Scalable and Efficient Non-adaptive Deterministic Group Testing

Dariusz R. Kowalski
School of Computer and Cyber Sciences
Augusta University, USA
dkowalski@augusta.edu

Dominik Pajak
Department of Pure Mathematics
Wroclaw University of Science and Technology,
Infermedica, Poland
dominik.pajak@pwr.edu.pl

Abstract

Group Testing (GT) is about learning a (hidden) subset K, of size k, of some large domain N, of size $n \gg k$, using a sequence of queries. A result of a query provides some information about the intersection of the query with the unknown set K. The goal is to design efficient (polynomial time) and scalable (polylogarithmic number of queries per element in K) algorithms for constructing queries that allow to decode *every* hidden set K based on the results of the queries. A vast majority of the previous work focused on randomized algorithms minimizing the number of queries; however, in case of large domains N, randomization may result in a significant deviation from the expected precision of learning the set K. Others assumed unlimited computational power (existential results) or adaptiveness of queries (next query could be constructed taking into account the results of the previous queries) – the former approach is less practical due to non-efficiency, and the latter has several drawbacks including non-parallelization. To avoid all the abovementioned drawbacks, for Quantitative Group Testing (QGT) where query result is the size of its intersection with the hidden set, we present *the first* efficient and scalable *non-adaptive deterministic* algorithms for constructing queries and decoding a hidden set K from the results of the queries – these solutions *do not* use any randomization, adaptiveness or unlimited computational power.

1 Introduction

In the Group Testing (GT) field, the goal is to identify all elements of an unknown set K by asking queries. Originally GT was applied to identifying infected individuals in large populations using pooled tests [26] which regained interest during COVID-19 pandemic [2, 52, 56]. Recently GT found applications in various areas of Machine Learning including approximating the nearest neighbor [28], simplifying multi-label classifiers [58] and accelerating forward pass of a deep neural network [51].

All we initially know about set K is that $|K| \leq k$, for some known parameter $k \leq n$, and that it is a subset of some much larger set N with $|N| = n$. The answer to a query Q depends on the intersection between K and Q and equals to $\mathsf{Result}(K \cap Q)$, where Result is some result function (also called feedback function in this paper).

The sequence of queries is a correct solution to Group Testing if and only if for any two different sets K_1, K_2 (satisfying some cardinality restriction), the sequence of answers for K_1 and K_2 is different. Note that although this allows to uniquely identify the hidden set K based on the results of the queries, in some cases decoding of set K could be a hard computational problem. The objective is: for a given deterministic feedback function $\mathsf{Result}(\cdot)$, to find a fixed sequence of queries that will identify any set K, and the length of this sequence, called the query complexity, is as short as possible. In particular, we are interested in solutions that have query complexity logarithmic in n and polynomial in k.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

α	Constructive algorithms	Lower bound
1	$O(k^2 \log n)$ [54]	$\Omega\left(k^2 \frac{\log n}{\log k}\right)$ [12]
k	$O(k^2 \log n)$ [54] $\widetilde{O}(k)$ **Thm 1** (for $\alpha = k$)	$\Omega\left(k \frac{\log n}{\log k}\right)$ (folklore)
$*$	$\widetilde{O}\left(\min\left\{\left(\frac{k}{\alpha}\right)^2, \frac{n}{\alpha}\right\} + k\right)$ **(Thm 1)**	$\Omega\left(\min\left\{\left(\frac{k}{\alpha}\right)^2, \frac{n}{\alpha}\right\} + k\frac{\log n}{\log k}\right)$ **(Thm 2)**

Table 1: Asymptotic bounds on query complexity of solutions to non-adaptive deterministic QGT with $\mathcal{F}_\alpha$ feedback. By constructive upper bound we mean query construction and decoding of hidden set in time $poly(n)$. Symbol $*$ stands for any valid value of the parameter $\alpha \in \{1, \ldots, k\}$, notations $\widetilde{O}$ and $\widetilde{\Omega}$ disregard polylogarithmic factors.

The most popular classical variant, present in the literature, considers function $\mathsf{Result}(\cdot)$ that answers whether the intersection between K and Q is empty or not, c.f., [27]; it is also known under the name of *beeping*. Another popular result function returns the intersection size; this variant has also been studied under the name of *coin weighting* [4, 24] and Quantitative Group Testing (QGT) [32, 29].

In this paper we study the problem of Group Testing under a more general *capped quantitative result*, where the result (feedback) is the size of the intersection up to some parameter $\alpha \in \{1, \ldots, k\}$, and α for larger intersections. It subsumes and generalizes the two previously described classical result functions: the smallest case of $\alpha = 1$ corresponds to the classical empty/non-empty feedback (beeping), while the case $\alpha = k$ captures the (full) quantitative feedback (classical QGT). Motivation for our research is twofold: first, to better understand the intermediate settings between the two classical GT models; second, to find efficient solutions for the capped settings that could be applied to other problems (see the Appendix).

Our focus is on *non-adaptive deterministic* solutions, in which queries are fixed (i.e., the same sequence applies to all hidden sets) and allow to discover (decode) any hidden set based on α-capped results. The primary goal is to minimize the number of queries, called *query complexity*, and the second objective is to compute the queries and decode the hidden set in polynomial time. All existing polynomial construction/decoding algorithms for beeping or quantitative feedbacks use $\Theta(\min\{n, k^2 \log n\})$ queries, c.f., [54]. General constructions developed in this work employ the concept of *dispersion*, c.f., [57], and combine it appropriately with other techniques. When instantiated for a specific feedback parameter $\alpha \in \{1, \ldots, k\}$, (e.g., $\alpha = \sqrt{k}$), they shrink the gap between the upper and lower bounds on the number of efficiently constructed queries nearly exponentially: from linear to polylogarithmic gap per hidden element. Together with our lower bound, they explain why sometimes a much smaller feedback is sufficient for decoding sets with similar efficiency. We show that parameter α has also different interpretations – it inversely relates to the number of occurrences of an element in the queries and to fault-tolerance (c.f., Section 6).

1.1 Our Results (see also Table 1)

We generalize the classical beeping and QGT models of Group Testing by considering an α-*capped results*, also called an α-*capped feedback* function, for an $\alpha \in \{1, \ldots, k\}$: for a hidden set K and query Q it returns $\mathcal{F}_\alpha(Q \cap K) = \min\{|Q \cap K|, \alpha\}$. We study the query complexity of non-adaptive deterministic QGT in this general model. We focus on scalable (polylogarithmic number of queries per element) and efficient (polynomial-time) constructing/decoding algorithms, and we show that it could be achieved *if and only* if the information cap $\alpha = \widetilde{\Omega}(\sqrt{k})$. In particular, it holds for classical QGT.

Main result – Polynomial-time construction/decoding algorithms using almost optimal number of queries (Algorithms 1 and 2 in Section 4). Here almost-optimality means that the length of the constructed query sequence is only polylogarithmically longer than the shortest possible sequence. The previous best polynomial-time solution used $\Theta(\min\{n, k^2 \log n\})$ queries for all $\alpha \geq 1$ [54], and we shrink it by factor $\Theta(\min\{\alpha^2, k\} \operatorname{polylog}^{-1} n)$. To achieve this goal, we define and build new types of selectors, called (Strong) Selectors under Interference (SSuI and SuI, for short) based on the concept of (superimposed) dispersion. Although dispersion has been used before in GT, c.f., [10, 44], as well as the classical superimposed codes, our technical novelty is twofold: first, we identified additional properties of selectors (e.g., avoiding interfering set with capped intersections), that could

not be formally deducted from the definitions of previous types of selectors, but allow to improve the solutions to QGT; second, we applied careful mathematical analysis to choose specific parameters of dispersers and superimposed codes and to prove that the new selectors' properties hold.

We also generalize the concept of Round-Robin query systems, where each query is a singleton, to α-Round-Robin query systems, containing sets of size at most α. Such sequences are shorter than the simple Round-Robin, *i.e.,* have length $O((n/\alpha)\ \mathrm{polylog}\ n)$, and, unlike a simple Round-Robin singletons' structure, are challenging to construct in a way to allow correct decoding based on α-capped feedback.

Theorem 1. *There is an explicit polynomial-time algorithm constructing non-adaptive queries $Q_1, \ldots, Q_m$, for $m = O\left(\min\left\{\left(\frac{k}{\alpha}\right)^2, \frac{n}{\alpha}\right\}\ \mathrm{polylog}\ n + k\ \mathrm{polylog}\ n\right)$, that solve Group Testing under feedback $\mathcal{F}_\alpha$ with polynomial-time decoding. Moreover, every element occurs in $O(\frac{k}{\alpha}\ \mathrm{polylog}\ n)$ queries, and the decoding time is $O(m + \frac{k^2}{\alpha}\ \mathrm{polylog}\ n + k\ \mathrm{polylog}\ n)$.*

One of the consequences is that having the result function capped at $\alpha = \sqrt{k}$, we obtain similar number of queries as with the actual result (i.e., returning the size of the whole intersection, up to k), which together with our next lower bound establishes an interesting inherited property of GT. The value $\alpha = \sqrt{k}$ seems to be a sweet spot balancing two different phenomena – superimposed coding and dispersion. Enriching the beeping model (i.e., α increases starting from 1) allows to improve the superquadratic query complexity of classical superimposed coding (used for the beeping feedback), and the improvement is roughly quadratic in terms of α, due to the nature of superimposed codes. On the other hand, limiting the full quantitative model (i.e., α decreases starting from k) increases the number of queries, starting from slightly superlinear, roughly linearly in α, as the nature of dispersion is linear. Both these phenomena equalize query complexities at $\alpha = \sqrt{k}$. This is illustrated in our construction – in the beginning, the progress in learning relies on dispersion-based SuI's for small α, then with the growth of α the length of used SuIs becomes similar to the length of the code-based SSuI; it happens for $\alpha = \sqrt{k}$. The SSuI allows to discover all the remaining elements and finishes the learning process. The last fact explains why further increase of α is not needed.

