Node-Based Verification-Based Recovery Algorithms in Compressed Sensing: Asymptotic Analysis and Design of Irregular Sensing Matrices (Graphs)

by

Yaser Eftekhari Roozbehani, **M.Sc.**

A dissertation submitted to the
Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario
September, 2012

The undersigned hereby recommends to the
Faculty of Graduate Studies and Research
acceptance of the dissertation

Node-Based Verification-Based Recovery Algorithms in Compressed Sensing: Asymptotic Analysis and Design of Irregular Sensing Matrices (Graphs)

submitted by **Yaser Eftekhari Roozbehani**, **M.Sc.**

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical and Computer Engineering

Professor Amir H. Banihashemi, Thesis Supervisor

Professor Ioannis Lambadaris, Thesis Co-supervisor

Professor Richard Baraniuk, External Examiner

Professor Howard Schwartz, Chair,
Department of Systems and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems and Computer Engineering
Carleton University
September, 2012

Abstract

In this thesis, we study the asymptotic behavior of iterative node-based verification-based (NB-VB) recovery algorithms over random sparse matrices in the context of compressed sensing. Such algorithms are particularly interesting due to their low complexity (linear in the signal dimension n). Let α, here referred to as density factor, be the probability that a signal element is non-zero. It is known that there exists a success threshold on the density factor, before which the recovery algorithms are successful, and beyond which they fail with probability one.

We propose a mathematical framework that predicts the average fraction of unverified signal elements at each iteration ℓ in the asymptotic regime, where the average is taken over the ensembles of input signals and sensing matrices as a function of ℓ as $n \to \infty$. The asymptotic analysis is similar in nature to the well-known density evolution technique commonly used to analyze iterative decoding algorithms. To perform the analysis, a message-passing interpretation of NB-VB algorithms is provided. This interpretation lacks the extrinsic nature of standard message-passing algorithms to which density evolution is usually applied. This requires a number of non-trivial modifications in the analysis. We first discuss the analysis of recovery algorithms of interest over random regular sensing graphs. Later we generalize the analysis to include random irregular sensing graphs as well. We also discuss concentration results that ensure the performance of the recovery algorithms applied to any choice of the input signal over any realization of the sensing matrix follows the deterministic results of the analysis closely.

We also demonstrate that the proposed asymptotic analysis matches the performance of recovery algorithms for large but finite values of n. Compared to the sole existing technique for the analysis of NB-VB algorithms, which is based on numerically solving a large system of coupled differential equations, the proposed analysis is simpler to implement and more accurate. Moreover, we use the proposed analysis in an optimization loop in order to design irregular sensing matrices (graphs) that outperform previously reported results. The maximum density factor that designed irregular graphs can handle exceeds that of the regular graphs substantially.

I dedicate this dissertation to my wonderful family. Particularly to my understanding and patient wife, Tara, who has given me her fullest support during all these years of research.

Table of Contents

List of Tables

List of Figures

Nomenclature

AMP	Approximate Message-Passing
BP	Belief Propagation
CN	Check Node
CS	Compressed Sensing
D1CN	Degree One Check Node
d.d.	Degree Distribution
ECN	Equal Check Nodes
HR	Half Round
LDPC	Low-Density Parity-Check
LM	Luby and Mitzenmacher (algorithm)
MB	Message-Based
MECN	Modified Equal Check Nodes
MRI	Magnetic Resonance Imaging
MSE	Mean Square Error
NB	Node-Based
RIP	Restricted Isometry Property
SBB	Sarvotham, Baron and Baraniuk (algorithm)
SNR	Signal to Noise Ratio
VB	Verification-Based
VN	Variable Node
XH	Xu and Hassibi (algorithm)
ZCN	Zero Check Node

Chapter 1

Introduction

1.1 Motivation

Many signals in real life applications, if defined in the proper domain, will have a small support set with respect to their dimension [2, 3]. For example, when an ordinary image is represented in the wavelet domain, most of the information lies in the low frequency components, while high frequency components can be neglected with a small perceptual loss [3]. This simple observation is the key factor in many compression algorithms based on transform coding. In these schemes, a signal is represented in the domain in which it has many negligible coefficients. The few large transform coefficients are coded along with their location in the transform. This whole process of calculating all transform coefficients and then throwing away a large quantity of them is very wasteful.

