SAFFRON: A Fast, Efficient, and Robust Framework for Group Testing Based on Sparse-Graph Codes

Kangwook Lee, Kabir Chandrasekher, Ramtin Pedarsani, and Kannan Ramchandran

***Abstract*—Group testing is the problem of identifying K defective items among n items by *pooling* groups of items. In this paper, we design group testing algorithms for approximate recovery with order-optimal sample complexity by leveraging design and analysis tools from modern sparse-graph coding theory. Our algorithm, SAFFRON, recovers at least $(1-\varepsilon)K$ defective items w.p. $1-K/n^r$ with $m=2(1+r)C(\varepsilon)K\log_2 n$ tests, where ε is an arbitrarily small constant, $C(\varepsilon)$ is a precisely characterizable constant, and r is any positive integer. The decoding complexity is $\Theta(K\log n)$. We also propose variations of SAFFRON, which are robust to noise and unknown offsets. For example, for $n\simeq 4.3\times 10^9$ and $K=128$, our algorithm is observed to recover all defective items with $m\simeq 8.3\times 10^5$ tests, even in the presence of noisy test results. Moreover, the decoding time takes less than 4 seconds on a laptop with a 2 GHz Intel Core i7 and 8 GB memory.**

***Index Terms*—Group testing, compressed sensing, sparse-graph codes, sub-linear decoding complexity.**

I. Introduction

GROUP testing tackles the problem of identifying a population of K defective items from a set of n items by *pooling* groups of items in order to reduce the number of tests needed. The result of a pooled test is *positive* if any of the items in the pool is defective and *negative* otherwise. The goal is to judiciously group subsets of items such that defective items can be reliably recovered using a small number of tests, while having a low-complexity decoding procedure.

Group testing arose during the Second World War [3]: in order to detect all soldiers infected with the syphilis virus without individually testing them, the blood samples of subsets of soldiers were pooled together and tested as groups. This testing mechanism is cost-efficient since a negative test result can tell that all the individuals in the tested group are *not* infected with the virus. Since then, varied theoretical aspects of group testing have been studied, and more applications of group testing have been discovered in a variety of fields spanning across signal processing [4]–[7], computer science [8], biology [9], machine learning [10], [11], and medicine [12].

Among a variety of applications, group testing is particularly relevant to compressed sensing (CS) [13], [14]. The main difference between group testing and CS is that Boolean OR over binary signals is assumed in group testing problems while addition over real-valued or complex-valued signals is assumed in CS. Except for that difference, the two problems are identical: the goal is to recover a high-dimensional signal from the fewest number of low-dimensional projections or measurements. Indeed, quantitative group testing is one variant of group testing that is even more relevant to CS: Each pool reveals the number of defective items in the pool, i.e., the pool operator is addition instead of Boolean OR [15, Part III]. Note that this problem is essentially identical to a variant of CS called Binary CS (BCS), where the input signal is binary-valued [16]. An adaptive version of quantitative group testing problem has been also studied in [7], [17]–[19].

A. Problem Formulation

We now formally define the problem. Consider n items, among which exactly K items are defective. The locations of the defective items can be compactly written as a length-n binary vector $\boldsymbol{x}=(x_1,x_2,\ldots,x_n)^\intercal\in\{0,1\}^n$, where x_i is 1 if and only if item i is defective for $1\le i\le n$. Defining supp $(\cdot)$ as the set of indices of non-zero elements, $|\text{supp}\,(\boldsymbol{x})\,|=K$.

A subset of items can be pooled and tested, and the test result is either 1 (positive) if it includes at least one defective item, or 0 (negative) otherwise. For notational simplicity, we denote a subset by a binary vector, $\boldsymbol{a}=(a_1,a_2,\ldots,a_n)^\intercal\in\{0,1\}^n$, where a_i is 1 if and only if item i belongs to the subset. Following the notation of [5], a group testing result y can be denoted by

$$y=\boldsymbol{a}^\intercal\boldsymbol{x}:=\bigvee_{i=1}^{n}a_ix_i=\bigvee_{a_i=1}x_i, \tag{1}$$

where multiplication of 0's and 1's is the usual multiplication but addition is replaced by the logical OR. We represent m pools $\boldsymbol{a}_1,\boldsymbol{a}_2,\ldots,\boldsymbol{a}_m$ by a matrix $A=(\boldsymbol{a}_1^\intercal,\boldsymbol{a}_2^\intercal,\ldots,\boldsymbol{a}_m^\intercal)^\intercal\in$

Manuscript received July 30, 2018; revised December 3, 2018, March 19, 2019, and June 21, 2019; accepted June 25, 2019. Date of publication July 24, 2019; date of current version August 9, 2019. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Vaughan Clarkson. This work was supported by the National Science Foundation under Grants CCF-1527767 and CCF-1755808. This paper was presented in part at the IEEE International Symposium on Information Theory, Barcelona, Spain, July 2016 [1] and in part at the IEEE International Conference on Communications, Paris, France, May 2017. *(Corresponding author: Kangwook Lee.)*

K. Lee is with the Department of Electronics and Communication Engineering, University of Wisconsin–Madison, Madison, WI 53706 USA (e-mail: kangwook.lee@wisc.edu).

K. Chandrasekher is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: kabirchandrasekher@gmail.com).

R. Pedarsani is with the Department of Electronics and Communication Engineering, University of California Santa Barbara, Santa Barbara, CA 93106 USA (e-mail: ramtin@ece.ucsb.edu).

K. Ramchandran is with the Department of Electrical Engineering and Computer Science, University of California—Berkeley, Berkeley, CA 94720 USA (e-mail: kannanr@eecs.berkeley.edu).

Digital Object Identifier 10.1109/TSP.2019.2929938

$\{0,1\}^{m\times n}$. We call this matrix *the group testing matrix*. Then, the group testing results can be compactly written as $\boldsymbol{y} := A\boldsymbol{x}$.

The goal of group testing is to design a matrix A such that 1) the number of tests m is as low as possible, and at the same time, 2) one can efficiently recover $\boldsymbol{x}$ from $\boldsymbol{y}$. More specifically, there should exist a decoding function, denoted by $g_A : \{0,1\}^m \to \{0,1\}^n$, such that $\hat{\boldsymbol{x}} := g_A(\boldsymbol{y})$ is efficiently computable, and $\hat{\boldsymbol{x}} \approx \boldsymbol{x}$ with provable guarantees. In this work, we focus on the *non-adaptive* setting, i.e., A needs to be designed before observing any test outcomes.

Depending on different applications, one can define different recovery objectives. The most stringent objective is *exact recovery*: $\hat{\boldsymbol{x}} = \boldsymbol{x}$. A slightly looser objective is *ε-partial-recovery without false detections*: one wants to make sure that $|\text{supp}(\hat{\boldsymbol{x}}) \setminus \text{supp}(\boldsymbol{x})| = 0$ and $|\text{supp}(\boldsymbol{x}) \setminus \text{supp}(\hat{\boldsymbol{x}})| \leq \varepsilon K$.

B. Our Contributions

Our goal is to design an efficient group testing matrix A that allows for computationally efficient decoding algorithm for partial recovery without false detections. More specifically, our goal is to have a decoding complexity of sublinear in n.

In this work, we propose SAFFRON, a non-adaptive, approximate group testing framework. Leveraging the tools of sparse-graph coding theory and powerful tools like density evolution techniques [20], we precisely characterize the measurement complexity of SAFFRON. We also show that SAFFRON can be made robust to noise in a modular method. Furthermore, we also formulate the asynchronous group testing problem, targeting the neighbor discovery application in the Internet of Things (IoT) systems. We then propose a SAFFRON-based solution, dubbed as A-SAFFRON, and show that it can reliably find the location of defective items even when items are pooled with *unknown offsets*, which will be formally described in Section V-A.

In particular, SAFFRON can achieve ε-partial-recovery without false defectives with $m = 2(1+r)C(\varepsilon)K\log_2 n$ tests, w.p. $1 - K/n^r$, where $C(\varepsilon)$ is a precisely characterizable constant (See Table II). Here, r is an arbitrary positive integer, so one can easily trade off the measurement complexity with the error probability. The decoding complexity is $\Theta(K\log n)$.

To show the usefulness of SAFFRON, consider a case where $K = 128$ defective items have to be recovered from a set of $n \approx 4.3\times 10^9$ items. Let us also assume that each test result is randomly flipped with probability of 2%, i.e., noisy group testing. Under this setting, our simulation results show that SAFFRON recovers *all* the $K = 128$ defective items with $m \approx 8.3\times 10^5$ tests in 3.8 seconds on average, when run on a standard laptop. To the best of our knowledge, this is the first group testing algorithm that can solve *billion-scale* noisy group testing problems in seconds.

Remark 1 (Practical Impact of SAFFRON): While SAFFRON features efficient decoding (sublinear in n) algorithm, it has a few limitations. First, though order-optimal for a fixed value of ε, there exists a constant factor gap between the number of tests required by SAFFRON and the fundamental limit. (See Remark 6 for a detailed discussion.) On the other hand, there exist other group testing algorithms that 1) run 'linear' in n and 2) closely match the fundamental limit in terms of test complexity [21], [22]. When n is extremely large, the decoding complexity will clearly become the bottleneck of the entire group testing procedure in any cases, and this is when SAFFRON can play a critical role.

Furthermore, under the SAFFRON scheme, the average number of items included in each pool is $\Theta(n/K)$ while the average number of defective items in each pool is $\Theta(1)$. Thus, SAFFRON is useful when even a single defective item pooled with a large number of non-defective items can make the test result positive without being diluted.

The neighbor discovery problem in IoT systems is a particular application that satisfy both conditions: n is large, and non-defective items correspond to inactive IoT nodes, so they do not *dilute* test results. Another critical application is digital forensics [8]. In such applications, n is large since it corresponds to the size of database. Furthermore, a pooled test corresponds to an integrity check of concatenated data blocks (or concatenated code blocks), so a single error alone makes the test result positive.

C. Related Work

We briefly overview the existing work in the literature, focusing on the non-adaptive group testing problem. We refer the readers to [9], [21], [23] for a more comprehensive survey.

1) Combinatorial Group Testing: For group testing algorithms with zero-error reconstruction, i.e., combinatorial group testing, the best asymptotic lower bound (for fixed K and asymptotic n) on the number of required tests is $\Theta(K^2\log n/\log K)$, and the best asymptotic upper bound is $\Theta(K^2\log n)$ [24], [25]. Although the zero-error reconstruction property is definitely a desired property, such group testing schemes typically involve exhaustive table searches in their reconstruction procedures, and hence require a high computational and memory complexity of $O(Kn\log n)$ [26]. In [27], Cheraghchi develops a noisy group testing matrix with $O(K\log n)$, allowing for $O(Kn\log n)$ decoding, which may yield up to $O(K)$ false positive. In [28], the authors propose a scheme that requires $O(K^2\log n)$ tests, while having an efficient decoding algorithm of computational complexity $\text{poly}(K)\log n\log^2(K^2\log n) + O(K^4\log^2 n)$, which is sublinear in n if K is a sufficiently small power of n.

2) Probabilistic and Approximate Group Testing: Several relaxations of the group testing problem have been studied in the literature. One such relaxation is allowing a small error probability as well as relaxing the requirement of perfect identification. That is, the goal is to design a group testing scheme that identifies an approximate answer with high probability. Many approaches have been proposed to design group testing schemes that allow an efficient decoding algorithm for these relaxed yet important problems. One such approach is random pooling design based on random bipartite graphs. For instance, randomized group testing schemes based on constant test-per-item design and constant items-per-test design are studied, respectively in [29] and [30]. Other lines of work have made use of existing pooling designs. In [26], [31], the authors use randomly chosen pools from carefully designed pools that are initially tailored for a zero-error reconstruction setting. With certain success probabilities, these schemes find a large fraction of the K defective items, while wrongly identifying a small fraction of normal items as defective items, which can be efficiently identified as false positive in a subsequent round of trivial tests. Despite the simplicity of this class of constructions, performance analysis is rather convoluted

and cumbersome. Further, these schemes are generally difficult to make robust to noise.

Another line of work takes an information-theoretic approach. That is, one assumes a prior distribution on the set of defective items, and designs a group testing scheme that succeeds with probability approaching one. In [32], the authors study the noisy group testing problem under this setup. Recently, Atia and Saligrama show that $\Theta(K \operatorname{polylog} n)$ tests is sufficient [33]. Chan *et al.* propose novel group testing algorithms and compare their performances with the fundamental lower bounds [34]. Scarlett and Cevher precisely characterize the phase transitions in group testing when K is a small power of n [35], [36]. Mazumdar presents near-optimal explicit constructions in [21]. Aldrige *et al.* propose two new algorithms that are superior under certain regimes of sparsity [22] compared to [34].

In [37], generalizing the definition of *disjunct* matrices, i.e., testing matrices that allow for efficient non-adaptive group testing, the authors propose (K, ε)-disjunct matrices, and reveal their connection to error correcting codes. While most of these works assume that the number of defectives is binomially distributed, Emad and Milenkovic study the case where the number of defective follows the Poisson distribution and propose a near-optimal test matrix design [6].

3) Computationally-Efficient Group Testing: Several works have proposed group testing schemes with efficient decoding algorithms. In [5], the authors present a simple group testing procedure that efficiently recovers a large fraction of K defective items with $O(K \log^2 n)$ tests. While the proposed decoding algorithm runs in time $(K \log n)^{O(1)}$, the algorithm returns $O(K \log n)$ false positives, which need to be double-checked using a two-stage algorithm.

In [38], Cai *et al.* propose GROTESQUE, a class of efficient group testing schemes, which is the first adaptive group testing algorithm that achieves both an order-optimal number of tests and an order-optimal decoding complexity, but at the cost of using $O(\log K)$ adaptive stages. Their non-adaptive scheme requires $O(K \log K \log n)$ tests, and has a decoding complexity of $O(K(\log n + \log^2 K))$. When applied to the synchronous case, our A-SAFFRON algorithm recovers the sample complexity (up to a constant factor) and the computational complexity of the GROTESQUE algorithm, and hence one can view it as an asynchronous extension of the GROTESQUE algorithm.

