Compressed Sensing Using Sparse-Graph Codes for the Continuous-Alphabet Setting

Dong Yin, Ramtin Pedarsani, Xiao Li, Kannan Ramchandran
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Email: {dongyin, ramtin, xiaoli, kannanr}@eecs.berkeley.edu

***Abstract*—In this paper, we consider the compressive sensing (CS) problem in the presence of noise. The problem is to recover a K-sparse signal $\boldsymbol{s} \in \mathbb{R}^n$ from noisy linear measurements $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{w}$. We propose a fast recovery algorithm that can reconstruct any K-sparse signal $\boldsymbol{s}$ with time complexity that grows linearly in K and sublinearly in n. Specifically, with high probability, our algorithm is able to recover an arbitrarily large fraction of the support of the sparse signal using $\Theta(K\log(n)\log\log(n))$ samples and $\Theta(K\log^{1+r}(n))$ computational cost, where $r > 0$ is an arbitrarily small constant. The sample and time complexities are near order-optimal. Further, our algorithm is able to recover the exact support with $\Theta(K\log(K)\log(n)\log\log(n))$ measurements and time complexity of $\Theta(K\log(K)\log^{1+r}(n))$. With a mild technical assumption on the existence of a code with universal decoding algorithm and small decoding complexity, our algorithm can achieve $\Theta(K\log(n))$ sample and time complexities for the large fraction recovery, and furthermore, $\Theta(K\log(K)\log(n))$ sample and time complexities for the full support recovery. The design of measurements and the recovery algorithm are based on sparse graph codes. We also justify our theoretical results with numerical experiments.**

I. Introduction

Sparse signal processing has become an important area in both theory and applications [1], [2]. Compressive sensing (CS) [3] is one of the major problems in this area that has been largely studied in the literature. The goal of compressive sensing is to recover a sparse signal from linear measurements using a small number of samples, and in addition, the computational cost of the algorithm should also be as small as possible. Most of the common approaches to compressive sensing, such as ℓ_1 norm minimization [3], Iterative Hard Thresholding [4], and the OMP algorithm [5], require $m = \mathcal{O}(K\log(n))$ samples to guarantee successful recovery, where K and n denote the sparsity and dimension of the signal, respectively. However, the time complexities of these algorithms are usually $\Omega(n)$, and this computational cost can be prohibitive when processing signals with high dimensions. Therefore, it is becoming increasingly important to study fast algorithms for compressive sensing. Some fast algorithms [6]–[8], for compressive sensing have been proposed in recent years. Among these algorithms, the peeling style algorithm based on sparse graph codes [9] is shown to have almost order-optimal sample and time complexities $\Theta(K\log(n))$ for noisy compressive sensing.

One shortcoming of the peeling style algorithm in [9] is that it is based on the assumption that the non-zero elements in the signal lie in finite constellation points, which is impractical in some applications. In this paper, we propose an algorithm that relaxes this assumption and is able to reconstruct signals with components coming from a continuous alphabet, while maintaining sublinear sample and time complexities. Specifically, we consider two types of recovery guarantees: one is to recover an arbitrarily large fraction of the support of the sparse signal (large fraction recovery), and the other is to recover the full support of the signal (full support recovery). We provide the recovery guarantees using ℓ_∞ norm bounds with error probability approaching 0. Specifically, our algorithm can recover an arbitrarily large fraction of the support of the sparse signal using $\Theta(K\log(n)\log\log(n))$ samples and $\Theta(K\log^{1+r}(n))$ computational cost, where $r > 0$ is an arbitrarily small constant. The sample and time complexities are near order-optimal. To recover the exact support, our algorithm needs $\Theta(K\log(K)\log(n)\log\log(n))$ measurements and time complexity of $\Theta(K\log(K)\log^{1+r}(n))$. With a mild technical assumption on the existence of a code with universal decoding algorithm and small decoding complexity, our algorithm can achieve $\Theta(K\log(n))$ sample and time complexities for the large fraction recovery, and furthermore, $\Theta(K\log(K)\log(n))$ sample and time complexities for the full support recovery.[1]

Our algorithm also provides a general framework for the signal processing and machine learning algorithms which are based on sparse graph codes to provably work for continuous-valued signals in the presence of noise. Specifically, we point out that the robustified algorithms for sparse DFT [11], compressive phase retrieval [12] and sparse covariance estimation [13] can also work for continuous-valued signals using the methodology that we propose in this paper.

A. Problem Formulation

We formally define the problem as follows. Let $\boldsymbol{s} \in \mathbb{R}^n$ be a K-sparse unknown signal. With measurement matrix $\boldsymbol{A} \in$

[1]While preparing this paper, we were aware of another work which uses a scheme similar to our algorithm [10]. However, our algorithm has stronger theoretical guarantee than the algorithm in [10] due to the following two reasons. First, the algorithm in [10] guarantees a constant error probability by using $\Theta(K\log^2(n))$ samples while our algorithm can use a similar amount of samples to achieve vanishing error probability. This is due to our truncation scheme which will be detailed in the following sections. Second, we propose a concatenated code which gives rigorous guarantee of the channel coding subroutine of the algorithm.

$\mathbb{R}^{m\times n}$, we can get the linear measurements $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{w}$, where $\boldsymbol{w} \sim \mathcal{N}(0, \sigma^2\boldsymbol{I})$ denotes the i.i.d. Gaussian noise, with known variance σ^2. We define the *support* of the signal $\boldsymbol{s}$, denoted by $\mathrm{supp}(\boldsymbol{s})$, as the set of indices of the non-zero elements of $\boldsymbol{s}$, i.e., $\mathrm{supp}(\boldsymbol{s}) = \{j : s_j \neq 0\}$. Consequently, we have $|\mathrm{supp}(\boldsymbol{s})| = K$ in our problem. We assume that the sparse signal $\boldsymbol{s}$ satisfies the following minimum magnitude assumption:

Assumption 1. *For each $j \in \mathrm{supp}(\boldsymbol{s})$, $|s_j| \geq \beta$ for some constant $\beta > 0$.*

We consider the asymptotic regime where n and K approach infinity, and we assume no further assumption on n and K. Our goal is to design the measurement matrix $\boldsymbol{A}$ and the decoding algorithm such that we can decode $\boldsymbol{s}$ with small sample and time complexity.

B. Notations

In this paper, for any positive integer k, $[k]$ denotes the set $\{1, 2, \ldots, k\}$. For a vector $\boldsymbol{x}$, x_j denotes the jth element of $\boldsymbol{x}$. For $\boldsymbol{x} \in \mathbb{R}^n$ and a subset $\Gamma \subset [n]$, $\boldsymbol{x}_\Gamma$ is a vector consisting of the elements of $\boldsymbol{x}$ indexed by Γ. We use standard BIG-O notations $\mathcal{O}(\cdot)$, $\Theta(\cdot)$, and $\Omega(\cdot)$. The notation $\mathrm{poly}(n)$ denotes a polynomial with an arbitrarily large constant degree.

II. Main Results

We state our main results in this section. Our first result provides the guarantee for recovering an arbitrarily large fraction $1-\delta$ of the support (we call this kind of recovery guarantee the *large fraction recovery*).

