Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 6340430, 10 pages
http://dx.doi.org/10.1155/2016/6340430

Research Article

Peeling Decoding of LDPC Codes with Applications in Compressed Sensing

Weijun Zeng and Huali Wang

Department of Electrical and Computer Engineering, PLA University of Science and Technology, Nanjing 210007, China

Correspondence should be addressed to Weijun Zeng; zwj3103@126.com

Received 8 October 2015; Revised 10 January 2016; Accepted 24 January 2016

Academic Editor: Erik Cuevas

We present a new approach for the analysis of iterative peeling decoding recovery algorithms in the context of Low-Density Parity-Check (LDPC) codes and compressed sensing. The iterative recovery algorithm is particularly interesting for its low measurement cost and low computational complexity. The asymptotic analysis can track the evolution of the fraction of unrecovered signal elements in each iteration, which is similar to the well-known density evolution analysis in the context of LDPC decoding algorithm. Our analysis shows that there exists a threshold on the density factor; if under this threshold, the recovery algorithm is successful; otherwise it will fail. Simulation results are also provided for verifying the agreement between the proposed asymptotic analysis and recovery algorithm. Compared with existing works of peeling decoding algorithm, focusing on the failure probability of the recovery algorithm, our proposed approach gives accurate evolution of performance with different parameters of measurement matrices and is easy to implement. We also show that the peeling decoding algorithm performs better than other schemes based on LDPC codes.

1. Introduction

Compressed sensing (CS) is a novel signal sampling theory that exploits the sparsity of signal [1, 2]. In the noiseless settings, the CS problem can be considered as estimating an original sparse signal $\mathbf{x} \in \mathbb{R}^n$ from the linear measurement vector $\mathbf{y} \in \mathbb{R}^m$, $m \ll n$; that is, $\mathbf{y} = H\mathbf{x}$, where H is referred to as measurement matrix; this measurement process is also referred to as *encoding*. Generally, without additional information, it is impossible to recover $\mathbf{x}$ from $\mathbf{y}$ in the case $m \ll n$. The signal $\mathbf{x} \in \mathbb{R}^n$ is called k-sparse if the signal has no more than k nonzero entries, for k-sparse signal and with an appropriate measurement matrix H, even though when $m \ll n$, $\mathbf{y}$ essentially contains enough information to recover the original signal $\mathbf{x}$; this recovery process is also referred to as *decoding*.

Candés et al. in [1, 2] used the Gaussian random matrix as the measurement matrix and the ℓ_1 norm minimization to recover the original signal. However, most of elements in Gaussian random matrix are nonzero, which requires a lot of storage space. Furthermore, these random matrices are often difficult or expensive in hardware implementation. Recently, many researchers have exploited some excellent sparse matrices of channel coding into the CS system [3–5]; compared with random measurement matrices, the sparse matrices could reduce the storage space and are easy to implement in hardware. As a special class of sparse matrices, the parity-check matrices of Low-Density Parity-Check (LDPC) codes can be used as good measurement matrices for CS under ℓ_1 norm minimization [3]. In [3], the authors have pointed out that the ℓ_1 norm minimization decoding of LDPC codes is very similar to the ℓ_1 norm minimization of CS. Inspired by the work of [3], the authors of [4, 5] constructed a class of deterministic measurement matrices from finite geometry LDPC (FG-LDPC) codes. However, these works only focus on the construction of measurement matrices.

From the class of measurement matrix, decoding algorithms can be assorted as algorithms with dense matrices and sparse matrices. Generally, decoding algorithms based on sparse matrices have lower complexity than the algorithms associated with dense matrices. As shown in our analysis in Section 3, the decoding algorithm with sparse matrices

is faster than the algorithms using dense matrices. Since the random dense matrices satisfy the well-known restricted isometry property (RIP) with high probability, the standard decoding algorithms for these matrices are in line with greedy or convex programming [1, 2, 6, 7]. The RIP is a sufficient condition which guarantees unique and perfect recovery of sparse signals via ℓ_1-minimization [2]. It has been shown that the sparse matrices do not satisfy RIP when $m < \Omega(k^2)$ [8]. However, the decoding algorithms with sparse matrices explore the insights from coding theory to assist in sparse recovery. In fact, through the design of sparse measurement matrices and decoding algorithm, both the measurement cost and decoding complexity can be simultaneously reduced.

For the complexity of recovery algorithm, the ℓ_1 norm minimization has a computation complexity of $\mathcal{O}(n^3)$. To reduce this complexity, with the sparse measurement matrices, various low computational complexity message-passing algorithms have been introduced for reconstruction of sparse signals in CS [9–22]. In [9], the authors used the random sparse matrix as a new sparse measurement matrix and proposed an iterative algorithm of complexity of $\mathcal{O}(k\log(k)\log(n))$, while it only required $m = \mathcal{O}(k\log(n))$ measurements. It was essentially the same as the ideal of verification decoding of packet-based LDPC codes in [23] and is rediscovered by Zhang in [11]. These verification-based (VB) recovery algorithms in the context of CS have been analyzed rigorously via the density evolution in [10, 11]. In [12], the VB algorithm was applied to the wideband spectrum sensing, where the measurement matrix is designed based on a block sparse matrix. Based on special nonnegative sparse signals, a new message-passing algorithm, called the Interval-Passing (IP) algorithm, was proposed in [13], which has a better performance than the VB algorithm in [11]. In [14], a combination scheme of both the IP algorithm and the VB algorithm was proposed for nonnegative sparse signals, which can perform better than either the IP algorithm or the VB algorithm. Based on expander graphs of the sparse measurement matrix, Jafarpour et al. in [15, 16] proposed a different iterative algorithm with a complexity of $\mathcal{O}(n\log(n/k))$ using $\mathcal{O}(k\log(n/k))$ measurements in the noiseless case. It should be worth noting that the above fast recovery algorithms only can get rid of the noiseless case.

For the noise case, in [17], the authors assumed signal as Gaussian mixture priors and proposed a belief propagation (BP) algorithm. The main disadvantage with this approach is that decoding complexity grows exponentially. In a similar work [18], the authors assumed signal as Jeffreys' priors and applied well-known least-squares algorithms to recover the signal. However, in [18], the sparsity k was assumed to be known.

To overcome the restriction of the decoding algorithms discussed above, Pawar et al. in [19–22] designed a hybrid mix of the LDPC codes and Discrete Fourier Transform (DFT) framework (LDPC-DFT) and proposed a fast Short-and-Wide Iterative Fast Transform (SWIFT) peeling decoding recovery algorithm. It only needs $\mathcal{O}(k)$ measurements and $\mathcal{O}(k)$ iterative step to achieve the exact signal recovery in the noiseless case. In the presence of noise, both measurements and computational complexity for exact signal recovery are $\mathcal{O}(k\log^{1.3}(n))$. These frameworks also have been used in spectrum sensing in [24]; Hassanieh et al. in [24] presented a prime sampling technology that can capture GHz of spectrum in real-time with low speed analog-to-digital converters (ADCs); this prime sampling technology is essentially identical to the framework of [19–22, 25]. These frameworks of [19–22] also have been used in compressed sensing phase retrieval [26].

