 4. *Every (n, k, ℓ)-LocAS has length*

$$\Omega(\ell \cdot \min\{k^3 \log_k n, kn\}) .$$

Proof. Consider (n, k, ℓ)-LocAS of length m. Consider a set $L \subseteq [n]$ of size $\ell + 1$. For any $i \in [n]$, consider a set of pairs $\{(i, j) : i \in [n]\}$, that is, a set of rows in the (n, k, ℓ)-LocAS corresponding/labeled to/by these pairs. The number of columns that

- have at least one 1 in these rows (i.e., take part in the local selection of any set of size k of pairs with i in their second coordinate), and
- do not have any 1 from any row labeled by an element in $L \setminus \{i\}$ (i.e., avoids other elements in L on the second coordinate)

is at least $\Omega(\min\{k^3 \log_k n, kn\})$, by the fact that they must be an (n, k)-LocS, to which Theorem 2 applies. Denote the set of such columns C_i.

Next, if we consider analogous set $C_{i'}$ of columns defined for an element $i' \in L$ different from L, it is disjoint with C_i, due to the second bullet in the definition of set C_i above (avoidance property). Hence, the total number of columns in the (n, k, ℓ)-LocAS is
$|L| \cdot \Omega(\min\{k^3 \log_k n, kn\}) \geq \Omega(\ell \cdot \min\{k^3 \log_k n, kn\}) .$ □

5 Extensions and Applications

This section presents interesting extensions and applications of the technical results obtained in the previous sections, with corresponding future directions. Other open problems are discussed in Section 6.

5.1 Fault-tolerance of local GT

Suppose one would like to be able to decode the hidden set correctly even if some α positions in the feedback vector would be altered by a *worst-case* adversary. More precisely, assume that the adversary could change a feedback from specific id to zero (but cannot produce/forge an id as feedback). Observe that if we use a larger constant c in the constructions, for instance, $c \geq 2 + \frac{\alpha}{kd}$, there will be always some column with correct feedback. Specifically, in the proofs of Theorems 1 and 3, the number of "witnessed" segments (and thus, also columns) for a pair i, j is

$$q - 2(k-1)d = ckd - 2(k-1)d = \left(2 + \lceil \frac{\alpha}{kd} \rceil\right) kd - 2(k-1)d,$$

which subtracted by the number of adversarially changed ones, α, is still at least 1. Enhancing constant c increases the lengths of selectors by factor $\lceil \frac{\alpha}{kd} \rceil$. Decreasing the number of queries with respect to the number of tolerated faults or generalizing to other types of adversarial/stochastic failures is an interesting open direction.

5.2 Other feedback functions

We show an example how to convert the obtained results to another popular GT feedback functions – the classic one-threshold setting in which the feedback vector is a 0-1 vector where the value on a position z is equal 1 if the intersection of the z-th query with the hidden set is non-empty, and is equal to 0 otherwise. It has also been named beeping feedback in recent literature, see e.g., (Afek et al. 2011). The method is similar to the one guarantying fault-tolerance.

If we extend parameter c to be a constant sufficiently bigger than 2, the asymptotic length and analysis of our constructions become intact. However, the number of "witnessed" columns for any pair i, j, which is $q - 2(k-1) \cdot d$, could be made bigger than d. Each such column corresponds to a different segment, and thus – to different arguments for which polynomials are evaluated. Obviously, the feedback now is only 1 in such columns, however, j could create as many equations as the number of such columns (i.e., bigger than d) in order to find the polynomial corresponding any element i in the hidden set by classic interpolation over the algebraic field $\mathcal{F}_q$ of q elements $\{0, \ldots, q-1\}$, with addition and multiplication modulo q. Now, since the number of such equations is at least $d+1$, the polynomial for i could be interpolated successfully, as its degree is at most d. Hence, each element of the hidden set could be successfully found.

Note here that the lower bounds from Sections 3 and 4 hold automatically for the beeping feedback, because the previously considered feedback function is richer than the beeping feedback.