D. Sparse-Graph Codes for Group Testing

Sparse-graph codes form the backbone of reliable modern communication systems [20]. In [39], the authors analyze the performance of a belief propagation algorithm for group testing in a similar way one would analyze the belief propagation decoding for Low-Density-Parity-Check (LDPC) codes [20]. As a result, the authors characterize the phase transition behaviors of certain algorithms. In [40], the author derives some impossibility results for the group testing problem by leveraging LDPC-inspired designs and proof techniques, but the results were later retracted due to some errors. While the existing works directly apply sparse-graph codes as a group testing matrix, SAFFRON is based on a novel combination of a sparse-graph code and efficient singleton/doubleton recovery codes, allowing for efficient sublinear decoding complexity.

TABLE I
SUMMARY OF THE NOTATION

Notation	Definition
n	Number of items
K	Number of defective items
m	Number of pools (tests)
h	Number of tests in each test group
M	Number of right nodes (test groups)
$\boldsymbol{x}$	Binary representation of the set of defective items
$\boldsymbol{y}$	Binary representation of the group test results
$\boldsymbol{z}_i$	Measurements from the i^{th} right node
$\boldsymbol{z}_i^j$	Measurements from the j^{th} stage of the i^{th} right node
$\boldsymbol{u}_j^{(i)}$	Signature vector of the j^{th} item for the i^{th} right node
$U^{(i)}$	$[\boldsymbol{u}_1^{(i)}, \boldsymbol{u}_2^{(i)}, \ldots, \boldsymbol{u}_{n-1}^{(i)}, \boldsymbol{u}_n^{(i)}]$
$\boldsymbol{b}_i$	Binary representation of $i-1$
L	The length of $\boldsymbol{b}_i$
$\boldsymbol{e}_i$	The i^{th} standard basis vector
$\mathcal{G}$	Bipartite graph representation of a sparse-graph code
$T_{\mathcal{G}}$	The incidence matrix of $\mathcal{G}$
q	Probability of each test outcome being wrong

E. Paper Organization

The rest of the paper is organized as follows. We introduce our SAFFRON framework in Section II, and provide its theoretical guarantees in Section III. In Section IV, we robustify SAFFRON so that they can reliably recover the defective items even with noisy test results. In Section V, we formulate the asynchronous group testing problem, targeting the neighbor discovery application in IoT systems. We then propose A-SAFFRON and analyze its performance. Lastly, we provide simulation results in Section VI, corroborating our guarantees and demonstrating the practicality of the SAFFRON framework.

F. Notations

For a positive integer n, $[n] := \{1, 2, \ldots, n\}$. Following the MATLAB notation, $[A; B] := [A^\intercal, B^\intercal]^\intercal$. We denote the set of the indices of the non-zero elements by $\operatorname{supp}(\cdot)$. The Hamming weight or the number of ones of a binary vector is denoted by $w(\cdot)$. The i^{th} element of vector $\boldsymbol{x}$ is denoted by x_i, and the bit-wise complement vector of $\boldsymbol{x}$ is denoted by $\overline{\boldsymbol{x}}$. Table I summarizes our notation, which will be defined throughout the paper.

II. THE SAFFRON SCHEME

Before we describe our scheme, we first describe the high-level difference between our approach and the existing ones. Many of the efficient decoding algorithms proposed in the literature are designed to identify non-defective items. For instance, Combinatorial Orthogonal Matching Pursuit (COMP), proposed in [34], finds a set of definite non-defective items: If an item is pooled in a test but the test result is negative, the item must be non-defective. Definite Defectives (DD), proposed in [22], first runs COMP, obtaining a set of items that are *not* marked as non-defective. Note that if some positive test contains exactly one item from this set, the item is definitely defective. DD finds a list of such items, declaring them to be defective items. While these algorithms are shown to succeed with a small number of tests close to the lower bound, their computational complexities are inevitably $\Omega(n)$. Particularly, if $K = o(n)$, finding a list of non-defective items takes at least $\Theta(n - K) = \Theta(n)$ operations. Furthermore, these algorithms require a full scan of A, so the computational complexities are linear in the number of ones in it. Under a reliable recovery condition, this leads to

$\Omega(n \log n)$ complexity. In order to have a sublinear decoding time, we judiciously design our group testing matrix such that 1) a full scan of A is not required while decoding and 2) defective items can be identified without needing to find non-defective items beforehand.

To achieve this properties, we rely on an architectural philosophy that is similar to the ones in [41]–[47]. The key idea of SAFFRON is the adoption of a design principle called '*sparse signal recovery via sparse-graph codes*', which has been applied to a varied class of problems such as computing a sparse Fast Fourier Transform [48], a sparse Walsh Hadamard Transform [49], and the design of systems for compressive sensing [50] and compressive phase-retrieval [51]. In all these problems, one designs an efficient way of sensing or *measuring* an unknown sparse signal such that the decoder can estimate the unknown signal with a low decoding complexity. The overarching design principle is to 1) design a sensing matrix based on a sparse bipartite graph and to 2) decode the measurements using a simple peeling decoder. Since the group testing problem can also be viewed as a sparse signal recovery problem, we will follow this design principle to design an efficient solution.

In our framework, the m tests are arranged into M groups of size h, i.e., $m = Mh$. We then design a bipartite graph with n left nodes and M right nodes, in which left nodes correspond to items, and right nodes correspond to test groups. Particularly, we will construct our bipartite graph based on left-regular construction. That is, each left node is connected to constant number d of right nodes uniformly at random. We note that left-regular bipartite graphs have been shown useful for improving sample efficiency in group testing [39], [52].

We denote the incidence matrix of a bipartite graph $\mathcal{G}$ by $T_{\mathcal{G}} \in \{0,1\}^{M \times n}$, or simply T if $\mathcal{G}$ is clear from the context. Let $\boldsymbol{t}_i$ be the i^{th} row of $T_{\mathcal{G}}$. We associate each left node with M *signature* (column) vectors of length h. We denote the M signature vectors of item j by $[\boldsymbol{u}_j^{(i)}]_{i=1}^{M}$, where $\boldsymbol{u}_j^{(i)} \in \{0,1\}^h$. We define M signature matrices $[U^{(i)}]_{i=1}^{M}$, where $U^{(i)} := [\boldsymbol{u}_1^{(i)}, \boldsymbol{u}_2^{(i)}, \ldots, \boldsymbol{u}_{n-1}^{(i)}, \boldsymbol{u}_n^{(i)}] \in \{0,1\}^{h \times n}$ for all i. Here, $U^{(i)}$ is called the signature matrix for the i^{th} right node.

Given a graph $\mathcal{G}$ and signature matrices $[U^{(i)}]_{i=1}^{M}$, we design our group testing matrix A to be $[A_1; A_2; \ldots; A_M] \in \{0,1\}^{hM \times n}$, where $A_i = U^{(i)}\text{diag}(\mathbf{t}_i) \in \{0,1\}^{h \times n}$, and $\text{diag}(\cdot)$ is the diagonal matrix constructed by the input vector. As an example, the signature matrix designed with $T = [0,1,0; 1,1,0; 0,0,1]$ and $U^{(1)} = U^{(2)} = U^{(3)} = [1,0,1; 0,1,1]$ is

$$A = \left[\begin{array}{ccc} 0 & \mathbf{0} & 0 \\ 0 & \mathbf{1} & 0 \\ \hline \mathbf{1} & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{1} & 0 \\ \hline 0 & 0 & \mathbf{1} \\ 0 & 0 & \mathbf{1} \end{array}\right]. \tag{2}$$

For notational simplicity, we define the observation vector corresponding to right node i as $\boldsymbol{z}_i := \boldsymbol{y}_{(i-1)h+1:ih}$ for $1 \le i \le M$. Then,

$$\boldsymbol{z}_i = U^{(i)} \cdot \text{diag}(\boldsymbol{t}_i)\boldsymbol{x} = \bigvee_{(\boldsymbol{t}_i)_j = 1, x_j = 1} \boldsymbol{u}_j^{(i)},\ 1 \le i \le M. \tag{3}$$

That is, $\boldsymbol{z}_i$ is the bitwise logical ORing of all the signatures of the active left nodes that are connected to right node i.

Our decoding algorithm simply iterates through all the right node measurement vectors $\{\boldsymbol{z}_i\}_{i=1}^{M}$, and checks whether a right node is *resolvable* or not. A right node is resolvable if exactly one new defective item can be detected by processing the right node, i.e., the location index of the defective item is found. The decoding algorithm is terminated when there are no more resolvable right nodes.

We now present the following terminologies. A right node that is connected to one and only one defective item is called a *singleton*. A right node that is connected to two defective items is called a *doubleton*. Later, we show that with the aid of our signature matrix, 1) a singleton is resolvable, and 2) a doubleton is resolvable if one of the two defective items is already identified (in the previous iterations of the algorithm).

A. Detecting and Resolving a Singleton

Consider the following signature matrix V where the i^{th} column is a vertical concatenation of $\boldsymbol{b}_i$ and its complement $\overline{\boldsymbol{b}_i}$, where $\boldsymbol{b}_i$ is the L-bits binary representation of an integer $i-1$, for $i \in [n]$, where $L = \lceil \log_2 n \rceil$.[1]

$$V = \left[\begin{array}{cccc} \boldsymbol{b}_1 & \boldsymbol{b}_2 & \cdots & \boldsymbol{b}_n \\ \overline{\boldsymbol{b}_1} & \overline{\boldsymbol{b}_2} & \cdots & \overline{\boldsymbol{b}_n} \end{array}\right] = \left[\begin{array}{cccc} 0 & 0 & \ldots & 1 \\ 0 & 0 & \ldots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \\ 0 & 1 & \ldots & 1 \\ \hline 1 & 1 & \ldots & 0 \\ 1 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \ldots & 0 \\ 1 & 0 & \ldots & 0 \end{array}\right] \tag{4}$$

We now show that a singleton can be detected and resolved with the aid of this signature matrix. First, note that the sum of the weight of any binary vector and the weight of its complement is always the length of the vector. Thus, given a singleton, the weight of the measurement vector is L. Further, one can also read the first half of the measurement of the detected singleton to find the index location of the defective item. We refer to this procedure as *the singleton recovery algorithm. The following proposition characterizes the recovery guarantee of the singleton recovery algorithm.*

Proposition 1: Consider a fixed right node whose signature matrix is V. The singleton recovery algorithm correctly detects a singleton and identifies the defective item connected to the right node w.p. 1. If the right node is not a singleton, the algorithm does not declare a false defective item.

Proof: The first part of the statement is evident from the algorithm description, and hence we focus on the case where the right node is a zeroton or a multiton. If the node is a zeroton, the weight of its vector will be 0, and hence the algorithm will not declare a defective item. For the case of multiton, we will show that the weight of a doubleton's measurement vector is always

[1] For simplicity, the rest of paper will assume that n is a power of 2, and hence $L = \log_2 n$.

strictly larger than L. To see this, consider a doubleton consisting of item i and j, $i \neq j$. Denote the set of 1's locations in $\boldsymbol{b}_i$ by S_i. Since $i \neq j$, either $S_j \setminus S_i \neq \emptyset$ or $S_j \subsetneq S_i$ holds. If the former holds, $\|\boldsymbol{b}_i + \boldsymbol{b}_j\|_0 > \|\boldsymbol{b}_i\|$. If the latter holds, $[L] \setminus S_i \subsetneq [L] \setminus S_j$, and hence $\|\overline{\boldsymbol{b}}_i + \overline{\boldsymbol{b}}_j\|_0 > \|\overline{\boldsymbol{b}}_i\|$. This proves that the weight of a doubleton's measurement vector is strictly larger than L. Hence, the recovery algorithm does not declare a false defective item when the right node is a doubleton. Since multitons with degree larger than 2 will have at least the same weight as that of a doubleton, the statement is proved. ■

B. Detecting and Peeling Doubletons

We now design the full signature matrix U_r by expanding V so that one can detect and resolve both singletons and resolvable doubletons. Here, r denotes the number of extra check stages used, determining the reliability of the doubleton decoding part. A formal guarantee will be given in Lemma 2 later in this section.

We first define some useful notations. Given a mapping $i : [n] \to [n]$, we define

$$V_i := \begin{bmatrix} \boldsymbol{b}_{i(1)} & \boldsymbol{b}_{i(2)} & \boldsymbol{b}_{i(3)} & \dots & \boldsymbol{b}_{i(n-1)} & \boldsymbol{b}_{i(n)} \\ \overline{\boldsymbol{b}_{i(1)}} & \overline{\boldsymbol{b}_{i(2)}} & \overline{\boldsymbol{b}_{i(3)}} & \dots & \overline{\boldsymbol{b}_{i(n-1)}} & \overline{\boldsymbol{b}_{i(n)}} \end{bmatrix}. \quad (5)$$

That is, V_i is a column-permuted version of V as per i. We are now ready to define the full signature matrix U_r. We first randomly draw r random mappings: $i_1, i_2, \dots, i_r$. Specifically, for all $j \in [r]$ and $k \in [n]$, $i_j(k)$ is drawn uniformly at random from $[n]$, independently of others. Then, we define $U_r := [V; V_{i_1}; V_{i_2}; \cdots; V_{i_r}]$.

For illustrative purpose, let us consider U_1.

$$U_1 = \begin{bmatrix} V \\ V_{i_1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 & \boldsymbol{b}_2 & \boldsymbol{b}_3 & \dots & \boldsymbol{b}_{n-1} & \boldsymbol{b}_n \\ \overline{\boldsymbol{b}_1} & \overline{\boldsymbol{b}_2} & \overline{\boldsymbol{b}_3} & \dots & \overline{\boldsymbol{b}_{n-1}} & \overline{\boldsymbol{b}_n} \\ \boldsymbol{b}_{i_1(1)} & \boldsymbol{b}_{i_1(2)} & \boldsymbol{b}_{i_1(3)} & \dots & \boldsymbol{b}_{i_1(n-1)} & \boldsymbol{b}_{i_1(n)} \\ \overline{\boldsymbol{b}_{i_1(1)}} & \overline{\boldsymbol{b}_{i_1(2)}} & \overline{\boldsymbol{b}_{i_1(3)}} & \dots & \overline{\boldsymbol{b}_{i_1(n-1)}} & \overline{\boldsymbol{b}_{i_1(n)}} \end{bmatrix}$$

Then, the measurement vector for the k^{th} right node, $\boldsymbol{z}_k$, is

$$\boldsymbol{z}_k = U_1 \cdot \mathrm{diag}(\boldsymbol{t}_k)\boldsymbol{x} = \begin{bmatrix} V \cdot \mathrm{diag}(\boldsymbol{t}_k)\boldsymbol{x} \\ V_{i_1} \cdot \mathrm{diag}(\boldsymbol{t}_k)\boldsymbol{x} \end{bmatrix} = \begin{bmatrix} \boldsymbol{z}_k^0 \\ \boldsymbol{z}_k^1 \end{bmatrix}, \quad (6)$$

where we defined $\boldsymbol{z}_k^0 := V \cdot \mathrm{diag}(\boldsymbol{t}_k)\boldsymbol{x}$ and $\boldsymbol{z}_k^j := V_{i_j} \cdot \mathrm{diag}(\boldsymbol{t}_k)\boldsymbol{x}$. We call $\boldsymbol{z}_k^j$ the j^{th} section of the k^{th} right-node measurement vector.