In this study, we are interested in the sparse LDPC-DFT measurement matrices associated with SWIFT recovery algorithm [19–22], which can reduce both the measurement cost and computational complexity simultaneously. The authors in [19] analyzed the successful recovery probability of SWIFT algorithm via exploiting the connection between CS system and packet-communication system. Furthermore, in [22] the authors investigated the generalized family of LDPC-DFT measurement matrices in terms of the measurement cost, computational complexity, and recovery performance via a rough density evolution, where a local neighborhood of every edge in the graph should be cycle-free (tree-like). However, they only focused on reducing the number of measurements m required for successful recovery, for given k and n.

In this paper, we investigate the performance measures of SWIFT recovery algorithm in the asymptotic case ($n \to \infty$) in order to give an estimate of the performance for given sparse measurement matrix and k in the finite n case, where our proposed approach gives accurate evolution of performance with different parameters of measurement matrices and is easy to implement. The analysis shows that there exists a threshold on the density factor $\alpha = k/n$; if under this threshold, the recovery algorithm is successful; otherwise it will fail. It is shown that the threshold is dependent on the parameter of LDPC matrix, which provides a basis for us to design the sparse measurement matrices.

2. A Class of Hybrid LDPC-DFT Measurement Matrix and Their Fast Recovery Algorithm

In this section, we consider a special class of hybrid mix of the parity-check matrix of binary LDPC codes and DFT matrix (referred to as "LDPC-DFT measurement matrix") and their fast recovery algorithm. We firstly introduce the sparse parity-check matrix of LDPC codes (referred to as "LDPC-based measurement matrix") and its bipartite graph.

2.1. LDPC-Based Measurement Matrix and Measurement Graph. In channel coding, it is well known that the LDPC codes can be defined by the null space of its sparse parity-check matrix; the codes are classified as binary codes and nonbinary codes. In this paper, we only consider the case of binary codes. A sparse parity-check matrix can be represented by a bipartite graph. In a bipartite graph, two sets of nodes are defined as the code symbols and parity-check equations; they are labeled as "variable nodes" and "check nodes," respectively. There is an edge between the variable node v and check node c if and only if v appears in parity-check equation c.

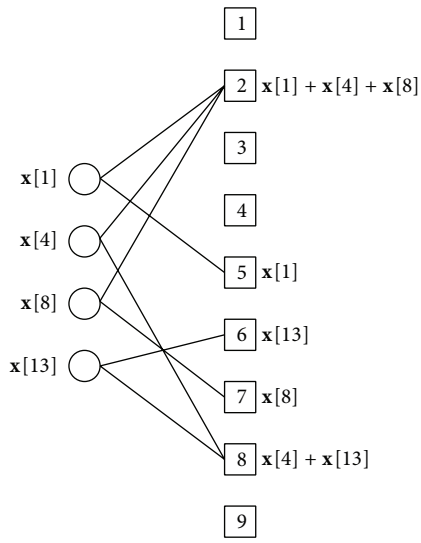

Figure 1: Sparse measurement bipartite graph.

In CS, when the measurement matrix is sparse parity-check matrix, the measurement also has its own bipartite graph (referred to as "measurement graph"). We denote this measurement graph as $G = (\mathscr{V} \cup \mathscr{C}, \partial)$, where the set of variable nodes $\mathscr{V}$ represents the original signal and the check nodes $\mathscr{C}$ represent measurement observations and ∂ is set of edges. The number of edges associated with every node ($\mathscr{V}$ or $\mathscr{C}$) is called node degree. In this paper, we consider the case in which every variable node (check node) has same nodes degree d_v (d_c), which is also referred to as regular degree (d_v, d_c). Additionally, we denote the check nodes set $\mathscr{M}(v)$ as all the check nodes incident to variable node v by edges. Similarly, the set of variable nodes $\mathscr{N}(c)$ is denoted as all the variable nodes incident to check node c. In particular, due to the mathematical connection between channel coding and CS, we can analyze the fast recovery algorithm over the measurement graph. Hereinafter, we refer to the recovery process as decoding.

2.2. Decoding Algorithm Based on Sparse Measurement Graph. We firstly introduce the principle of the decoding algorithm based on sparse measurement graph through an example. Consider an example consisting of a sparse signal $\mathbf{x}$ of length $n = 15$ with $k = 4$ nonzero elements. Let the 4 nonzero elements of signal be as follows: $\mathbf{x}[1] = 2$, $\mathbf{x}[4] = 1$, $\mathbf{x}[8] = 4$, and $\mathbf{x}[13] = 7$; let the number of measurements be $m = 9$. According to the definition of measurement graph, we have sparse measurement graph, shown in Figure 1.

In Figure 1, it is shown that the check nodes can be classified into three observation nodes.

Zero-Ton. A check node does not contain any nonzero elements of signal, for example, nodes 1, 3, 4, and 9 in the right side of Figure 1.

Single-Ton. A check node only contains one nonzero element of signal, for example, nodes 5, 6, and 7 in the right side of Figure 1.

Multiton. A check node contains more than one nonzero element of signal, for example, nodes 2 and 8 in the right side of Figure 1.

Assuming that there exists an "oracle" can decide exactly which check nodes are zero-ton, single-ton, and multiton, and the "oracle" can further decode the corresponding value and index of variable nodes incident to the single-ton. In the above example, the "oracle" can decide that the nodes 5, 6, and 7 are single-ton and decode the value and index of variable nodes 1, 13, and 8, respectively. Then, the "oracle" could subtract variable nodes contributions from other check nodes and form new single-tons; for example, in Figure 1 check node 8 subtracts the contribution of the variable node $\mathbf{x}[13]$; then check node 8 becomes a new single-ton.

Therefore, with the information of single-ton, the "oracle" repeats the following steps:

(1) Selecting all the edges in sparse measurement graph with single-ton check nodes.

(2) Removing these edges and the corresponding variable and check nodes connected to these edges.

(3) Removing the other edges adjacent variable nodes that have been removed in step (2).

(4) Subtracting the value of variable nodes that have been removed in step (3) from the corresponding check nodes.

When all the edges have been removed, the decoding is successful. The decoding algorithm is called "peeling decoding" in traditional erasure channel [23, 27].

Luby et al. in [23, 27] analyzed the performance of peeling decoding algorithm; it has been shown that the peeling decoding algorithm terminates successfully with probability of at least $1 - \mathcal{O}(k^{-3/4})$ with good variable degree distribution. However, in the decoding algorithm, the "oracle" can (1) decide the type of check nodes and (2) exactly estimate the value and index of single-ton incident to variable node. In order to get rid of the oracle, Pawar and Ramchandran introduced the DFT matrix to decode the single-ton in [19]; they also proved that the corresponding SWIFT algorithm with $m = 2k(1 + \varepsilon)$ measurements could exactly recover the original signal with probability of at least $1 - \mathcal{O}(k^{-3/4})$ in the noiseless setting, where $\varepsilon > 0$, arbitrarily close to 0.