Extending our constructions and lower bounds to other types of feedback function, considered in the literature (cf., (Klonowski, Kowalski, and Pajak 2022)) is another

interesting research direction. For some of these feedback functions, local selectors (with or without avoiding) could be potentially shorter then the ones studied in this work for most fundamental and simple feedback functions.

5.3 Local GT as codes

By definition of non-adaptive GT, one should be able to decode a hidden set from the feedback vector – recall that each position of this vector has been created by applying a given feedback function to the intersection of the query corresponding to this position (queries are designed by the "coding" algorithm) and the hidden set (an arbitrary set, fixed by the adversary). For instance, in case of the beeping feedback function, the feedback vector is computed by applying bitwise OR on the vectors of participating elements (this is the coding part). Each such vector, corresponding to an element i, is of length equal to the number of queries and has 1 in position x if and only if $i \in Q_x$, otherwise it has 0.

From the locality property of LocS and LocAS, if another element want to retrieve hidden elements, it gets the feedback locally and decodes it – more precisely, it applies bitwise AND to its own vector and the feedback vector, and follows the interpolation procedure as described in Section 5.2 with respect to the beeping feedback. We call codes with such "local vector decoding" property *local superimposed codes*. If we want to get avoidance property, corresponding to LocAS, we apply the abovementioned bitwise coding/decoding procedures and interpolations with respect to the vectors of elements with the same second coordinate, while for those with a different second coordinate – we apply AND_NOT bitwise operation.

Applying similar methodology to other feedback functions, which are symmetric Boolean functions, could result in interesting results in the area of information theory and codes, in particular, when the function is applied to the participating codewords and the feedback is decoded locally (by combining the feedback vector with the original element's vector, using another symmetric Bolean function), with avoidance of codewords from different clusters/groups (e.g., applying the bitwise AND_NOT to such elements, as described above).

5.4 Application: Testing neighborhoods in graphs

In this part we give more details on how to apply local testing with avoidance to test more complex objects, such as graphs. Assume there is a hidden graph G, in which we would like to learn up to $\ell + 1$ neighborhoods, assuming each of them is of size at most k. (Here, a neighborhood is understood as the set of neighbors of a node in a simple graph G.) Let us apply (n, k, ℓ)-LocAS to the edges (i, j) in the union of edges in the tested $\ell + 1$ neighborhoods, where the second coordinate j stands for one of the $\ell + 1$ vertices in G whose neighborhoods we test (corresponds to a cluster number in the notation used in LocAS), and the first coordinate i stands for a node in the neighborhood of j. Once we get feedback from testing the unknown $\ell + 1$ neighborhoods by (n, k, ℓ)-LocAS, we reveal these neighborhoods as follows. For each pair (i, j), we check its local feedback (i.e., the feedback for queries to which (i, j) belongs) and if there are no elements (i', j) revealed, we ignore it. Otherwise, we assign such revealed elements (i', j) as edges of the neighborhood of vertex j, that is, i' is a neighbor of j in G. Avoidance property guarantees that, despite of other ℓ neighborhoods (again, in the terminology of LocAS these are clusters) there is a "witness" (i, j) in the cluster/neighborhood j that reveals all elements in the cluster (i.e., all neighbors of vertex j). The length of (n, k, ℓ)-LocAS is $O((k + \ell)\ell k^2 \operatorname{polylog}(n))$, by Theorem 3, which for parameters $k, \ell << n$ could be much shorter than the methods of revealing the whole graph G, requiring $\Omega(nk)$ queries (by the information theory argument, see e.g., (Grebinski and Kucherov 2000)).

6 Discussion and More Open Directions

This work introduced to the GT area new concepts of local learning and avoidance of elements from different clusters. We designed polynomially constructable and nearly optimal, in terms of the number of queries, non-adaptive query systems that observe both new properties. To justify near-optimality of our constructions, we also proved the corresponding lower bounds. Apart of applications and open problems already suggested in Section 5, there are even more interesting directions to follow.

Complexity gaps. The most straightforward future direction is to improve the remaining polylogarithmic gaps on the lengths of LocS and LocAS selectors, since the constructions and corresponding lower bounds are not strictly matching, as well as analyzing their other properties, e.g., query sizes.