Lower bound (Section 5). The almost-optimality of our algorithms from Theorem 1 is justified by proving an absolute lower bound on the length of sequences allowing to decode a hidden set based on feedback $\mathcal{F}_\alpha$ to the queries. Here by "absolute" we mean that it holds for all query systems that allow for decoding of the hidden sets based on feedback $\mathcal{F}_\alpha$, not restricted to polynomially constructed queries with polynomial decoding algorithm. Even more, some components of the lower bound are general: they hold for *any α-capped* feedback function, which will be formally defined later in Section 3. The lower bound has three components: $(k/\alpha)^2$, n/α, and $k\frac{\log \frac{n}{k}}{\log \alpha}$. For different ranges of k, different components determine the value of the lower bound. Note that these components match the corresponding components in our constructive upper bound (Main result in Theorem 1), up to a polylogarithmic factor.

Theorem 2. *Any non-adaptive algorithm solving Group Testing needs:*

- $\Omega\left(\min\left\{\left(\frac{k}{\alpha}\right)^2, \frac{n}{\alpha}\right\} + k\frac{\log \frac{n}{k}}{\log \alpha}\right)$ *queries under feedback $\mathcal{F}_\alpha$.*
- $\Omega\left(\min\left\{\left(\frac{k}{\alpha}\right)^2, \frac{n}{\alpha}\right\}\right)$ *queries under any feedback capped at α.*

Document structure. We discuss related work on various variants of GT in Section 2. In Section 3 we formally define the QGT problem and the generalized α-capped-results model. In Section 4 we present our polynomial-time query construction and decoding algorithms. Section 5 proves the lower bound. Discussion of extensions, applications and future directions is given in Section 6. Additional Appendix contains more details: Appendix A discusses other related work; Appendix B shows details of extensions and applications; Our new selector tools are constructed and analyzed in Appendix C.

2 Previous and Related Work (see also Tables 1 and 2)

Group Testing with general feedback functions have been studied in [47], where two parameters of feedback functions (cap and expressiveness) are shown to influence the necessary number of queries.

In the standard feedback model, considered in most of the Group Testing literature [27], the feedback tells whether the intersection between query Q and set K is empty or not (sometimes it is also

α	Upper bound	First work with asymptotic bound	Difference from this paper
k	$O\left(k\frac{\log n}{\log k}\right)$	[37]	existential (super-exponential construction time)
	$O\left(k\frac{\log n}{\log k}\right)$	[4]	adaptive
	$O\left(k\frac{\log n}{\log k}\right)$	[55]	randomized (only with some probability)

Table 2: Other related work on *adaptive*, *randomized* or *existential* solutions to QGT for $\alpha = k$.

called a *beeping model*). It it is a special case $\mathcal{F}_1$ of our feedback function. In this feedback model, Group Testing is known to be solvable using $O(k^2 \log(n/k))$ queries [21] and an explicit polynomial-time construction of length $O(k^2 \log n)$ exists [54]. Best known lower bound (for $k < \sqrt{n}$) is $\Omega(k^2 \log n/\log k)$ [12]. This model is also similar to Angluin's notion of concept learning with disjointness queries [1]. The main differences are that in our model the learner is non-adaptive and that in the disjointness queries the feedback might include an element from the intersection.

The setting considered in this paper is also a generalization of an existing problem of coin weighting. In this problem, we have a set of n coins of two distinct weights w_0 (true coin) and w_1 (counterfeit coin), out of which up to k are counterfeit ones. We are allowed to weigh any subset of coins on a spring scale, hence we can deduce the number of counterfeit coins in each weighting. The task is to identify all the counterfeit coins. Such a feedback is a special case $\mathcal{F}_k$ of our feedback function. The problem is solvable with $O(k \log(n/k)/\log k)$ [37] non-adaptive (i.e., fixed in advance) queries and matching a standard information-theoretic lower bound of $\Omega(k \log(n/k)/\log k)$, as well as its stronger version proved for randomized strategies [24]. [4] considers the problem of explicit polynomial-time construction of $O(k \log(n/k)/\log k)$ queries that allows for polynomial time identification of the counterfeit coins. However, the algorithm presented in [4] is adaptive, which means that the subsequent queries can depend on the feedback from the previous ones. The only existing, constructive, non-adaptive solution would be using the explicit construction of the superimposed codes [46] but the resulting query complexity would be $O(k^2 \operatorname{polylog} n)$. Thus, the solution in our paper is the first explicit polynomial time algorithm constructing non-adaptive queries allowing for fast decoding of set K, with $O(k \operatorname{polylog} n)$ fixed queries.

An important line of work on non-adaptive *randomized* solutions to Quantitative Group Testing, started by [55] and being continued until recently [32, 14, 29, 3], resulted in a number of algorithms nearing the lower bound of $2k\frac{\ln(n/k)}{\ln k}$ from [25]. However, these results always assume some restriction on k (typically $k \sim n^\theta$ for some $0 < \theta < 1$), and similarly as above, they may result in significant deviation from the actual set if n is large. They also require a large number of truly random bits (in practice, algorithms have access only to small truly random seeds, while randomized solutions typically have large number of $\Omega(n \log n \log k)$ random bits to construct the queries) and typically do not offer any provable guarantees in case when an adaptive adversary could create/change the hidden set or introduce errors online during the learning process.

The bounds obtained in this paper match (up to polylogarithmic factors) the best existing results for the extreme cases of $\alpha = 1$ and $\alpha = k$. The paper also bounds how the query complexity depends on the value of α between these extremes. Interestingly we show, that the shortest-possible query complexity of $k \operatorname{polylog} n$ is already possible for $\alpha = \sqrt{k}$ and increasing α from $\sqrt{k}$ to k does not result in further decrease of the query complexity.

Other interesting feedback function is a *Threshold Group Testing* [20, 23], where the feedback model includes a set of thresholds and the feedback function returns whether or not the size of the intersection is larger or smaller than each threshold. In [22] the authors show that it is possible to define an interval of $\sqrt{k \log k}$ thresholds resulting in an algorithm with $O(k \log(n/k)/\log k)$ queries. Note that both those feedbacks are "inefficient" in view of our setting of α-capped feedbacks, because their feedback functions are not capped at any $\alpha < k$, but they achieve similar query complexity as our capped $\mathcal{F}_{\sqrt{k}}$ feedback.

Superimposed codes and dispersers were already used in Group Testing. However, they either led to a super-quadratic (in k) number of queries, c.f., [46, 9], or decoded only a fraction of elements of the hidden set, see [44, 41]. Recall that in solutions where query complexity depends on the number of identified elements, decoding of all the elements requires over k^2 queries, as proved in [21, 10]. [45]

presents the first Group Testing solution with $poly(k, \log n)$ decoding time, but super-quadratic query complexity. It is worth noting that our generalized solution achieves almost-linear number of queries (for certain values of α) and our decoding algorithm identifies all the elements in time polynomial in k and logarithmic in n. Our use of the known tools is different then in previous approaches: we define new properties (SuI, SSuI), which we prove to be satisfied by some combinations of those tools, and lead to efficient solutions in both query complexity and construction/decoding time.

3 The Model and the Problem

We assume that the universe of all elements N, with $|N| = n$, is enumerated with integers $1, 2, \ldots, n$. Throughout the paper, we will associate an element with its identifier. Let K, with $|K| \leq k$, denote a hidden subset of N chosen arbitrarily by an adversary. Let $\mathcal{Q} = \langle Q_1, \ldots, Q_m \rangle$ be a non-adaptive algorithm, represented by a sequence of m queries fixed prior to an execution.

A general feedback function $\mathcal{F}$ is a function from subsets of N into an arbitrary domain. A function is applied to $K \cap Q$. A meaningful feedback function can range from simple empty/non-empty feedback up to returning all elements of the intersection. Consider a feedback function $\mathcal{F}_\alpha$, that returns the size of an intersection if it is at most α and α for larger intersections, *i.e.,* $\mathcal{F}_\alpha(Q \cap K) = \min\{|Q \cap K|, \alpha\}$. Parameter α in feedback $\mathcal{F}_\alpha$ is called a *feedback cap*. A general class of feedback functions (used in our lower bound) with feedback cap α includes all deterministic functions that take subsets of N as input and for sets with more than α elements output some arbitrary fixed value.

We say that $\mathcal{Q}$ solves Quantitative Group Testing (QGT), if the feedback vector allows for unique identification of set K. The feedback vector is defined as $\langle \mathcal{F}_\alpha(Q_1 \cap K), \mathcal{F}_\alpha(Q_2 \cap K), \ldots, \mathcal{F}_\alpha(Q_t \cap K) \rangle$. Thus, in order to solve QGT, the feedback vectors for any two sets K_1 and K_2 have to be different. We say that $\mathcal{Q}$ solves QGT with polynomial-time reconstruction if there exists a polynomial-time algorithm that, given the feedback vector outputs all the identifiers of the elements from K. Finally we say that $\mathcal{Q}$ is constructible in polynomial time if there exists a polynomial-time algorithm, that given parameters n, k, α outputs an appropriate sequence of queries.

We assume that both coupled algorithms, construction and decoding, know $n, k, \mathcal{F}_\alpha$. W.l.o.g., in order to avoid rounding in the presentation, we assume that n and other crucial parameters are powers of 2.

4 Polynomial-time Constructions and Decoding

4.1 Combinatorial Tools

In this section we introduce two new combinatorial tools (Selectors-under-Interference and Strong-Selectors-under-Interference) for our QGT solutions, and recall Dispersers and Balanced IDs.

For given sets $K_1, K_2 \subseteq N$ and an element $v \in K_1$, we say that $S \subseteq N$ *selects* v *from* K_1 *under* α*-interference from* K_2 if $S \cap K_1 = \{v\}$ and $|S \cap K_2| < \alpha$. Intuitively, v is a unique representative of K_1 in S and the number of representatives of K_2 in S is smaller than α.