2.3. Measurement Matrix Based on LDPC-DFT. Denote $F \in \mathbb{C}^{2\times n}$ as the first two rows of $n \times n$ DFT matrix; that is,

$$F = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{-j2\pi/n} & e^{-j4\pi/n} & \cdots & e^{-j2(n-1)\pi/n} \end{bmatrix}. \quad (1)$$

Since the matrix F can decide the type of check nodes and decode the value as well as index of corresponding variable nodes, the matrix F is also referred to as "detection matrix." We design the required measurement matrix via the following definition.

Definition 1. Let $m = 2R$ for some positive integers R. Given $R \times n$ sparse parity-check matrix P of LDPC, denoting p_j^T as

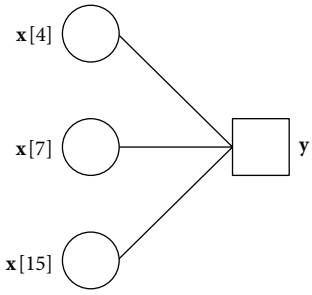

Figure 2: Multiton check node and its observation vector $\mathbf{y}_i$.

jth row of P ($1 \le j \le R$), and given a detection matrix $F \in \mathbb{C}^{2\times n}$, then the $m \times n$ measurement matrix H is designed by

$$H = P \odot F = \begin{bmatrix} p_1^T \otimes F \\ \vdots \\ p_R^T \otimes F \end{bmatrix}, \tag{2}$$

where $\odot$ is row-tensor product and $\otimes$ is standard Kronecker product.

From the definition of Kronecker product, it is shown that the value of every check node (observation) is a 2-length vector, for example, the following check nodes i, also shown in Figure 2:

$$\begin{aligned} \mathbf{y}_i = {} & \mathbf{x}[4] \begin{pmatrix} 1 \\ e^{-j8\pi/n} \end{pmatrix} + \mathbf{x}[7] \begin{pmatrix} 1 \\ e^{-j14\pi/n} \end{pmatrix} \\ & + \mathbf{x}[15] \begin{pmatrix} 1 \\ e^{-j30\pi/n} \end{pmatrix}. \end{aligned} \tag{3}$$

We use the following criterions to decide the type of check nodes and estimate the value and index of corresponding variable nodes.

Zero-Ton. ith check node is a zero-ton if $\mathbf{y}_i = 0$.

Single-Ton. ith check node is a single-ton if $|\mathbf{y}_i[1]| = |\mathbf{y}_i[2]|$ and $\angle(\mathbf{y}_i[1]\mathbf{y}_i[2]^\dagger) = 2\pi p/n$, $0 \le p \le n-1$; its neighbor indexed by $(n/2\pi)\angle(\mathbf{y}_i[1]\mathbf{y}_i[2]^\dagger)$ is nonzero with value given by $\mathbf{y}_i[1]$.

Multiton. ith check node is a multiton if $|\mathbf{y}_i[1]| \ne |\mathbf{y}_i[2]|$ or $\angle(\mathbf{y}_i[1]\mathbf{y}_i[2]^\dagger) \ne 2\pi p/n$.

The VB algorithm with decoding successfully claims that the nonzero elements of signal must satisfy continuous distribution [10, 11]. However, the above criterions are identified with probability of one whether nonzero elements of signal obey continuous distribution or not. Hence, the measurement matrix based on LDPC-DFT could be more applicable for various signal types.

Theorem 2. *Whether the nonzero elements of original signal obey a continuous distribution f or take values from the finite discrete set, the above criterions of zero-ton, single-ton, and multiton are identified with probability of one.*

Proof. Firstly, we prove the case of nonzero elements obeying the continuous distribution f. Here, we only prove the type of zero-ton, which is valid for the other types with no major changes. If the nonzero elements of original signal obey the continuous distribution, for arbitrary $a \ne b$, we have $\Pr(\mathbf{x}[a] = -\mathbf{x}[b]) = 0$. For the sparse measurement graph, $\mathbf{y}_i = \sum_{j\in\mathcal{N}(i)} \mathbf{x}[j]$, when $\mathbf{y}_i = 0$; that is, $\sum_{j\in\mathcal{N}(i)} \mathbf{x}[j] = 0$; by $\Pr(\mathbf{x}[a] = -\mathbf{x}[b]) = 0$, we have $\Pr(\mathbf{x}[j] = 0) = 1$, where $j \in \mathcal{N}(i)$; the decoder can decide the type of check node as zero-ton via the definition of zero-ton.

Then, we consider the case of the nonzero elements of original signal taking values from the finite discrete set. Similarly, we only prove the type of zero-ton. Considering the worst case, for $a \ne b$, $\Pr(\mathbf{x}[a] = -\mathbf{x}[b]) \to 1$. However the $\mathbf{y}_i = \sum_{j\in\mathcal{N}(i)} \mathbf{x}[j]$ is 2-length complex vector; that is, when $\mathbf{y}_i = 0$, it claims not only $\mathbf{y}_i[1] = 0$, but also $\mathbf{y}_i[2] = 0$. Due to the sparse measurement matrix, $\mathbf{y}_i = 0$ holds, if and only if $\mathbf{x}[j] = 0$, $j \in \mathcal{N}(i)$. Hence, the decoder can decide the type of check node as zero-ton via the definition of zero-ton. □

In the context of analysis of framework and simulation, we adopt the following way to generate the original signal. Let $\mathcal{T}$ be the set of nonzero elements of original signal and let α be the density factor, that is, the probability that a signal element belongs to $\mathcal{T}$. For a given α, every element of original signal takes value as follows: every element takes 0 with probability of $1-\alpha$ or the value from the continuous distribution f (discrete sets) with probability of α. In this sense, the sparsity k and density ratio $\gamma = k/n$ are random variables. Furthermore, $E[k] = \alpha n$ and $E[\gamma] = \alpha$, where $E[\cdot]$ is referred to as expected value.

2.4. Decoding Algorithm. In the above subsections, we describe an iterative decoding algorithm to decode the sparse signal with the help of "oracle" that is implemented via LDPC-DFT measurement matrix and single-ton criterions. In this subsection, the iterative decoding algorithm is showed by using the step described in Section 2.2 (also see pseudocode in Algorithms 1 and 2).

Following the discussions in [19], it has been proven that the SWIFT decoding algorithm satisfies the criterions with probability of at least $1 - \mathcal{O}(k^{-3/4})$ via the corresponding Theorem 2 in [27]. In this paper, we propose a general framework for the asymptotic analysis of SWIFT algorithm. Using the asymptotic analysis, we can track the evolution of density factor α and the corresponding threshold can be determined for different (d_v, d_c).

3. Asymptotic Analysis of Decoding Algorithm

3.1. Analysis Model. In this paper, we only consider the regular (d_v, d_c) measurement bipartite graph. However, the results can be generalized to irregular case.