Multi-dimensional and structured systems. Second and main future direction regards direct generalization of the proposed concepts into multi-dimensional system, and further into more structured systems (e.g., graphs, hypergraphs, matroids, polytopes, etc.).

Adversarial settings. There are many adversarial models that could be analyzed. In case of false positives: even in case of a malicious Byzantine adversary, surprisingly, if the number of false positives is bounded by $\alpha = O(kd)$, our constructions tolerate them if we adjust constant c with $\alpha/(kd)$. Then, our constructions guarantee more than α "good" occurrences of every hidden element in the feedback (i.e., occurrences that are local and avoiding), thus if the adversary introduces a false-positive, it could repeat it at most α times and our query system recognizes it and rejects. It also holds for more benign adversaries with smaller adaptivity. If one wants to tolerate more than $O(k \log_k n)$ faults without asymptotic increase on the number of queries, the problem remains open. We suspect that it could work for less adaptive adversaries (e.g., by introducing some randomness to the model), but for Byzantine adversary an impossibility result is more likely.

Randomization. One could ask if randomization helps in local learning, especially against an adaptive adversary who may dynamically tailor the hidden set (as long as it is compatible with the obtained feedback at any time)? If the answer is yes, what is the minimum amount of randomness (entropy) needed and how it affects learning time?

Acknowledgments

This work was supported by the Polish National Science Centre project no. 2020/39/B/ST6/03288.

References

Afek, Y.; Alon, N.; Bar-Joseph, Z.; Cornejo, A.; Haeupler, B.; and Kuhn, F. 2011. Beeping a Maximal Independent Set. In Peleg, D., ed., *Distributed Computing - 25th International Symposium, DISC 2011,*, 32–50.

Aggarwal, C. C. 2001. On the effects of dimensionality reduction on high dimensional similarity search. In *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '01, 256–266. New York, NY, USA: Association for Computing Machinery. ISBN 1581133618.

Augenblick, N.; Kolstad, J. T.; Obermeyer, Z.; and Wang, A. 2020. Group testing in a pandemic: The role of frequent testing, correlated risk, and machine learning. Technical report, National Bureau of Economic Research.

Cheraghchi, M.; and Ribeiro, J. 2019. Simple codes and sparse recovery with fast decoding. In *2019 IEEE International Symposium on Information Theory (ISIT)*, 156–160. IEEE.

Clementi, A. E. F.; Monti, A.; and Silvestri, R. 2001. Selective families, superimposed codes, and broadcasting on unknown radio networks. In Kosaraju, S. R., ed., *SODA*, 709–718. ACM/SIAM. ISBN 0-89871-490-7.

Cormode, G.; and Hadjieleftheriou, M. 2008. Finding frequent items in data streams. *Proceedings of the VLDB Endowment*, 1(2): 1530–1541.

Cormode, G.; Korn, F.; Muthukrishnan, S.; and Srivastava, D. 2003. Finding hierarchical heavy hitters in data streams. In *Proceedings 2003 VLDB Conference*, 464–475. Elsevier.

Cormode, G.; and Muthukrishnan, S. 2005. What's hot and what's not: tracking most frequent items dynamically. *ACM Transactions on Database Systems (TODS)*, 30(1): 249–278.

De Bonis, A.; Gasieniec, L.; and Vaccaro, U. 2003. Generalized Framework for Selectors with Applications in Optimal Group Testing. In *Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, Eindhoven, The Netherlands, June 30 - July 4, 2003. Proceedings*, volume 2719 of *Lecture Notes in Computer Science*, 81–96. Springer.

Dorfman, R. 1943. The detection of defective members of large populations. *The Annals of Mathematical Statistics*, 14(4): 436–440.

Du, D.; Hwang, F. K.; and Hwang, F. 2000. *Combinatorial group testing and its applications*, volume 12. World Scientific.

Engels, J.; Coleman, B.; and Shrivastava, A. 2021. Practical Near Neighbor Search via Group Testing. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 9950–9962. Curran Associates, Inc.