In certain applications we can not afford to acquire a vast number of measurements and throw most of them away. The reason could be that the number of sensors present is limited, or sampling (sensing) the signal is time consuming, costly or slow. Imaging based on neutron scattering and Magnetic Resonance Imaging (MRI) are considered examples of costly and slow sensing processes, respectively [3]. In these cases, a logical question to ask then would be: is it possible to sense the original signal in an already compressed fashion? As we will see in the sequel, compressed sensing (CS) answers this question in affirmative.

1.2 Basic Notations and Definitions

Compressed sensing was introduced with the idea to represent a signal $\boldsymbol{v} \in \mathbb{R}^n$ with k non-zero elements with measurements $\boldsymbol{c} \in \mathbb{R}^m$, where $k < m \ll n$, and yet to be able to recover back the original signal $\boldsymbol{v}$ [4, 5]. In the measuring process, also referred to as *encoding*, signal elements are mapped to measurements through a linear transformation represented by the matrix multiplication $\boldsymbol{c} = \boldsymbol{G}\boldsymbol{v}$, where the matrix $\boldsymbol{G} \in \mathbb{R}^{m \times n}$ is referred to as the *sensing matrix*. This linear mapping can also be characterized by a bipartite graph [2], referred to as the *sensing graph*.

In the recovery process, also referred to as *decoding*, based on the knowledge of the measurements and the sensing matrix, we estimate the original signal. The decoding process is successful if $\boldsymbol{v}$ is estimated correctly. Three performance measures namely, density ratio $\gamma \triangleq k/n$, compression ratio $r_c \triangleq m/n$, and oversampling ratio $r_o \triangleq m/k$ are used in order to measure and compare the performance of the recovery algorithms in the context of compressed sensing.

For successful decoding clearly we need $r_o \geq 1$. It is desirable to have this parameter as small as possible. Indeed, in [6] the authors proved that for sparse signals and in noiseless scenarios $r_o = 1$ is achievable in the asymptotic case ($n \to \infty$). This means that the highest density ratio that an algorithm can possibly handle is $\gamma^* = r_c$. Authors in [7] have shown that if the sensing matrix consists of i.i.d. Gaussian elements, then a decoder based on the ℓ_0 norm can recover the original signal with $m = k + 1$ measurements; i.e., $r_o \approx 1$. To find the solution based on the ℓ_0 recovery, however, one has to perform an exhaustive search, which is computationally too complex [8].

1.3 Previous Works

The sensing matrix in compressed sensing can be either *dense* or *sparse*. A sensing matrix is considered dense if it has few, or none, zero entries. Sparse matrices, on the other hand, have few non-zero entries in each row and column. One major difference between these two types of matrices is the encoding complexity associated with each class. For sparse matrices, the number of operations needed to calculate the measurements is considerably lower than the one needed for dense matrices.

Decoding algorithms can be classified based on the class of sensing matrix they use. The decoding algorithms in each class have certain properties in common. (For a comprehensive study on the topic, we refer the interested readers to [9].) Decoding algorithms associated with dense matrices have, generally, high complexity (between $O(n^2)$ and $O(n^3)$) compared to the lower complexity of algorithms utilizing sparse matrices (between $O(n)$ and $O(n^2)$). To have a better feeling about the complexity and running time of these two classes of algorithms, we have included in Fig. 4.5 of Section 4.8 the comparison between two standard recovery algorithms for dense matrices (ℓ_1 minimization and weighted ℓ_1 minimization) and one algorithm (SBB) for sparse matrices. As can be seen, the decoding algorithm for sparse matrices is faster by about two orders of magnitude. Decoding algorithms for dense matrices are mostly based on linear or convex programming [4, 5, 10, 11]. The reason is that random dense matrices satisfy *restricted isometry property* (RIP) with overwhelming probability [12, 13]. The RIP was introduced by Candès and Tao in [14] as the main restriction on the sensing matrix so that the recovery based on linear programming will be able to successfully recover the signal. Sparse matrices, on the other hand, do not satisfy RIP unless $m = \Omega(k^2)$ [15].[1] In fact, many of the decoders based on