Definition 1 (Selector under Interference (SuI))**.** *An* $(n, \ell, \epsilon, \kappa, \alpha)$*-Selector-under-Interference,* $(n, \ell, \epsilon, \kappa, \alpha)$*-SuI for short, is a sequence of queries* $\mathcal{S} = (S_1, \ldots, S_x)$ *satisfying: for every set* $K_1 \subseteq N$ *of at most* ℓ *elements and set* $K_2 \subseteq N$ *of at most* κ *elements, there are at most* $\epsilon\ell$ *elements* $v \in K_1$ *that* are not *selected from* K_1 *under* α*-interference from* K_2 *by any query* $S_i \in \mathcal{S}$*, i.e., set* $\{v \in K_1 \ : \ \forall_{i \leq x} \ S_i \cap K_1 \neq \{v\}$ *or* $|S_i \cap K_2| \geq \alpha\}$ *has less than* $\epsilon\ell$ *elements.*

Disperser. Consider a bipartite graph $G = (V, W, E)$, where $|V| = n$, which is an $(\ell^* = \epsilon\ell, d, \epsilon)$-*disperser with entropy loss* δ, i.e., it has left-degree d, $|W| = \ell d/\delta$, and satisfies the following dispersion condition: for each $L \subseteq V$ such that $|L| \geq \ell^*$, the set $N_G(L)$ of neighbors of L in graph G is of size at least $(1 - \epsilon)|W|$. Note that it is enough for us to take as ϵ in the dispersion property the same value as in the constructed $(n, \ell, \epsilon, \kappa, \alpha)$-SuI $\mathcal{S}$.[1] An explicit construction (i.e., in time polynomial in n) of dispersers was given by [57], for any $n \geq \ell$, and some $\delta = O(\log^3 n)$, where $d = O(\text{polylog } n)$.[2] In the following the notation d and δ will always denote the parameters of the disperser. Note that any improved construction of dispersers will immediately reduce the complexities of our algorithms. In Appendix C.1 we will describe two disperser-based polynomial-time constructions of SuI and prove:

[1] If someone considers $\epsilon \geq 1/3$, we could use construction for $\epsilon = 1/3$ to get solution with better guarantees.

[2] Optimal dispersers have $\delta = O(1)$ and $d = O(\log n)$, but their polynomial-time construction is an open problem.

Theorem 3. *There is an explicit polynomial-time construction of an $(n, \ell, \epsilon, \kappa, \alpha)$-SuI, for any ℓ, any $\alpha \leq k$ such that $\ell\alpha/(3\delta) > \kappa$, for any constant $\epsilon \in (0, 1/3)$, of size $O(\min\{n, \ell d\delta \log^2 n\})$. Moreover, every element occurs in $O(d\delta \log n)$ queries.*

Theorem 4. *There is an explicit polynomial-time construction of an $(n, \ell, \epsilon, \kappa, \alpha)$-SuI, for any ℓ, any $\alpha \leq k$ such that $\ell \leq 3\delta\kappa/\alpha$ for any constant $\epsilon \in (0, 1/3)$, of size $O\left(\min\left\{n, \frac{\kappa d\delta}{\alpha} \log^2 n\right\} + \frac{nd\delta}{\alpha} \log n\right)$. Moreover, every element occurs in $O(d\delta \log n)$ queries.*

Definition 2 (Strong Selector under Interference (SSuI))**.** *An $(n, \ell, \kappa, \alpha)$-Strong-Selector-under-Interference, $(n, \ell, \kappa, \alpha)$-SSuI for short, is a sequence of queries $\mathcal{T} = (T_1, \ldots, T_x)$ satisfying: for every set $K_1 \subseteq N$ of at most ℓ elements and set $K_2 \subseteq N$ of at most κ elements,* every *element $v \in K_1$ is selected from K_1 under α-interference from K_2 by some query $T_i \in \mathcal{T}$, i.e., set $\{v \in K_1 \ : \ \forall_{i \leq x}\, T_i \cap K_1 \neq \{v\}$ or $|T_i \cap K_2| \geq \alpha\}$ is empty.*

An $(n, \ell, \kappa, \alpha)$-Strong-Selector-under-Interference could be also viewed as $(n, \ell, 0, \kappa, \alpha)$-Selector-under-Interference. In Appendix C.2 we describe a polynomial-time construction of SSuI, which essentially is a [46] construction for adjusted parameters, and prove that it satisfies the SSuI property.

Theorem 5. *There is an explicit polynomial-time construction of an $(n, \ell, \kappa, \alpha)$-SSuI of length $O(\ell^2 \log_\ell^2 n)$, provided $\ell \geq 3\delta\kappa/\alpha$. Moreover, every element occurs in $O(\ell \log_\ell n)$ queries.*

Balanced IDs. Each element i in N has a unique ID represented by $2\log_2 n$ bits, in which the number of 1's is the same as the number of 0's; e.g., take a binary representations of elements i and $n - i$, each in $\log_2 n$ bits, and concatenate them (recall $n = |N|$). Balanced IDs have previously been used in algorithms for decoding elements in Group Testing (see e.g., [50]).

4.2 Construction of Queries, Decoding and Analysis (Proof of Theorem 1)

Algorithm constructing queries. Assume $\alpha \geq 2$ and $k > 3\delta k/(\alpha - 1)$ (the opposite case will be considered later) and let us take $(n, \ell, 1/2, k, \alpha - 1)$-SuI $\mathcal{S}^{(\ell)}$, for ℓ being a power of 2 ranging down from k to $3\delta k/(\alpha - 1)$ (w.l.o.g. we could also assume that $3\delta k/(\alpha - 1)$ is a power of 2). Next, for each set S in these selectors we add the following family $\mathcal{R}(S) = \{R_i(S)\}_{i=1}^{2\log_2 n}$ of sets $R_i(S) = \{v \in S \ : \ \lfloor v/2^{i-1} \rfloor = 1 \mod 2\}$. Intuitively $R_i(S)$ is the set of elements from S that have 1 on i-th least significant bit of Balanced ID. Let us call the obtained enhanced selectors (i.e., with additional families $\mathcal{R}(S)$, for every set S in the original selector) $\bar{\mathcal{S}}^{(\ell)}$. Then we concatenate selectors $\mathcal{R}(\mathcal{S}^{(\ell)})$, starting from the largest $\ell = k$, to the smallest value $\ell = 3\delta k/(\alpha - 1)$. An $(n, 3\delta k/(\alpha - 1), k, \alpha - 1)$-SSuI $\mathcal{T}$ is concatenated at the end, with the same replacement of bits 1 and 0 that gives $\mathcal{R}(\mathcal{T})$ as in the above $(n, \ell, 1/2, k, \alpha - 1)$-SuI's. In the case, where $\alpha = 1$ or $k \leq 3\delta k/(\alpha - 1)$ then the sequence equals to (n, k, k, α)-SSuI $\mathcal{T}$ concatenated as in the above with $\mathcal{R}(\mathcal{T})$. Algorithm 1 presents a pseudocode of the construction algorithm for $\alpha > 1$.

Algorithm 1: Construction of a sequence of queries solving QGT with α-capped feedback.

Input: Disperser G of degree d and entropy loss δ

```
ℓ ← k, Q ← ⟨⟩;
while ℓ > 3δk/(α−1) do
    S ← (n, ℓ, 1/2, k, α − 1)-SuI;
    foreach S ∈ S do
        Q.append(S);
        for i ← 1 to 2 log_2 n do
            /* Add a set of elements
               from S that have 1 on
               i-th least significant
               bit of Balanced ID.    */
            R_i(S) ← {v ∈ S : ⌊v/2^(i−1)⌋ = 1
              mod 2};
            Q.append(R_i(S));
    ℓ ← ℓ/2;
T ← (n, 3δk/(α − 1), k, α − 1)-SSuI;
foreach T ∈ T do
    Q.append(T);
    for i ← 1 to 2 log_2 n do
        R_i(T) ← {v ∈ T : ⌊v/2^(i−1)⌋ = 1
          mod 2};
        Q.append(R_i(T));
return Q
```

$$\langle S_1^{(16)}, S_2^{(16)}, \ldots\rangle = \mathcal{S}^{(16)} = (64, 16, 1/2, 32, 16)\text{-SuI}$$

$$\langle S_1^{(32)}, S_2^{(32)}, \ldots\rangle = \mathcal{S}^{(32)} = (64, 32, 1/2, 32, 16)\text{-SuI} \qquad \langle T_1, T_2, \ldots\rangle = \mathcal{T} = (64, 8, 32, 16)\text{-SSuI}$$

$$\Big\langle \underbrace{S_1^{(32)}, R\left(S_1^{(32)}\right), S_2^{(32)}, R\left(S_2^{(32)}\right), \ldots,}_{\text{for decoding 16 (out of 32) hidden elements}} \underbrace{S_1^{(16)}, R\left(S_1^{(16)}\right), S_2^{(16)}, R\left(S_2^{(16)}\right), \ldots,}_{\text{for decoding 8 (out of remaining 16) hidden elements}} \underbrace{T_1, R\left(T_1\right), T_2, R\left(T_2\right), \ldots}_{\text{for decoding the remaining 8 hidden elements}} \Big\rangle$$

Figure 1: An example of sequence of queries produced by Algorithm 1 for $n = 64, k = 32, \alpha = 17$. For simplicity we assume that $16 > \frac{3\delta k}{\alpha-1} \geq 8$.

Decoding algorithm. During the decoding algorithm we process, in subsequent iterations, the feedbacks from enhanced selectors $\bar{\mathcal{S}}^{(\ell)}$ for $\ell = k, k/2, k/4, \ldots, 3\delta k/(\alpha - 1)$. We will later prove, by induction, that during processing $\bar{\mathcal{S}}^{(\ell)}$, $\ell/2$ new elements from K are decoded. To show this, we consider any iteration and let set K_1 be the set of the elements that have been decoded in previous iterations while set $K_2 = K \setminus K_1$ be the set of unknown elements. We treat K_1 as the interfering set and, by the properties of $\mathcal{S}^{(\ell)}$, we know that for at least $\ell/2$ elements v, there exists a query $S \in \mathcal{S}^{(\ell)}$, such that $v \in S$, $|K_1 \cap S| < \alpha - 1$, $|K_2 \cap S| = 1$. We observe that since we already know the identifiers of all the elements from the interfering set K_1, then using feedbacks from the additional queries $\mathcal{R}(S)$ (corresponding to balanced IDs) we can exactly decode the identifier of v. We do this for all $\ell/2$ elements that are possible to decode in this iteration and we proceed to the next iteration. After considering all Selectors-under-Interference, we have only at most $3\delta k/(\alpha - 1)$ unknown elements. To complete the decoding we use a Strong-Selector-under-Interference, where the decoding procedure is exactly the same as in the case of SuI (the interfering set is also the set of already decoded elements). The properties of SSuI guarantee that we decode the identifiers of all the remaining elements from K. See the pseudocode of decoding Algorithm 2 for details of deocding for $\alpha > 1$. In the case where $\alpha = 1$ the decoding is straightforward, as the SSuI guarantees in this case no interference, hence the results (due to our enhancement $\bar{\mathcal{T}}$) contain exactly the identifiers of the elements from set K.