Consider a sparse bipartite graph $G_k^m = (\mathcal{V}^* \cup \mathcal{C}, \partial^*)$, where $\mathcal{V}^* \subset \mathcal{V}$ is the subset of variable nodes, $\mathcal{C}$ is the set of m check nodes, and ∂^* is the set of corresponding edges. Since

(1) **Input**: observation vectors $\mathbf{y}$, length of original signal n, measurement matrix H.
(2) **Output**: estimation of original signal $\widehat{\mathbf{x}}$.
(3) **Initialization**: $\widehat{\mathbf{x}} = 0$.
(4) **for** each iteration **do**
(5) **for** $i = 1 : R$ **do**
(6) denote $\mathbf{y}_i$ for observation vector of ith check node
(7) **if** $\|\mathbf{y}_i\|^2 == 0$ **then**
(8) i is zero-ton.
(9) **else**
(10) [flag value location] = SingletonEstimator($\mathbf{y}_i$)
(11) **if** flag **then**
(12) $\mathbf{y} = \mathbf{y} - \text{value} \cdot H(:, \text{location})$, $\widehat{\mathbf{x}}(\text{location}) = \text{value}$
(13) **end if**
(14) **end if**
(15) **end for**
(16) **end for**

Algorithm 1: Short-and-Wide Iterative Fast Transform, SWIFT, algorithm.

(1) **Input**: observation vector $\mathbf{y}_i$.
(2) **Output**: a boolean *flag* "single-ton", estimation of signal: *value* and *location*;
(3) **Initialization**: flag = "false".
(4) **if** $(n/2\pi)\angle(\mathbf{y}_i[1]\mathbf{y}_i[2]^\dagger) \in \{0, 1, \ldots, (n-1)\}$ **then**
(5) flag = "true"
(6) value = $\mathbf{y}_i[1]$
(7) location = $(n/2\pi)\angle(\mathbf{y}_i[1]\mathbf{y}_i[2]^\dagger)$
(8) **end if**

Algorithm 2: Single-ton estimator algorithm.

the locations of nonzero elements in signal are random, the graph G_k^m is left-regular, as shown in Figure 1. In the SWIFT decoding algorithm, the criterions employed by the peeling decode algorithm depend on the degree of check nodes. To describe the asymptotic analysis, we classify the set of check nodes $\mathscr{C}$ as $\mathscr{C}_i$ being the set of check nodes having degree i, $1 \le i \le d_c$. Furthermore, the set $\mathscr{T}$ of nonzero elements of signal $\mathbf{x}$ is classified as $\mathscr{T}_i$ being the set variable nodes having i edges connected to the set $\mathscr{C}_1$, $0 \le i \le d_v$.

Based on the above classification and criterions employed in Section 2, in each iteration of SWIFT algorithm, a variable node can be decoded successfully if the following condition holds.

Theorem 3. *In each iteration, a variable node v can be decoded successfully with probability of almost 1 if and only if $v \in \bigcup_{i=1}^{d_v} \mathscr{T}_i$.*

Proof.

Sufficiency. When $v \in \mathscr{T}_1$, from the definition of single-ton, it is trivial that the variable node v can be decoded successfully with probability of almost 1. When $v \in \bigcup_{i=2}^{d_v} \mathscr{T}_i$, that is, the variable node v being connected to more than one single-ton, since the decoder can decode successfully, with probability of almost 1, the corresponding variable nodes from the single-ton, we can conclude that value of these single-tons must be identical. Hence, this case is identical to the case $v \in \mathscr{T}_1$.

Necessity. For the SWIFT algorithm, when a variable node v is decoded successfully, there must at least exist a single-ton check node that connected to the variable node v; that is, $v \in \bigcup_{i=1}^{d_v} \mathscr{T}_i$. □

We remodel the sparse measurement graph via the definition of $\mathscr{C}_i$ as well as $\mathscr{T}_i$ and Theorem 3, which allows us to analyze the evolution of $\mathscr{C}_i$ and $\mathscr{T}_i$. Let $p_{\mathscr{C}_i}^{(\ell)}$ denote the set of probability that a check node belongs to $\mathscr{C}_i$ and let $p_{\mathscr{T}_i}^{(\ell)}$ denote the set of probability that a variable node belongs to $\mathscr{T}_i$, where ℓ is the iteration number. Furthermore, we denote $\alpha^{(\ell)}$ by the probability that a variable node belongs to $\mathscr{T}^{(\ell)}$. Given $p_{\mathscr{C}_i}^{(\ell)}$, $p_{\mathscr{T}_i}^{(\ell)}$, and $\alpha^{(\ell)}$ at iteration $\ell \ge 1$, our asymptotic analysis can calculate $p_{\mathscr{C}_i}^{(\ell+1)}$, $p_{\mathscr{T}_i}^{(\ell+1)}$, and $\alpha^{(\ell+1)}$ at next iteration $\ell + 1$. For given initial density factor α, we can track the evolution of $\alpha^{(\ell)}$ via the analysis approach. Once the probability $\alpha^{(\ell)}$ decreases monotonically to zero with the iteration number increase, we refer to this case as the decoding algorithm recovering

the original signal successfully; while the probability $\alpha^{(\ell)}$ is bounded away from 0 with any iteration number, we call this case decoding algorithm failure. In what follows, we provide a rigorous calculation of the above sets of probability via combinatorial and probabilistic arguments.

3.2. Details of Asymptotic Analysis. Define $\mathcal{S}_{\mathcal{T}} = \bigcup_{i=1}^{d_v} \mathcal{T}_i$, $\mathcal{S}_{\mathcal{T}}^c = \mathcal{T}_0$. From Theorem 3, it is known that a variable node that belongs to $\mathcal{S}_{\mathcal{T}}$ can be decoded successfully and a variable node that belongs to $\mathcal{S}_{\mathcal{T}}^c$ is left intact. Thus, the probability $p_v^{(\ell)}$ that a variable node in $\mathcal{T}^{(\ell)}$ is being decoded successfully is

$$p_v^{(\ell)} = \sum_{\mathcal{T}_i \in \mathcal{S}_{\mathcal{T}}} p_{\mathcal{T}_i}^{(\ell+1)}. \tag{4}$$

Hence, after ℓth iteration, the probability of variable node v remaining undecoded is

$$\alpha^{(\ell+1)} = \alpha^{(\ell)} \left(1 - p_v^{(\ell)}\right). \tag{5}$$

Once the variable node is decoded successfully, as discussed in SWIFT algorithm, the d_v edges along with the variable node will be removed, which also results in the degree of check nodes being incident to these removed edges. Let $p_{\mathcal{C}_{ij}}^{(\ell)}$ denote the probability that the degree number i of a check node c reduces to number j, where $j \le i$. Based on Theorem 3, when a variable node $v \in \mathcal{T}_i$ is decoded successfully, i edges along with the set $\mathcal{C}_1$ as well as $d_v - i$ edges along with the set $\mathcal{C}_1^c = \{\mathcal{C}_2 \cup \mathcal{C}_3 \cup \cdots \cup \mathcal{C}_{d_c}\}$ are removed. To calculate the probability $p_{\mathcal{C}_{ij}}^{(\ell)}$ conveniently, we use the total probability law:

$$p_{\mathcal{C}_j}^{(\ell+1)} = \sum_{i=j}^{d_c} p_{\mathcal{C}_i}^{(\ell)} p_{\mathcal{C}_{ij}}^{(\ell)}, \quad j = 0, \ldots, d_c. \tag{6}$$