[1] Authors in [9] extended the definition of RIP and showed that sparse matrices satisfy a generalized RIP constraint. The generalized RIP suffices for the linear programming decoders to succeed

sparse sensing matrices are iterative [1, 2, 16–27]. Although more computationally complex, decoding algorithms for dense matrices tend to recover signals with larger number of non-zero elements (higher density ratio) compared to decoders for sparse matrices. Nevertheless, the high complexity of decoding algorithms on dense matrices hinders their application to high-dimensional signal recovery (signals with large n). In such cases, using sparse sensing graphs is clearly beneficial. Moreover, in certain compressed sensing applications such as computer networks [25,28,29], channel coding [14], spectrum sensing [30], and identification of linear operators [31], the nature of the problem results in a formulation with a sparse sensing graph.

Focusing on recovery algorithms based on sparse matrices (or sparse graphs), we can further divide them into two major groups. In one group, we have algorithms that use group testing and similar techniques from estimation theory [16–19]. These are referred to as *combinatorial algorithms*. In the other group, recovery algorithms work with the bipartite graph associated with the sensing matrix by passing messages over the edges of the graph [1, 2, 20–27, 32, 33]. These are referred to as *message-passing* algorithms. Combinatorial algorithms, generally, assume that the decoder knows the parameter k [16–19]. These algorithms, have two main steps. In the first step, the algorithm outputs an estimate which has more non-zero values than the original signal. In the next step, knowing the parameter k, the estimate is pruned so that it has the same number of non-zero elements as the original signal (i.e., k non-zero elements). Combinatorial algorithms are, in general, more computationally complex than message-passing algorithms. For example, the algorithm introduced in [16] has complexity $O(k^2\text{polylog}(n))$, which translates to $O(n^2\text{polylog}(n))$ in the regime where k scales linearly with n. Message-passing algorithms, on the other hand, have computational complexity $O(n)$.

For the reasons discussed above, we are interested in low-complexity recovery algorithms that exploit the sparsity of the sensing matrix. In particular, we are interested in message-passing recovery algorithms. In [32, 33], the authors proposed and analyzed a belief propagation (BP) algorithm to recover the non-zero elements of the signal for the scenario where the location of such elements is known in the asymptotic regime. It was shown in [34] that if each signal element contributes to infinitely many measurements (subject to certain criteria), then the BP algorithm is asymptotically optimal in the case of sparse noisy measurements. In [25], the authors proposed a simple message-passing algorithm to reconstruct non-negative signals. This algorithm assumes lower and upper bounds for the values of the signal elements. It then shrinks the difference between the two bounds through iterations. An analysis for the same algorithm that yields uniform guarantee on signal reconstruction was then provided in [21]. Another approach in message-passing algorithms is to assume a prior distribution for the values of the signal elements and try to maximize the a-posteriori distribution of the values of the elements based on the observed

in recovering sparse signals. However, the resulting bound on the reconstruction error is weaker compared to the case where the sensing matrix satisfies the original RIP condition.

measurements. In [27], the authors assumed Gaussian mixture priors. The main problem associated with this approach is that the length of the messages passed over the edges of the graph grows exponentially fast with the number of iterations. In another work [26], the authors assumed Jeffreys' priors [35] and aimed at recovering the non-zero elements of the original signal using message-passing algorithms. Then, they applied well-known least-square algorithms, such as LSQR [36], to estimate the value of the non-zero signal elements. Moreover, in [26], it was assumed that the parameter k is known. In a more recent work, authors in [37] applied the approximate message-passing (AMP) algorithm proposed in [38] for dense sensing matrices to band-diagonal (sparse) matrices. In their approach, the authors assume that the recovery algorithm has access to the distribution of non-zero elements of the signal. Algorithms discussed so far are either restrictive, in the sense that they assume some knowledge about the set of non-zero elements of the signal at the decoder, or have a high computational complexity that makes them impractical in applications with large n.