Theorem 1. *Let $\boldsymbol{s} \in \mathbb{R}^n$ be a K-sparse signal with support Γ, satisfying Assumption 1. Let $\epsilon > 0$ be a preassigned accuracy parameter, and suppose that there exist universal constants c, c_3, and c_4 such that $\beta > \max\{c\epsilon, (c_3\sigma + c_4\epsilon)^2\}$. Let $\hat{\boldsymbol{s}}$ be the signal recovered by our proposed algorithm and $\hat{\Gamma} = \mathrm{supp}(\hat{\boldsymbol{s}})$. Then, for an arbitrarily small $\delta > 0$, with $m = \Theta(\log(1/\delta)K\log(n)[\log\log(n) + \max\{\sigma^2/\epsilon^2, 1\}])$ samples, our algorithm satisfies $\hat{\Gamma} \subset \Gamma$ and*

$$\begin{aligned}&\mathbb{P}\left\{|\hat{\Gamma}|/|\Gamma| \geq 1-\delta \text{ and } \|\hat{\boldsymbol{s}}_{\hat{\Gamma}} - \boldsymbol{s}_{\hat{\Gamma}}\|_\infty \leq C\epsilon\right\}\\ \geq& 1 - \mathcal{O}(\exp\{-c_1(\delta)K^{c_2(\delta)}\} + 1/\mathrm{poly}(n)),\end{aligned}$$

where C is a universal constant and $c_1(\delta)$ and $c_2(\delta)$ are determined only by δ. The time complexity is $\Theta(\log(1/\delta)K\log(n)[\log^r(n) + \max\{\sigma^2/\epsilon^2, 1\}])$, where $r > 0$ is an arbitrarily small constant.

Here, the probability statement comes from the randomness in measurement design of the algorithm. We can see that since σ, ϵ, and δ are constants, the sample and time complexities of the algorithm are $\Theta(K\log(n)\log\log(n))$ and $\Theta(K\log^{1+r}(n))$, respectively, and the probability of accurate large fraction recovery (in terms of ℓ_∞ norm) approaches 1 as n and K approach infinity. We need several steps to prove Theorem 1. These steps are given in Sections IV, V, and VI. The final proof will be concluded in Section VII.

In addition, since one single trial of the our algorithm can recover an arbitrarily large fraction of the support, we can make $\Theta(\log(K))$ trials of the algorithm with independent randomness to recover the full support of the sparse signal (we call this recovery guarantee the *full recovery*). We call this version of our algorithm the *repeated* algorithm, and provide the recovery guarantee here.

Theorem 2. *Assume the same conditions as in Theorem 1. Then, with $m = \Theta(K\log(K)\log(n)[\log\log(n) + \max\{\sigma^2/\epsilon^2, 1\}])$ samples, the repeated algorithm satisfies*

$$\begin{aligned}&\mathbb{P}\left\{\hat{\Gamma} = \Gamma \text{ and } \|\hat{\boldsymbol{s}} - \boldsymbol{s}\|_\infty \leq C\epsilon\right\}\\ \geq& 1 - \mathcal{O}(\exp\{-c_1K^{c_2}\}\log(K) + 1/\mathrm{poly}(n)),\end{aligned}$$

where c_1 and c_2 are universal constants. The time complexity is $\Theta(K\log(K)\log(n)[\log^r(n) + \max\{\sigma^2/\epsilon^2, 1\}])$.

We can see that since σ and ϵ are as constants, the sample and time complexities of the algorithm are $\Theta(K\log(K)\log(n))$ and $\Theta(K\log(K)\log(n))$, respectively, and the probability of accurate full recovery (in terms of ℓ_∞ norm) approaches 1 as n and K approaches infinity.

We will see in Section IV that one part of the recovery algorithm can be formulated as a decoding algorithm in a binary symmetric channel (BSC) with symbols ± 1 (we use the terminology BSC even though the conventional definition of BSC involves symbols $\{0, 1\}$), and that we only know an upper bound of the bit flip probability. We construct a *concatenated* code for this BSC, and the encoding and decoding algorithms only need the knowledge of an upper bound of the bit flip probability. The block length of the codeword is $\Theta(\log(n)\log\log(n))$ and the decoding complexity is $\Theta(\log^{1+r}(n))$. The decoding algorithm of the concatenated code provably succeeds with probability $\mathcal{O}(1/\mathrm{poly(n)})$. This is the reason for the appearance of the $\log\log(n)$ and $\log^r(n)$ factors in the sample and time complexities. More details will be provided in Section IV. Here, we mention that this code is used only for theoretical purpose. To the best of our knowledge, the decoding algorithms of the existing codes with decoding complexity linear in the block length of the codewords, such as the LDPC codes, all need the exact knowledge of the bit flip probability of the BSC channel to provably succeed, and therefore, we need this concatenated code. However, in practice, the existing state-of-the-art codes can work well in BSC with an upper bound of the bit flip probability. If we believe that there exists a code with universal decoding algorithm, decoding complexity linear in the block length of the code, and error probability exponentially small in the block length, then our results will be improved to $\Theta(K\log(n))$ sample and time complexities for the large fraction recovery, and $\Theta(K\log(K)\log(n))$ sample and time complexities for full recovery.

We summarize our results in Table I. Here, all the recovery guarantees have success probabilities approaching 1 as n and K approaches infinity.

TABLE I: Sample and time complexities

Type	large fraction	large fraction with conjecture	full recovery	full recovery with conjecture
Sample	$\Theta(K\log(n)\log\log(n))$	$\Theta(K\log(n))$	$\Theta(K\log(K)\log(n)\log\log(n))$	$\Theta(K\log(K)\log(n))$
Time	$\Theta(K\log^{1+r}(n))$	$\Theta(K\log(n))$	$\Theta(K\log(K)\log^{1+r}(n))$	$\Theta(K\log(K)\log(n))$

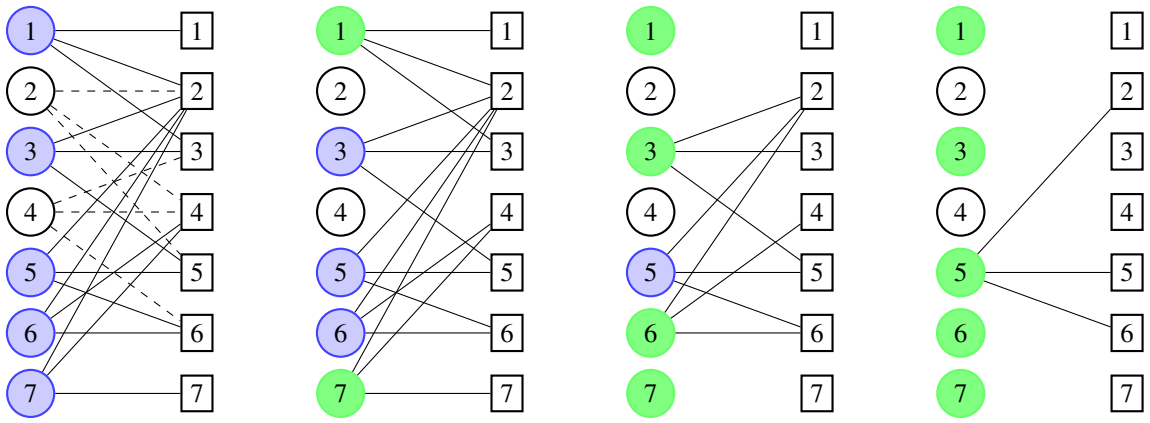

Fig. 1: Peeling style algorithm. Left nodes correspond to the signal elements; white, blue, and green nodes represent the zero elements, unrecovered non-zero elements, and recovered non-zero elements, respectively. (a) The bipartite graph representation of the design of measurements. The support of the signal is $\{1, 3, 5, 6, 7\}$. The connection between the zero elements and the bins are shown in dashed edges, since the zero elements do not have actual influence on the measurements. (b) Bins 1 and 7 are singletons, so the singleton balls s_1 and s_7 are recovered. (c) The decoder peels s_1 and s_7 from other bins. Then bins 3 and 4 become singletons with singleton balls s_3 and s_6, which can be recovered. (d) The decoder peels s_3 and s_6 from other bins and s_5 becomes a singleton ball. Then, all the non-zero elements are recovered.