For the asymptotic analysis, we assume the length of signal $n \to \infty$ and the designed sparse matrix is random. Hence, both the distribution of i edges between variable node v and the set $\mathcal{C}_1$ and the $d_v - i$ edges between variable node v and the set $\mathcal{C}_1^c$ can be regarded as uniform distributions. Let $p_{\mathcal{C}_1}$ and $p_{\mathcal{C}_1^c}$ be the probabilities that an edge is removed given that the edge is incident to a check node in the sets $\mathcal{C}_1$ and $\mathcal{C}_1^c$, respectively. We have

$$\begin{aligned} p_{\mathcal{C}_{10}}^{(\ell)} &= C_1^1 \left(p_{\mathcal{C}_1}\right)^1 \left(1 - p_{\mathcal{C}_1}\right)^0 = p_{\mathcal{C}_1}, \\ p_{\mathcal{C}_{11}}^{(\ell)} &= 1 - p_{\mathcal{C}_{10}}^{(\ell)}, \end{aligned} \tag{7}$$

$$p_{\mathcal{C}_{ij}}^{(\ell)} = C_i^{i-j} \left(p_{\mathcal{C}_1^c}\right)^{i-j} \left(1 - p_{\mathcal{C}_1^c}\right)^j, \quad i = 2, \ldots, d_c, \ j = 0, \ldots, i. \tag{8}$$

Before calculating $p_{\mathcal{C}_1}$ and $p_{\mathcal{C}_1^c}$, we should introduce a significant conditional probability. Let $p^{(\ell)}$ denote the probability that a check node incident to an edge e belongs to $\mathcal{C}_1$ given a variable node incident to the edge in $\mathcal{T}^\ell$; its diagram is shown in Figure 3. Then, the $p^{(\ell)}$ can be calculated by

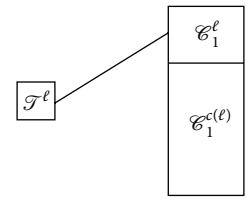

Figure 3: Diagram of calculation of conditional probability $p^{(\ell)}$.

$$\begin{aligned} p^{(\ell)} &= \Pr\left[c \in \mathcal{C}_1^{(\ell)} \mid v \in \mathcal{T}^\ell\right] = \frac{\Pr\left[v \in \mathcal{T}^\ell, c \in \mathcal{C}_1^{(\ell)}\right]}{\Pr\left[v \in \mathcal{T}^\ell\right]} \\ &= \frac{p_{\mathcal{C}_1}^{(\ell)}}{d_c \alpha^{(\ell)}}. \end{aligned} \tag{9}$$

From the definition of $p_{\mathcal{C}_1}$, we have

$$\begin{aligned} p_{\mathcal{C}_1} &= \Pr\left[v \in \mathcal{S}_{\mathcal{T}}^{(\ell)} \mid c \in \mathcal{C}_1^{(\ell)}, v \in \mathcal{T}^{(\ell)}\right] \\ &= \sum_{\mathcal{T}_i^\ell \in \mathcal{S}_{\mathcal{T}}^{(\ell)}} \Pr\left[v \in \mathcal{T}_i^\ell \mid c \in \mathcal{C}_1^{(\ell)}, v \in \mathcal{T}^{(\ell)}\right] \\ &= \sum_{\mathcal{T}_i^\ell \in \mathcal{S}_{\mathcal{T}}^{(\ell)}} \frac{\Pr\left[c \in \mathcal{C}_1^{(\ell)} \mid v \in \mathcal{T}_i^\ell, v \in \mathcal{T}^{(\ell)}\right] \Pr\left[v \in \mathcal{T}_i^\ell \mid v \in \mathcal{T}^\ell\right]}{\Pr\left[c \in \mathcal{C}_1^{(\ell)} \mid v \in \mathcal{T}^\ell\right]} \\ &= \frac{\sum_{\mathcal{T}_i^\ell \in \mathcal{S}_{\mathcal{T}}^{(\ell)}} (i/d_v)\, p_{\mathcal{T}_i}^{(\ell)}}{p^{(\ell)}} = \frac{d_c \alpha^{(\ell)} \sum_{\mathcal{T}_i^\ell \in \mathcal{S}_{\mathcal{T}}^{(\ell)}} i p_{\mathcal{T}_i}^{(\ell)}}{d_v p_{\mathcal{C}_1}^{(\ell)}}. \end{aligned} \tag{10}$$

Similarly, the calculation of $p_{\mathcal{C}_1^c}$ can be obtained by

$$\begin{aligned} p_{\mathcal{C}_1^c} &= \Pr\left[v \in \mathcal{S}_{\mathcal{T}}^{(\ell)} \mid c \in \mathcal{C}_1^{c(\ell)}, v \in \mathcal{T}^{(\ell)}\right] \\ &= \sum_{\mathcal{T}_i^\ell \in \mathcal{S}_{\mathcal{T}}^{(\ell)}} \Pr\left[v \in \mathcal{T}_i^\ell \mid c \in \mathcal{C}_1^{c(\ell)}, v \in \mathcal{T}^{(\ell)}\right] \\ &= \sum_{\mathcal{T}_i^\ell \in \mathcal{S}_{\mathcal{T}}^{(\ell)}} \frac{\Pr\left[c \in \mathcal{C}_1^{c(\ell)} \mid v \in \mathcal{T}_i^\ell, v \in \mathcal{T}^{(\ell)}\right] \Pr\left[v \in \mathcal{T}_i^\ell \mid v \in \mathcal{T}^\ell\right]}{\Pr\left[c \in \mathcal{C}_1^{c(\ell)} \mid v \in \mathcal{T}^\ell\right]} \\ &= \frac{\sum_{\mathcal{T}_i^\ell \in \mathcal{S}_{\mathcal{T}}^{(\ell)}} (1 - i/d_v)\, p_{\mathcal{T}_i}^{(\ell)}}{1 - p^{(\ell)}} = \frac{d_v p_v^{(\ell)} - \sum_{\mathcal{T}_i^\ell \in \mathcal{S}_{\mathcal{T}}^{(\ell)}} i p_{\mathcal{T}_i}^{(\ell)}}{d_v \left(1 - p_{\mathcal{C}_1}^{(\ell)} / d_c \alpha^{(\ell)}\right)}. \end{aligned} \tag{11}$$

To sum up, we can calculate the set of probability $p_{\mathcal{C}_i}^{(\ell+1)}$ of a check node in $\mathcal{C}_i$ via (6)–(11).

Next, the $p_{\mathcal{T}_i}^{(\ell+1)}$ is discussed. When a variable node is decoded successfully, with the corresponding edges removal, it is also shown that these undecoded sets $\mathcal{T}^{(\ell+1)}$ should be reclassified into the set $\mathcal{T}_i^{\ell+1}$. Let $p_{\mathcal{T}_{ij}}^{(\ell)}$ denote the probability that a variable node $v \in \mathcal{T}_i^\ell$ changes into $v \in \mathcal{T}_j^{(\ell+1)}$; from Theorem 3, we have $i = 0$. On the other hand, check nodes in the set $\mathcal{C}_1^c$ also should be reclassified into the set $\mathcal{C}_1$. Let the set $\mathcal{C}_1^\Delta$ denote the check nodes that reclassified $\mathcal{C}_1^c$ to $\mathcal{C}_1$. Similarly, we assume the length of signal $n \to \infty$

Table 1: Threshold for different (d_v, d_c) with SWIFT and VB algorithms.