In the sequel, we are interested in a sub-class of message-passing algorithms called *Verification-Based* (VB) algorithms. An instance of VB algorithms was first introduced for compressed sensing in [20]. Later, [22, 23] observed that this algorithm is essentially identical to the earlier idea of verification decoding from [39]. This observation allowed a rigorous analysis of the VB reconstruction for compressed sensing. The class of VB algorithms has certain properties that make it perhaps one of the most interesting classes of recovery algorithms in compressed sensing. The VB algorithms recover signal elements in iterations. When an element is verified, its value is kept unchanged in future iterations. The algorithms in this class have decoding complexity $O(n)$, which makes them suitable for applications involving recovery of signals with large n. Moreover, these algorithms operate on sparse sensing graphs, which translates to less computations in the encoding process. Another main advantage of VB algorithms is that they are not sensitive to the distribution of non-zero elements of the sensing matrix as well as the distribution of non-zero elements of the signal, if certain conditions are satisfied. We will elaborate on this topic further in Section 2.4. These properties make the VB algorithms a suitable choice for low-complexity recovery of sparse signals. The VB algorithms are, however, sensitive to the presence of noise in the measured data. One can always argue that: i) the noise-free analysis of recovery algorithms could serve as an upper bound for the performance of the noisy versions, and ii) noiseless compressed sensing has applications in computer networks as shown in [25, 29]. Nevertheless, we will comment on using standard thresholding techniques to deal with noisy measurements in Section 4.7. This technique is very effective in the high signal to noise ratio (SNR) regime (such as the scenario in [28]). An in-depth analysis of the approach, however, is beyond the scope of this thesis.

Another interesting feature of VB algorithms is that their performance can be analyzed in the asymptotic case ($n \to \infty$). Assume a probabilistic input model, in which a signal element is non-zero (and takes a value from a certain distribution) with probability α and is zero with probability $1 - \alpha$. In the sequel, we refer to parameter

α as the *density factor*. Furthermore, let $\alpha^{(\ell)}$ denote the probability that a signal element is non-zero and unverified before iteration ℓ over the ensemble of all sensing graphs and inputs of interest. So, $\alpha^{(0)} = \alpha$. If $\lim_{\ell\to\infty} \alpha^{(\ell)} = 0$, then the algorithm is called successful for the initial density factor α.[2] On the other hand, if there exists $\epsilon > 0$, such that $\lim_{\ell\to\infty} \alpha^{(\ell)} > \epsilon$, then the algorithm is said to fail for the initial density factor α.

Authors in [1,22,24,39–42] have shown that for each VB recovery algorithm in the asymptotic regime as $n \to \infty$ and $\ell \to \infty$, a limiting value exists for α, before which the recovery algorithm is successful and beyond which it is not. We refer to this limit as the *success threshold*. The success threshold serves as an asymptotic measure of performance for VB algorithms. It can also be used to estimate the performance of these algorithms for finite but large values of n. To this end, researchers have analyzed VB algorithms in the asymptotic regime ($n \to \infty, \ell \to \infty$) in order to find the success threshold associated with each VB algorithm. There are two categories of VB algorithms: *node-based (NB)* and *message-based (MB)* [1]. The two categories yield different success thresholds and are analyzed using different techniques. Algorithms considered in [22,39] are of MB type, while the authors in [1] considered the NB type recovery algorithms. In general, NB algorithms have higher success thresholds and are harder to analyze. We elaborate on the differences between the two categories and their corresponding analytical tools in Section 2.3. The focus of this work is on the analysis of NB algorithms.