III. Related Work

Compressive sensing has been extensively studied in recent years and various algorithms have been proposed for different problem settings. We do not attempt to provide a thorough review of all the related works. We refer the readers to [9] for a summary of the main classes of compressive sensing. Since our proposed algorithm is built upon of the peeling style sublinear time complexity algorithm for compressive sensing proposed in [9], we provide a brief review of the algorithm in [9], which is based on sparse graph codes, and the basic idea of the algorithm is to use a "divide-and-conquer" approach.

We use a bipartite graph to model the problem. We divide the m measurements into M sets, and call each set of measurements a *bin*. By careful design of the measurement matrix, we can make the measurements in each bin only influenced by a small number of signal elements. For example, for a 7 dimensional signal, if we design the ith row of the measurement matrix to be $\boldsymbol{a}_i^{\mathrm{T}} = [1\ 0\ 1\ 0\ 0\ 0\ 1]$, then the measurement y_i is only influenced by the first, third, and the seventh signal elements. Then, we can draw a bipartite graph with n left nodes representing the elements of the sparse signal, M right nodes representing the bins, and the edges representing the influence of the signal elements on the bins.

Further, we know that only the non-zero elements can have actual influence on the measurements. Therefore, we prune the graph and only focus on the K left nodes which correspond to the non-zero signal elements. We call these K left nodes *balls*. Then, we can classify the bins according to the number of balls connected to them (or equivalently, the number of non-zero elements which have influence on the bins). We say that the bins connected to zero, one, and more than one balls *zerotons*, *singletons*, and *multitons*, respectively. The single signal element connected to a singleton is called a *singleton ball*.

In the peeling style algorithm, the measurements in a bin are carefully designed such that one can detect whether a bin is a singleton or not, and if a bin is a singleton, one can find the location index and the value of the singleton ball[2]. Then we can subtract (or peel) the influence of this element from other bins. By doing this process iteratively, one can show that if the bipartite graph is properly designed, we can recover all the non-zero elements. We provide an example in Figure 1, and in Section V, we will revisit this example with a slightly different peeling strategy.

In the following sections, we will provide the details of the design of the measurement matrix, the peeling algorithm, and the singleton detection procedure of the our algorithm.

IV. Design of Measurements

The measurement matrix $\boldsymbol{A}$ of our algorithm is a row tensor of a *modulation* matrix $\boldsymbol{T} \in \{-1, 1\}^{R\times n}$ and a *code* matrix $\boldsymbol{H} \in \{0, 1\}^{M\times n}$ with rows $\boldsymbol{h}_i$ and entries $h_{i,j}$, $i \in [M]$,

[2]There are various ways to find the location index and the value. For example, in the noiseless case, one can use *ratio tests*. We do not provide details of the singleton detection process of the peeling style algorithm in [9] here. In later parts of this paper, we will provide the methods that we use in our algorithm.

$j \in [n]$, denoted by $\boldsymbol{A} = \boldsymbol{T} \otimes \boldsymbol{H}$. Specifically, the row tensor means that $\boldsymbol{A} = [\boldsymbol{A}_1^{\mathrm{T}}\ \boldsymbol{A}_2^{\mathrm{T}} \cdots \boldsymbol{A}_M^{\mathrm{T}}]^{\mathrm{T}}$, where $\boldsymbol{A}_i = \boldsymbol{T}\mathrm{diag}(\boldsymbol{h}_i)$ and $\mathrm{diag}(\boldsymbol{h}_i)$ is a square diagonal matrix with the elements of $\boldsymbol{h}_i$ on the main diagonal. Consequently, the total number of measurements is $m = MR$.

A. Code Matrix

The code matrix $\boldsymbol{H}$ is the biadjacency matrix of a d-left regular random bipartite graph, where d is a parameter of the algorithm. Specifically, we construct a random bipartite graph with n left nodes and M right nodes, and each left node is connected to d right nodes uniformly at random. The left nodes correspond to the n signal elements and the M right nodes correspond to the M sets of measurements generated by sub-matrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_M$. The i-th row of $\boldsymbol{H}$ represents the connection between the i-th right node and all the left nodes. For example, in Figure 1(a), the third right node is connected to the third and fourth left nodes, and therefore, the third row of $\boldsymbol{H}$ is $\boldsymbol{h}_3 = [1\ 0\ 1\ 1\ 0\ 0\ 0]$.

We still use the terminologies in Section III. We call the left nodes which correspond to the K non-zero signal elements *balls*, and call the M right nodes *bins*, and we still classify the bins according to the number of balls in them, using the terminologies *zerotons*, *singletons*, *singleton balls*, and *multitons*. We also define the measurements of bin i as $\boldsymbol{y}_i = \boldsymbol{A}_i\boldsymbol{s} + \boldsymbol{w}_i$, where $\boldsymbol{w}_i$ is the noise. Consequently, $\boldsymbol{y} = [\boldsymbol{y}_1^{\mathrm{T}}\ \boldsymbol{y}_2^{\mathrm{T}} \cdots \boldsymbol{y}_M^{\mathrm{T}}]^{\mathrm{T}}$ and $\boldsymbol{w} = [\boldsymbol{w}_1^{\mathrm{T}}\ \boldsymbol{w}_2^{\mathrm{T}} \cdots \boldsymbol{w}_M^{\mathrm{T}}]^{\mathrm{T}}$. Since $\boldsymbol{A}_i = \boldsymbol{T}\mathrm{diag}(\boldsymbol{h}_i)$, we have $\boldsymbol{y}_i = \boldsymbol{T}\boldsymbol{x}_i + \boldsymbol{w}_i$, where $\boldsymbol{x}_i = \mathrm{diag}(\boldsymbol{h}_i)\boldsymbol{s}$ is called the *true signal* of bin i. Our algorithm is a process of finding all the non-zero signal elements iteratively, or equivalently, finding the locations and values of the K balls. One should notice that the type of a bin can change during the decoding process due to the iterative peeling. For instance, if a bin contains two balls and one ball is peeled in a certain iteration, the bin becomes a singleton.

B. Modulation Matrix

Now we describe how we design the modulation matrix $\boldsymbol{T}$. The matrix $\boldsymbol{T}$ consists of two parts, the *location* matrix $\boldsymbol{L} \in \{-1, 1\}^{R_1 \times n}$ and the *verification* matrix $\boldsymbol{V} \in \{-1, 1\}^{R_2 \times n}$, i.e., $\boldsymbol{T} = [\boldsymbol{L}^{\mathrm{T}}\ \boldsymbol{V}^{\mathrm{T}}]^{\mathrm{T}}$, and thus, we have $R = R_1 + R_2$. We let $\boldsymbol{T}_j$, $\boldsymbol{L}_j$, and $\boldsymbol{V}_j$ be the columns of $\boldsymbol{T}$, $\boldsymbol{L}$, and $\boldsymbol{V}$, $j \in [n]$.