Example 1: Assume $\boldsymbol{x} = \boldsymbol{e}_2 + \boldsymbol{e}_3 + \boldsymbol{e}_{n-1}$ and $\boldsymbol{t}_k = \boldsymbol{e}_2 + \boldsymbol{e}_{n-2} + \boldsymbol{e}_{n-1}$. That is, items 2, 3, and $n-1$ are defective items, and right node k is connected to items 2, $n-2$, and $n-1$. Thus, this right node is a doubleton connected with items 2 and $n-1$. Assuming U_1, the measurement vector consists of 2 sections: $\boldsymbol{z}_k^0 = [\boldsymbol{b}_2 \vee \boldsymbol{b}_{n-1}; \overline{\boldsymbol{b}_2} \vee \overline{\boldsymbol{b}_{n-1}}]$ and $\boldsymbol{z}_k^1 = [\boldsymbol{b}_{i_1(2)} \vee \boldsymbol{b}_{i_1(n-1)}; \overline{\boldsymbol{b}_{i_1(2)}} \vee \overline{\boldsymbol{b}_{i_1(n-1)}}]$.

We now describe how one can resolve a doubleton using U_r. We first describe the doubleton recovery procedure assuming $r = 1$. Assume that right node k is connected to exactly one identified defective item, say ℓ. The decoder first *assumes* that the right node is a resolvable doubleton that is connected to item ℓ and an unidentified defective item ℓ'. Let us first consider the first section: $\boldsymbol{z}_k^0 = [\boldsymbol{b}_\ell \vee \boldsymbol{b}_{\ell'}; \overline{\boldsymbol{b}_\ell} \vee \overline{\boldsymbol{b}_{\ell'}}]$. Note that one can always recover every bit of $\boldsymbol{b}_{\ell'}$ from $\boldsymbol{z}_k^0$ as follows. Consider $(b_{\ell'})_1$, the first bit of $\boldsymbol{b}_{\ell'}$. If $(b_\ell)_1 = 0$, then $(b_{\ell'})_1 = (z_k^0)_1$. If not, one can read the first bit from the second half of the section: $(b_{\ell_0})_1 = \overline{(z_k^0)_{L+1}} = 1 - (z_k^0)_{L+1}$. This procedure can be applied to every other bit of $\boldsymbol{b}_{\ell'}$. Denote the result of this procedure by s_0.

After the decoder recovers s_0, it then computes $\mathbf{b}_{i(\ell)}$. By applying the same procedure to the second section, the decoder can obtain another index, which we call s_1.

The decoder now checks whether $i_1(s_0) = s_1$. If this is the case, it claims that the assumption was correct, declaring a new defective item s_0. In general cases where $r > 1$, the doubleton recovery algorithm declares a new defective item only when all of the r additional check equations hold.

While this procedure always outputs a correct index when applied to a resolvable doubletons, i.e., $s_0 = \ell'$ w.p. 1, it could wrongly declare a defective item when applied to a right node that is connected to more than 2 defective items. We now state a formal performance guarantee.

Lemma 2: Consider a fixed right node whose signature matrix is U_r, whose index mappings are drawn at random, independently of others. Further assume that it is connected to one identified defective item. If the right node is indeed a doubleton, the doubleton recovery algorithm recovers the index of the other defective item w.p. 1. If the right node is connected to more than 2 defective items, the doubleton recovery algorithm declares a wrong defective item w.p. $1/n^r$.

Proof: The first half of the statement is shown in the above decoding algorithm description, and hence we focus on proving the second part. Consider a right node connected to more than 2 defective items. Assume that the index read from the j^{th} stage is s_j for $j \in \{0, 1, \dots, r\}$. Note that s_0 is not random: It is a deterministic function of the indices of defective items connected to the right node. We consider two cases. Assume that item s_0 is indeed one of those defective items connected to the right node. In this case, the doubleton recovery algorithm may declare item s_0 as defective but this is a benign error. If item s_0 is not a defective item or not connected to the right node, $[s_i]_{i=0}^r$ are mutually independent. Hence, all r check equations hold w.p. $1/n^r$. ■

C. SAFFRON: Algorithm Description and Complexities

Before we begin the description of our SAFFRON algorithm, we first describe a simplified version of it, which we call *Singleton-Only-SAFFRON*. While the main purpose of introducing Singleton-Only-SAFFRON is to help the readers better understand the key idea behind the full SAFFRON algorithm, we also note that this simplified version itself has its own value due to its sheer simplicity. Indeed, the A-SAFFRON algorithm, which we will present in Section V, is a variation of this algorithm.

Consider the case where $U^{(1)} = \cdots = U^{(M)} = V$, where V is defined as in (4). By Proposition 1, a right node can detect a singleton without making an error if the right node is a singleton. Thus, as long as we design a graph such that most of the K defective items are connected to at least one singleton, we can have some recovery guarantee. We first pick positive integers r and d and a positive real number $C > 0$. The number of right nodes is set as $M = CK$. Fix a left node and consider d right nodes that are connected to it. The probability that a connected right node is a singleton is $(1 - \frac{d}{CK})^{K-1}$, which can be approximated by $e^{-d/C}$. Thus, the probability that all of them are not a singleton is approximately $(1 - e^{-d/C})^d$. Thus, by the

linearity of expectation, the expected number of defective items that are not connected to any singleton is $K(1-e^{-d/C})^d$. While some approximation is made in this informal derivation, one can formally show that the actual number of missed defective items is concentrated around this expectation value by applying McDiarmid's inequality with high probability.

The full SAFFRON algorithm is an advanced version of the Singleton-Only-SAFFRON algorithm as it not only recovers singletons but also recovers doubletons, which in turn improves the number of tests by a constant factor. We now formally describe the full SAFFRON algorithm. Again, we first pick positive integers r and d and a positive real number $C > 0$. To clearly show the choice of these parameters, we will denote this algorithm by SAFFRON(r, d, C). The number of right nodes is set as $M = CK$. One then generates a random d-left-regular bipartite graph $T_{\mathcal{G}}$ and M signature matrix $[U_r^{(i)}]_{i=1}^M$. Then, we test items in groups according to $A = [U_r^{(1)}\text{diag}(\mathbf{t}_1); \ldots ; U_r^{(M)}\text{diag}(\mathbf{t}_M)]$.

Given the observation vector $\boldsymbol{y}$ of length $m = hM$, SAFFRON first obtains M right-node measurements of length h. It then inspects the i^{th} right-node measurement with the aid of $\boldsymbol{t}_i$ and $U_r^{(i)}$ for each i. Each right-node measurement maintains a list of defective items that are declared to be connected to the right node. If this list is empty, one applies the single detection/recovery algorithm; If it has exactly one item, one applies the doubleton recovery algorithm. If a new defective item is identified, it is appended to d right node measurements that are connected to this item. We do not append an item to lists that already have two items. This is run for N_{iter} iterations.

The total runtime of our decoding algorithm is $\Theta(K \log n)$. Each iteration consists of $\Theta(K)$ right-node decoding, and it is run for a constant time. The singleton detection measures the weight of the measurement vector, which runs in time $\Theta(\log n)$. The doubleton recovery also runs in time $\Theta(\log n)$. Hence, the right-node decoding procedures run in time $\Theta(K \log n)$. In between right-node decoding procedures, the algorithm updates the list of identified items associated with each right node. When a new defective item is identified, SAFFRON updates d lists. By construction, the length of a list is no greater than 2, so appending an item to a list takes a constant time, and this updates occurs at most K times. Thus, the total complexity of updating lists is $\Theta(K)$. As the total complexity is the sum of the two costs, it becomes $\Theta(K \log n) + \Theta(K) = \Theta(K \log n)$.

D. An Example

In this section, we provide an illustrative example of the decoding algorithm of SAFFRON. Assume $n = 8$ and $K = 3$, and let $\boldsymbol{x} = (1, 0, 1, 0, 0, 0, 0, 1)$, i.e., item 1, item 3 and item 8 are defective items. Recall that A is a row tensor product of $T_{\mathcal{G}}$ and U. Assume that a bipartite graph $\mathcal{G}$ is randomly drawn as follows.[2]

$$T_{\mathcal{G}} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \in \{0,1\}^{M \times n}$$

[2]In the interest of conceptual clarity of the toy example, here we present a bipartite graph that is not left-regular. Furthermore, we note that the number of tests m is larger than the number of items n in this artificial example.

We have $M = 4$ right nodes, and $n = 8$ items are connected to them according to $T_{\mathcal{G}}$. Assume $r = 2$ and $U = U_2^{(1)} = U_2^{(2)} = U_2^{(3)} = U_2^{(4)}$ for the ease of explanation. Consider the following realizations of random mappings:

$$(i_1(1), i_1(2), \ldots, i_1(8)) = (5, 2, 4, 8, 7, 1, 3, 6),$$
$$(i_2(1), i_2(2), \ldots, i_2(8)) = (3, 1, 5, 6, 3, 8, 2, 7).$$

Then, the signature matrix of SAFFRON is as follows.

$$U = \begin{bmatrix} V \\ V_{i_1} \\ V_{i_2} \end{bmatrix} = \left[\begin{array}{cccccccc} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ \hline 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ \hline\hline 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ \hline\hline 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ \hline 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right] \tag{7}$$

$$= [\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3, \boldsymbol{u}_4, \boldsymbol{u}_5, \boldsymbol{u}_6, \boldsymbol{u}_7, \boldsymbol{u}_8] \tag{8}$$

Using (3), we have the following equations for the four right-node measurement vectors.

$$\begin{aligned} \boldsymbol{z}_1 &= \boldsymbol{u}_3 \\ &= (0,1,0,1,0,1|0,1,1,1,0,0|1,0,0,0,1,1)^\intercal, \\ \boldsymbol{z}_2 &= \boldsymbol{u}_1 \vee \boldsymbol{u}_3 \vee \boldsymbol{u}_8 \\ &= (1,1,1,1,1,1|1,1,1,1,1,1|1,1,0,1,1,1)^\intercal, \\ \boldsymbol{z}_3 &= \boldsymbol{u}_1 \vee \boldsymbol{u}_8 \\ &= (1,1,1,1,1,1|1,0,1,0,1,1|1,1,0,1,0,1)^\intercal, \\ \boldsymbol{z}_4 &= \boldsymbol{u}_3 \vee \boldsymbol{u}_8 \\ &= (1,1,1,1,0,1|1,1,1,1,1,0|1,1,0,0,1,1)^\intercal, \end{aligned}$$

where the 3 sections of each right node's measurement vector are separated by vertical bars. We are now ready to decode these measurements. The decoding algorithm first finds all the singletons by finding right nodes whose weight of the first section is $L = \log_2 n$. Since $w(\boldsymbol{z}_1^0) = \log_2 n$, the decoder declares that right node 1 is a singleton. Then, it can read off the first 3 bits of $\boldsymbol{z}_1$, declaring item 3 as defective.

In the second iteration, the algorithm inspects right nodes that are potentially resolvable doubletons including defective item 3. Since $T_{2,3} = T_{4,3} = 1$, the defective item 3 is connected to right nodes 2 and 4. Hence, the decoder will inspect these two right nodes, which are potentially resolvable.

Consider right node 2. We hypothesize that the right node is a doubleton consisting of defective item 3 and exactly one other unknown defective item. That is, we guess that $\boldsymbol{z}_2 = \boldsymbol{u}_3 \vee \boldsymbol{u}_{\ell'}$,

and recover the index as in Section II-B. Then, we get $s_0 = 6,\ s_1 = 8,\ s_2 = 3$. By noticing that $i_1(s_0) = 1 \neq s_1,\ i_2(s_0) = 8 \neq s_2$, the decoder declares that the right node is *not* a resolvable doubleton.

Consider right node 4. The decoder again makes a guess that $\boldsymbol{z}_4 = \boldsymbol{u}_3 \vee \boldsymbol{u}_{\ell'}$. Then, it obtains three indices as follows: $s_0 = 8,\ s_1 = 6,\ s_2 = 7$. Since $i_1(s_0) = s_1,\ i_2(s_0) = s_2$, the decoder concludes that right node 4 is a resolvable doubleton, and item 8 as a new defective item.

In the third iteration, the decoder knows that right node 2 is not resolvable anymore as it already includes two identified defective items. However, right node 3 now has a possibility of being a resolvable doubleton as the decoder found defective item 8 in the previous iteration, and defective item 8 is also connected to right node 3. The decoder hypothesizes that right node 3 is a doubleton, i.e., $\boldsymbol{z}_3 = \boldsymbol{u}_8 \vee \boldsymbol{u}_{\ell_1}$. Similarly, the decoder reads three indices, and the recovered three indices are as follows: $s_0 = 1,\ s_1 = 5,\ s_2 = 3$. Because $i_1(s_0) = s_1,\ i_2(s_0) = s_2$, the decoder will conclude that right node 3 also is a doubleton, declaring item 1 as a new defective item.

The algorithm is terminated as there are no more resolvable right nodes, declaring that items 1, 3 and 8 are defective.

III. Main Results

In this section, we analyze the SAFFRON scheme. The main theoretical result of this paper is the following theorem.

Theorem 3: Assume the noiseless group testing scenario. As $n, K \to \infty$, for positive integers $r \geq 1$ and $d \geq 2$ and a positive constant $C > 0$, w.p. at least $1 - O(\frac{K}{n^r})$, SAFFRON(r, d, C) recovers at least $(1-\varepsilon)K$ defective items with $m = 2(r+1)CK \log_2 n$ tests, where ε is the unique solution of

$$p = [1 - (\rho_1 + \rho_2(1-p))]^{d-1}, \quad 0 < p < 1, \tag{9}$$

where $\rho_i = \binom{K-1}{i-1} \left(\frac{d}{M}\right)^{i-1} \left(1 - \frac{d}{M}\right)^{K-i}$.