The analysis of NB-VB algorithms discussed in [1] results in a system of coupled differential equations. Due to the lack of closed form solution for the resulting differential equations, the authors used numerical methods to approximate the asymptotic results. Since the analysis is only valid for $n \to \infty$, one has to choose very large n for the numerical approximation. This translates to long running time and high computational complexity. The other challenge is that numerical errors can affect the accuracy of the results, making it hard to evaluate how close the success threshold reported by this analysis (even for large values of n) is to the real success threshold. In comparison, the analysis proposed in this thesis lends itself to closed form update equations consisting of only simple operations (addition/subtraction and multiplication/division). As a result, the analysis is faster and, more importantly, more accurate.

1.4 Our Contributions

One of the main goals of this work is to develop a simple and accurate framework for the asymptotic analysis (as $n \to \infty, \ell \to \infty$) of NB-VB algorithms over sparse random sensing graphs and extend it to include recovery algorithms of similar nature such as that of [2]. In our analysis, we assume that the measurements are noiseless.

[2] It is easy to prove that the probability of a zero-valued signal element being unverified at iteration ℓ is upper bounded by $d_c \frac{\alpha^{(\ell)}}{1-\alpha^{(\ell)}}$. Hence, when $\alpha^{(\ell)}$ tends to zero, this probability also tends to zero.

We demonstrate that the recovery algorithms can be described by a first order time-varying Markov chain. We thus track the distribution of the states of this Markov chain as well as the transition probabilities between different states through iterations in the analysis. We refer to the analysis as *density evolution*, a terminology used for the analysis of iterative message-passing algorithms of low-density parity-check (LDPC) codes [43]. We however note that the analysis presented here fundamentally differs from the conventional density evolution where the message passing is *extrinsic* [43]. The computational complexity of the proposed analysis increases linearly with the number of iterations. The calculation of transition probabilities includes simple mathematical operations, more specifically addition and multiplication, as opposed to solving complex systems of coupled differential equations, as is the case in [1]. To be more precise, in [1], the variables involved in the equations are the expected values of certain random variables, such as the normalized number of edges of certain type. These random variables determine the status of the decoder. In our analysis, we calculate the probabilities of events involving similar or related random variables, and/or their densities.

As part of our asymptotic analysis, we also discuss concentration results which certify that the performance of a recovery algorithm for a random choice of the input signal and the sensing matrix is very close to what is predicted by the density evolution results at the limit of $n \to \infty$.

Using the proposed analysis, we can determine the distribution of the decoder states at any desired iteration. By tracking the distribution of the decoder states with iterations, we then find the success threshold of different NB-VB algorithms. The analysis is presented first for random regular sensing graphs and then generalized to include random irregular graphs as well. Moreover, using the proposed density evolution analysis, we perform a comprehensive study and comparison of performance of different VB recovery algorithms over a variety of sparse graphs. We also use the analysis for irregular graphs as the core in an optimization setup to study and find optimal sensing graphs under certain constraints such as fixed compression ratio and fixed maximum degrees in the sensing graph. Our simulations show that the behavior of VB algorithms, when applied to signals with large lengths (in the order of 10^5), are in good agreement with the asymptotic analytical results. They also demonstrate that the success threshold associated with irregular graphs can be substantially higher than the one for regular graphs.

1.5 Organization of the Thesis

The rest of the thesis is organized as follows. In Chapter 2, we introduce the class of bipartite graphs and input signals of interest in this thesis. We also provide a detailed description of VB algorithms along with the verification rules for each VB algorithm in this chapter. Also in this chapter, we discuss the important notion of false verification and its probability for VB algorithms. A message-passing interpretation

of the recovery algorithms is presented in Chapter 3. The analysis framework will be introduced in Chapters 4 and 5 for regular and irregular graphs, respectively. We propose a simple modification of VB algorithms to deal with noisy measurements in Chapter 4. Simulation results for regular and irregular graphs will be presented in Sections 4.8 and 5.4, respectively. In Chapter 6, we will discuss the design of irregular graphs that are optimal under certain constraints. We conclude in Chapter 7 with some guidelines for further research. Appendices A and B are devoted to the derivation of the transition probabilities for the regular and irregular graphs, respectively.