Before we demonstrate the design of the location matrix and the verification matrix, we briefly describe the functions of the two matrices. The modulation matrix $\boldsymbol{T}$ is used to check whether a bin is a singleton, and if a bin is a singleton, the modulation matrix finds the location index and estimate the value of the singleton ball. Suppose bin i is a singleton and the singleton ball is located at j, $j \in [n]$, i.e., $\boldsymbol{x}_i = s_j\boldsymbol{e}_j$, where $\boldsymbol{e}_j$ is the j-th vector of the standard basis. Then, the measurements of this bin is $\boldsymbol{y}_i = s_j\boldsymbol{T}_j + \boldsymbol{w}_i$. We divide the measurements into two parts, location measurements $\boldsymbol{y}_i^\ell$ and verification measurements $\boldsymbol{y}_i^v$, which correspond to the location matrix and verification matrix, respectively. Then, we have $\boldsymbol{y}_i^\ell = s_j\boldsymbol{L}_j + \boldsymbol{w}_i^\ell$ and $\boldsymbol{y}_i^v = s_j\boldsymbol{V}_j + \boldsymbol{w}_i^v$, where $\boldsymbol{w}_i^\ell$ and $\boldsymbol{w}_i^v$ are the corresponding noise vectors. The location matrix is used to find the location index j, and the verification matrix is used to check whether this bin is indeed a singleton and estimate the value of s_j.

The design of the verification matrix is relatively simple. The entries of the verification matrix $\boldsymbol{V}$ are i.i.d. Rademacher distributed, i.e., all the entries are independent and equally likely to be either 1 or -1. The detailed procedures of singleton verification and value estimation are shown in Section V. The design of the location matrix $\boldsymbol{L}$ is more complicated. Here, we demonstrate the basic idea. As we can see, if bin i is indeed a singleton, then the location measurements $\boldsymbol{y}_i^\ell$ is a scaled version of $\boldsymbol{L}_j$ with additive noise $\boldsymbol{w}_i^\ell$. Let $\zeta_i = \Phi(-|s_j|/\sigma)$, where $\Phi(\cdot)$ is the CDF of standard Gaussian distribution. Taking the sign of all the location measurements and considering the randomness of the Gaussian noise, we can see that for each element $y_{i,k}^\ell$ in the location measurements, $k \in [R_1]$, we have

$$\mathrm{sgn}(y_{i,k}^\ell) = \begin{cases} \mathrm{sgn}(s_j)L_{j,k} & \text{with probability } 1-\zeta_i \\ -\mathrm{sgn}(s_j)L_{j,k} & \text{with probability } \zeta_i. \end{cases}$$

Now the problem becomes a channel coding problem in a symmetric channel with symbols $\{+1, -1\}$. The channel is similar to the binary symmetric channel (BSC) except the fact that we are using $\{+1, -1\}$ rather than $\{0, 1\}$. For simplicity we will still call this channel a BSC in the following context. Consider the n possible locations of the singleton balls as n *messages*. We encode the n messages by R_1-bit *codewords* with symbols ± 1, or equivalently, we design a map $f : [n] \to \{1, -1\}^{R_1}$, and the columns of the location matrix are the codewords of all the messages, i.e., $\boldsymbol{L}_j = f(j)$, $j \in [n]$. If $s_j < 0$, the codeword gets a global sign flip and then we get the modified codeword $\mathrm{sgn}(s_j)\boldsymbol{L}_j$. Transmitting this modified codeword through a BSC with bit flip probability ζ_i, we get the received sequence, $\mathrm{sgn}(\boldsymbol{y}_i^\ell)$. Then we need a decoding algorithm to decode the original codeword $\boldsymbol{L}_j$, up to a global sign flip, and then, there are at most two possible locations at which the singleton ball can be. The verification measurements can check whether the bin is indeed a singleton, find the correct location among the two possible choices, and estimate the value of the singleton ball.

Now we describe the encoding and decoding scheme of the location matrix. The code should satisfy four properties:

(i) The block length of the codewords should be as small as possible. Since we need at least $\Theta(\log(n))$ bits to encode n messges, R_1 should be as close to $\Theta(\log(n))$ as possible.

(ii) The decoding complexity should be as close to $\Theta(\log(n))$ as possible.

(iii) The decoding algorithm succeeds with high probability; specifically, when there are $\Theta(1)$ bits flipped, we need the probability of successful decoding to be $\mathcal{O}(1/\mathrm{poly}(n))$.

(iv) The decoding algorithm should be *universal*, i.e., it should not rely on the exact knowledge of the bit flipping probability.

Many of the state-of-the-art capacity achieving codes, such

as LDPC codes and Polar codes, satisfy the first three properties. The decoding algorithms in these codes need exact knowledge of the channel, meaning that these algorithms need the flip probability ζ_i as a known input parameter. However, in our problem, $\zeta_i = \Phi(-|s_j|/\sigma)$, where $|s_j|$ is unknown. This is the reason that we need universal decoding algorithm. In practice, since we have an upper bound of the bit flip probability, $\zeta_i \leq \Phi(-\beta/\sigma)$, it is reasonable to believe that if we use the upper bound as the bit flip probability, the state-of-the-art capacity achieving codes still work well, although there is no theoretical guarantee. For theoretical interests, here we propose a *concatenated* code which satisfies all the four properties provably. The results are given in Lemma 1. This code is based on Justesen's concatenation scheme [14], linear complexity expander codes [15], and the Wozencraft's ensemble [16]

Lemma 1. *There exists a concatenated code*

$$f_c : [n] \to \{1, -1\}^{R_1}$$

for BSC with block length $R_1 = \Theta(\log(n)\log\log(n))$ *and universal decoding algorithm, which can successfully decode with probability* $\mathcal{O}(1/\mathrm{poly}(n))$. *The decoding complexity is* $\Theta(\log^{1+r}(n))$, *where* $r > 0$ *is an arbitrarily small constant.*

We refer the readers to the Appendix B for the details of the code and the proof of Lemma 1. With this concatenated code, we can construct the location matrix $\boldsymbol{L}$ by setting the j-th column as the codeword of j, i.e., $\boldsymbol{L}_j = f_c(j)$.

V. Peeling Decoder with Truncation

The basic idea of the peeling decoder is to use the location matrix and verification matrix to find the location and estimate the value of singleton balls. After identifying a singleton, the decoder peels this singleton ball from the bins that it is connected to. Then, more bins become singletons. The decoder continues the peeling process iteratively until no singletons can be found. The major difference between our work and [9] is that the signal components can be any $\Theta(1)$ real number with absolute value lower bounded by β, which implies that we cannot obtain the exact value of the singleton balls. Then, the error propagation in the peeling process is inevitable. In this paper, we propose a truncation peeling strategy that controls the error propagation.

We now revisit the example in Figure 1 using the truncation peeling strategy, and demonstrate the algorithm in Figure 2. The key in the truncated peeling strategy is that we fix the maximum number of balls that can be peeled from a bin. Denote this maximum number by D, which is an input constant parameter of the algorithm. This means that if at least D balls have been peeled from a particular bin, we stop using this bin in following iterations, i.e., we "truncate" large multitons with more than D balls. We set $D = 2$ in the example in Figure 2.