Proof: We first provide a brief outline of the proof elements, highlighting the main technical components needed to show that SAFFRON recovers an arbitrarily-close-to-one fraction of non-zero defective items with high probability.

- *Density evolution:* We analyze the performance of SAFFRON on a random bipartite graph for a fixed number of iterations, ℓ. First, we assume that a local neighborhood of depth 2ℓ of every edge in the graph is tree-like, i.e., cycle-free. Under this assumption, which we refer to as the *tree-like assumption*, all the messages between right and left nodes, in the first ℓ iterations of the algorithm, are independent. Using this assumption, we derive a recursive equation that represents the evolution of the expected number of unresolved components at each iteration.
- *Convergence to the cycle-free case:* Using a Doob martingale argument as in [53], we show that the 2ℓ neighborhood of most of the edges of a randomly chosen graph from the ensemble is cycle-free with high probability. This proves that SAFFRON decodes all but a small fraction of the left nodes with high probability in a constant number of iterations. The main difference of our convergence analysis compared to [53] is that the right edge degree distribution in our graphs is binomial distributed, while the right degree is regular in [53].

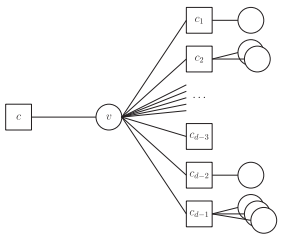


Fig. 1. A tree-like neighborhood of an edge between left node v and right node c. At iteration $j+1$, a 'not-recovered' message is passed through this edge if and only if none of the other neighbors of v, $\{c_i\}_{i=1}^{d-1}$, have been identified as either a singleton or a resolvable doubleton at iteration j.

Partial recovery guarantee: We need to show that with $M = CK$ right nodes, SAFFRON can recover $(1-\varepsilon)K$ defective items with high probability. Recall that by design, our bipartite graph is a d-left-regular bipartite graph with n left nodes and M right nodes. For the analysis, we focus on the *pruned* bipartite graph constructed by the K defective left nodes and the M right nodes. Denote the degree of a random right node by D. Clearly, D is distributed as $\text{Bin}(K, \frac{d}{M})$, i.e., $\Pr(D = i) = \binom{K}{i}(\frac{d}{M})^i(1-\frac{d}{M})^{K-i}$.

Denoting by ρ_i the probability that an edge is connected to a right node of degree i, we have $\rho_i = \frac{iM}{Kd}\Pr(D = i) = \binom{K-1}{i-1}(\frac{d}{M})^{i-1}(1-\frac{d}{M})^{K-i}$. We now analyze the fraction of defective items that cannot be resolved at the end of the iterative decoding procedure via density evolution, a tool to analyze a message-passing algorithm [20], [53], [54]. (For more details about the density evolution technique, we refer the interested readers to [53].) Note that if no error occurs during the entire iterative decoding procedure, this fraction will be equal to the fraction of unidentified defective items. At iteration j of the algorithm, an unidentified defective item passes a message to its neighbor right nodes that have not been recovered yet. Let p_j be the probability that a random defective item is not identified at iteration j. Under the tree-like assumption, the density evolution relates p_j to p_{j+1} as follows.

$$p_{j+1} = [1 - (\rho_1 + \rho_2(1-p_j))]^{d-1}. \tag{10}$$

By defining $\eta(p) := [1 - (\rho_1 + \rho_2(1-p))]^{d-1}$, the update equation can be compactly written as $p_{j+1} = \eta(p_j)$. To see the above equation, consider the graph shown in Fig. 1. At iteration $j+1$, left node v passes a 'not-recovered' message to right node c if none of its other neighbor right nodes $\{c_i\}_{i=1}^{d-1}$ has been identified as either a singleton or a resolvable doubleton at iteration j. The probability that each neighborhood right node has been resolved (either as a singleton or doubleton) at iteration j is $\rho_1 + \rho_2(1-p_j)$. To see this, recall the definition of ρ's. With probability ρ_1, a neighborhood right node is a singleton. With probability ρ_2, a neighborhood right node is a doubleton, and it is a resolvable doubleton if the other connected left node is already resolved. This happens with probability $1 - p_j$. Thus, a neighborhood right node is resolvable w.p. $\rho_1 + \rho_2(1-p_j)$. Since all the messages are independent given the tree-like neighborhood of v, the above equation follows.

We now state a property of a sequence $\{p_j\}_{j=1}^{\infty}$, deferring the proof to the appendix.

Corollary 4: If $p_0 = 1, \{p_j\}_{j=1}^{\infty}$ is a decreasing sequence and it converges to ε, which is the unique solution of $p = \eta(p), 0 < p < 1$. Furthermore, for any $\varepsilon' > 0$, there exists a constant $\ell(\varepsilon')$ such that $p_\ell \leq \varepsilon + \varepsilon'$.

In the density evolution analysis so far, we have derived the *average* fraction of defective items that cannot be recovered after running j iterations of the algorithm is p_j assuming *the tree-like neighborhood assumption*. It remains to show that the actual fraction of unidentified defective items is close to p_j with high probability without assuming the tree-like neighborhood assumption. This can be proved in two steps. First, one can first show that the tree-like neighborhood assumption indeed holds with high probability. Given the tree-like neighborhood, one can prove that the actual fraction is highly concentrated around p_j using Doob's martingale inequality [53]. We provide a formal guarantee in the following lemma, deferring the full proof to the appendix.

Lemma 5: Over the probability space of the ensemble of d-left-regular graphs, let Z be the number of unresolved items after ℓ iterations of the SAFFRON algorithm. Then, there exists a constant $\beta > 0$ such that for any $\varepsilon'' > 0$,

$$|\mathbb{E}[Z] - Kp_\ell| < K\varepsilon''/2, \tag{11}$$

$$\mathbb{P}(|Z - Kp_\ell| > K\varepsilon'') < 2e^{-\beta\varepsilon''^2 K^{1/(4\ell+1)}}, \tag{12}$$

for all K sufficiently large.

Remark 2: If the tree-like neighborhood assumption holds, we have $\mathbb{E}[Z] = Kp_\ell$. In (11), we have $\mathbb{E}[Z] < K(p_\ell + \varepsilon''/2)$, which has a small additional term. This comes from a small probability of not having a tree-like neighborhood.

Remark 3: Combined with Corollary 4, this lemma asserts that the number of unidentified defective items can be made arbitrarily close to εK by running a constant number of iterations of SAFFRON.

By applying this lemma, we obtain the partial recovery guarantee in the theorem.

Error probability: We now analyze the probability of error. We fix the input signal x and analyze the error probability over random testing matrix, which consists of a random bipartite graph and random signatures. By Lemma 5, the concentration does not hold with probability less than $2e^{-\beta\varepsilon''^2 K^{1/(4\ell+1)}}$, which is asymptotically smaller than any polynomial in K. Denoting the concentration event by C, $\Pr(\bar{C}) \leq 2e^{-\beta\varepsilon''^2 K^{1/(4\ell+1)}}$.

Conditioned on the concentration event, we still need to bound the probability of making any errors during the iterative decoding, i.e., missing *any* defective items and the probability of declaring *any* false defective items. Given a realization of a random bipartite graph and assuming no error (missed defective item or false defective item), the order of right-node decoding procedures is fixed. According to this order, we index the error event associated with each right-node decoding procedure as $E_1, E_2, \ldots, E_{\ell M}$. By Proposition 1 and Lemma 2, $\Pr(E_i|\overline{E_{i-1}}, \ldots, \overline{E_1}) \leq 1/n^r$. Thus,

$$\begin{aligned}\Pr(E) &= \Pr(E_1) + \Pr(E_2, \overline{E_1}) + \cdots \\ &\quad + \Pr(E_{\ell M}, \overline{E_{\ell M-1}}, \ldots, \overline{E_1}) \\ &\leq \Pr(E_1) + \Pr(E_2|\overline{E_1}) + \cdots \\ &\quad + \Pr(E_{\ell M}|\overline{E_{\ell M-1}}, \ldots, \overline{E_1}) \\ &\leq \frac{\ell M}{n^r}.\end{aligned}$$

Here, note that the above bound holds even though E_i's are not independent. The total error probability is bounded by $\frac{\ell M}{n^r} + 2e^{-\beta\varepsilon''^2 K^{1/(4\ell+1)}} = O(K/n^r)$. ■

Remark 4 (Model Limitation): We note that our main theorem assumes a *perfect* Boolean ORing measurement. While this perfect model holds in some digital applications (such as digital forensic [8]), it may not be appropriate for a variety of natural group testing applications, especially when n is large. A more general way of modeling a measurement is based on an absolute threshold, i.e., a test result is positive if and only if the number of defective items in a pool is larger than a certain threshold, which does not have to be one. Another way is based on a relative threshold, i.e., a test result is positive if and only if the fraction of defective items in a pool is larger than a critical constant. It is not clear whether or not our "divide and conquer" principle can be applied to these more general models. We note a recent successful attempt [55]. In this work, the authors apply the sparse-graph-code framework to design a group testing algorithm for the absolute threshold model, where the threshold can be larger than one.

A. Parameter Optimization

In Theorem 3, the recovery guarantee is given in terms of ε, which is the solution of a high-order polynomial equation. In this section, we show how one can approximately find the value of ε, and then provide a method for choosing parameters.

By the definition of ρ_i, we have $\rho_1 = \left(1 - \frac{d}{M}\right)^{K-1}$. Since d is fixed and $K, M \to \infty$, $\rho_1 = (1 - \frac{d}{M})^K(1 + o(1))$. Furthermore,

$$\begin{aligned}&\left(1 - \frac{d}{M}\right)^K = \left(e^{-d/M} + O(d^2/M^2)\right)^K \\ &= \left(e^{-d/M} + O(1/K^2)\right)^K \leq e^{-dK/M} + \sum_{i=1}^{K} O(1/K^i) \\ &\leq e^{-dK/M} + O(1/K) = e^{-dK/M} + o(1).\end{aligned}$$

Here, the first equality holds since $M = \Theta(K)$ and d is constant, and the last inequality holds since $i = 1$ term dominates the summation. Thus, $\rho_1 = (e^{-dK/M} + o(1))(1 + o(1)) = e^{-dK/M}(1 + o(1))$. By defining $\lambda := dK/M$, $\rho_1 = e^{-\lambda}(1 + o(1))$. Similarly, $\rho_2 = \rho_1 \frac{(K-1)\frac{d}{M}}{1-\frac{d}{M}} = \rho_1(\lambda + o(1))(1 + o(1)) = \lambda e^{-\lambda}(1 + o(1))$.

For the purpose of numerical evaluations of the required number of tests for a targeted missed fraction ε, we next provide an approximate optimization framework for designing the parameters of the algorithm given ε. First, we approximately find ε by finding the solution of the following equation.

$$p = [1 - \rho_1 - \rho_2(1 - p))]^{d-1}, \;\; 0 < p < 1.$$

Here, by approximating $1 - p \approx 1$, we have

$$\varepsilon \approx [1 - \rho_1 - \rho_2)]^{d-1} \approx [1 - e^{-\lambda} - \lambda e^{-\lambda}]^{d-1}. \tag{13}$$

TABLE II
PAIRS OF ε AND $C(\varepsilon)$

ε	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}
$C(\varepsilon)$	6.13	7.88	9.63	11.36	13.10	14.84	16.57
$d^\star$	7	9	10	12	14	15	17
$\lambda^\star$	1.14	1.14	1.04	1.05	1.07	1.01	1.03

Using this approximation, we now find a pair of design parameters (d, M) that approximately minimizes the number of right nodes M (thus the number of tests) given a targeted reliability ε. By taking log of both sides in (13), we have

$$(d-1)\log\left(1-e^{-\lambda}-\lambda e^{-\lambda}\right)=\log\varepsilon. \tag{14}$$

To minimize the number of right nodes while satisfying the recovery guarantee, we formulate an optimization problem.

$$\min_{d\in\mathbb{N}^+,\lambda>0} \quad C=d/\lambda \tag{15}$$

$$\text{subject to } (d-1)\log\left(1-e^{-\lambda}-\lambda e^{-\lambda}\right)=\log\varepsilon. \tag{16}$$

One can numerically solve the optimization problem and attain the optimal $\lambda^\star$ and $d^\star$ as a function of ε. Some of the approximately optimal design parameters for different recovery targets are shown in Table II.

B. Storage Cost

Our theoretical guarantee assumes that $[U_r^{(i)}]_{i=1}^M$ are mutually independent, but storing rM random mappings incurs prohibitive storage cost. In practice, one can reduce this storage cost by setting $U_r^{(1)}=U_r^{(2)}=\cdots=U_r^{(M)}$. While the theoretical guarantee may not hold anymore in this setting, no performance degradation is observed in our simulation results. See Section VI for more details. In this setting, A can be fully specified by storing $T_{\mathcal{G}}$ and $U_r^{(1)}$, which can be also fully specified by r random mappings $i_1, i_2, \ldots, i_r$. Each column of an incidence matrix $T_{\mathcal{G}}$ can be efficiently stored since it can be represented a list of d integers, which costs $d\log_2 M$ bits. Hence, the total storage cost for storing an incidence matrix is $nd\log_2 M$. Storing each mapping costs $n\log_2 n$ bits. Therefore, the total storage cost is $n(d\log_2 M + r\log_2 n)$. Compared to the worst-case storage cost, which is nm bits, this is a significant storage saving. For instance, when $n=2^{32}, K=2^7, M=1454, d=12$ and $r=2$, the worst-case storage cost is 409.2TB while our storage cost is 93 GB.