Lemma 1. *There is an explicit polynomial-time algorithm constructing non-adaptive queries $Q_1, \ldots, Q_m$ and decoding any hidden set K of size at most $k \leq n$, from the feedback vector in polynomial time, under feedback $\mathcal{F}_\alpha$ and for $m = O((\frac{k}{\alpha})^2 \delta^2 \log^3 n + kd\delta \log^3 n)$ queries. Moreover, every element occurs in $O(\frac{k}{\alpha}\delta \log^2 n + d\delta \log^3 n)$ queries.*

Proof. We start from describing a procedure of revealing elements in any given set K of at most k elements, together with a formal (inductive) argument of its correctness. Our first goal is to show that by the beginning of $\bar{\mathcal{S}}^{(\ell)}$, for ℓ stepping down from k to $3\delta k/(\alpha - 1)$, we have not learned about the identity of at most ℓ elements from the hidden set K.

The proof is by induction – it clearly holds in the beginning of the computation, as the set K has at most $\ell = k$ elements. We prove the inductive step: by the end of $\bar{\mathcal{S}}^{(\ell)}$, at most $\ell/2$ elements are not learned. We set K_2 to be the set of learned elements and $K_1 = K \setminus K_2$. Clearly, $|K_2| \leq k$, and by the inductive assumption $|K_1| \leq \ell \leq k$. For such K_1 and K_2, by the definition of SuI, there are at most $\ell/2$ elements from K_1 that are not occurring in some round without other such elements or with at least $\alpha - 1$ of already learned elements from K_2. Consider a previously not learned element $v \in K_1$, for which there exists a *good query* in $\bar{\mathcal{S}}^{(\ell)}$, *i.e.,* a query $S \in \bar{\mathcal{S}}^{(\ell)}$ such that $S \cap K_1 = \{v\}$ and $|S \cap K_2| < \alpha - 1$. At this point of decoding of set K we know the Balanced IDs of all elements from set K_2. Hence we can calculate the $2\log_2 n$-bit feedback vector from sets $K_2 \cap R_1(S), K_2 \cap R_2(S), \ldots, K_2 \cap R_{2\log_2 n}(S)$. We compare this feedback vector with the output of the enhanced selector, which is the feedback vector for sets $K \cap R_1(S), K \cap R_2(S), \ldots, K \cap R_{2\log_2 n}(S)$. The difference between the latter and the former is exactly the Balanced ID of v. In case this difference does not form a Balanced ID of any element, *i.e.,* it has some value bigger than 1 or otherwise the number of 1's is different from $\log_2 n$, or in case $\mathcal{F}_\alpha(K \cap S) = \alpha$ (recall that S is also in the constructed selector) the feedback from this $\mathcal{R}(S)$ is ignored. This is done to avoid misinterpreting the feedback and false discovery of an element which is not in K. Indeed, first note that the fact $|K \cap S| \geq \alpha$ will automatically discard the part of the feedback from $K \cap R_1(S), K \cap R_2(S), \ldots, K \cap R_{2\log_2 n}(S)$, as it indicates that the intersection is too large to provide correct decoding of an element. Second, assuming $|K \cap S| < \alpha$, if there are no elements

Algorithm 2: Decoding of the elements for QGT with α-capped feedback.

Input: $\mathsf{Result}(Q)$ for queries $Q \in \mathcal{Q}$ constructed by Algorithm 1, based on SuI $\mathcal{S}^{(\ell)}$ for ℓ being powers of 2 stepping down from k to $\frac{3\delta l}{\alpha-1}$ and SSuI $\mathcal{T}$.

$\ell \leftarrow k, K_{acc} \leftarrow \emptyset$; `/* Set` K_{acc} `accumulates the decoded elements. */`
while $\ell > \frac{3\delta k}{\alpha-1}$ **do** `/* We decode` $l/2$ `elements in each iteration.*/`
  for $i \leftarrow 1$ **to** $l/2$ **do**
    `/* Look for query for which we can decode a new element. */`
    foreach $S \in \mathcal{S}^{(\ell)}$ **do**
      if $\mathsf{Result}(S) < \alpha - 1$ **and** $|S \cap K_{acc}| = \mathsf{Result}(S) - 1$ **then**
        $K_{acc} \leftarrow K_{acc} \cup \{DecodeElement(S, K_{acc})\}$;
  $\ell \leftarrow \ell/2$;
while $\ell > 0$ **do** `/* Iteratively decode all the remaining elements.*/`
  foreach $T \in \mathcal{T}$ **do**
    if $\mathsf{Result}(T) < \alpha - 1$ **and** $|T \cap K_{acc}| = \mathsf{Result}(T) - 1$ **then**
      $K_{acc} \leftarrow K_{acc} \cup \{DecodeElement(T, K_{acc})\}$;
      $\ell \leftarrow \ell - 1$;
return K_{acc}

Procedure *DecodeElement(Q,K_{acc})*
  $v \leftarrow 0$;
  for $j \leftarrow 2\log_2 n$ **downto** 0 **do**
    `/* Take the feedback from set` $R_j(Q)$`. Calculate the feedback from set` $R_j(Q)$`, if hidden set was exactly` K_{acc}`. The difference is` j`-th least significant bit of Balanced ID of new element` v`. */`
    $v \leftarrow 2 \cdot v + \mathsf{Result}(R_j(Q)) - |R_j(Q) \cap K_{acc}|$
  return v

in $K_1 \cap S$ then the difference between feedbacks gives vector of zeros, and if there will be at least two elements in $K_1 \cap S$, the difference between feedbacks will contain a value of at least 2 or all 1's, as it will be a bitwise sum of at least two Balanced IDs of $\log_2 n$ ones each. By the definition of SuI we can find $l/2$ such elements v. This shows that during decoding of enhanced $\bar{\mathcal{S}}^{(\ell)}$ we learn the identities of $\ell/2$ new elements. This completes the inductive proof. Note here that the inductive step, being one of $O(\log n)$ steps, defines a polynomial time algorithm decoding some elements one-by-one – indeed, it computes two feedbacks of polynomial number of queries, computes the difference and deducts based on the structure of subsequent blocks of $O(\log n)$ size.

The above analysis implies, that before applying $(n, 3\delta k/(\alpha-1), k, \alpha-1)$-SSuI we have not discovered at most $3\delta k/(\alpha-1)$ elements. Thus, by definition, $(n, 3\delta k/(\alpha-1), k, \alpha-1)$-SSuI combined with Balanced IDs reveals all the remaining elements in the same way as the SuI's above – the only difference in the argument is that instead of leaving at most $\ell/2$ undiscovered elements in the ℓ-th inductive step, due to the nature of SuI's, the SSuI guarantees that *every* undiscovered element will occur in a good query. The same argument as for SuI's proves that the decoding algorithm defined this way works in polynomial time. By Theorem 3, the length of $(n, \ell, 1/2, k, \alpha)$-SuI is $O(\min\{n, \ell d\delta \log^2 n\})$, which sums up to $O(kd\delta \log^2 n)$, and is multiplied by $\Theta(\log n)$ due to amplification by Balanced IDs. By Theorem 5, the length of $(n, 3\delta k/\alpha, k, \alpha)$-SSuI is $O((\frac{k}{\alpha})^2\delta^2 \log^2 n)$, and it is also increased by factor $\Theta(\log n)$ due to Balanced IDs. If we apply the above reasoning with respect to the number of queries containing an element, we get that every element occurs in $O(\frac{k}{\alpha}\delta \log^2 n + d\delta \log^3 n)$ queries. □

Implementing α-Round-Robin for large values of k/α. The question arises from the previous result if one could efficiently construct a shorter sequence of queries if $(k/\alpha)^2 > n/\alpha$? In the case of full feedback (*i.e.,* $\alpha = k$) the common way to deal with large values of k is via Round-Robin, which means that queries are singletons and consequently, the length of such query sequence is n. This also works for an arbitrary value of $\alpha \leq k$, however the lower bound in Theorem 2 suggest that in such case there could exist a shorter query system of length $O((k + (n/\alpha)) \operatorname{polylog} n)$. Indeed, if we modify our construction in such case, we could obtain such a goal. Namely, we concatenate:

- selectors $\bar{\mathcal{S}}^{(\ell)}$, for ℓ being decreasing powers of 2 from $\ell = k$ to $\ell = 6\delta k/(\alpha - 1)$;
- selectors $\bar{\mathcal{S}}|_{\alpha}^{(\ell)}$, for ℓ being decreasing powers of 2 from $\ell = 3\delta k/(\alpha - 1)$ to $\ell = 1$.

In the case, where $k > 6\delta k/(\alpha - 1)$, then the construction contains only selectors $\bar{\mathcal{S}}|_{\alpha}^{(\ell)}$. Then we enhance them based on Balanced IDs, as in the previous construction. Then, applying Theorem 3 for concatenated $\bar{\mathcal{S}}^{(\ell)}$ and Theorem 4 for concatenated $\bar{\mathcal{S}}|_{\alpha}^{(\ell)}$, instead of combination of Theorems 3 and 5 as it was in the proof of Lemma 1 with respect to $\bar{\mathcal{S}}^{(\ell)}$, we get the following result.