We first assume that by the location measurements and verification measurements, we can perfectly identify whether a bin is a singleton and find the location of the singleton ball. As we can see, in Figure 2, the bins 1 and 7 are singletons and the corresponding singleton balls are balls 1 and 7. In the first iteration, the two singleton balls are found and we let $\hat{s}_1$ and $\hat{s}_7$ be the estimated values. Then, we do peeling, meaning that we subtract the measurements contributed by the two singleton balls from the measurements in other bins. We get the *remaining* measurements of bins 2, 3, 4, 5, and 6 after the first iteration:

$$\begin{aligned}
\boldsymbol{y}_2^{(1)} &= \boldsymbol{y}_2 - \hat{s}_1\boldsymbol{T}_1 - \hat{s}_7\boldsymbol{T}_7 \\
\boldsymbol{y}_3^{(1)} &= \boldsymbol{y}_3 - \hat{s}_1\boldsymbol{T}_1 \\
\boldsymbol{y}_4^{(1)} &= \boldsymbol{y}_4 - \hat{s}_7\boldsymbol{T}_7 \\
\boldsymbol{y}_5^{(1)} &= \boldsymbol{y}_5 \\
\boldsymbol{y}_6^{(1)} &= \boldsymbol{y}_6.
\end{aligned}$$

Then we can see that bins 3 and 4 become singletons and the corresponding singleton balls are 3 and 6. We should also notice that since two balls have been peeled from bin 2, according to the truncated peeling strategy, no balls can be peeled from bin 2, and consequently bin 2 is not used in the following iterations. Let $\hat{s}_3$ and $\hat{s}_6$ be the estimated values of the two singleton balls. Then, the remaining measurements of bins 5 and 6 after the second iteration are:

$$\begin{aligned}
\boldsymbol{y}_5^{(2)} &= \boldsymbol{y}_5^{(1)} - \hat{s}_3\boldsymbol{T}_3 \\
\boldsymbol{y}_6^{(2)} &= \boldsymbol{y}_6^{(1)} - \hat{s}_6\boldsymbol{T}_6.
\end{aligned}$$

Then, bins 5 and 6 become singletons and the singleton ball is ball 5. We can estimate the value of ball 5 and get $\hat{s}_5$. So far, all the balls have been found, meaning that the all the non-zero elements are found.

We summarize the peeling procedure in Algorithm 1 in Appendix A. The following result of the peeling procedure guarantees that after the peeling process stops, an arbitrarily large fraction of balls are found.

Lemma 2. *Assume that we can always find the correct location indices of singleton balls. For any* $\delta > 0$, *when* K *is large enough, there exist proper parameters* d *and* $M = \Theta(\log(1/\delta)K)$, *such that after* $N(\delta)$ *iterations of truncation peeling, the fraction of non-zero signal elements which are not detected is less than* δ, *with probability* $\mathcal{O}(\exp\{-c_1(\delta)K^{c_2(\delta)}\})$. *Here,* $c_1(\delta), c_2(\delta) > 0$ *are two quantities determined by* δ.

We refer the readers to Appendix C for the proof of Lemma 2.

VI. Singleton Detection and Signal Estimation

In Section V, we have shown that if the singleton balls are always perfectly detected, an arbitrarily large fraction of non-zero signal elements can be found. Then, an important issue is to guarantee the correct singleton detection and accurate value estimation.

First, recall that we have shown in Section IV that, in the first iteration, if a bin is indeed a singleton, from the location measurements, one can decode the modified codeword

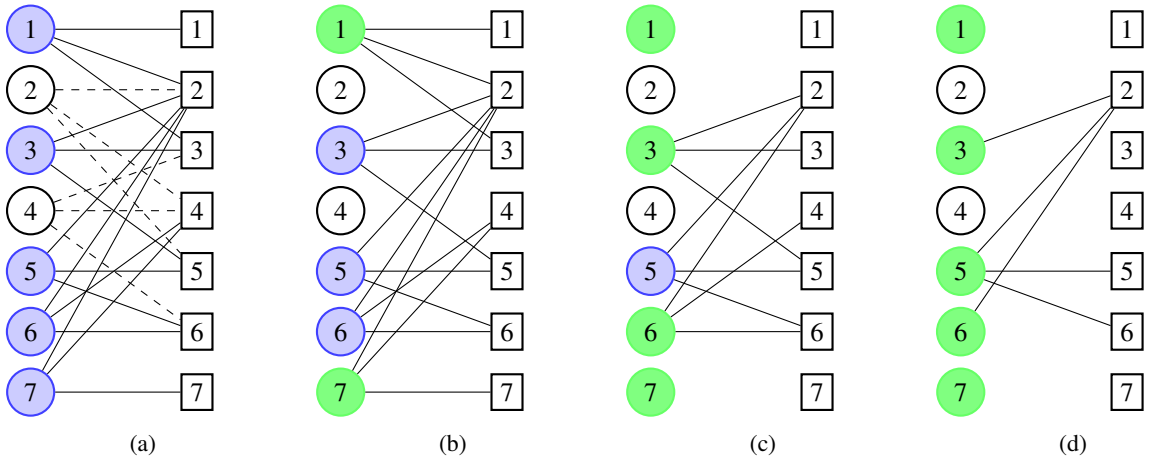

Fig. 2: Peeling with truncation. The decoder stops using Bin 2 in (c) and (d), since the number of balls peeled from this bin exceeds the maximum number.

corresponding to the location index of the singleton ball. Due to the sign ambiguity, there may be two possible locations and the true location is guaranteed to be one of them with high probability. We still need to find the correct location and estimate the value of the singleton ball. On the other hand, if the bin is not a singleton, the decoding algorithm of the concatenated code still returns at most two possible locations and we have to make sure that these bins are not considered as singletons. These problems are addressed by energy tests using the verification measurements, based on the same idea as in [12].

A. Signal Value Estimation

Consider the i-th bin in the $(t+1)$-th iteration, and let j be a possible location of the singleton ball that the decoding algorithm of the concatenated code suggests. We assume that the bin is indeed a singleton with the singleton ball located at j, and estimate s_j by the remaining verification measurements, i.e.,

$$\hat{s}_j = \frac{1}{R_2} \sum_{k=1}^{R_2} V_{k,j} y_{i,k}^{(t),v}. \tag{1}$$

Here, $y_{i,k}^{(t),v}$ is the k-th remaining verification measurement of the i-th bin after the t-th iteration. Intuitively, this estimation method is simply averaging over the measurements with corrected sign, meaning that we flip the sign if the corresponding entry in the verification matrix is -1. The theoretical guarantee of singleton detection and estimation is presented in Lemma 3.

Lemma 3. *There exists a constant $c > 0$ such that for an accuracy level $\epsilon > 0$, if $\beta \geq c\epsilon$, then, for a singleton in any constant iteration, we have:*

(i) *the location measurements can find the correct location of the singleton ball with probability $1 - \mathcal{O}(1/\mathrm{poly}(n))$.*

(ii) *by equation (1), the estimated value of singleton ball $\hat{s}_j$ satisfies $|\hat{s}_j - s_j| \leq C_j \epsilon$ for some constant $C_j > 0$ with probability $1 - \mathcal{O}(1/\mathrm{poly}(n))$ given the number of verification measurements in a bin is $R_2 = \Theta(\frac{\sigma^2}{\epsilon^2}\log(n))$.*

We should note that result (i) is a simple extension of the conclusion that we get in Section IV, since in Section IV, we focused on the first iteration. Result (ii) shows that for the preassigned accuracy level $\epsilon > 0$, if the number of verification measurements is $R_2 = \Theta(\frac{\sigma^2}{\epsilon^2}\log(n))$, then we can estimate the signal value within constant factor of ϵ with high probability.