C. Comparison With Other Methods

Remark 5 (Comparison with the Non-adaptive GROTESQUE Algorithm): SAFFRON is closely related to the non-adaptive GROTESQUE algorithm [38]. Particularly, when the non-adaptive GROTESQUE algorithm is used with a fixed number of stages, it requires $\Theta(K\log n)$ tests, matching the order of the test complexity of SAFFRON. The non-adaptive GROTESQUE algorithm is similar to SAFFRON since its testing matrix consists of a random bipartite graph combined with Bernoulli random signatures. However, GROTESQUE identifies singletons only, and its singleton detection is based on a randomized algorithm with Bernoulli$(1/2)$ tests, which is subject to false positives. On the other hand, SAFFRON does not make any false positives while performing singleton detections (for the noiseless case). Furthermore, SAFFRON detects and resolves both singletons and doubletons, achieving a lower number of tests. Lastly, our signature design allows us to robustify SAFFRON to unknown offsets, which cannot be easily handled with the GROTESQUE algorithm.

Remark 6 (Comparison with Maximum Likelihood Decoder): The low decoding complexity of SAFFRON comes at the cost of increased sample complexity. In [35], the authors analyze the performance of maximum likelihood decoder assuming a Bernoulli random test matrix. As a result, they show that $(1+o(1))K\log_2(n/K)$ tests are sufficient for partial recovery. Compared to this, the sample complexity of SAFFRON requires $2(r+1)C(\varepsilon)$ times more measurements. Since $C(\varepsilon)$ diverges as $\varepsilon\to 0$ (as proved in the appendix), this constant factor hit increases as $\varepsilon\to 0$.

It is an interesting open question whether or not one can reduce this constant factor hit in the number of tests. The main reason behind this overhead is due to the suboptimality of underlying signature matrices. Our current signature design allows for identifying and resolving only up to doubletons, and this limits the performance of our iterative decoding procedure. More specifically, recall that $\varepsilon\approx(1-e^{-\lambda}-\lambda e^{-\lambda})^{d-1}$, in which the term $e^{-\lambda}$ is due the fact that we can identify/resolve singletons and the other term $\lambda e^{-\lambda}$ is due to doubleton recovery. Therefore, if we could have identified and resolved right nodes with larger degrees, we would achieve a lower value of ε, or equivalently a lower measurement cost for the same value of ε. One natural way of achieving this goal is to combine the overall structure of SAFFRON with efficient disjunct matrices for designing a signature matrix.

Remark 7 (Exact recovery guarantee): It is clear that one can arbitrarily reduce the error fraction by concatenating more than one i.i.d. SAFFRON testing matrices. More specifically, if one concatenates SAFFRON matrices r times, the probability of missing a defective item becomes ε^r, so the average number of missed defective items is $K\varepsilon^r$. Thus, by setting $r=C'\log K$ for some $C'>1/\log(1/\varepsilon)$, one can make the average number of missed defective items vanish as $K\varepsilon^r<K^{1-C'\log(1/\varepsilon)}=o(1)$. That is, with $\Theta(K\log K\log n)$ tests, one can recover all K defective items with high probability.

IV. ROBUST-SAFFRON FOR NOISY GROUP TESTING

In this section, we robustify SAFFRON such that it can recover the set of K defective items with erroneous or *noisy* test results. Assuming an i.i.d. noise model, each test result is 'wrong' with probability q, i.e.,

$$\mathbf{y}=A\boldsymbol{x}+\boldsymbol{w}, \tag{17}$$

where the addition is over binary field, and $\boldsymbol{w}$ is an i.i.d. noise vector whose components are 1 with probability $0<q<\frac{1}{2}$ and 0 otherwise.[3]

[3]If $q>\frac{1}{2}$, one can always take the complement of all the test results, and redefine the crossover probability as $\tilde{q}=1-q<\frac{1}{2}$.

Our approach is simple: we design a robust signature matrix V_{robust} with the aid of error correction code, treating each index vector $\boldsymbol{b}_i$ as a message that needs to be transmitted over a noisy memoryless communication channel. Given an efficient channel code of rate $R < 1$, we map $\boldsymbol{b}_i$ (of length L) to a codeword of length L/R. By appropriately choosing a code, this redundancy makes sure that the index vector can be reliably transmitted over a noisy channel.

Particularly, spatially-coupled LDPC codes have the following properties [56].

- It has an encoding function $f(\cdot) : \{0,1\}^L \to \{0,1\}^{L/R}$ and a decoding function $g(\cdot) : \{0,1\}^{L/R} \to \{0,1\}^L$, and its decoding complexity is $O(L)$.
- If R satisfies

$$R < 1 - H(q) - \delta = q \log_2 q + (1-q) \log_2 (1-q) - \delta,$$

for an arbitrarily small constant $\delta > 0$, then there exists a constant $\zeta > 0$ such that $\Pr(g(\boldsymbol{x} + \boldsymbol{w}) \neq \boldsymbol{x}) < 2^{-\zeta L}$ as L approaches infinity.

Given such an error-correcting code, we design the signature matrix $V' \in \{0,1\}^{L/R \times n}$ for the robust SAFFRON scheme as follows:

$$V' = \left[\frac{f(\boldsymbol{b}_1)}{\overline{f(\boldsymbol{b}_1)}} \frac{f(\boldsymbol{b}_2)}{\overline{f(\boldsymbol{b}_2)}} \cdots \frac{f(\boldsymbol{b}_n)}{\overline{f(\boldsymbol{b}_n)}} \right] \tag{18}$$

For a positive integer r, the full robust signature matrix U'_r is defined similar to the full signature matrix U_r. With r random mappings $i_1, i_2, \ldots, i_r$, $U'_r := [V'; V'_{i_1}; V'_{i_2}; \ldots; V'_{i_r}]$.

We now describe how the robustified SAFFRON resolves singletons and doubletons. For illustration purpose, assume a robust signature matrix U'_1. Consider a singleton right node k. Assume that left node ℓ is connected to this singleton. The right-node measurement vector consists of two sections: $\boldsymbol{z}_k^0$ and $\boldsymbol{z}_k^2$. Let us consider the first section, which is of the following form: $\boldsymbol{z}_k^0 = [f(\boldsymbol{b}_\ell); \overline{f(\boldsymbol{b}_\ell)}] + \boldsymbol{w}_k^0$, where $\boldsymbol{w}_k^0$ is a slice of $\boldsymbol{w}$ corresponding to $\boldsymbol{z}_k^0$. Since the first half of each $\boldsymbol{z}_k^0$ is $f(\boldsymbol{b}_\ell)$ corrupted by some noise, one can apply the decoding function to estimate the index. Denote this estimate by s_0. Similarly, one can apply the decoding function to the other section, obtaining another estimate s_1. If $i_1(s_0) = s_1$, then the decoder declares that item s_0 is defective. The doubleton recovery algorithm proposed in Section II can be easily modified to the robust signature matrix.

Lemma 6: Assume a random robust signature matrix U'_r, whose index mappings are drawn at random, independently of others. The robust singleton recovery algorithm (or doubleton recovery algorithm) 1) fails to identify a singleton (or a resolvable doubleton) w.p. no greater than $\frac{r+1}{n^\zeta}$ and 2) declares a wrong defective item w.p. no greater than $\frac{1}{n^r}$.

Proof: The first error event (missed detection) happens if any of the $r+1$ decoding attempts fails, so the error probability is upper bounded by $\frac{r+1}{n^\zeta}$ by the union bound. Note that this is not a critical error since we can compensate this error by slightly increasing the number of measurements. (See below for more details.) The second error event (false positive) is critical since we do not allow for any false defective items. Denote the decoded index from $\boldsymbol{z}^0$ by s_0, and denote the index of the defective item connected to the right node by ℓ. If $s_0 = \ell$, then this is a benign error. If $s_0 \neq \ell$, then $i_i(s_0)$ is independent of the decoding results from $\boldsymbol{z}^1, \boldsymbol{z}^2, \ldots, \boldsymbol{z}^r$. Thus, the r check equations hold w.p. $1/n^r$. ■

Lemma 6 implies that the robustified SAFFRON scheme will miss a fraction of singletons that is fewer than $\frac{r+1}{n^\zeta}$. We compensate this loss by increasing the number of right nodes: instead of using M right nodes, we use $M\left(1 + \frac{r+1}{n^\zeta}\right)$ right nodes, so that the effective number of right nodes becomes $M\left(1 + \frac{r+1}{n^\zeta}\right)\left(1 - \frac{r+1}{n^\zeta}\right) \approx M$.

Using these lemmas, we now present our guarantee for noisy group testing. We omit the proof since it is essentially identical to that of Theorem 3.

Theorem 7: Assume the noisy group testing scenario with noise parameter q. As $K \to \infty$, for a positive integer $r \geq 1$, w.p. at least $1 - O(\frac{K}{n^r})$, Robust-SAFFRON(r) recovers at least $(1-\varepsilon)K$ defective items with $m = 2(r+1)\beta(q)C(\varepsilon)K \log_2 n$ tests, where ε is an arbitrarily-close-to-zero constant, $\beta(q) = \frac{1}{R} > \frac{1}{1-H(q)-\delta}$ for an arbitrarily small constant $\delta > 0$, and $C(\varepsilon)$ is a constant that depends only on ε. Table II shows some pairs of ε and $C(\varepsilon)$.

V. A-SAFFRON for Neighbor Discovery

The goal of neighbor discovery problem is to identify K active neighbors among n total nodes in the network through m bits of communication. During the discovery period, each active neighbor broadcasts a length-m binary signal, while every node is listening to the others' transmissions. For power efficiency and design simplicity, each node simply measures whether or not at least one neighbor node is transmitting a signal in each time slot, obtaining a binary received signal. After the discovery period, each distributed node decodes the received signal, which is a super-imposed (binary ORed) signal of all the neighbors' signals.

In this section, observing an implicit connection between the neighbor discovery problem and the group testing problem, we view the neighbor discovery problem as *an asynchronous group testing problem* and propose A-SAFFRON. Although group testing algorithms have been shown useful for neighbor discovery [57], our asynchronous formulation and solution, which were originally presented in our earlier conference paper [2], have not been proposed before.

A. Neighbor Discovery and Asynchronous Group Testing

Let $c_k = [c_k(i)]_{i=1}^m \in \{0,1\}^m$ be the signal of user k. Let $S = \{i_1, i_2, \ldots, i_K\}$ be the set of active users. Suppose that each bit corresponds to a time sub-slot. An active node transmits its signal in m sub-slots.

Due to the different locations of transmitters, active user k's signal is received with a delay of δ_k bits. Assume that δ_k is uniformly distributed in the set $\{0, 1, \ldots, \Delta\}$, where Δ is the maximum delay offset between the users.

The receiver performs a simple energy-based non-coherent detection at each sub-slot, and detects whether at least one user has transmitted signal (bit 1) in that sub-slot or not (bit 0). This corresponds to a bit-level OR channel as follows: the i^{th} component of the received signal $y \in \{0,1\}^{m+\Delta}$ is

$y_i = \bigvee_{k \in S} c_k(i - \delta_k)$, where $\vee$ is the Boolean OR operator. Note that $\Delta + m$ is the maximum transmission length.

The goal of neighbor discovery problem is to judiciously design the signals $\{c_k\}_{k=1}^n$ such that the receiver is able to recover any K active neighbors using as few number of bits, m, as possible, and the receiver has low decoding complexity. Here, δ_i's are not known to the receiver, i.e., the decoder needs to decode the signal without knowing the delays.

Note that the neighbor discovery problem can be viewed as *an asynchronous group testing problem*, where $y_i, 1 \leq i \leq m$ is the result of each test, and the K active neighbors correspond to the K defective items in group testing. Furthermore, the design of transmission signals can be viewed as the design of pools. Based on this equivalence, we will design an efficient algorithm for the asynchronous group testing problem.

B. A-SAFFRON: Algorithm Description and Analysis

We now propose our solution A-SAFFRON. A key observation about how a variation of the SAFFRON scheme can be used for the asynchronous case is as follows. When the SAFFRON algorithm determines whether a right node of the bipartite graph is a singleton or not, it does not use any information about the structure of the graph. Instead, it simply checks whether the weight of the right-node is $h/2$ or not. In the asynchronous case, the only difference is that there is no clear boundaries between the right nodes as signals are shifted by arbitrary bits. To resolve this issue, A-SAFFRON makes use of a sliding window, which continuously moves along the received signal to find the signatures.

We now explain the details of the A-SAFFRON scheme. The A-SAFFRON scheme attempts at detecting and resolving singletons only, and hence we may design our signature matrix as in (4). However, with this signature design, the boundaries of a signature cannot be uniquely determined in the asynchronous setting. For instance, consider the received signal $y = (0, 0, 0, 1, 1, 0, 0, 0)$ and assume that $n = 4$ and $K = 1$. If the signature matrix in (4) is used, the received signal can be decoded in three different ways. For instance, if the index of the defective item is 1, its signature is $(0, 0|1, 1)$, and one can interpret the received signal as $y = (0|\mathbf{0}, \mathbf{0}, \mathbf{1}, \mathbf{1}|0, 0, 0)$, where $|$ denotes the boundaries of the signature. Similarly, if the index of the defective item is 2, its signature is $(0, 1|1, 0)$, and one can interpret the received signal as $y = (0, 0|\mathbf{0}, \mathbf{1}, \mathbf{1}, \mathbf{0}|0, 0)$. If the index of the defective item is 4, its signature is $(1, 1|0, 0)$, and the received signal can be interpreted as $y = (0, 0, 0|\mathbf{1}, \mathbf{1}, \mathbf{0}, \mathbf{0}|0)$.

To resolve this issue, we first interleave the bits from the two sections and then prepend 1 to each signature so that one can uniquely determine the boundaries.[4] For instance, the signature vector $(0, 0|1, 1)$ becomes $(1|0, 1|0, 1)$, and $(0, 1|1, 0)$ becomes $(1|0, 1|1, 0)$. In the previous example, the received signal with this signature matrix would be $y = (0|\mathbf{1}, \mathbf{0}, \mathbf{1}, \mathbf{0}, \mathbf{1}|0, 0)$, $y = (0, 0|\mathbf{1}, \mathbf{0}, \mathbf{1}, \mathbf{1}, \mathbf{0}|0)$, or $y = (0, 0, 0|\mathbf{1}, \mathbf{1}, \mathbf{0}, \mathbf{1}, \mathbf{0}|)$, depending on which the index of the defective item. Another advantage of using interleaved signatures is that one can uniquely determine the boundaries of a transmitted signature (or a connected sequence of signatures). By design, three consecutive zeros occur only outside the region of signatures.

[4]This increases the signature length by 1, and the length of signature vectors is now $2 \log_2 n + 1$. For the ease of presentation, we will simply ignore this increment.