Lemma 2. *There is an explicit polynomial-time algorithm constructing non-adaptive queries $Q_1, \ldots, Q_m$ and decoding any hidden set K of size at most $k \leq n$, from the feedback vector in polynomial time, under feedback $\mathcal{F}_\alpha$ and for $m = O((k + n/\alpha)d\delta \log^3 n)$ queries. Moreover, each element occurs in $O(d\delta \log^3 n)$ queries.*

Proof. The proof is analogous to the proof of Lemma 1, except that we continue proving the invariant until $\ell = 6\delta\kappa/(\alpha - 1)$, using same properties guaranteed by Theorems 3, and continue the invariant until $\ell = 1$, using SuI's of slightly different length formula from Theorem 4. The correctness argument, as well as polynomial-time query construction and decoding of the elements, are the same as in the invariant proof in Lemma 1. Then, by Theorem 3 for concatenated $\bar{\mathcal{S}}^{(\ell)}$ and by Theorem 4 for concatenated $\bar{\mathcal{S}}|_{\alpha}^{(\ell)}$, we argue that the total length of the obtained sequence is $m = O((k + n/\alpha)d\delta \log^3 n)$. Indeed, the first part results from the telescoping sum for different ℓ and the second component is a logarithmic amplification of the original $O((nd\delta/\alpha) \log n)$ length of SuI's; all is amplified by $O(\log n)$ due to enhancement of the used SuI's by Balanced IDs. In all the components, every element belongs to $O(d\delta \log n)$ queries, by Theorems 3 and 4, in the final sequence it occurs in $O(d\delta \log^3 n)$ queries. □

Combining Lemma 1 with Lemma 2 gives Theorem 1. Note that in both lemmas the decoding algorithm proceeds query-by-query, each time spending polylogarithmic time on each of them; additionally, for each decoded element, an update of the feedback of next queries needs to be done, which takes time proportional to the number of occurrences of the discovered element in the queries. Thus, it is asymptotically upper bounded by the length of the sequence plus k times the upper bound on the number of occurrences of an element in the queries (polylogarithmic).

5 Lower Bound

Proof of Theorem 2. We will first show the $\min\{n/\alpha, k^2/\alpha^2\}$ component. Assume that a sequence of queries $Q_1, Q_2, \ldots, Q_t$ of length t solves Group Testing. We want to show the lower bound that holds for any feedback function capped at α hence we assume that the feedback function $\mathcal{F}$ returns the whole set (i.e., the identifiers of all the elements). Recall that $\mathcal{F}$ works only for sets with at most α elements. We begin by proving the following:
Claim A: For any set K, with $|K| \leq k$ and any $x \in K$, there must exist $\tau \in \{1, 2, \ldots, t\}$, such that $x \in Q_\tau$ and $|K \cap Q_\tau| \leq \alpha + 1$.

The proof is by contradiction. Assume that such a set K^* and element x^* exist for which there is no such query. Consider feedback vectors for sets K^* and $K^* \setminus \{x^*\}$. For any query that does not contain x^*, the feedback is clearly identical. For any query Q_τ, such that $x^* \in Q_\tau$, we have $|Q_\tau \cap K^*| \geq \alpha + 2$ and $|Q_\tau \cap (K^* \setminus \{x^*\})| \geq \alpha + 1$ and sets K^* and $K^* \setminus \{x^*\}$ are indistinguishable under any feedback capped at α hence the sequence of queries does not solve the problem. This completes the proof of Claim A. ■

Take all queries that have at most $\alpha + 1$ elements and all elements that belong to such queries. We have: $N_s = \bigcup_{\tau \in \{1,2,\ldots,t\} | Q_\tau| \leq \alpha+1} Q_\tau$ and remaining elements $N_l = N \setminus N_s$. Consider two cases:
Case 1: $|N_s| \geq n/2$.

Observe that: $t \geq |\{\tau \in \{1, 2, \ldots, t\} : |Q_\tau| \leq \alpha + 1\}| \geq \frac{N_s}{\alpha+1} \geq \frac{n}{2(\alpha+1)}$.

Case 2: $|N_l| \geq n/2$.

In this case, we take an arbitrary subset K_1 of $k/2$ elements from N_l. For every element $x \in K_1$, we consider a set of queries $Q(x) = \{Q_\tau \in \{Q_1, Q_2, \ldots, Q_t\} : x \in Q_\tau, |Q_\tau \cap K_1| \leq \alpha + 1\}$. We show:

Claim B: For every $x \in K_1$, we have $|Q(x)| \geq \frac{k}{2(\alpha+2)}$. The proof is by contradiction. Assume that for some $x^* \in K_1$ we have $|Q(x^*)| < \frac{k}{2(\alpha+2)}$. Then, for every query $Q \in Q(x^*)$, we take

$\alpha + 2 - |Q \cap K_1|$ elements from $Q \setminus K_1$. Such elements exist since $|Q| \geq \alpha + 2$. Choose such elements for each query in $Q(x^*)$ and gather them in set K_2. Note that since $|Q(x^*)| < \frac{k}{2(\alpha+2)}$, then $|K_2| \leq k/2$. Now observe that set $K_1 \cup K_2$ and element x^* violate Claim A. The obtained contradiction completes the proof of Claim B. ■

Now observe that each query belongs to at most $\alpha + 1$ sets $Q(x)$ for different values of $x \in K_1$. Thus: $t \geq \frac{\sum_{x \in K_1} |Q(x)|}{\alpha+1} \geq \frac{k^2}{4(\alpha+1)(\alpha+2)}$. To complete the proof observe that any algorithm must fall either into Case 1 or Case 2, hence any algorithm needs to use $\Omega\left(\min\{n/\alpha, k^2/\alpha^2\}\right)$ queries.

To see that any algorithm in $\mathcal{F}_\alpha$ feedback model at least $k\frac{\log \frac{n}{k}}{\log \alpha}$ queries, observe that the feedback vector must be unique for each set K with at most k elements. Hence we need at least $\binom{n}{k}$ different feedback vectors for different sets. Feedback has at most α values hence we get $\alpha^t \geq \binom{n}{k}$ and $t \in \Omega(k\frac{\log \frac{n}{k}}{\log \alpha})$. This completes the proof of Theorem 2. ■

6 Extensions, Applications and Open Directions

Observations. The size of response to a capped quantitative query is not much bigger than beeping – $\log \alpha$ bits vs one bit of beeping – whereas the number of queries is smaller by a factor of α^2 compared to beeping. Thus, even after normalization of the query complexity by the inverse of the size of the feedback, in the α-capped model it is nearly α^2 times better than the classic beeping, up to $\alpha = \sqrt{k}$.

Although optimizing the construction time is not our main focus, it is actually small. E.g., for $\alpha = \sqrt{k}$, the construction of all queries takes time $\widetilde{O}(n\sqrt{k})$. Most of the complexity comes from the construction of SSuI (Alg 4) – for each element of the universe, we consider $\widetilde{O}(\sqrt{k})$ zeros of some polynomial. We use a logarithmic number of SuIs in the construction, but each takes only $\widetilde{O}(n)$ time (Alg 3): in SuI for every element we list its neighborhood in disperser in polylogarithmic time [57] and take the Cartesian product with a polylogarithmic Reed-Solomon type of codeword.

Fault-tolerance. Algorithms 1 and 2 are fault-tolerant with respect to $f \leq c \cdot \frac{k}{\alpha}$ *adversarially* jammed results, for some sufficiently small constant $c > 0$ – here by a jammed result we understand receiving a null as a result of a query. It also tolerates *stochastic* failures, where a result is jammed with probability $p \leq c$. Indeed, in adversarial failures we observe that, by dispersion, $c \cdot \frac{k}{\alpha}$ could be made negligible (by right selection of constant c) with respect to the number of selected elements in SuI, and it is also smaller than the number of single occurrences (i.e., without interference from other $k - 1$ elements) of an element in SSuI. Thus the analysis of the algorithms hold (the original analysis have margins to accommodate those additional negligible jamming, as they were done with only asymptotic upper bounds). The tolerance of stochastic failures follows from the fact that a random p-fraction of results of queries coming from SuI's and SSuI are again negligible from perspective of the fraction of correctly encoded elements and can be accommodated by our asymptotic analysis.

Theorem 6. *Algorithms 1 and 2 tolerate $f \leq c \cdot \frac{k}{\alpha}$ adversarial jammings and stochastic jammings with probability $p \leq c$, for some sufficiently small constant $c > 0$.*

Other applications. Applications to efficient GT and dynamic maintenance of multi-sets, graphs, as well as to Private Parallel Information Retrieval and coding, are detailed in Appendix B.

Open directions. Considering only polynomially-constructible query systems leaves some interesting open directions. One such direction is whether optimal-length query sequence can be constructed in polynomial time or perhaps it is possible to show some reduction that constructing a nearly-shortest query sequence is computationally hard (even if we know that it exists). Shrinking polylogarithmic gaps between lower and (existential) upper bounds and improving constants is another challenging direction, as well as considering other interesting classes of GT models with an α-capped results, e.g., parity. We also believe that with some adjustment, capped GT codes could be applied to efficiently solve many open problems in online streaming, communication and graph learning fields.

Acknowledgments and Disclosure of Funding

Dominik Pająk was supported by the National Science Centre, Poland (NCN), grant UMO-2019/33/B/ST6/02988. Dariusz R. Kowalski was supported, in part, by the National Science Center, Poland (NCN), grant UMO-2017/25/B/ST6/02010.

References

[1] D. Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.

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

[3] W. H. Bay, E. Price, and J. Scarlett. Optimal non-adaptive probabilistic group testing in general sparsity regimes. *arXiv preprint arXiv:2006.01325*, 2020.

[4] N. H. Bshouty. Optimal algorithms for the coin weighing problem with a spring scale. In *COLT 2009 - The 22nd Conference on Learning Theory, Montreal, Quebec, Canada, June 18-21, 2009*, 2009.

[5] C.-C. Cao, C. Li, and X. Sun. Quantitative group testing-based overlapping pool sequencing to identify rare variant carriers. *BMC bioinformatics*, 15(1):1–14, 2014.

[6] J. Capetanakis. Generalized tdma: The multi-accessing tree protocol. *IEEE Transactions on Communications*, 27(10):1476–1484, 1979.

[7] J. Capetanakis. Tree algorithms for packet broadcast channels. *IEEE transactions on information theory*, 25(5):505–515, 1979.

[8] K. Censor-Hillel, B. Haeupler, N. A. Lynch, and M. Médard. Bounded-contention coding for the additive network model. *Distributed Computing*, 28(5):297–308, 2015.

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

[10] B. S. Chlebus and D. R. Kowalski. Almost optimal explicit selectors. In M. Liskiewicz and R. Reischuk, editors, *Fundamentals of Computation Theory, 15th International Symposium, FCT 2005, Lübeck, Germany, August 17-20, 2005, Proceedings*, volume 3623 of *Lecture Notes in Computer Science*, pages 270–280. Springer, 2005.

[11] S.-S. Choi and J. H. Kim. Optimal query complexity bounds for finding graphs. *Artificial Intelligence*, 174(9-10):551–569, 2010.