Chapter 2

Bipartite Graphs and Recovery Algorithms

2.1 Introduction

As discussed in the previous chapter, the focus of this work is on sparse sensing graphs and probabilistic input signals. In this chapter, we describe regular and irregular bipartite graphs, their random construction and the model used for the input signal. We then introduce the role of bipartite graphs in compressed sensing. Afterwards, we elaborate on verification-based algorithms, their history and their usage as recovery algorithms in the context of compressed sensing. We will discuss the verification rules employed by VB algorithms as well as false verification concept in VB algorithms.

2.2 Bipartite Graphs and Input Signal

A *bipartite graph* (or *bigraph*) $\mathcal{G}(V \cup C, E)$ is defined as a graph whose set of vertices $V \cup C$ is divided into two disjoint sets V and C, so that every edge in the set of edges E connects a vertex in V to one in C. Consider a bigraph $\mathcal{G}(V \cup C, E)$ with $|V| = n$ and $|C| = m$. Corresponding to each such graph, a *biadjacency matrix* $\boldsymbol{A}(\mathcal{G})$ of size $m \times n$ is formed as follows: the entry a_{ij} is 1 if there exists an edge $e_{ij} \in E$ connecting the vertex $c_i \in C$ to the vertex $v_j \in V$; and is 0, otherwise.

In general, a bigraph can be *irregular* and *weighted*. In a bigraph, a node in V (C) has degree i if it is neighbor (connected) to i nodes in C (V). Let $\lambda_i \in \mathbb{R}^+$ and $\rho_i \in \mathbb{R}^+$ denote the fraction of nodes in V and C with degree i, respectively. The polynomials $\lambda(x) = \sum_i \lambda_i x^i$ and $\rho(x) = \sum_i \rho_i x^i$ are referred to as *degree distributions* corresponding to nodes in V and C, respectively. Clearly, $\lambda(1) = \rho(1) = 1$. For mathematical convenience, we define $\bar{d}_v := \sum_i i\lambda_i$ and $\bar{d}_c := \sum_j j\rho_j$ and we refer to them as the *average variable degree* and the *average check degree*, respectively. For given $\lambda(x), \rho(x)$ and n, let $\mathcal{G}^n(\lambda(x), \rho(x))$ ($\mathcal{G}^n(\lambda, \rho)$ for short) denote the ensemble of all irregular bigraphs with degree distributions $\lambda(x)$ and $\rho(x)$ and $|V| = n, |C| = m = n\bar{d}_v/\bar{d}_c$.

Biregular graphs (or regular bigraphs) are considered special cases of irregular bigraphs as follows. Let d_v and d_c be two positive integers. Consider a bigraph $\mathcal{G}(V \cup C, E)$ so that each vertex in V (C) is incident to d_v (d_c) vertices in C (V). Clearly, $nd_v = md_c$. We refer to this bigraph as an (n, d_v, d_c)-*biregular graph*. The biadjacenecy matrix $\boldsymbol{A}(\mathcal{G})$ associated to an (n, d_v, d_c)-biregular graph has d_c non-zero entries in each row and d_v non-zero entries in each column. For given parameters d_v, d_c and n ($m = nd_v/d_c$), let $\mathcal{G}^n(d_v, d_c)$ denote the ensemble of all (n, d_v, d_c)-biregular graphs.

Moreover, a regular or irregular *bipartite weighted graph* (or *weighted bigraph*) $\mathcal{G}'(V \cup C, E, W(E))$ is a generalization of the bigraph $\mathcal{G}(V \cup C, E)$ in the sense that a weight $w_{ij} := W(e_{ij}) \in \mathbb{R}\backslash\{0\}$ is associated with each edge $e_{ij} \in E$. The biadjacency matrix $\boldsymbol{A}(\mathcal{G}')$ corresponding to the weighted bigraph $\mathcal{G}'$ is acquired from the biadjacency matrix $\boldsymbol{A}(\mathcal{G})$ of the underlying bigraph $\mathcal{G}$ by replacing non-zero a_{ij} values in $\boldsymbol{A}(\mathcal{G})$ with w_{ij}.