B. Energy Test

So far, we have seen that if a bin is indeed a singleton, the location measurements can find the correct location of the singleton ball and the verification measurements can give accurate estimation of the value. However, there are still several things left. As we have mentioned, we need to clarify sign ambiguity, and rule out non-singleton bins. These operations can be done by energy tests.

Consider the i-th bin in the t-th iteration. Let $\mathcal{B}$ be the set of location indices of balls that have been found before this iteration. Before using the location measurements to find the location index of a singleton ball, we use an energy test to check if this bin is a zeroton, i.e., check if $\mathrm{supp}(\boldsymbol{x}_i) = \mathcal{B}$. If it is, there is no need to run the decoding algorithm of the concatenated code. Specifically, we construct $\hat{\boldsymbol{y}}_1 = \sum_{g\in\mathcal{B}} \hat{s}_g \boldsymbol{V}_g$ and conduct the zeroton energy test with threshold $\tau > 0$:

$$\text{if } \frac{1}{R_2}\|\boldsymbol{y}_i^v - \hat{\boldsymbol{y}}_1\|_2^2 < \tau, \text{ bin } i \text{ is a zeroton;}$$
$$\text{else bin } i \text{ is not a zeroton.}$$

If the bin is not a zeroton, we use the location measurements to find a possible singleton location j and get the estimated the value $\hat{s}_j$. We need to verify if there is indeed $\mathrm{supp}(\boldsymbol{x}_i) = \mathcal{B}\cup\{j\}$. Similar to the zeroton test, we construct $\hat{\boldsymbol{y}}_2 = \sum_{g\in\mathcal{B}\cup\{j\}} \hat{s}_g \boldsymbol{V}_g$ and conduct the singleton energy test

with threshold $\tau > 0$:

if $\frac{1}{R_2}\|\boldsymbol{y}_i^v - \hat{\boldsymbol{y}}_2\|_2^2 < \tau$,

bin i is a singleton with singleton ball at j;

else

bin i is not a singleton with singleton ball at j.

The intuition behind both energy tests is simple. We actually make a hypothesis that the true signal of a bin is $\hat{\boldsymbol{x}}$ and construct the corresponding measurements $\hat{\boldsymbol{y}} = \boldsymbol{V}\hat{\boldsymbol{x}}$. If the support of $\hat{\boldsymbol{x}}$ and $\boldsymbol{x}_i$ are the same and the values are accurately estimated, i.e., $\|\hat{\boldsymbol{x}} - \boldsymbol{x}_i\|_\infty < C_0\epsilon$, for some constant $C_0 > 0$. Then the energy of the difference between the actual measurements and the constructed measurements should be small; otherwise, the energy should be large. For zeroton energy test, the hypothesis signal consists of the balls that are found before the t-th iteration, and for the singleton energy test, the hypothesis signal consists of the balls which are found before and including the t-th iteration. The theoretical guarantees of both energy tests are provided in Lemma 4.

Lemma 4. *There exist constants $c_3 > 0$ and $c_4 > 0$ such that when $\beta > (c_3\sigma + c_4\epsilon)^2$, there exists proper threshold $\tau > 0$ such that a particular energy test succeeds with probability $1 - \mathcal{O}(1/\text{poly}(n))$, when $R_2 = \Theta(\max\{\sigma^2/\epsilon^2, 1\}\log(n))$.*

VII. Proof of Theorem 1

In this section, we give the final proof of our main result. First, we analyze the error probability. There are three possible error events,

(i) E_1: the peeling algorithm does not find an arbitrarily large fraction of non-zero elements.

(ii) E_2: error in decoding algorithm of concatenated code (location decoding).

(iii) E_3: error in value estimation or energy test.

We have shown that

$$\mathbb{P}\{E_1|E_2^{\complement}, E_3^{\complement}\} = \mathcal{O}(\exp\{-c_1(\delta)K^{-c_2(\delta)}\}).$$

Similar to the analysis in [12], since we need to conduct $\Theta(K)$ times of location decoding and energy tests, using union bound, $\mathbb{P}\{E_2\} = \mathcal{O}(1/\text{poly}(n))$ and $\mathbb{P}\{E_3\} = \mathcal{O}(1/\text{poly}(n))$. Then by union bound and law of total probability, we get the error probability $\mathcal{O}(\exp\{-c_1(\delta)K^{-c_2(\delta)}\} + 1/\text{poly}(n))$. Then computational complexity can be analyzed by the same method as in [12]. Combining Lemmas 1, 2, 3, and 4, our main result is proved.

VIII. Experimental Results

In this section, we provide the results of our numerical experiments to justify our theoretical results. We conduct two experiments to test the sample and time complexities.

In the both experiments, we set the left degree of the bipartite graph to be $d = 10$, and the number of bins to be $M = 10K$. The maximum number of ball that can be peeled from a bin is set to be $D = 5$. The nonzero elements of the signal are generated from a uniform distribution in $[-10, -3] \cup [3, 10]$, and the locations of the nonzero elements are uniformly chosen from the n coordinates. The additive noise are all i.i.d. Gaussian distributed with zero mean. The inner code that we use for the singleton detection is a $(3, 6)$ regular LDPC code with rate 0.5.

In the first experiment, we choose $n = 4096$ and test the sample complexity of our algorithm. Specifically, we test how the empirical probability of successful recovery changes when we increase the number of verification measurements in each bin. We define the event of successful recovery as the cases when $\|\hat{\boldsymbol{s}} - \boldsymbol{s}\|_\infty \leq 0.1$, where $\hat{\boldsymbol{s}}$ and $\boldsymbol{s}$ denote the recovered signal and the original signal, respectively. The empirical success probability is computed by averaging 100 trials and the phase transition behavior under different noise power is shown in Figure 3.

In the second experiment, we fix the noise to be $\mathcal{N}(0, 0.1)$ distributed, and the number of verification measurements in each bin to be $2\log_2(n)$. We test the average running time with different (n, K) pairs over 100 trials. As we can see, the time cost of our algorithm is linear in K and do not have significant dependence on n, and this behavior justifies our theory.

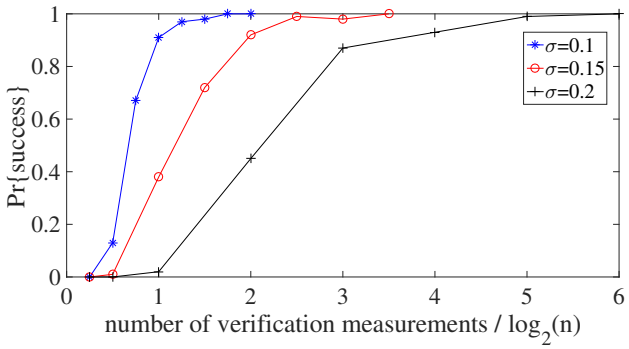

Fig. 3: Sample complexity.

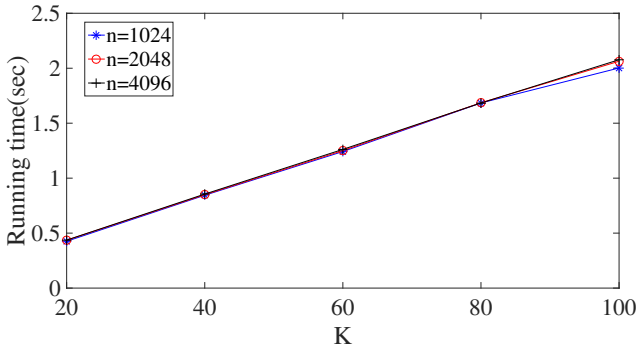

Fig. 4: Time complexity.