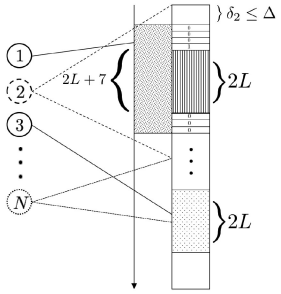


Fig. 2. A-SAFFRON and its decoding algorithm. Each of the colored nodes represents an active node, whose signature is transmitted at the indicated slot on the right. The receiving node sees the concatenation of the right nodes a contiguous bit string.

We now describe the A-SAFFRON algorithm in details. We first design the bipartite graph (matrix T) of size $n \times M$, whose entry is i.i.d. Bernoulli random variables with parameter p (to be determined). For decoding, we use a simple sliding-window decoder to detect the active users (as illustrated in Fig. 2). The decoder checks all the $2L + 7$ consecutive received bits. It first checks whether the first three bits and last three bits are 0, ensuring that it is an isolated singleton. It then checks whether the fourth bit is 1, indicating the start of a signature. Finally, it looks at the weight of the remaining $2L$ bits. If the weight of the $2L$-bits binary string is L, the decoder declares a singleton and finds the corresponding active user by observing the L bits at location $1, 3, \ldots, 2L - 1$ to obtain $\boldsymbol{b}_i$. This procedure is repeated until the sliding window reaches the end of the signal. We now analyze the performance of A-SAFFRON.

Theorem 8: Let $k_0 := \lceil \frac{\Delta}{2 \log n} \rceil$. For any $\alpha > 0$, if $m = 4e^{1+k_0/2}(1 + \alpha)K \log K \log_2 n$, then A-SAFFRON recovers all of the K active users with zero false positives, with probability at least $1 - \frac{1}{K^\alpha}$. The computational complexity of the decoding algorithm is linear in the length of the signals, i.e., it is $\Theta(K \log K \log_2 n)$.

Proof: The A-SAFFRON algorithm can make sure that there exists an isolated singleton within a window by checking the first four bits and the last three bits. Detecting and resolving a singleton reduces to the synchronous case, so no error occurs by Proposition 1.

We now prove that the A-SAFFRON scheme correctly recovers all of the K active users with probability at least $1 - \frac{1}{K^\alpha}$. Consider an edge of an active left node. This edge corresponds to a signature superimposed at some location in the received signal. In general, this signature spans two consecutive right nodes. In order to avoid collision, no signature of the other active users should reside in these right nodes. However, due to random delays, every signature can be shifted (rightward) by at Δ bits. By the definition of k_0, this maximum shift corresponds to k_0 right nodes. Therefore, if $2 + k_0$ right nodes are not chosen by any of the other active users, this signature is guaranteed to be isolated. Since each entry of our incidence matrix is an i.i.d. Bernoulli random variable with parameter p, this happens with probability $(1 - p)^{(2+k_0)(K-1)}$. By choosing

$p = \frac{c_1}{K}$, we have $p_0 := (1-p)^{(2+k_0)(K-1)} = (1+o(1))(1-p)^{(2+k_0)K} = (1+o(1))e^{-c_1(2+k_0)}$. Now, conditioning on the degree of an active left node being i, the probability that the corresponding active user is not detected is at most $(1-p_0)^i$. Thus, the probability of missing a certain defective item is:

$$\sum_{i=0}^{M} \binom{M}{i} p^i (1-p)^{M-i} (1-p_0)^i = (p(1-p_0) + (1-p))^M$$
$$= (1 - pp_0)^M$$
$$= (1+o(1))e^{-pp_o M}.$$

If $M = c_2 K \log K$, then $e^{-pp_o M} = K^{-c_1 c_2 p_0}$. By choosing $c_1 = 1/2$ and $c_2 = 2(1+\alpha)e^{1+k_0/2}$, we have $K^{-c_1 c_2 p_0} = K^{-(1+\alpha)}$. Union bounding over K defective items gives us the error probability of $1/K^{-\alpha}$.

As for the decoding complexity, the A-SAFFRON decoding algorithm processes the received signal of length $m + \Delta$ in one pass. Since $\Delta = o(m)$, the decoding cost is $\Theta(m)$. ■

Remark 8 (Counting Bound): One may derive a counting bound for the asynchronous group testing problem. Consider a situation one wants to detect all defective items as well as their offsets. Since the decoder receives $m + \Delta$ bits, this gives us $m + \Delta \geq \log_2 \left(\binom{n}{K} \Delta^K\right) \approx K \log_2 (n/K) + K \log_2 \Delta$. This implies that having unknown offsets does not increase the lower bound if Δ does not scale as fast as n/K. Our theorem asserts that the measurement complexity of A-SAFFRON remains constant if $\Delta = o(\log n)$ or $k_0 = o(1)$. When $\Delta = \Theta(\log n)$, $k_0 = \Theta(1)$, and the cost of asynchrony is a constant factor, which could be still large if k_0 is large. When $\Delta = \omega(\log n)$, $k_0 = \omega(1)$, the sample complexity increases by a scaling factor of $e^{k_0/2}$.

C. The Robustified A-SAFFRON

The A-SAFFRON scheme can be also made robust by replacing the signature matrix with a robust one. For robust prefix detection, we prepend $t \log_2 n$ many 1's to each signature vector, for some constant $t > 0$. The rest of the signature matrix is encoded via an efficient error-correcting code of rate R. The decoding algorithm of the robustified A-SAFFRON scheme is described as follows. The decoder slides a window of length $\left(t + \frac{6}{R}\right) \log_2 n$ and checks the energy of the first $t \log_2 n$ bits. If this energy is greater than an energy threshold $T_e = \frac{t \log_2 n}{2}$, the following $\frac{6 \log_2 n}{R}$ bits are added as a hypothesis, and the window keeps sliding. We now note that in order to ensure the singleton is isolated, the decoder skips the following $\frac{6 \log_2 n}{R}$ bits, only adding them to the hypothesis list. It keeps track of how many bits are left to "restart", and resets this counter once it sees a set of $t \log_2 n$ bits which pass the energy test. The decoder only marks a hypothesis when the "restart" counter hits 0 and it sees a set of contiguous $t \log_2 n$ bits that pass the energy test. Now, note that since we are still using the complement of each codeword, a valid singleton will still have approximately constant weight. This allows the decoder to iterate through the decoding map and check for an approximately constant weight of $3\left(\frac{\log_2 n}{R}\right)$. In this way, just as in the noiseless decoder, it may take one pass through the received bit string, and then carefully check the remaining hypotheses in the following manner. We first use the decoding map $g(\cdot)$ to find the hypothetical active node corresponding to the first $\left(\frac{\log_2 n}{R}\right)$ bits. Then, using the two random checks in the following two $\left(\frac{\log_2 n}{R}\right)$ length sections, the decoder checks the initial hypothesis. If the checks were inconsistent, it does not declare a resolved singleton, and continues decoding. Note that one can easily bound the number of hypotheses by $O(K \log K)$ using Hoeffding's inequalities. By applying Lemma 6 along with this, we have the following corollary.

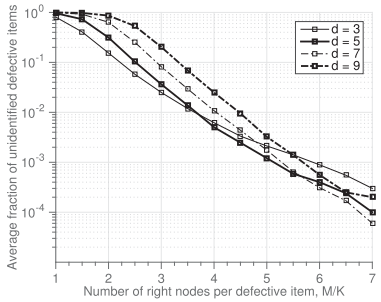


Fig. 3. Performance of SAFFRON with noiseless test results. For different pairs of (d, M), we run SAFFRON 1000 times and run the average fraction of unidentified defective items. We set $n = 2^{16}$, $K = 100$, $d \in \{3, 5, 7, 9\}$ and $K \leq M \leq 7K$.

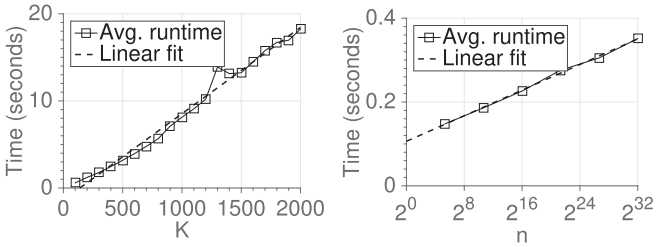
Corollary 9: The sample and decoding complexity of Robust-A-SAFFRON are both $\Theta(K \log (K) \log_2 (n))$.

VI. Simulation Results

We now evaluate the performance of the SAFFRON scheme and its variations via simulations.[5] The SAFFRON scheme recovers with high probability an arbitrarily-close-to-one fraction of K defective items with $\Theta(K \log n)$ tests, as stated in Theorem. 3. The theorem also characterizes the optimal pairs of $(d^\star, \lambda^\star)$ for a target recovery performance ε: as the fraction of unidentified defective items ε decreases, the corresponding optimal left-degree $d^\star$ increases. For different pairs of (d, M), we run SAFFRON 1000 times and measure the average fraction of unidentified defective items: we choose $n = 2^{16}$, $K = 100$, $d \in \{3, 5, 7, 9\}$ and $K \leq M \leq 7K$. Plotted in Fig. 3 are the average fractions of unidentified defective items obtained via simulations for different values of d. As expected, we can observe that if M is close to K, the average fraction of unidentified defective items can be minimized by setting $d = 3$, and if M is close to $7K$, higher values of d perform better.

We now observe how computationally-efficient SAFFRON's decoding algorithm is. We measure the average runtime of SAFFRON with $n = 2^{32}$, while increasing the value of K. In Fig. 4a, we plot the simulation results, and they clearly demonstrate the $\Theta(K)$ factor of the computational complexity. Similarly, we repeat simulations with $K = 2^5 = 32$, while increasing the value of n. In Fig. 4b, we plot the average runtime with a logarithmic x-axis: we can clearly observe that the computational complexity is linear in $\log n$.

[5] Simulators are written in Python and run on a laptop with 2 GHz Intel Core i7 and 8 GB memory.



(a) Runtime with varying K (b) Runtime with varying n

Fig. 4. Time complexity of SAFFRON. The run-time of SAFFRON for $n = 2^{32}$ (left) and for $K = 2^5$ (right).

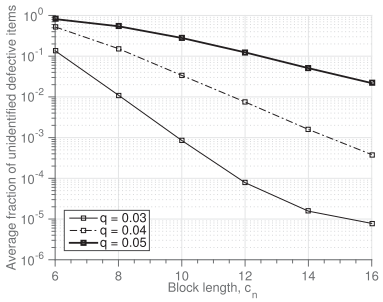


Fig. 5. Performance of Robust-SAFFRON with noisy test results. The average fraction of missed defective items for varying (q, c_n).

We also evaluate the robustified SAFFRON scheme. In our setting, we choose $n = 2^{32} \approx 4.3 \times 10^9$ and $K = 2^7 = 128$. For random bipartite graphs, we use $d = 12$ and $M = 11.36K$. The noise model is the one we described in Section IV. We vary the probability of error q from 0.03 to 0.05: a test result is flipped with probability from 3% to 5%.

While we made use of capacity-achieving codes in Theorem. 7, we use Reed-Solomon codes for simulations for simplicity [58]. A Reed-Solomon code takes a message of c_k symbols from a finite field of size $c_q \geq c_n$, for a prime power c_q, and then encodes the message into c_n symbols. This code can correct upto any $\lfloor \frac{c_n - c_k}{2} \rfloor$ symbol errors. By using a field of size $c_q = 2^8$, a binary representation of length $L\ (= \log_2 n)$ can be viewed as a 4-symbol message, i.e., $c_k = 4$. Thus, the overall number of tests is as follows:

$$m = \underbrace{11.36K}_{\text{Number of right nodes}} \times \underbrace{c_n/c_k}_{\text{Code redundancy}} \times \underbrace{6\log_2 n}_{\text{Size of message}} . \quad (19)$$

By having $c_n = c_k + 2t$, the robustified SAFFRON scheme can correct upto t symbol errors within each section of the right-node measurement vector. Thus, we evaluate the performance of the robustified SAFFRON scheme with $c_n \in \{6, 8, \ldots, 16\}$ for various noise levels. We measure the average fraction of unidentified defective items over 1000 runs for each setup. Fig. 5 shows the simulation results; the x-axis is the block-length of the used code, and the logarithmic y-axis is the average fraction of unidentified defective items. We can observe that for a higher noise level q, the minimum block length required to achieve a certain value of ε increases.

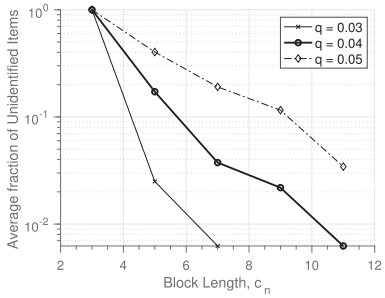


Fig. 6. Performance of Robust-A-SAFFRON with noisy test results. The average fraction of unidentified defective items.

We also observe that *the robustified SAFFRON perfectly recovers all defective items even with the presence of erroneous test results* with certain parameters. For instance, with $q = 0.02$, we test the robustified SAFFRON 1000 times with $c_n = 12$. For all the test cases, it successfully recovers all $K = 128$ defective items from the population of $n \approx 4.3 \times 10^9$ items with $m = 838080 \approx \frac{2}{10000} n$ tests. Further, the decoding time takes only about 3.8 seconds on average. Similarly, we observe the perfect recovery with $q = 0.01$ and $c_n \geq 12$ and with $q = 0.005$ and $c_n \geq 6$.

Lastly, we report the experimental results for the A-SAFFRON algorithm. Shown in Fig. 6 is the average fraction of unidentified items when $n = 2^{24}$, $K = 2^5$, and $\Delta = 200$. We test the performance of A-SAFFRON with different noise levels (bit flip probabilities): $q \in \{0.03, 0.04, 0.05\}$. Observe that as long as a large enough block length is chosen, i.e., sufficient error-correcting capability is provisioned, the Robust-A-SAFFRON algorithm can recover a large fraction of the defective items even when items are pooled with unknown offsets and test results are randomly flipped.

VII. Conclusion

In this paper, we have described the design and analysis of SAFFRON based on the modern coding-theoretic tools of sparse-graph coding and density evolution. We show that 1) SAFFRON recovers a $(1 - \varepsilon)$-fraction of K defective items w.h.p. with $\Theta(K \log_2 n)$ tests, and 2) its decoding complexity is $\Theta(K \log n)$. We also robusitifed SAFFRON to noisy and asynchronous tests.