(d_v, d_c)	(3, 6)	(4, 8)	(5, 10)	(6, 12)	(3, 4)	(5, 6)	(5, 7)	(7, 8)
SWIFT $\alpha^{(0)}$	0.4294	0.3829	0.3411	0.3074	0.6476	0.5504	0.4782	0.4711
VB $\alpha^{(0)}$	0.2574	0.2394	0.2179	0.1992	0.4488	0.3892	0.3266	0.3335

and the designed sparse matrix is random. We can exploit the following: the distribution of edges incident to the set of check nodes $\mathscr{C}_1^c$ and $\mathscr{C}_1^\Delta$ is uniform. It can be seen that the probability $p_{\mathscr{T}_{0j}}^{(\ell)}$ is

$$\begin{aligned} p_{\mathscr{T}_{0j}}^{(\ell)} &= \Pr\left[v \in \mathscr{T}_j^{(\ell+1)} \mid v \in \mathscr{T}_0^{\ell}, v \in \mathscr{T}^{(\ell)}\right] \\ &= C_{d_v}^{j}\left(p_\Delta^{(\ell)}\right)^j\left(1-p_\Delta^{(\ell)}\right)^{d_v-j}, \quad j=0,\ldots,d_v, \end{aligned} \tag{12}$$

where $p_\Delta^{(\ell)}$ is the probability of an edge incident to the set $\mathscr{C}_1^c$ moving to the set $\mathscr{C}_1^\Delta$; that is,

$$\begin{aligned} p_\Delta^{(\ell)} &= \Pr\left[c \in \mathscr{C}_1^{(\ell)} \mid v \in \mathscr{T}^{(\ell)}, c \notin \mathscr{C}_{11}^{(\ell)}\right] \\ &= \frac{1\cdot\sum_{j=2}^{d_c} p_{\mathscr{C}_j}^{(\ell)} p_{\mathscr{C}_{j1}}^{(\ell)}}{1\cdot\sum_{j=2}^{d_c} p_{\mathscr{C}_j}^{(\ell)} p_{\mathscr{C}_{j1}}^{(\ell)} + \sum_{i=2}^{d_c} i p_{\mathscr{C}_i}^{(\ell+1)}}. \end{aligned} \tag{13}$$

Based on the total probability law, $p_{\mathscr{T}_i}^{(\ell+1)}$ can be expressed as

$$p_{\mathscr{T}_i}^{(\ell+1)} = \frac{p_{\mathscr{T}_0}^{(\ell)} p_{\mathscr{T}_{0i}}^{(\ell)}}{1-p_v^{(\ell)}}, \quad i=0,\ldots,d_v, \tag{14}$$

where $1-p_v^{(\ell)}$ is a normalization factor, which makes the probability $p_{\mathscr{T}_i}^{(\ell+1)}$ satisfying standardization of probability; that is,

$$\sum_{i=0}^{d_v} p_{\mathscr{T}_i}^{(\ell+1)} = \sum_{\mathscr{T}_i \in \mathscr{S}_{\mathscr{T}}^c} \sum_{j=i}^{d_v} p_{\mathscr{T}_i}^{(\ell)} p_{\mathscr{T}_{ij}}^{(\ell)} = \sum_{\mathscr{T}_i \in \mathscr{S}_{\mathscr{T}}^c} p_{\mathscr{T}_i}^{(\ell)} = 1 - p_v^{(\ell)}. \tag{15}$$

Thus, according to the calculation of (12)–(15), the probability $p_{\mathscr{T}_i}^{(\ell+1)}$ can be obtained.

In what follows, we provide a precise summary of the above updated rules.

(1) Initialization. Setting the initial value α to $\alpha^{(0)}$, in the random sparse bipartite graph, it can be seen that the probability of a check node that belongs to the set $\mathscr{T}_i$ can be given by the following binomial distribution:

$$p_{\mathscr{C}_i}^{(0)} = C_{d_c}^{i}\left(\alpha^{(0)}\right)^i\left(1-\alpha^{(0)}\right)^{d_c-i}. \tag{16}$$

From Figure 3 and the probability $p^{(\ell)}$ in (9), the probability $p_{\mathscr{T}_i}^{(0)}$ is given by the following binomial distribution:

$$p_{\mathscr{T}_i}^{(0)} = C_{d_v}^{i}\left(p^{(0)}\right)^i\left(1-p^{(0)}\right)^{d_v-i}. \tag{17}$$

(2) Calculating $p_{\mathscr{C}_j}^{(\ell+1)}$. Substituting (7)–(11) into (6), then $p_{\mathscr{C}_j}^{(\ell+1)}$ is calculated.

(3) Calculating $p_{\mathscr{T}_i}^{(\ell+1)}$. Substituting (13)–(15) into (12), then $p_{\mathscr{T}_i}^{(\ell+1)}$ is calculated.

(4) Calculating $\alpha^{(\ell+1)}$. Substituting (4) and the probability $p_{\mathscr{T}_i}^{(\ell+1)}$ in step (3) into (5), $\alpha^{(\ell+1)}$ is calculated.

3.3. Comparison of VB Algorithm in the Asymptotic Regime. In this subsection, we compare the above analytical success thresholds of SWIFT algorithm with VB algorithm in the asymptotic regime. In Table 1, we have listed the analytical success thresholds of the SWIFT and VB algorithms for different graphs.

As seen in Table 1, for every graph, the SWIFT algorithm has a better performance than the VB algorithm. Overall, it is shown that the oversampling ratio $\gamma = k/m = d_v/(\alpha^{(0)}d_c)$ improves when decreasing both d_v and d_c. Furthermore, when we keep compression ratio constant—that is, d_v/d_c is a constant—such as (3, 6), (4, 8), and (5, 10), as shown in Table 1, in general, when decreasing d_v, SWIFT and VB algorithms perform better. Based on the relationship between the threshold and (d_v, d_c), the sparse measurement matrix with superior performance can be designed.

4. Simulation Results and Analysis

In this section, simulation results are provided for analyzing the performance of the SWIFT decoding algorithm. We also provide asymptotic analysis described in Section 3 with $n \to \infty$. According to the comparison of SWIFT decoding algorithm and asymptotic analysis, it is shown that there is a good agreement between them.

4.1. Asymptotic and Finite-Length Results for SWIFT Algorithm. Two scenarios are adopted for verifying Theorem 2: (1) nonzero elements obey Gaussian distribution; (2) nonzero elements take value from the discrete set $\{\pm 1\}$. The reconstruction signal-to-noise ratio is defined as $\mathrm{SNR}(\mathbf{x}) = 10 \cdot \log_{10}(\|\mathbf{x}\|_2/\|\mathbf{x}-\hat{\mathbf{x}}\|_2)$. We declare that recovery is successful if $\mathrm{SNR}(\mathbf{x}) \geq 100$; for each point, 1000 Monte Carlo trials are performed. In the asymptotic analysis, we declare that the recovery is successful if $\alpha^{(\ell)} < 10^{-8}$. To calculate the threshold of α, the search step is set to $\Delta = 10^{-4}$.