[12] A. E. F. Clementi, A. Monti, and R. Silvestri. Selective families, superimposed codes, and broadcasting on unknown radio networks. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA*, pages 709–718. ACM/SIAM, 2001.

[13] R. Clifford, K. Efremenko, E. Porat, and A. Rothschild. Pattern matching with don't cares and few errors. *Journal of Computer and System Sciences*, 76(2):115–124, 2010.

[14] A. Coja-Oghlan, O. Gebhard, M. Hahn-Klimroth, and P. Loick. Optimal group testing. In J. D. Abernethy and S. Agarwal, editors, *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pages 1374–1388. PMLR, 2020.

[15] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi. Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 25–36, 2005.

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

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

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

[19] G. Cormode and S. Muthukrishnan. Combinatorial algorithms for compressed sensing. In *International colloquium on structural information and communication complexity*, pages 280–294. Springer, 2006.

[20] P. Damaschke. Threshold group testing. *Electronic Notes in Discrete Mathematics*, 21:265 – 271, 2005. General Theory of Information Transfer and Combinatorics.

[21] A. De Bonis, L. Gasieniec, and U. Vaccaro. 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*, pages 81–96. Springer, 2003.

[22] G. De Marco, T. Jurdzinski, and D. R. Kowalski. Optimal channel utilization with limited feedback. *J. Comput. Syst. Sci.*, 119:21–33, 2021.

[23] G. De Marco, T. Jurdzinski, D. R. Kowalski, M. Rózanski, and G. Stachowiak. Subquadratic non-adaptive threshold group testing. *J. Comput. Syst. Sci.*, 111:42–56, 2020.

[24] G. De Marco and D. R. Kowalski. Searching for a subset of counterfeit coins: Randomization vs determinism and adaptiveness vs non-adaptiveness. *Random Structures & Algorithms*, 42(1):97–109, 2013.

[25] A. Djackov. On a search model of false coins. In *Topics in Information Theory (Colloquia Mathematica Societatis Janos Bolyai 16). Budapest, Hungary: Hungarian Acad. Sci*, pages 163–170, 1975.

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

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

[28] J. Engels, B. Coleman, and A. Shrivastava. Practical near neighbor search via group testing. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 9950–9962. Curran Associates, Inc., 2021.

[29] U. Feige and A. Lellouche. Quantitative group testing and the rank of random matrices. *CoRR*, abs/2006.09074, 2020.

[30] L. Galanis, Y. Wang, S. R. Jeffery, and D. J. DeWitt. Locating data sources in large distributed systems. In J. C. Freytag, P. C. Lockemann, S. Abiteboul, M. J. Carey, P. G. Selinger, and A. Heuer, editors, *Proceedings of 29th International Conference on Very Large Data Bases, VLDB 2003, Berlin, Germany, September 9-12, 2003*, pages 874–885. Morgan Kaufmann, 2003.

[31] R. Gallager. A perspective on multiaccess channels. *IEEE Transactions on information Theory*, 31(2):124–142, 1985.

[32] O. Gebhard, M. Hahn-Klimroth, D. Kaaser, and P. Loick. Quantitative group testing in the sublinear regime. *CoRR*, abs/1905.01458, 2019.

[33] P. B. Gibbons, Y. Matias, and V. Poosala. Fast incremental maintenance of approximate histograms. *ACM Transactions on Database Systems (TODS)*, 27(3):261–298, 2002.

[34] A. C. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 389–398, 2002.

[35] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. How to summarize the universe: Dynamic maintenance of quantiles. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*, pages 454–465. Elsevier, 2002.

[36] S. Gollakota and D. Katabi. Zigzag decoding: combating hidden terminals in wireless networks. In V. Bahl, D. Wetherall, S. Savage, and I. Stoica, editors, *Proceedings of the ACM SIGCOMM 2008 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Seattle, WA, USA, August 17-22, 2008*, pages 159–170. ACM, 2008.

[37] V. Grebinski and G. Kucherov. Optimal reconstruction of graphs under the additive model. *Algorithmica*, 28(1):104–124, 2000.

[38] A. G. Greenberg, P. Flajolet, and R. E. Ladner. Estimating the multiplicities of conflicts to speed their resolution in multiple access channels. *Journal of the ACM (JACM)*, 34(2):289–325, 1987.

[39] A. G. Greenberg and S. Winograd. A lower bound on the time needed in the worst case to resolve conflicts deterministically in multiple access channels. *Journal of the ACM (JACM)*, 32(3):589–596, 1985.

[40] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. *ACM SIGMOD Record*, 30(2):58–66, 2001.

[41] E. Hradovich, M. Klonowski, and D. R. Kowalski. Contention resolution on a restrained channel. In *26th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2020, Hong Kong, December 2-4, 2020*, pages 89–98. IEEE, 2020.

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

[43] P. Indyk. 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*, pages 127–136. IEEE Computer Society, 1997.

[44] P. Indyk. Explicit constructions of selectors and related combinatorial structures, with applications. In D. Eppstein, editor, *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA*, pages 697–704. ACM/SIAM, 2002.

[45] P. Indyk, H. Q. Ngo, and A. Rudra. Efficiently decodable non-adaptive group testing. In M. Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1126–1142. SIAM, 2010.

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

[47] M. Klonowski, D. R. Kowalski, and D. Pajak. Generalized framework for group testing: Queries, feedbacks and adversaries. *Theor. Comput. Sci.*, 919:18–35, 2022.

[48] J. Komlós and A. G. Greenberg. An asymptotically fast nonadaptive algorithm for conflict resolution in multiple-access channels. *IEEE Trans. Inf. Theory*, 31(2):302–306, 1985.

[49] D. R. Kowalski and D. Pajak. Light agents searching for hot information. In L. D. Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 363–369. ijcai.org, 2022.

[50] K. Lee, K. Chandrasekher, R. Pedarsani, and K. Ramchandran. Saffron: A fast, efficient, and robust framework for group testing based on sparse-graph codes. *IEEE Transactions on Signal Processing*, 67(17):4649–4664, 2019.

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

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

[53] J. L. Massey. Collision-resolution algorithms and random-access communications. In *Multi-user communication systems*, pages 73–137. Springer, 1981.

[54] E. Porat and A. Rothschild. Explicit nonadaptive combinatorial group testing schemes. *IEEE Trans. Inf. Theory*, 57(12):7982–7989, 2011.

[55] A. Sebő. On two random search problems. *Journal of Statistical Planning and Inference*, 11(1):23–31, 1985.

[56] N. Sinnott-Armstrong, D. L. Klein, and B. Hickey. Evaluation of group testing for sars-cov-2 rna. *MedRxiv*, 2020.

[57] A. Ta-Shma, C. Umans, and D. Zuckerman. Loss-less condensers, unbalanced expanders, and extractors. In J. S. Vitter, P. G. Spirakis, and M. Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 143–152. ACM, 2001.

[58] S. Ubaru, S. Dash, A. Mazumdar, and O. Gunluk. Multilabel classification by hierarchical partitioning and data-dependent grouping. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22542–22553. Curran Associates, Inc., 2020.

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

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Our abstract and introduction states the theoretical results that are later proven in our paper.
   (b) Did you describe the limitations of your work? [Yes] Our assumed model (including the result function) is described in Section 3.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Our result is theoretical hence does not have any direct negative impact. Also none of the existing solutions to the Group Testing (which is a field with almost 80 years of history) were harmful to the society.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] The assumptions are given in a statement of each theoretical result.
   (b) Did you include complete proofs of all theoretical results? [Yes] Some combinatorial constructions (used in proofs of Theorems 3 4 5), are included Appendix C.
3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Other Related Work

Our algorithms have direct application in stream processing and set reconstruction. There is a number of existing algorithms retrieving various informations from streams, such as: extracting the most frequent elements [17, 18, 16, 59, 49], quantile tracking [15, 35, 40], or approximate histogram maintenance and reconstruction [33, 34]. Some of these existing algorithms use Group Testing (e.g., [18]) but in a randomized variant. This leads to a small probability of error that might become significant in very large streams.

Solutions to Group Testing were also applied in many other domains, including pattern matching [13, 43], compressed sensing [19], reconstructing graphs [11, 37], identifying genetic carriers [5], resolving conflicts on multiple-access channels and wireless networks [6, 7, 31, 38, 39, 48, 53, 36]. Our setting of capped quantitative feedback could be applied to more subtle versions of the above-mentioned problems, for example: in wireless communication, where the devices could filter out up to α interfering signals; in bio-chemical testing with capped scale; in streaming algorithms with limited processing time.

The problem of Group Testing has also been considered in various different feedback models. For instance, [8] shows that $O(k \log \frac{n}{k})$ queries are sufficient for a feedback that only returns whether the size of the intersection $|Q \cap K|$ is odd or even.

B Additional Extensions and Applications of Our Results

Group testing on multi-sets. Assume instead of a hidden set, there is a hidden multi-set K, containing at most k elements from $[N]$. Multi-set means that each element may have several multiplicities. Let κ be the sum of multiplicities of elements in K, and we assume it is unknown to the algorithm. We could decode all elements in K with their multiplicities using similar approach as in Section 4, with the following modifications.

First, we need to have a sufficiently large cap α to decode each multiplicity, i.e., α should be not smaller than κ.

Second, in the construction Algorithm 1, instead of applying SuI's only while $\ell > \frac{3\delta l}{\alpha - 1}$ and then SSuI, we need to keep applying SuI's while $\ell > 1$. Analogously in the structure of decoding Algorithm 2 – updating the condition of the first while-loop and removing the second while-loop. This is because the multiplicities of elements not decoded by SuI's in the While-loop could still be larger than $\sqrt{\kappa}$, therefore switching to SSuI may not be enough to decode their multipicities (note that κ plays in this part a similar role to k in the original algorithm for sets without multiplicities). The correctness still holds, as each consecutive SuI combined with balanced IDs reveals full multiplicities of a fraction of remaining elements in K (instead of just presence of elements, as in the original proof of Theorem 1). The asymptotic query complexity (codeword length) stays the same as the part coming from the first while-loop (the sum of SuI lengths multiplied by $2 \log n$ coming from balanced IDs), since we just add a negligible tail in the sum of lengths of SuI's considered in Theorem 1.

Third, polynomial time is now with respect to n and $\log \kappa$, to deal with multiplicities.