Appendix

A. Proofs

1) Proof of Corollary 4:

Proof: Let us define $f(x) := \eta(x) - x = (1 - \rho_1 - \rho_2 + \rho_2 x)^{d-1} - x$. Note that $f(0) = (1 - \rho_1 - \rho_2)^{d-1} > 0$ and $f(1) = (1 - \rho_1)^{d-1} - 1 < 0$. Thus, there exists at least one solution of $f(x) = 0$ in $x \in (0, 1)$.

In order to show the uniqueness of the solution, consider its derivative $f'(x) = \rho_2(d-1)(1 - \rho_1 - \rho_2 + \rho_2 x)^{d-2} - 1$. We first show that $f(x) = 0$ has a unique solution in $x \in (0, 1)$ if $f'(x) = 0$ occurs at most once in $x \in (0, 1)$. To see this, we prove the contrapositive statement. That is, we show that if $f(x) =$

0 has more than one solution in $x \in (0,1)$, $f'(x) = 0$ occurs more than once in $x \in (0,1)$. Assume that $f(x_1) = f(x_2) = 0$ for $0 < x_1 < x_2 < 1$. Since $f(0) > 0, f(x_1) = 0$, we have $f'(x_1') < 0$ for some $0 < x_1' < x_1$ by the mean value theorem. Similarly, since $f(x_2) = 0$ and $f(1) < 0$, we have $f'(x_2') < 0$ for some $x_2 < x_2' < 1$. Thus, if $f'(x_1) > 0$ or $f'(x_2) > 0$ is true, $f'(x) = 0$ must hold true at least twice in $x \in (x_1', x_2')$ by the continuity of f'. Consider the other case where $f'(x_1) \leq 0$ and $f'(x_2) \leq 0$. Since $f(x_1) = f(x_2) = 0$, this clearly implies that $f'(x) = 0$ occurs at least twice in $x \in [x_1, x_2]$. This proves the contrapositive statement.

We now conclude the uniqueness proof by showing that $f'(x) = 0$ occurs at most once in $x \in (0,1)$.

$$(d-1)\rho_2(1-(\rho_1+\rho_2(1-x)))^{d-2} = 1$$

$$\Rightarrow\ 1-(\rho_1+\rho_2(1-x)) = \pm\left(\frac{1}{\rho_2(d-1)}\right)^{1/(d-2)}$$

$$\Rightarrow\ \frac{x}{\rho_1} = \frac{1}{\rho_1}+\frac{1}{\rho_2}-\frac{1}{\rho_1\rho_2}\pm\frac{1}{\rho_1\rho_2}\left(\frac{1}{\rho_2(d-1)}\right)^{1/(d-2)}.$$

Here, $f'(x) = 0$ has only one solution if d is odd and has two solutions if d is even, corresponding to the $\pm$ above.

We now lower bound the distance between two solutions, say x_1 and x_2: $|x_1 - x_2| = \frac{2}{\rho_2}(\frac{1}{\rho_2(d-1)})^{1/(d-2)}$. Here, I) $\frac{2}{\rho_2} > e$ since $\rho_2 = (1+o(1))\lambda e^{-\lambda} \leq 2\max_{t>0} te^{-t} = 2/e$, II) $(1/\rho_2)^{1/(d-2)} > 1$, and III) $(1/(d-1))^{1/(d-2)} \geq 1/2$ for all $d \geq 3$. The last inequality holds as $(1/(d-1))^{1/(d-2)}$ is an increasing function for $d > 0$ and the value of this function at $d = 3$ is $1/2$. Thus, $|x_1 - x_2| > e/2 > 1$. Hence, $f'(x) = 0$ cannot have more than one solution in $x \in (0,1)$, implying the uniqueness of the solution.

Since ε is the unique solution of $f(x) = 0, f(1) < 0$, and $\eta(x)$ is an increasing function for all $0 \leq x \leq 1$, we have $f(x) < 0$ for all $x \in (\varepsilon, 1]$, i.e., $\eta(x) < x$ for all $x \in (\varepsilon, 1]$. This proves that the sequence is strictly decreasing.

We now show that one can get arbitrarily close to ε in a constant number of iterations. Define $\xi(\varepsilon') := \min_{p\in[\varepsilon+\varepsilon',1]}[p-\eta(p)]$. Since $\eta(p) < p$ for all $p \in (\varepsilon, 1]$, $\xi > 0$. Since $p_{j+1} - p_j = \eta(p_j) - p_j \leq -\xi(\varepsilon')$, it takes at most $1/\xi(\varepsilon')$ iterations to reach $\varepsilon + \varepsilon'$. This also implies the convergence of the sequence to the fixed point. ■

2) Proof of $C(\varepsilon) \to \infty$ as $\varepsilon \to 0$:

Proof: From the constraint of the optimization problem, we have $d = 1 + \frac{\log\varepsilon}{\log(1-e^{-\lambda}-\lambda e^{-\lambda})}$. Thus,

$$C(\varepsilon) = \min_{\lambda>0}\frac{d}{\lambda} > \min_{\lambda>0}\frac{\log\varepsilon}{\lambda(\log(1-e^{-\lambda}-\lambda e^{-\lambda})} = \log\varepsilon\cdot\frac{1}{\min_{\lambda>0}\lambda(\log(1-e^{-\lambda}-\lambda e^{-\lambda})} > \frac{\log(1/\varepsilon)}{2}.$$

Here, the last inequality is due to

$$\min_{\lambda>0}\lambda(\log(1-e^{-\lambda}-\lambda e^{-\lambda})) \approx -1.33301 > -2.$$

Hence, as $\varepsilon \to 0$, $C(\varepsilon) \to \infty$. ■

3) Proof of Lemma 5:

Proof: Note that the proof is nearly identical to that of [51]. We provide the proof of this lemma (with slight modification) for the sake of self-containedness. Consider a directed edge $\vec{e} = (v, c)$ from a left-node v to a right-node c. We call an edge *identified* if the right-node c is resolved. Define the directed neighborhood of depth ℓ of $\vec{e}$ as $\mathcal{N}_{\vec{e}}^{\ell}$, that is the subgraph of all the edges and nodes on paths having length less than or equal to ℓ, that start from v and the first edge of the path is not $\vec{e}$. We now prove the following lemma.

Lemma 10: For a fixed ℓ^*, $\mathcal{N}_{\vec{e}}^{2\ell^*}$ is a tree-like neighborhood with probability at least $1 - O(\log(K)^{\ell^*}/K)$.

Proof: Let C_ℓ be the number of right-nodes and V_ℓ be the number of left-nodes in $\mathcal{N}_{\vec{e}}^{2\ell}$. Since the ensemble of the graphs we consider is left-regular, we cannot immediately use the result of [53]. Note that the degree distribution of right nodes is binomial distribution. The key idea is to show that the size of the tree is bounded by $O(\log(K)^{\ell})$ with high probability. This is intuitively clear since binomial distribution has a tail decaying faster than exponential decay. To formally show this, we keep unfolding the tree up to level ℓ^*, and at each level ℓ we upper bound the probability that the size of the tree grows larger than $(\log(K)^{\ell})$. Fix some constant c_1. We upper bound the probability of not having a tree as follows.

$$\begin{aligned}\Pr(\mathcal{N}_{\vec{e}}^{2\ell^*}\text{is not a tree}) \leq{}& \Pr(V_{\ell^*} > c_1\log(K)^{\ell^*}) \\ &+ \Pr(C_{\ell^*} > c_1\log(K)^{\ell^*}) \\ &+ \Pr(\mathcal{N}_{\vec{e}}^{2\ell^*}\text{is not a tree}|V_{\ell^*} < c_1\log(K)^{\ell^*}, \\ &C_{\ell^*} < c_1\log(K)^{\ell^*}).\end{aligned}$$

Note that since the left degree is a constant, d, if V_{ℓ^*} is of order $\log(K)^{\ell^*}$ or less, C_{ℓ^*} is also of order $\log(K)^{\ell^*}$ or less. Let $\alpha_\ell = \Pr(V_\ell > c_1\log(K)^{\ell})$. Then,

$$\begin{aligned}\alpha_\ell &\leq \alpha_{\ell-1} + \Pr(V_\ell > c_1\log(K)^{\ell}|V_{\ell-1} < c_1\log(K)^{\ell-1}) \\ &\leq \alpha_{\ell-1} + \Pr(V_\ell > c_1\log(K)^{\ell}|C_\ell < c_2\log(K)^{\ell-1}),\end{aligned}$$

where the second inequality is due to the fact that every left node has exactly d edges connected to right nodes so if $V_{\ell-1} < c_1\log(K)^{\ell-1}$, there exists some constant c_2 such that $C_\ell < c_2\log(K)^{\ell-1}$. To count the number of left nodes in depth ℓ, let $n_\ell < C_\ell$ be the number of right nodes exactly at depth ℓ after unfolding the tree. Let X_i, $1 \leq i \leq n_\ell$ be the degree of these right nodes. Given that $V_{\ell-1} < c_1\log(K)^{\ell-1}$, one has $V_\ell > c_1\log(K)^{\ell}$, only if $X = \sum_{i=1}^{n_\ell} X_i > c_3\log(K)^{\ell}$ for some constant c_3. While the original distribution of X is binomial distribution with parameters $(n_\ell K, d/M)$, we will bound the tail probability of a Poisson random variable with parameter $n_\ell K d/M$. Since the total variation between a Poisson random variable with parameter np and a binomial random variable with parameter (n,p) is bounded above by p [59], the difference between the Poisson$(n_\ell K d/M)$ tail probability and the binomial$(n_\ell K, d/M)$ tail probability is bounded above by $2d/M = O(1/K)$.

We know that the tail probability of a Poisson random Y variable with parameter λ can be upper bounded as follows:

$\Pr(Y \geq y) \leq (\frac{e\lambda}{y})^y$ [60, Theorem 5.4]. Thus,

$$\Pr(X > c_3 \log(K)^\ell) \leq \left(\frac{c_4}{\log(K)}\right)^{c_3 \log(K)^\ell} \leq O(1/K).$$

Thus, $\alpha_\ell \leq \alpha_{\ell-1} + \frac{c_5}{K}$, for some constant c_5. Now since ℓ^* is a constant, summing up these inequalities, we show that

$$\alpha_{\ell^*} = \Pr(V_{\ell^*} > c_1 \log(K)^{\ell^*}) \leq O(1/K).$$

Similarly one can show $\Pr(C_{\ell^*} > c_1 \log(K)^{\ell^*}) \leq O(1/K)$.

To complete the proof, we need to show that with high probability, we have a tree-like neighborhood, given that the number of nodes is bounded by order of $\log(K)^{\ell^*}$. First, we find a lower bound on the probability that $\mathcal{N}_{\vec{e}}^{2\ell+1}$ is a tree-like neighborhood if $\mathcal{N}_{\vec{e}}^{2\ell}$ is a tree-like neighborhood, when $\ell < \ell^*$. Assume that t additional edges have been revealed at this stage without forming a cycle. The probability that the next edge from a left node does not create a cycle is the probability that it is connected to one of the right nodes that is not already in the subgraph which is lower bounded by $1 - \frac{C_{\ell^*}}{m}$. Thus, the probability that $\mathcal{N}_{\vec{e}}^{2\ell+1}$ is a tree-like neighborhood if $\mathcal{N}_{\vec{e}}^{2\ell}$ is a tree-like neighborhood, is lower-bounded by $(1 - \frac{C_{\ell^*}}{M})^{C_{\ell+1} - C_\ell}$. Similarly, the probability that $\mathcal{N}_{\vec{e}}^{2\ell+2}$ is a tree-like neighborhood if $\mathcal{N}_{\vec{e}}^{2\ell+1}$ is a tree-like neighborhood, is lower-bounded by $(1 - \frac{V_{\ell^*}}{K})^{V_{\ell+1} - V_\ell}$. Thus, the probability that $\mathcal{N}_{\vec{e}}^{2\ell^*}$ is a tree-like neighborhood is lower-bounded by $(1 - \frac{V_{\ell^*}}{K})^{V_{\ell^*}} (1 - \frac{C_{\ell^*}}{M})^{C_{\ell^*}}$, which is lower bounded by $1 - (V_{\ell^*}^2/K + C_{\ell^*}^2/M) + o(1)$. Since V_{ℓ^*} and C_{ℓ^*} are upper-bounded by $\Theta(\log(K)^{\ell^*})$, the probability of having a tree-like neighborhood is at least $1 - O(\log(K)^{\ell^*}/K)$. ■

We are now ready to prove the lemma. The proof follows similar steps as in [53], with the difference that the right degree is irregular and binomial-distributed. First, we prove (11). Let $Z_i = 1_{\{\vec{e}_i \text{ is identified}\}}$, $1 \leq i \leq Kd$ be the indicator that $\vec{e}_i$ is identified after ℓ iterations of the algorithm. Let B be the event that $\mathcal{N}_{\vec{e}_1}^{2\ell}$ is tree-like. Then,

$$\begin{aligned}\mathbb{E}[Z_1] &= \mathbb{E}[Z_1|B]\Pr(B) + \mathbb{E}[Z_1|\bar{B}]\Pr(\bar{B}) \\ &\leq \mathbb{E}[Z_1|B] + \Pr(\bar{B}) \leq p_\ell + \frac{\gamma \log(K)^\ell}{K},\end{aligned}$$

for some constant γ, where the last inequality is by Lemma 10. Since $|\mathbb{E}[Z_1|B]| \leq 1$, $\mathbb{E}[Z] = Kd\mathbb{E}[Z_1]$. Hence,

$$Kd\left(1 - \frac{\gamma \log(K)^\ell}{K}\right) < \mathbb{E}[Z] < Kd\left(p_\ell + \frac{\gamma \log(K)^\ell}{K}\right).$$

Then, (11) follows from choosing K large enough such that $\frac{K}{\log(K)^\ell} > \frac{2\gamma}{\varepsilon}$.