IX. Conclusions

In this paper, we propose an algorithm for noisy compressive sensing based on sparse graph codes. The algorithm can recover an arbitrarily large fraction of the support of the

unknown signal with ℓ_∞ norm recovery guarantee and near optimal sample and time complexities. The repeated version of the algorithm can guarantee full support recovery with high probability with an additional $\log(K)$ factor in the sample and time complexities. Future work may include generalizing our framework to other fast signal processing algorithms based on sparse graph codes.

REFERENCES

[1] M. Lustig, D. Donoho, and J. M. Pauly, "Sparse mri: The application of compressed sensing for rapid mr imaging," *Magnetic resonance in medicine*, vol. 58, no. 6, pp. 1182–1195, 2007.

[2] J. Romberg, "Imaging via compressive sampling [introduction to compressive sampling and recovery via convex programming]," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 14–20, 2008.

[3] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *Information Theory, IEEE Transactions on*, vol. 52, no. 2, pp. 489–509, 2006.

[4] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis*, vol. 27, no. 3, pp. 265–274, 2009.

[5] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *Information Theory, IEEE Transactions on*, vol. 53, no. 12, pp. 4655–4666, 2007.

[6] S. Sarvotham, D. Baron, and R. G. Baraniuk, "Sudocodes-fast measurement and reconstruction of sparse signals," in *Information Theory, 2006 IEEE International Symposium on*. IEEE, 2006, pp. 2804–2808.

[7] H. V. Pham, W. Dai, and O. Milenkovic, "Sublinear compressive sensing reconstruction via belief propagation decoding," in *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*. IEEE, 2009, pp. 674–678.

[8] P. Indyk and M. Ružić, "Near-optimal sparse recovery in the l1 norm," in *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*. IEEE, 2008, pp. 199–207.

[9] X. Li, S. Pawar, and K. Ramchandran, "Sub-linear time support recovery for compressed sensing using sparse-graph codes," *arXiv preprint arXiv:1412.7646*, 2014.

[10] X. Chen and D. Guo, "A generalized ldpc framework for robust and sublinear compressive sensing," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4623–4627.

[11] S. Pawar and K. Ramchandran, "A robust r-ffast framework for computing a k-sparse n-length dft in o (k log n) sample complexity using sparse-graph codes," in *2014 IEEE International Symposium on Information Theory*. IEEE, 2014, pp. 1852–1856.

[12] D. Yin, K. Lee, R. Pedarsani, and K. Ramchandran, "Fast and robust compressive phase retrieval with sparse-graph codes," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2015.

[13] R. Pedarsani, K. Lee, and K. Ramchandran, "Sparse covariance estimation based on sparse-graph codes," in *Annual Allerton Conference on Communication, Control, and Computing*, 2015.

[14] J. Justesen, "Class of constructive asymptotically good algebraic codes," *IEEE Transactions on Information Theory*, vol. 18, no. 5, pp. 652–656, 1972.

[15] D. A. Spielman, "Linear-time encodable and decodable error-correcting codes," in *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*. ACM, 1995, pp. 388–397.

[16] J. L. Massey, "Threshold decoding," DTIC Document, Tech. Rep., 1963.

[17] M. Cheraghchi, "Capacity achieving codes from randomness conductors," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2009, pp. 2639–2643.

[18] R. Pedarsani, K. Lee, and K. Ramchandran, "Phasecode: Fast and efficient compressive phase retrieval based on sparse-graph-codes," *arXiv preprint arXiv:1408.0034*, 2014.

[19] ——, "Capacity-approaching phasecode for low-complexity compressive phase retrieval," *arXiv preprint arXiv:1412.5694*, 2014.

APPENDIX

A. Pseudocode of Peeling Decoder with Truncation Strategy

Algorithm 1 Peeling decoder with truncation strategy

Input: $\boldsymbol{y}$, $\boldsymbol{T}$, $\boldsymbol{H}$, D
Output: Estimated signal $\hat{\boldsymbol{s}}$

```
ŝ ← 0,
number of peeled balls in each bin: B_i ← 0, i ∈ [M],
Indicator of utilizability of bins: U_i ← true, i ∈ [M],
y_i^(0) ← y_i, i ∈ [M], stop ← false, t ← 1
while stop = false do
  Find singleton balls.
  I_t ← {indices of all singletons found in the iteration t}.
  U_i ← false, for all i ∈ I_t.
  J_t ← {locations of singleton balls found in iteration t}.
  y_i^(t) ← y_i^(t−1), i ∈ [M].
  if J_t ≠ ∅ then
    for all j ∈ J_t do
      Estimate ŝ_j.
      for all i ∈ [M] such that U_i = true and h_{i,j} = 1
      do
        y_i^(t) ← y_i^(t) − ŝ_j T_j.
        B_i ← B_i + 1.
        if B_i = D then
          U_i ← false
        end if
      end for
    end for
  else
    stop ← true
  end if
  t ← t + 1
end while
return ŝ
```

B. Proof of Lemma 1

In this section, we give a brief proof of Lemma 1. The construction of the concatenated code in Lemma 1 is based on Justesen's concatenation scheme [14] and similar method is also analyzed in [17]. The concatenated code consists of an outer code f_{out} and an ensemble of inner codes $\mathcal{I}$. For the outer codes, we use an expander-based code proposed in [15]. The outer code maps the message to a codeword with length p on an alphabet with size 2^k, i.e., $f_{\text{out}} : [n] \to [2^k]^p$. Recall that by definition, the rate of the outer code is $R_{\text{out}} = \lceil \log(n) \rceil / p$. We make essential use of the Theorem in [15].

Theorem 3. *For every integer $k > 0$ and every absolute constant $R' < 1$, there is an explicit family of expander-based linear codes with alphabet $[2^k]$ and rate $R_{out} = R'$ that is error-correcting for a $\Theta(1)$ fraction of errors. The running time of the encoder and the decoder is linear in the block length of the codewords.*

Note that here, the $\Theta(1)$ fraction of error can be adversarially chosen, and that the decoding algorithm of the outer code does not rely on the knowledge of the channel. Now let $(c_1, c_2, \ldots, c_p) \in [2^k]^p$ be the codeword that we obtained from the outer code, and we call it the outer codeword. As we have mentioned, we use an ensemble of inner codes $\mathcal{I}$, which means that $\mathcal{I} = \{g_1, \ldots, g_p\}$ is a collection of p codes which encode the symbols in the outer codeword as a new q-bit codeword with alphabet $\{1, -1\}$. Specifically, each code g_i in $\mathcal{I}$ is a map $g_i : [2^k] \to \{1, -1\}^q$, and we encode the i-th symbol in the outer codeword by the i-th code in $\mathcal{I}$. This gives us the final codeword $(g_1(c_1), g_2(c_2), \ldots, g_p(c_p)) \in \{1, -1\}^{qp}$, which also implies that the block length of the concatenated code is $R_1 = qp$.