Therefore we get:

Theorem 7. *There is an explicit polynomial-time (in n and $\log \kappa$) algorithm constructing non-adaptive queries $Q_1, \ldots, Q_m$, for $m = O\left(\min\{n, k \operatorname{polylog} n\}\right)$, that correctly decode a multi-set K of at most k elements and multiplicity κ (where k is known but κ could be unknown) under feedback $\mathcal{F}_\alpha$ with polynomial-time (in n and $\log \kappa$) decoding, where α is not smaller than the largest multiplicity of an element in K. Moreover, every element occurs in $O(\operatorname{polylog} n)$ queries, and the decoding time is $O(m + k \operatorname{polylog} n)$.*

Maintaining and reconstructing a (multi) set. Consider the following problem. We have an incoming very large stream of insertion or deletions of elements from some domain N. The objective is to propose a datastructure that processes such operations and at any step (i.e., after processing a certain number of operations) it can answer a request and provide information about the set

specified by the operations that have been processes so far. This is a commonly studied setting (see e.g., [18, 42]) and extracting information from such stream of operations has applications to database systems. Our algorithms lead directly to an explicit formulation of a datastructure capable of extracting the whole (multi) set, however only conditioned that (at the moment of the request) the sum of multiplicities of the hidden set does not exceed k. The space complexity of the datastructure would equal to the number of queries of the algorithm, which is $O\left(\min\left\{n, k \operatorname{polylog} n\right\}\right)$.

Maintaining and reconstructing a graph with dynamically added or removed edges. Consider a graph G with a fixed set of nodes and an online stream of operations on G, where a single operation could be either adding or removing an edge to/from G. Assume for the ease of presentation that after each operation, the maximum node degree is bounded by some parameter k. [3] Consider a sequence of queries from Theorem 1 on the set of all possible $\frac{n(n-1)}{2}$ edges. For each added/removed edge, we increase/decrease (resp.) a counter associated with each query containing this edge. As each edge occurs in $O(\frac{k}{\alpha}\log^2 n + \operatorname{polylog} n)$ queries, and thus this is an upper bound (up to some additional logarithmic factor) on the time of each *graph update*, which is *polylogarithmic* for α close to k. Whenever one would like to recover the whole graph, a *reconstruction* algorithm is applied, which takes $O(m + \frac{k^2}{\alpha}\log^2 n + k \operatorname{polylog} n)$ steps, which for α close to k is $O(nk \operatorname{polylog} n)$. Note that the latter formula corresponds to (the upper bound on) the number of edges in G. To summarize, we implemented graph updates operations in $\operatorname{polylog} n$ time per (edge-)operation, and the whole graph recovery in time proportional to the graph size (number of edges) times $\operatorname{polylog} n$.

Private Parallel Information Retrieval (PPIR). One of techniques to speed-up Information Retrieval from a large dataset is to employ autonomous agents searching parts of the datasets, c.f., [30]. Our Capped Quantitative Group Testing algorithms could be applied to achieve this goal, additionally providing a level of privacy. Assume that there are $m = O(k \operatorname{polylog} n)$ simple autonomous agents, where m also denotes the number of queries in our Capped QGT system. Each agent i is capable to search only through records captured by the corresponding query set Q_i, and only count the number of occurrences of records satisfying the search criteria, but only up to $\sqrt{k}$. If all agents share privately their results with the user, he can decompose the set K of at most k elements satisfying the searching criteria, while each of the agents has knowledge about at most $\sqrt{k}$ of these elements. It follows from the construction of our queries that each of them is of size $O(n/\sqrt{k} \operatorname{polylog} n)$, which is worst-case number of records that a single agent needs to check – thus equal to parallel time. Note also that agents perform very simple counting operations, thus the PPIR algorithm could be efficient in practice.

Non-adaptive QGT as codes. Alternatively, we can reformulate the problem of non-adaptive Group Testing under $\mathcal{F}_\alpha$ feedback into the language of codes. A query sequence translates to code as follows: each element $v \in [N]$ corresponds to a binary codeword with i-th position being 1 or 0 depending on whether v belongs to the i-th query or not. Then the hidden set K is a subset of at most k codewords, for which we calculate the feedback vector by taking the function $\min\{\cdot, \alpha\}$ from elementwise sum of all the codewords corresponding to set K. I.e., the feedback is computed for each position i, and the whole feedback vector is an input to the decoding algorithm. The objective is to decode the elements of K from the feedback vector.

C Constructions of Combinatorial Tools

C.1 Polynomial-time Construction of Selectors-under-Interference

In this section, we show how to construct, in time polynomial in n, an $(n, \ell, \epsilon, \kappa, \alpha)$-SuI $\mathcal{S}$ of size $O(\min\{n, \ell \operatorname{polylog} n\})$, for any integer parameters $\ell, \kappa \leq n$, $\alpha \leq \kappa$, and any (arbitrarily small) constant $\epsilon \in (0, 1/2)$. Let ℓ^* denote $\ell\epsilon$. The construction combines dispersers with strong selectors, see also the pseudocode Algorithm 3. We start from reminding the disperser needed for the construction (defined in Section 4.1), and then specifying the strong selectors.

[3] This assumption could be waved by using Group Testing codes for different parameters k, depending on the actual size of G, hence k could play role of an average size of a neighborhood.

The disperser. We will use a bipartite (ℓ^*, d, ϵ)-disperser graph $G = (V, W, E)$ with entropy loss δ, where $|V| = n$. It has left-degree d, $|W| = \ell d/\delta$, and satisfies the following dispersion condition: for each $L \subseteq V$ such that $|L| \geq \ell^*$, the set $N_G(L)$ of neighbors of L in graph G is of size at least $(1-\epsilon)|W|$. Note that it is enough for us to take as ϵ in the dispersion property the same value as in the constructed $(n, \ell, \epsilon, \kappa, \alpha)$-SuI $\mathcal{S}$ – ff someone considers $\epsilon \geq 1/3$, we could use construction for $\epsilon = 1/4$ to get solution with better guarantees. An explicit construction (i.e., in time polynomial in n) of dispersers was given by [57], for any $n \geq \ell$, and some $\delta = O(\log^3 n)$, where $d = O(\text{polylog } n)$. Optimal dispersers have $\delta = O(1)$ and $d = O(\log n)$, but their polynomial-time construction is an open problem.

Strong selector. Let $\mathcal{T} = \{T_1, \ldots, T_m\}$ be an explicit $(n, 3\delta)$-strong-selector (also called strongly-selective family) of length $m \leq \min\{n, 2 \cdot (3\delta)^2 \log^2 n\} = O(\min\{n, \delta^2 \log^2 n\})$, as constructed by Kautz and Singleton [46].

Construction of $(n, \ell, \epsilon, \kappa, \alpha)$-SuI $\mathcal{S}$. We define an $(n, \ell, \epsilon, \kappa, \alpha)$-SuI $\mathcal{S}$ of size $\min\{n, m|W|\}$, which consists of sets S_i, for $1 \leq i \leq \min\{n, m|W|\}$. There are two cases to consider, depending on the relation between n and $m|W|$. The case of $n \leq m|W|$ is simple: take the singleton containing only the i-th element of V as S_i. Consider a more interesting case when $n > m|W|$. For $i = am + b \leq m|W|$, where a and b are non-negative integers satisfying $a + b > 0$, let S_i contain all the nodes $v \in V$ such that v is a neighbor of the a-th node in W and $v \in T_b$.

Algorithm 3: Construction of $(n, \ell, \epsilon, \kappa, \alpha)$-Selectors-under-Interference (SuI).

Data: $(\ell\epsilon, d, \epsilon)$-disperser $G = (V, W, E)$, $V = \{v_1, \ldots, v_n\}$, $W = \{w_1, \ldots, w_{|W|}\}$,
$|W| = \ell d/\delta$, $\delta = O(\log^3 n)$, $d = O(\text{polylog } n)$,
$(n, 3\delta)$-strong-selector $\mathcal{T} = \{T_1, \ldots, T_m\}$
Result: $(n, \ell, \epsilon, \kappa, \alpha)$-SuI $\mathcal{S}$

```
for i ← 1 to min{n, m|W|} do
    if n > m|W| then
        S_i ← {v_i}
    else
        Find a, b > 0, such that i = am + b ≤ m|W| ;
        S_i ← T_b ∩ N_G(w_a)
return ⟨S_1, S_2, ..., S_{min{n,m|W|}}⟩
```

Proof. **of Theorem 3** First we show that the constructed $\mathcal{S}$ is an $(n, \ell, \epsilon, \kappa, \alpha)$-SuI. The case $n \leq m|W|$ is clear, since each element in a set K_1 of size at most ℓ occurs as a singleton in some set S_i (here it does not matter what the set K_2 is).

Consider the case $n > m|W|$. Let a set $K_1 \subseteq V$ be of size at most ℓ and a set K_2 of at most κ elements. Suppose, to the contrary, that there is a set $L \subseteq K_1$ of size ℓ^* such that none among the elements in L is K_1-selected by $\mathcal{S}$ under α-interference from K_2, that is, $S_i \cap L \neq \{v\}$ or $|S_i \cap K_2| \geq \alpha$, for any $v \in L$ and $1 \leq i \leq m|W|$. (Recall that $\ell^* = \ell\epsilon$.)

Claim: Every $w \in N_G(L)$ has more than 3δ neighbors in K_1 or at least α neighbors in K_2.

The proof is by contradiction. Suppose, to the contrary, that there is $w \in N_G(L)$ which has at most 3δ neighbors in K_1 and less than α neighbors in K_2, that is, $|N_G(w) \cap K_1| \leq 3\delta$ and $|N_G(w) \cap K_2| < \alpha$. By the former property and the fact that $\mathcal{T}$ is an $(n, 3\delta)$-strong-selector, we get that, for every $v \in N_G(w) \cap K_1$, the equalities

$$S_{w \cdot m + b} \cap K_1 = (T_b \cap N_G(w)) \cap K_1 = T_b \cap (N_G(w) \cap K_1) = \{v\}$$

hold, for some $1 \leq b \leq m$. This holds in particular for every $v \in L \cap N_G(w) \cap K_1$. There is at least one such $v \in L \cap N_G(w) \cap K_1$ because set $L \cap N_G(w) \cap K_1$ is nonempty, since $w \in N_G(L)$ and $L \subseteq K_1$. Additionally, recall that $N_G(w) \cap K_2$ is smaller than α. The existence of such v is in contradiction with the choice of L. Namely, L contains only elements which are not K_1-selected by sets from $\mathcal{S}$ under α-interference from K_2, but $v \in L \cap N_G(w) \cap K_1$ is selected from K_1 by some

set $S_{w \cdot m+b}$ and the interference from K_2 on this set is smaller than α. This contradiction makes the proof of the Claim complete. ■

Recall that $|L| = \ell^* = \ell\epsilon$. By dispersion, the set $N_G(L)$ is of size larger than $(1-\epsilon)|W|$. Consider two cases below – they cover all possible cases because of the above Claim.