Second, we prove that

$$\Pr(|Z - Kdp_\ell| > Kd\varepsilon/2) < 2e^{-\beta\varepsilon^2 K^{1/(2\ell+1)}}. \tag{20}$$

Then, (12) follows from (11) and (20). To prove (20), we use the standard Martingale argument and Azuma's inequality provided in [53] with some modifications to account for the right irregular degree. Suppose that we expose the Kd edges of the graph one at a time. Let $Y_i = \mathbb{E}[Z|e_1^i]$. By definition, $Y_0, Y_1, \ldots, Y_{Kd}$ is a Doob's martingale process, where $Y_0 = \mathbb{E}[Z]$ and $Y_{Kd} = Z$. To use Azuma's inequality, we find the appropriate upper bound: $|Y_{i+1} - Y_i| \leq \alpha_i$. If the right-degree is regular and equal to d_c, it is shown in [53] that α_i can be chosen as $8(d_v d_c)^\ell$. We show that when the right degree has binomial distribution, the degree of all of the right nodes can be upper bounded by $K^{\frac{1}{2\ell+0.5}}$ with probability $K(e^{-\beta_1 K^{\frac{1}{2\ell+0.5}}})$ for some constants β_1.

To show this, let X be a binomial random variable with parameters $(K, d/M)$. By applying the standard Chernoff bound, we know that the tail probability of a binomial random Y variable with parameter (n, p) can be upper bounded as follows: $\Pr(Y \geq y) \leq \exp(-nD(\frac{y}{n} \parallel p))$, where $D(p \parallel q) = p \log p/q + (1-p)\log(1-p)/(1-q)$, i.e., the Kullback–Leibler divergence between Bern(p) and Bern(q). When $p = o(1)$ and $q = o(1)$, the first term dominates, and hence $D(p \parallel q) \geq \beta_1 p \log p/q$ for some constant $0 < \beta_1 < 1$. Thus,

$$\begin{aligned}\Pr\left(X > K^{\frac{1}{2\ell+0.5}}\right) &\leq e^{-KD\left(K^{\frac{1}{2\ell+0.5}-1} \parallel \frac{d}{M}\right)} \\ &\leq e^{-K\left(\beta_1 K^{\frac{1}{2\ell+0.5}-1} \log \frac{C}{d} K^{\frac{1}{2\ell+0.5}}\right)} \\ &= e^{-\beta_1 K^{\frac{1}{2\ell+0.5}} \log \frac{C}{d} K^{\frac{1}{2\ell+0.5}}} \leq e^{-\beta_1 K^{\frac{1}{2\ell+0.5}}}.\end{aligned}$$

Now considering $M = \Theta(K)$ right nodes and using union bound, one can see that the probability that all the right nodes have degree less than $K^{\frac{1}{2\ell+0.5}}$ is at least $1 - K(e^{-\beta_1 K^{\frac{1}{2\ell+0.5}}})$. Let E be the event that at least one right node has degree larger than $K(e^{-\beta_1 K^{\frac{1}{2\ell+0.5}}})$. Given E has not happened, one can upper bound α_i^2 by $K^{\frac{2\ell}{2\ell+0.5}}$. Then,

$$\begin{aligned}&\Pr(|Z - Kdp_\ell| > Kd\varepsilon/2) \\ &\leq \Pr(|Z - Kdp_\ell| > Kd\varepsilon/2|\bar{E}) + \Pr(E) \\ &\leq 2e^{-\frac{K^2 d^2 \varepsilon^2/4}{2\sum_i \alpha_i^2}} + K(e^{-\beta_1 K^{\frac{1}{2\ell+0.5}}}) \leq 2e^{-\beta\varepsilon^2 K^{1/(4\ell+1)}}.\end{aligned}$$

■

ACKNOWLEDGMENT

The authors would like to thank anonymous reviewers for providing us with invaluable comments. The authors would also like to thank P. Kairouz for discussions on the formulation of A-SAFFRON.

REFERENCES

[1] K. Lee, R. Pedarsani, and K. Ramchandran, "Saffron: A fast, efficient, and robust framework for group testing based on sparse-graph codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2016, pp. 2873–2877.

[2] K. Chandrasekher, K. Lee, P. Kairouz, R. Pedarsani, and K. Ramchandran, "Asynchronous and noncoherent neighbor discovery for the IoT using sparse-graph codes," in *Proc. IEEE Int. Conf. Commun.*, May 2017, pp. 1–6.

[3] R. Dorfman, "The detection of defective members of large populations," *Ann. Math. Statist.*, vol. 14, no. 4, pp. 436–440, 1943.

[4] L. Tong, V. Naware, and P. Venkitasubramaniam, "Signal processing in random access," *IEEE Signal Process. Mag.*, vol. 21, no. 5, pp. 29–39, Sep. 2004.

[5] A. C. Gilbert, M. A. Iwen, and M. J. Strauss, "Group testing and sparse signal recovery," in *Proc. 42nd Asilomar Conf. Signals, Syst. Comput.*, Oct. 2008, pp. 1059–1063.

[6] A. Emad and O. Milenkovic, "Poisson group testing: A probabilistic model for Boolean compressed sensing," *IEEE Trans. Signal Process.*, vol. 63, no. 16, pp. 4396–4410, Aug. 2015.

[7] C. Wang, Q. Zhao, and C. N. Chuah, "Optimal nested test plan for combinatorial quantitative group testing," *IEEE Trans. Signal Process.*, vol. 66, no. 4, pp. 992–1006, Feb. 2018.

[8] M. Goodrich, M. Atallah, and R. Tamassia, "Indexing information for data forensics," in *Applied Cryptography and Network Security* (Series Lecture Notes in Computer Science), J. Ioannidis, A. Keromytis, and M. Yung, Eds., vol. 3531. Berlin, Germany: Springer, 2005, pp. 206–221.

[9] H.-B. Chen and F. Hwang, "A survey on nonadaptive group testing algorithms through the angle of decoding," *J. Combinatorial Optim.*, vol. 15, no. 1, pp. 49–59, 2008.

[10] D. Malioutov and K. Varshney, "Exact rule learning via Boolean compressed sensing," in *Proc. 30th Int. Conf. Mach. Learn.*, Feb. 2013, pp. 765–773.

[11] S. Ubaru and A. Mazumdar, "Multilabel classification with group testing and codes," in *Proc. 34th Int. Conf. Mach. Learn.*, Aug. 2017, vol. 70, pp. 3492–3501.

[12] A. Ganesan, S. Jaggi, and V. Saligrama, "Learning immune-defectives graph through group tests," *IEEE Trans. Inf. Theory*, vol. 63, no. 5, pp. 3010–3028, 2017.

[13] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[14] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[15] F. K. Hwang *et al.*, *Combinatorial Group Testing and its Applications*. Singapore: World Scientific, 2000.

[16] U. Nakarmi and N. Rahnavard, "BCS: Compressive sensing for binary sparse signals," in *Proc. IEEE Military Commun. Conf.*, Oct. 2012, pp. 1–5.

[17] S.-S. Choi and J. H. Kim, "Optimal query complexity bounds for finding graphs," in *Proc. 40th Annu. ACM Symp. Theory Comput.*, 2008, pp. 749–758.

[18] F. H. Hao, "The optimal procedures for quantitative group testing," *Discrete Appl. Math.*, vol. 26, no. 1, pp. 79–86, Dec. 1989.

[19] M. Aigner and M. Schughart, "Determining defectives in a linear order," *J. Statistical Planning Inference*, vol. 12, pp. 359–368, 1985.

[20] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[21] A. Mazumdar, "Nonadaptive group testing with random set of defectives," *IEEE Trans. Inf. Theory*, vol. 62, no. 12, pp. 7522–7531, 2016.

[22] M. Aldridge, L. Baldassini, and O. Johnson, "Group testing algorithms: Bounds and simulations," *IEEE Trans. Inf. Theory*, vol. 60, no. 6, pp. 3671–3687, 2014.

[23] H. Q. Ngo and D.-Z. Du, "A survey on combinatorial group testing algorithms with applications to DNA library screening," *Discrete Math. Problems Med. Appl.*, vol. 55, pp. 171–182, 2000.

[24] A. G. D'yachkov and V. V. Rykov, "Bounds on the length of disjunctive codes," *Problemy Peredachi Informatsii*, vol. 18, no. 3, pp. 7–13, 1982.

[25] A. G. D'yachkov and V. V. Rykov, "Superimposed distance codes," *Problems Control Inform. Theory/Problemy Upravlen. Teor. Inf.*, vol. 18, no. 4, pp. 273–250, 1989.

[26] H. Q. Ngo and D.-Z. Du, "New constructions of non-adaptive and error-tolerance pooling designs," *Discrete Math.*, vol. 243, no. 1–3, pp. 161–170, 2002.

[27] M. Cheraghchi, "Noise-resilient group testing: Limitations and constructions," in *Proc. Int. Symp. Fundam. Comput. Theory*, 2009, pp. 62–73.

[28] P. Indyk, H. Q. Ngo, and A. Rudra, "Efficiently decodable non-adaptive group testing," in *Proc. 21st Annu. ACM-SIAM Symp. Discrete Algorithms*, 2010, pp 1126–1142.

[29] W. J. Bruno *et al.*, "Efficient pooling designs for library screening," *Genomics*, vol. 26, no. 1, pp. 21–30, 1995.

[30] F. K. Hwang, "Random k-set pool designs with distinct columns," *Probability. Eng. Inf. Sci.*, vol. 14, no. 1, pp. 49–56, Jan. 2000.

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

[32] M. B. Malyutov, "The separating property of random matrices," *Math. Notes Acad. Sci. USSR*, vol. 23, no. 1, pp. 84–91, 1978.

[33] G. K. Atia and V. Saligrama, "Boolean compressed sensing and noisy group testing," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, pp. 1880–1901, Mar. 2012.

[34] C. L. Chan, S. Jaggi, V. Saligrama, and S. Agnihotri, "Non-adaptive group testing: Explicit bounds and novel algorithms," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 3019–3035, 2014.

[35] J. Scarlett and V. Cevher, "Phase transitions in group testing," in *Proc. 27th Annu. ACM-SIAM Symp. Discrete Algorithms*, Jan. 2016, pp. 40–53.

[36] J. Scarlett and V. Cevher, "Limits on support recovery with probabilistic models: An information-theoretic framework," *IEEE Trans. Inf. Theory*, vol. 63, no. 1, pp. 593–620, Jan. 2017.

[37] A. Barg and A. Mazumdar, "Group testing schemes from codes and designs," *IEEE Trans. Inf. Theory*, vol. 63, no. 11, pp. 7131–7141, Nov. 2017.

[38] S. Cai, M. Jahangoshahi, M. Bakshi, and S. Jaggi, "GROTESQUE: Noisy group testing (quick and efficient)," in *Proc. 51st Annu. Allerton Conf. Commun., Control, Comput.*, Oct. 2013, pp. 1234–1241.

[39] M. Mézard, M. Tarzia, and C. Toninelli, "Group testing with random pools: Phase transitions and optimal strategy," *J. Statistical Phys.*, vol. 131, no. 5, pp. 783–801, Jun. 2008.

[40] T. Wadayama, "Nonadaptive group testing based on sparse pooling graphs," *IEEE Trans. Inf. Theory*, vol. 63, no. 3, pp. 1525–1534, Mar. 2017.

[41] S. Pawar and K. Ramchandran, "FFAST: An algorithm for computing an exactly k-sparse DFT in $O(k \log k)$ time," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 429–450, 2017.

[42] S. Pawar and K. Ramchandran, "R-FFAST: A robust sub-linear time algorithm for computing a sparse DFT," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 451–466, 2017.

[43] X. Li, J. Bradley, S. Pawar, and K. Ramchandran, "The SPRIGHT algorithm for robust sparse Hadamard Transforms," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2014, pp. 1857–1861.

[44] X. Li, D. Yin, S. Pawar, R. Pedarsani, and K. Ramchandran, "Sub-linear time support recovery for compressed sensing using sparse-graph codes," *IEEE Trans. Inf. Theory*, 2019.

[45] R. Pedarsani, K. Lee, and K. Ramchandran, "Phasecode: Fast and efficient compressive phase retrieval based on sparse-graph-codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 6, pp. 3663–3691, 2017.

[46] R. Pedarsani, K. Lee, and K. Ramchandran, "Capacity-approaching phasecode for low-complexity compressive phase retrieval," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2015, pp. 989–993.

[47] D. Yin, K. Lee, R. Pedarsani, and K. Ramchandran, "Fast and robust compressive phase retrieval with sparse-graph codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2015, pp. 2583–2587.

[48] S. Pawar and K. Ramchandran, "Ffast: An algorithm for computing an exactly k -sparse dft in $o(k \log k)$ time," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 429–450, Jan. 2018.

[49] X. Li and K. Ramchandran, "An active learning framework using sparse-graph codes for sparse polynomials and graph sketching," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2170–2178.

[50] X. Li and K. Ramchandran, "Recovering k-sparse n-length vectors in o(k log n) time: Compressed sensing using sparse-graph codes," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Mar. 2016, pp. 4049–4053.

[51] R. Pedarsani, D. Yin, K. Lee, and K. Ramchandran, "Phasecode: Fast and efficient compressive phase retrieval based on sparse-graph codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 6, pp. 3663–3691, Jun. 2017.

[52] O. Johnson, M. Aldridge, and J. Scarlett, "Performance of group testing algorithms with near-constant tests per item," *IEEE Trans. Inf. Theory*, vol. 65, no. 2, pp. 707–723, Feb. 2019.

[53] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[54] A. Shokrollahi, "LDPC codes: An introduction," in *Coding, Cryptography and Combinatorics*. Berlin, Germany: Springer, 2004, pp. 85–110.

[55] A. Reisizadeh, P. Abdalla, and R. Pedarsani, "Sub-linear time stochastic threshold group testing via sparse-graph codes," in *Proc. IEEE Inf. Theory Workshop*, Nov. 2018, pp. 1–5.

[56] S. Kudekar, T. Richardson, and R. Urbanke, "Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 803–834, Feb. 2011.

[57] J. Wolf, "Born again group testing: Multiaccess communications," *IEEE Trans. Inf. Theory*, vol. 31, no. 2, pp. 185–191, Mar. 1985.

[58] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. London, U.K.: Pearson, 2004.

[59] A. D. Barbour and P. Hall, "On the rate of Poisson convergence," *Math. Proc. Cambridge Philosophical Soc.*, vol. 95, no. 3, pp. 473–480, May 1984.

[60] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2005.