In the first experiment, (8, 32) regular sparse measurement matrix is designed. The length of signal is set to $n = 10000$, a signal element belongs to the support set with probability of α, and each nonzero signal element is drawn

Table 2: Asymptotic threshold and simulation success thresholds for SWIFT algorithm.

n	1×10^3	1×10^4	1×10^5	5×10^5	Asymptotic
Threshold	$\alpha^{(0)} = 0.0851$	$\alpha^{(0)} = 0.0946$	$\alpha^{(0)} = 0.106$	$\alpha^{(0)} = 0.1204$	$\alpha^{(0)} = 0.1246$

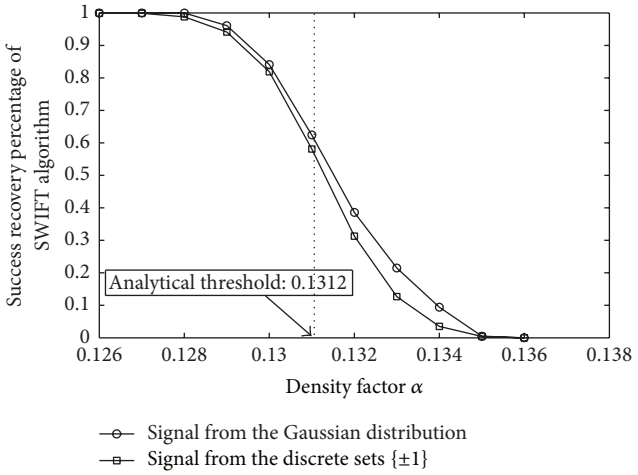

Figure 4: Success recovery percentage of SWIFT algorithm versus the initial density factor α; analysis threshold is curved by dotted line.

from the Gaussian distribution (or the discrete set $\{\pm 1\}$). The success recovery percentage of SWIFT algorithm versus the initial density factor α is shown in Figure 4. From Figure 4, it can be observed that, with the initial density factor α increasing, the performance of recovery decreases. In the figure, we can also see that the performance of two types of signals is almost identical. In addition, the threshold of the algorithm for the $(8, 32)$ regular sparse measurement is obtained by the asymptotic analysis, portrayed by dotted line. As shown in Figure 4, the threshold has a good agreement with the simulation curves.

In the second experiment, we verify that the SWIFT only requires $\mathcal{O}(k)$ measurements and $\mathcal{O}(k)$ iterative step. In order to examine the measurement cost and run-time, a random sparse matrix with regular degree $d_v = 3$ is adopted. Figure 5 illustrates the measurement costs and run-time as a function of the length of signal n. As seen in the figure, the measurement and run-time costs remain almost constant independently of n.

To investigate the level of agreement between the asymptotic analysis and SWIFT algorithm, we provide the evolution of α^{ℓ} with iterations ℓ for $(8, 32)$ regular sparse measurement in Figure 6. In this experiment, two cases of $\alpha^{(0)}$ value are adopted: one above the threshold and another below the threshold. As shown in Figure 6, there is also a good agreement between simulation results and asymptotic analysis for the two cases of $\alpha^{(0)}$.

To further show the degree of agreement between the asymptotic analysis and simulation results, we apply SWIFT algorithm to $(3, 20)$ regular sparse graph with $n = \{1\times10^3, 1\times10^4, 1\times10^5, 5\times10^5\}$. The successful recovery of thresholds $\alpha^{(0)}$

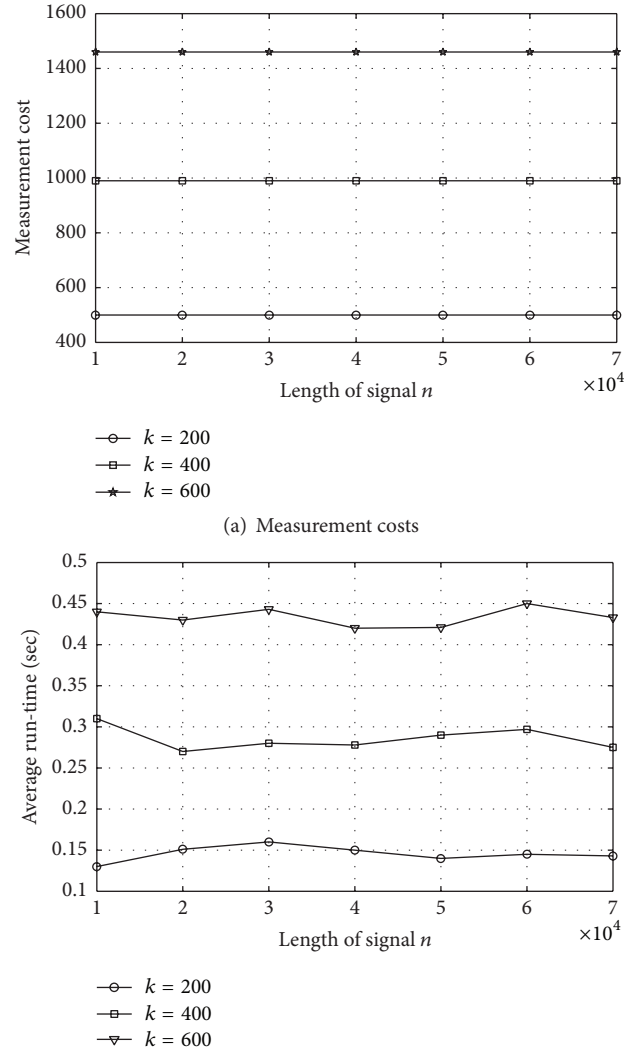

Figure 5: Measurement costs and average running time versus length of signal n.

is shown in Table 2. As shown in Table 2, when increasing n, the simulation results are closer to the asymptotic threshold.

4.2. Comparison of SWIFT Algorithm and VB Algorithm at Finite-Length. In this subsection, we compare the SWIFT algorithm with the VB algorithm in the noiseless setting. In the VB algorithm, two, $(3, 15)$ and $(3, 72)$, regular sparse matrices are designed with signal length $n = 2500, 12000$, respectively. To have a fair comparison, in the SWIFT algorithm, two, $(3, 30)$ and $(3, 144)$, regular sparse matrices are generated with signal length $n = 2500, 12000$, respectively. Each nonzero signal element is taken from a standard

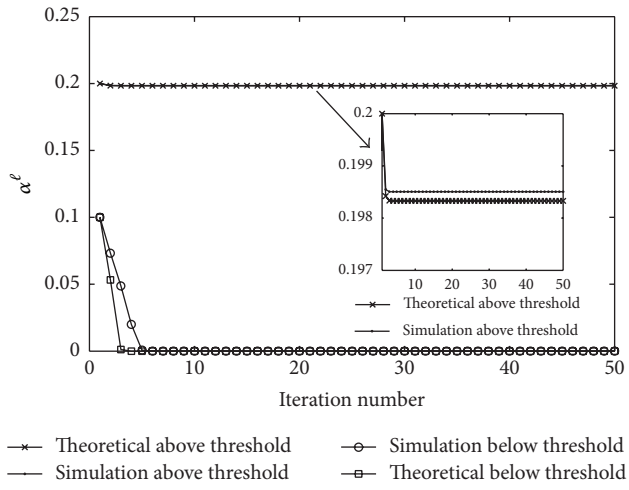

Figure 6: Evolution of α^ℓ versus iteration number ℓ with $(8, 32)$ regular sparse measurement.