Case 1: At least half of the nodes w in $N_G(L)$ have more than 3δ neighbors in K_1.

In this case, the total number of edges between the nodes in K_1 and $N_G(L)$ in graph G is larger than

$$\frac{1}{2}(1-\epsilon)|W| \cdot 3\delta = \frac{1}{2}(1-\epsilon)(\ell d/\delta) \cdot 3\delta > \ell d \,,$$

since $\frac{1}{2}(1-\epsilon) \cdot 3 > 1$ for $\epsilon < 1/3$. This is a contradiction, since the total number of edges in graph G incident to nodes in K_1 is at most $|K_1|d = \ell d$.

Case 2: More than half of the nodes w in $N_G(L)$ have at least α neighbors in K_2.

In this case, the total number of edges between the nodes in K_2 and $N_G(L)$ in graph G is larger than

$$\frac{1}{2}(1-\epsilon)|W| \cdot \alpha = \frac{1}{2}(1-\epsilon)(\ell d/\delta) \cdot \alpha > \kappa d \,,$$

because of assumptions $\epsilon < 1/3$ and $\ell > \kappa \cdot (3\delta)/\alpha$. This is a contradiction, since the total number of edges in graph G incident to nodes in K_2 is at most $|K_2|d = \kappa d$.

Thus, it follows from the contradictions in both cases that $\mathcal{S}$ is an $(n, \ell, \epsilon, \kappa, \alpha)$-SuI.

The size of this selector is

$$\begin{aligned} \min\{n, m|W|\} &= O\left(\min\left\{n, \delta^2 \log^2 n \cdot \ell^* d/\delta\right\}\right) \\ &= O\left(\min\left\{n, \ell\delta d \log^2 n\right\}\right), \end{aligned}$$

since $\ell = \Theta(\ell^*)$. It follows directly from the construction that each element is in $O(d\delta \log n)$ queries: d neighbors in the disperser multiplied by the number $O(\delta \log n)$ of occurrences in the strong selector. □

Sparser SuI for small ℓ compared to κ/α – Modification and proof of Theorem 4. What if $\alpha\ell \leq 3\delta\kappa$? We could modify the above construction as follows. If $\frac{\alpha\ell}{\kappa} \leq 3\delta$, we take the $(n, 3\delta\kappa/\alpha, \epsilon, \kappa, \alpha)$-SuI $\mathcal{S}$ from Theorem 3 and partition each set $S \in \mathcal{S}$ into the smallest number of sets of size at most α each. We denote the new sequence obtained from $\mathcal{S}$ by $\mathcal{S}|_\alpha$. To bound the total number η of sets in $\mathcal{S}|_\alpha$ we first note that $\eta \leq |\mathcal{S}| + \frac{\sum_{S \in \mathcal{S}} |S|}{\alpha}$. The total number of occurrences of elements in sets $S \in \mathcal{S}$ is upper bounded by $O(n \cdot d\delta \log n)$, by Theorem 3. Therefore, we obtain that $\eta = O\left(\min\left\{n, (\kappa/\alpha)d\delta \log^2 n\right\} + \frac{nd\delta}{\alpha} \log n\right)$. To show that $\mathcal{S}|_\alpha$ is an $(n, \ell, \epsilon, \kappa, \alpha)$-SuI, note that if in the original $(n, 3\delta\kappa/\alpha, \epsilon, \kappa, \alpha)$-SuI $\mathcal{S}$ an element v is α-selected from a set K_1 under interference from K_2 by some set $S_i \in \mathcal{S}$, where $|K_1| \leq \ell \leq 3\delta\kappa/\alpha$, it occurs in some of the new sets being in the partition of the original selecting set S_i. Since each partition (made during the construction of $\mathcal{S}|_\alpha$) is a subset of the original set, element v is also α-selected in $\mathcal{S}|_\alpha$ from K_1 under interference from K_2. Note that in the above decoding, the number of queries containing any element remains $O(d\delta \log n)$, as in the original SuI from Theorem 3 (as in the construction we only partition sets). Hence, by taking the construction of family $\mathcal{S}|_\alpha$, we proved Theorem 4.

C.2 Polynomial-time Construction of Strong-Selectors-under-Interference

In order to construct an $(n, \ell, \kappa, \alpha)$-SSuI $\mathcal{S}$, we use the following variation of a Reed-Solomon superimposed code, analogous to the construction used in [46], however here we prove an additional property of these objects.

1. Let $\mathsf{deg} = \lceil \log_\ell n \rceil$ and $q = c \cdot \ell \cdot \mathsf{deg}$ for some constant $c > 0$ such that $q^{\mathsf{deg}+1} \geq n$ and q is prime. Note that, by distribution of prime numbers, we could assume $c \in [3/2, 3)$.
2. Consider all polynomials of degree deg over field $\mathbb{F}_q$; there are $q^{\mathsf{deg}+1}$ such polynomials. Remove $q^{\mathsf{deg}+1} - n$ arbitrary polynomials and denote the remaining polynomials by $P_1, P_2, \ldots, P_n$.

3. Create the following matrix M of size $q \times n$. Each column i, for $1 \leq i \leq n$, stores values $P_i(x)$ of polynomial P_i for arguments $x = 0, 1, \ldots, q-1$; the arguments correspond to rows of M. Next, matrix M^* is created from M as follows: each value $y = P_i(x) \in \{0, 1, \ldots, q-1\}$ is represented and padded in q consecutive rows containing 0s and 1s, where 1 is exactly in $y+1$-st padded row, while in all other padded rows there are 0s. Notice that each column of M^* has q^2 rows (q rows per each argument), therefore M^* is of size $q^2 \times n$.
4. Set $T_i \subseteq [n]$, for $1 \leq i \leq q^2$, is defined based on row i of matrix M^*: it contains all elements $v \in [n]$ such that $M^*[j, v] = 1$. (Recall that each such v corresponds to some polynomial.) For a fixed constant $c > 0$, $\{T_i\}_{i=1}^{q^2}$ forms a family $\mathcal{T}^{(c)}$ of subsets of set $\{1, \ldots, n\}$.

The above construction could be presented as a simplified pseudocode, see Algorithm 4.

Algorithm 4: Construction of Strong-Selectors-under-Interference

```
deg ← ⌈log_ℓ n⌉;
q ← c · ℓ · deg for some constant c ∈ [3/2, 3) such that q^(deg+1) ≥ n and q is prime;
/* There are q^(deg+1) ≥ n such polynomials.                                  */
P_1, P_2, ..., P_n ← arbitrary n polynomials of degree deg over field F_q;
T^(c) ← sequence of q^2 empty sets {T_i}_{i=1}^{q^2};
for i ← 1 to n do
    for x ← 0 to q − 1 do
        /* Value of i-th polynomial for argument x.                            */
        value ← P_i(x);
        /* Encode value in unary on positions x·q+1, x·q+2, ... (x+1)·q        */
        index ← x · q + value + 1;
        /* Add element i to the corresponding set T.                           */
        T_index.add(i)
return T^(c)
```

Proof. **of Theorem 5** Consider the constructed family $\mathcal{T}^{(c)}$. Polynomial time of this construction follows directly from the fact that the space of polynomials over field $[q]$ is of polynomial size in n and all the operations on them are polynomial. The length follows from the fact that it is $q^2 < 3^2 \ell^2 \log_\ell^2 n = O(\ell^2 \log_\ell^2 n)$.

Recall that each element $v \in [n]$ correspond to some polynomial of degree at most deg over $\mathbb{F}_q$. Note that two polynomials P_i and P_j of degree deg with $i \neq j$, can have equal values for at most deg different arguments. This is because they have equal values for arguments x for which $P_i(x) - P_j(x) = 0$. However, $P_i - P_j$ is a polynomial of degree at most deg, so it can have at most deg zeroes. Hence, $P_i(x) = P_j(x)$ for at most deg different arguments x.

Take any polynomial P_i and any other at most $\ell - 1$ polynomials P_j, which altogether form set K_1 of at most ℓ polynomials. There are at most $(\ell - 1) \cdot \mathsf{deg}$ different arguments where one of the other $\ell - 1$ polynomials can be equal to P_i. Hence, for at least $q - (\ell - 1) \cdot \mathsf{deg}$ different arguments, the values of the polynomial P_i are different than the values of the other polynomials in K_1. Let us call the set of these arguments A.

Consider any set $K_2 \subseteq \{1, \ldots n\}$ of at most κ elements (corresponding to polynomials). Consider arguments from set A for which P_i has the same value as at least α other polynomials in K_2. The number of such arguments is at most

$$\frac{\kappa \cdot \mathsf{deg}}{\alpha} \leq \frac{\ell}{3\delta} \cdot \mathsf{deg} < \frac{\ell}{2} \cdot \mathsf{deg} \leq (c-1)\ell \cdot \mathsf{deg} < q - (\ell - 1) \cdot \mathsf{deg}\ ,$$

which means it is smaller than $|A|$, for $c \in [3/2, 3)$ in the definition of $q = c\ell \cdot \mathsf{deg}$. Therefore, there is an argument (in set A) such that the value of P_i is different from the values of all other $\ell - 1$ polynomials in K_1 *and* less than α polynomials in set K_2. As this holds for an arbitrary polynomial

P_i in an arbitrary set K_1 of at most ℓ polynomials (in total) and an arbitrary set K_2 of at most κ polynomials, $\mathcal{T}^{(c)}$ is an $(n, \ell, \kappa, \alpha)$-SSuI. Finally, it follows directly from the construction that every element occurs in $q = O(\ell \log_\ell n)$ queries. $\square$