Then we show the details of the inner code ensemble. We choose the inner code ensemble to be the Wozencraft's ensemble [16]. The Wozencraft's ensemble satisfies the property that all but a $o(1)$ fraction of the codes in the ensemble are capacity achieving, where the asymptotic is with respect to the block length q. Specifically, for the capacity achieving codes in the ensemble, the probability of decoding error is exponentially small in the block length q, i.e., $e^{-\alpha q}$ for some constant $\alpha > 0$, as long as the rate of the codes $R_{\text{in}} = k/q$ is below the capacity of the BCS. Here, we should notice that we do need an upper bound of the bit flip probability in the design of the inner code since we need to get a lower bound of the capacity of the BSC, however, we do not need the exact value of the bit flip probability. Then, it is shown in [17] that using brute force maximum likelihood decoder for the inner code and the decoding algorithm of the expander-based outer code, the error probability is exponentially small in the block length of the concatenated code, i.e., $e^{-\alpha' R_1}$ for some constant $\alpha' > 0$.

Now we analyze the block length and decoding complexity of the concatenated code. The number of codes in the Wozencraft's ensemble is 2^q, meaning that $p = 2^q$. Since rate of the outer code is a constant $R_{\text{out}} = \lceil \log(n) \rceil / p$ which can be arbitrarily close to 1, we know that $p = \Theta(\log(n))$. Then $q = \Theta(\log \log(n))$ and the block length of the concatenated code is $R_1 = qp = \Theta(\log(n) \log \log(n))$. Consequently the error probability is $e^{-\alpha' R_1} = \mathcal{O}(1/\text{poly}(n))$, where $\text{poly}(n)$ is a polynomial of n which can have arbitrarily large degree. Consider the decoding complexity. For the inner code, the complexity of testing each possible message is $\Theta(q)$ and there are $2^k = 2^{qR_{\text{in}}}$ messages. Therefore, for each inner code, the computational complexity of the brute force maximum likelihood decoding is $\Theta(2^{qR_{\text{in}}} q)$. Since there are p inner codes, the complexity of decoding all the inner codes is $\Theta(2^{qR_{\text{in}}} qp) = \Theta(p^{1+R_{\text{in}}} q) = \Theta(\log^{1+R_{\text{in}}}(n) \log \log(n))$. Since we do not require the inner code to be capacity achieving, R_{in} can be arbitrarily close to 0, we can conclude that complexity of decoding all the inner codes is $\Theta(\log^{1+r}(n))$, where $r > 0$ can be arbitrarily small. Since the complexity of decoding the outer code is linear in its block length, which is $\Theta(p) = \Theta(\log(n))$, we know that the decoding complexity of the concatenated code is $\Theta(\log^{1+r}(n))$.

C. Proof of Lemma 2

The proof of Lemma 2 is based on density evolution, which is a common and powerful tool in modern coding theory. Similar ideas have been applied in [9], [18], [19]. The basic idea is to get a recursive equation to analyze the fraction of singleton balls that are not recovered in a particular iteration. We will only give brief proof here and focus on the truncation peeling strategy, which is main difference between our algorithm and previous works. We refer the readers to [9], [18] for more details.

From now on, we do not consider the connection between the zero signal elements and the bins, meaning that we only focus on the d-left regular random bipartite graph with K balls on the left and M bins on the right. We let $M = \eta K$ for some constant $\eta > 0$. Using the same Poisson approximation as in [18], we get the expected fraction of edges which are connected to right nodes with degree i is

$$\rho_i \approx \frac{(d/\eta)^{i-1} e^{-d/\eta}}{(i-1)!}.$$

Now we consider the peeling process as a message passing process on the bipartite graph. According to our peeling decoding algorithm, a singleton can send a "peeling" message to the ball that is connected to it, and a peeled singleton ball sends "peeled" message to all the bins that are connected to it. In a particular iteration, a bin sends a "peeling" message to a ball through an edge if other edges connected to this bin all send "peeled" messages in the previous iteration and a ball sends a "peeled" message to a bin through an edge if if at least one of the bins that is connected to it sends a "peeling" message to it. We should also notice that the bins with degree greater than D never send "peeling" message to the balls due to the truncation strategy.

As in [9], [18], we still need to first assume that the neighborhood of each edge with a constant depth is a tree (tree-like assumption). Let p_j be the probability that in the jth iteration, a randomly chosen edge is *not* peeled, i.e., sending a "not peeled" message. Then, under the tree-like assumption, we have the density evolution equation:

$$p_{j+1} = F(p_j) = \left(1 - \sum_{i=1}^{D} \rho_i (1-p_j)^{i-1}\right)^{d-1}.$$

Similar to the analysis in [18], we need to consider the fix point of $F(t)$, i.e., the point such that $F(t) = t$, and show that the fix point can be arbitrarily small by choosing proper parameters. We have

$$\begin{aligned} F(t) &= \left(1 - \sum_{i=1}^{D} \frac{(d/\eta(1-t))^{i-1} e^{-d/\eta}}{(i-1)!}\right)^{d-1} \\ &= \left(1 - \sum_{i=0}^{D-1} \frac{(d/\eta(1-t))^{i} e^{-d/\eta}}{i!}\right)^{d-1} \\ &= \left(1 - e^{-d/\eta}(e^{d(1-t)/\eta} - \frac{e^{\xi}(d(1-t)/\eta)^{D}}{D!})\right)^{d-1}, \end{aligned}$$

where $0 < \xi < d(1-t)/\eta$. We can choose D to be large enough such that $\frac{(d(1-t)/\eta)^D}{D!} < \frac{1}{2}$. Then we have

$$F(t) < \left(1 - \frac{1}{2} e^{-dt/\eta}\right)^{d-1} := G(t).$$

Then we know that the fix point of $F(t)$ should be upper bounded by that of $G(t)$. Further, if we keep d/η to be a constant and enlarge d, the fix point of $G(t)$ can be arbitrarily small, and consequently, the fix point of $F(t)$ can be arbitrarily small. Specifically, let $p^\star \in (0,1)$ be the fix point of $F(t)$, then for any $\delta > 0$, there exist parameters d and η such that $p^\star < \delta$. Here, we briefly analyze the relationship between η and the fix point of $G(t)$, denoted by $t^\star$. Since $t^\star = G(t^\star) = (1 - \frac{1}{2} e^{-dt^\star/\eta})^{d-1}$, and $t^\star$ is close to 0, we have $e^{-dt^\star/\eta} \approx 1$ and thus $t^\star \approx (\frac{1}{2})^{d-1}$. Therefore, $d = \Theta(\log(1/t^\star))$, and further, since we keep d/η as a constant, $\eta = \Theta(\log(1/t^\star))$. Since the fix point of $F(t)$, $p^\star$ is upper bounded by $t^\star$, we have $\eta = \Theta(\log(1/p^\star))$. We can choose parameters such that $p^\star = \Theta(\delta)$ and then, $\eta = \Theta(\log(1/\delta))$. Using the same argument as in [18], we can show that for any $\delta > 0$, there exist a constant $N(\delta)$ and proper parameters d and η such that $p_N < \delta$.

By the same martingale argument as in [18] and taking the event that the tree-like assumption does not hold into consideration, we can show that the fraction of balls which are not peeled is highly concentrated around p_N. Let Z be the fraction of balls which are not peeled after N-th iteration, when K is large enough, we have for any $\delta > 0$,

$$\mathbb{P}\{|Z - p_N| > \delta\} < 2\exp\{-C\delta^2 K^{1/(4N+1)}\},$$

where $C > 0$ is a universal constant. The proof of Lemma 2 is completed by choosing N such that $p_N < \delta$.