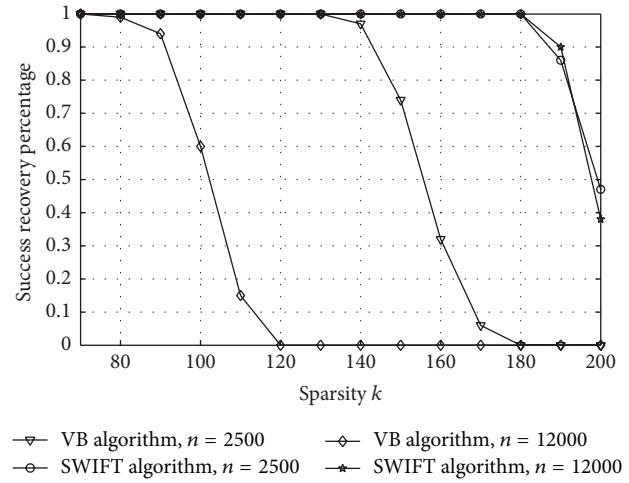

Figure 7: Success recovery percentage of SWIFT and VB algorithms versus the sparsity k.

Gaussian distribution. The success recovery percentage of SWIFT and VB algorithms versus the sparsity k is shown in Figure 7. As shown in Figure 7, the SWIFT algorithm performs better than the VB algorithm especially for large signal length, which coincides with the analysis in Section 3.3.

5. Conclusion

In this paper, we investigate a new class of LDPC-DFT measurement matrices and their recovery algorithm. These new measurement matrices exploit the sparsity of parity-check matrices of LDPC and the phase of DFT matrices to recover the original signal. In addition, a general framework for the asymptotic analysis of this fast recovery algorithm is proposed. The analysis shows that there exists a threshold on the density factor $\alpha = k/n$; if below this threshold, the recovery algorithm is successful; otherwise it will fail. Moreover, it can be found that the threshold is dependent on the parameter of LDPC matrix which provides a basis for us to design the measurement matrices. We also observe that the SWIFT algorithm performs better than other schemes based on LDPC codes.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This research was supported by a research grant from the National Natural Science Foundation of China (no. 61271354).

References

[1] E. J. Candés and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[2] E. J. Candés, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.

[3] A. G. Dimakis, R. Smarandache, and P. O. Vontobel, "LDPC codes for compressed sensing," *IEEE Transactions on Information Theory*, vol. 58, no. 5, pp. 3093–3114, 2012.

[4] S.-T. Xia, X.-J. Liu, Y. Jiang, and H.-T. Zheng, "Deterministic constructions of binary measurement matrices from finite geometry," *IEEE Transactions on Signal Processing*, vol. 63, no. 4, pp. 1017–1029, 2015.

[5] X. J. Liu, S. T. Xia, and T. S. Dai, "Deterministic constructions of binary measurement matrices with various sizes," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '15)*, pp. 3641–3645, IEEE, South Brisbane, Australia, April 2015.

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

[7] D. Needell and R. Vershynin, "Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit," *Foundations of Computational Mathematics*, vol. 9, no. 3, pp. 317–334, 2009.

[8] A. Gilbert and P. Indyk, "Sparse recovery using sparse matrices," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 937–947, 2010.

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

[10] Y. Eftekhari, A. Heidarzadeh, A. H. Banihashemi, and I. Lambadaris, "Density evolution analysis of node-based verification-based algorithms in compressed sensing," Tech. Rep. SCE-11-07, Carleton University, Ottawa, Canada, 2011.

[11] F. Zhang, *LDPC codes over large alphabets and their applications to compressed sensing and flash memory [Doctoral Dissertation]*, Texas A&M University, College Station, Tex, USA, 2010.

[12] Z. Zeinalkhani and A. H. Banihashemi, "Low-complexity detection of zero blocks in wideband spectrum sensing," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT '15)*, pp. 2578–2582, IEEE, Hong Kong, June 2015.

[13] V. Chandar, D. Shah, and G. W. Wornell, "A simple message-passing algorithm for compressed sensing," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT '10)*, pp. 1968–1972, Austin, Tex, USA, June 2010.

[14] X. Wu and Z. Yang, "Verification-based interval-passing algorithm for compressed sensing," *IEEE Signal Processing Letters*, vol. 20, no. 10, pp. 933–936, 2013.

[15] S. Jafarpour, W. Xu, B. Hassibi, and R. Calderbank, "Efficient and robust compressed sensing using optimized expander graphs," *IEEE Transactions on Information Theory*, vol. 55, no. 9, pp. 4299–4308, 2009.

[16] W. Xu and B. Hassibi, "Efficient compressive sensing with deterministic guarantees using expander graphs," in *Proceedings of the IEEE Information Theory Workshop (ITW '07)*, pp. 414–419, Tahoe City, Calif, USA, September 2007.

[17] D. Baron, S. Sarvotham, and R. G. Baraniuk, "Bayesian compressive sensing via belief propagation," *IEEE Transactions on Signal Processing*, vol. 58, no. 1, pp. 269–280, 2010.

[18] M. Akçakaya, J. Park, and V. Tarokh, "Low density frames for compressive sensing," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '10)*, pp. 3642–3645, IEEE, Dallas, Tex, USA, March 2010.

[19] S. Pawar and K. Ramchandran, "A hybrid DFT-LDPC framework for fast, efficient and robust compressive sensing," in *Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton '12)*, pp. 1943–1950, IEEE, October 2012.

[20] S. Pawar and K. Ramchandran, "Computing a k-sparse n-length discrete Fourier transform using at most 4k samples and O(k log k) complexity," http://arxiv.org/abs/1305.0870.

[21] S. Pawar and K. Ramchandran, "A robust sub-linear time R-FFAST algorithm for computing a sparse DFT," http://arxiv.org/abs/1501.00320.

[22] X. Li, S. Pawar, and K. Ramchandran, "Sub-linear time support recovery for compressed sensing using sparse-graph codes," http://arxiv.org/abs/1412.7646.

[23] M. Mitzenmacher, "A survey of results for deletion channels and related synchronization channels," *Probability Surveys*, vol. 6, pp. 1–33, 2009.

[24] H. Hassanieh, L. Shi, O. Abari, E. Hamed, and D. Katabi, "GHz-wide sensing and decoding using the sparse fourier transform," in *Proceedings of the IEEE INFOCOM*, pp. 2256–2264, Toronto, Canada, May 2014.

[25] B. Ghazi, H. Hassanieh, P. Indyk, D. Katabi, E. Price, and L. Shi, "Sample-optimal average-case sparse fourier transform in two dimensions," in *Proceedings of the 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton '13)*, pp. 1258–1265, Monticello, Ill, USA, October 2013.

[26] D. Yin, K. Lee, R. Pedarsani, and K. Ramchandran, "Fast and robust compressive phase retrieval with sparse-graph codes," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT '15)*, pp. 2583–2587, IEEE, Hong Kong, China, June 2015.

[27] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 569–584, 2001.