Let us assume an arbitrary, but fixed, labeling scheme for vertex sets V and C over the ensemble of graphs of interest. Further, let $\boldsymbol{W}$ be a matrix of size $m \times n$ of weights w drawn i.i.d. according to a distribution $f(w)$, and $\mathcal{W}_f^{m\times n}$ be the ensemble of all such matrices. Now for any bigraph $\mathcal{G}(V \cup C, E)$ ($\in \mathcal{G}^n(d_v, d_c)$ for biregular graphs and $\in \mathcal{G}^n(\lambda, \rho)$ for irregular bigraphs) and any weight matrix $\boldsymbol{W} \in \mathcal{W}_f^{m\times n}$, we form the corresponding weighted bigraph $\mathcal{G}'(V \cup C, E, W(E))$ as follows. To every edge $e_{ij} \in E$, $1 \le i \le m, 1 \le j \le n$, connecting ith vertex from C and jth vertex from V, we assign the weight $W(e_{ij}) = w_{ij}$; i.e., the weight in row i and column j of the weight matrix $\boldsymbol{W}$. Thus, we construct the ensemble of all (n, λ, ρ)-weighted irregular bigraph, denoted by $\mathcal{G}_f^n(\lambda, \rho)$, by freely combining elements in $\mathcal{G}^n(\lambda, \rho)$ and $\mathcal{W}_f^{m\times n}$. Similarly, we construct the ensemble of all (n, d_v, d_c)-weighted biregular graphs, denoted by $\mathcal{G}_f^n(d_v, d_c)$, by freely combining elements in $\mathcal{G}^n(d_v, d_c)$ and $\mathcal{W}_f^{m\times n}$.

Thus far, we have described the ensemble of graphs that are of interest in this work. In what follows, we describe the ensemble of inputs of interest. Let $\alpha \in [0, 1]$ be a fixed real number and $\boldsymbol{v}$ be a vector of length n with elements v_i drawn i.i.d. according to a probability distribution function defined as follows: the element is zero with probability $1 - \alpha$, or follows a distribution g with probability α (i.e., $P_{v_i}(v) = \alpha g(v) + (1 - \alpha)\delta(v)$, where δ is the Dirac delta function). We denote the ensemble of all such vectors by $\mathcal{V}_g^n(\alpha)$.[1]

In compressed sensing, each measurement is a linear combination of the signal elements. With a slight abuse of notation, we use c_i and v_j for both the label and the value of the ith measurement and the jth signal element, respectively. We denote by $\boldsymbol{c}$ and $\boldsymbol{v}$, the column vectors of the measurements c_i's ($1 \le i \le m$), and the signal elements v_j's ($1 \le j \le n$), respectively. The underlying system of linear combinations can then be represented by the matrix multiplication $\boldsymbol{c} = \boldsymbol{G}\boldsymbol{v}$. In the sequel, the

[1]It is worth noting that the expected fraction of non-zero elements in such a vector is α. Using the Chernoff bound, it can be shown that the actual fraction of non-zero elements in a randomly chosen vector from this ensemble is tightly concentrated around its expected value (α) with high probability.

sensing matrix $\boldsymbol{G}$ is the biadjacency matrix of a weighted bigraph $\mathcal{G}(V \cup C, E, W(E))$ drawn uniformly at random from the ensemble of interest (biregular or irregular). Henceforth, we refer to the graph $\mathcal{G}$ as the *sensing graph*. Moreover, the signal vector $\boldsymbol{v}$ is drawn uniformly at random from the ensemble $\mathcal{V}_g^n(\alpha)$. The sets of signal elements and measurements are respectively mapped to the vertex sets V and C ($|V| = n$, $|C| = m$). The coefficient of the j^{th} signal element ($v_j \in V$) in the linear combination associated with the i^{th} measurement $c_i \in C$, the entry g_{ij} in $\boldsymbol{G}$, is the entry w_{ij} of the biadjacency matrix $\boldsymbol{A}(\mathcal{G})$ of $\mathcal{G}$. Figure 2.1 pictorially represents the relation between signal elements, measurements and the sensing graph.