Grebinski, V.; and Kucherov, G. 2000. Optimal Reconstruction of Graphs under the Additive Model. *Algorithmica*, 28(1): 104–124.

Hahn-Klimroth, M.; and Kaaser, D. 2022. Distributed Reconstruction of Noisy Pooled Data. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, 89–99.

Hajiaghayi, M. T.; Kowalski, D. R.; Krysta, P.; and Olkowski, J. 2024. Online Sampling and Decision Making with Low Entropy. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, 4080–4088. ijcai.org.

Ilyas, I. F.; Aref, W. G.; and Elmagarmid, A. K. 2004. Supporting top-k join queries in relational databases. *VLDB J.*, 13(3): 207–221.

Indyk, P. 1997. Deterministic Superimposed Coding with Applications to Pattern Matching. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, 127–136. IEEE Computer Society.

Jurdzinski, T.; Kowalski, D. R.; Rozanski, M.; and Stachowiak, G. 2018. Deterministic Digital Clustering of Wireless Ad Hoc Networks. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018*, 105–114.

Jurdzinski, T.; Kowalski, D. R.; Rózanski, M.; and Stachowiak, G. 2020. Token traversal in ad hoc wireless networks via implicit carrier sensing. *Theor. Comput. Sci.*, 811: 3–20.

Kautz, W.; and Singleton, R. 1964. Nonrandom binary superimposed codes. *IEEE Transactions on Information Theory*, 10(4): 363–377.

Klonowski, M.; Kowalski, D. R.; and Pajak, D. 2022. Generalized framework for Group Testing: Queries, feedbacks and adversaries. *Theor. Comput. Sci.*, 919: 18–35.

Kowalski, D. R.; and Pajak, D. 2022a. Light Agents Searching for Hot Information. In Raedt, L. D., ed., *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, 363–369. ijcai.org.

Kowalski, D. R.; and Pajak, D. 2022b. Scalable and Efficient Non-adaptive Deterministic Group Testing. In *NeurIPS*.

Liang, W.; and Zou, J. 2021. Neural group testing to accelerate deep learning. In *2021 IEEE International Symposium on Information Theory (ISIT)*, 958–963. IEEE.

Mallapaty, S.; et al. 2020. The mathematical strategy that could transform coronavirus testing. *Nature*, 583(7817): 504–505.

Porat, E.; and Rothschild, A. 2011a. Explicit Nonadaptive Combinatorial Group Testing Schemes. *IEEE Transactions on Information Theory*, 57(12): 7982–7989.

Porat, E.; and Rothschild, A. 2011b. Explicit Nonadaptive Combinatorial Group Testing Schemes. *IEEE Trans. Inf. Theory*, 57(12): 7982–7989.

Ravi Kanth, K. V.; Agrawal, D.; and Singh, A. 1998. Dimensionality reduction for similarity searching in dynamic databases. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, 166–176. New York, NY, USA: Association for Computing Machinery. ISBN 0897919955.

Shen, H. T.; Zhou, X.; and Zhou, A. 2007. An adaptive and dynamic dimensionality reduction method for high-dimensional indexing. *VLDB J.*, 16(2): 219–234.

Sinnott-Armstrong, N.; Klein, D. L.; and Hickey, B. 2020. Evaluation of group testing for SARS-CoV-2 RNA. *MedRxiv*.

Tomasic, A.; and Garcia-Molina, H. 1993. Query Processing and Inverted Indices in Shared-Nothing Document Information Retrieval Systems. *VLDB J.*, 2(3): 243–275.

Ubaru, S.; Dash, S.; Mazumdar, A.; and Gunluk, O. 2020. Multilabel Classification by Hierarchical Partitioning and Data-dependent Grouping. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 22542–22553. Curran Associates, Inc.

Yu, J. X.; Chong, Z.; Lu, H.; and Zhou, A. 2004. False positive or false negative: Mining frequent itemsets from high speed transactional data streams. In *VLDB*, volume 4, 204–215.