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 $O(\min\left\{n, \ell \operatorname{polylog} n\right\})$, 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 $\alpha\ell \le 3\delta\kappa$? We could modify the above construction as follows. If $\frac{\alpha\ell}{\kappa} \le 3\delta$, 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 $\eta \le |\mathcal{S}| + \frac{\sum_{S\in\mathcal{S}}|S|}{\alpha}$. 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 $|K_1| \le \ell \le 3\delta\kappa/\alpha$, 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,ℓ,κ,α)-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 $q^{\mathsf{deg}+1} \ge n$ 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. $\square$