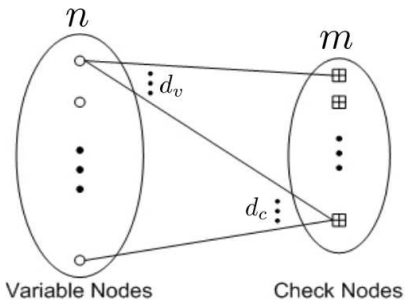


Figure 2.1: A regular bipartite graph with variable nodes on the left, check nodes on the right, variable node degree d_v and check node degree d_c. Check nodes are represented by the summation symbol indicating that the value of a check node is a linear combination of the values of variable nodes connected to it.

In the context of coding, a linear code can be represented by a bigraph, where the two sets of nodes represent the code symbols, and the linear parity-check constraints that the symbols have to satisfy [44]. Following the terminology frequently used in the context of coding, we refer to the sets V and C as the *variable nodes* and *check nodes*, respectively. We will interchangeably use the terms variable nodes and signal elements as well as check nodes and measurements.

2.3 Verification Based Algorithms and Verification Rules

Luby and Mitzenmacher [39] proposed and analyzed two iterative algorithms over bigraphs for packet-based error correction in the context of channel coding. In these algorithms, a variable node can be in one of the two states: "verified" or "unverified". Under certain circumstances, a variable node is verified and a value is assigned to it. This node then contributes to the verification of other variable nodes. The decoding process continues until either the entire set of unverified variable nodes is verified, or the process makes no further progress while there are still some unverified variables.

Due to the verification nature of the process, the two algorithms in [39] are called *verification-based* (VB) algorithms. If the assigned value to a variable node at a certain iteration is different from its true value, a *false verification* has occurred. In Section 2.4, we discuss sufficient conditions for VB algorithms so that the probability of false verification is zero.

In what follows, we describe four recovery algorithms that can be classified as VB algorithms. The first algorithm, here referred to as "Genie", is a benchmark VB algorithm in which the support set of the signal is known at the decoder. We use the Genie algorithm and its analysis to motivate and explain the analytical framework. The success threshold associated with this algorithm serves as an upper bound for the performance of other VB algorithms.[2]

In other recovery algorithms, the decoder has no information about the support set. The next two decoders considered here, are the two main VB decoding algorithms in the context of compressed sensing. The first algorithm is referred to as LM, to stand for Luby and Mitzenmacher [39]. The same algorithm is called LM1 in [22]. The second main VB algorithm is the algorithm introduced in [20], which is the same as the second algorithm discussed in [22]; LM2. We refer to this algorithm as SBB, for Sarvotham, Baron and Baraniuk. The algorithm in [2], here referred to as XH, for Xu and Hassibi, also falls into the category of VB algorithms, and can also be analyzed using the proposed framework. This algorithm serves as the fourth and last VB algorithm considered in this thesis.

In the VB algorithms, the check node values are initialized with the measurements. When a variable node is verified at an iteration, its verified value is subtracted from the value of its neighboring check nodes. The variable node, then, is removed from the sensing bigraph along with all its adjacent edges. Hence, all the neighboring check nodes of the verified variable node face a reduction in their degree. In the next iteration, some variable nodes may be verified based on the degree and/or the value of their neighboring check nodes. The rules based on which the variable nodes are verified at each iteration are called *verification rules* and are as follows:

- Zero Check Node (ZCN): If a check node has a zero value, all its neighboring variable nodes are verified with a zero value.
- Degree One Check Node (D1CN): If a check node has degree 1 in a graph, its unique neighboring variable node is verified with the value of the check node.
- Equal Check Nodes (ECN): Suppose we have N check nodes with the same non-zero value, then 1) all variable nodes neighboring a subset of these N check nodes (not all of them) are verified with the value zero; 2) if there exists a unique variable node neighboring all N check nodes, then it is verified with the common value of the check nodes.

[2]The Genie algorithm is essentially the same as the peeling algorithm over the erasure channel proposed in [45]. In particular, the two algorithms have the same threshold.

Verification rules ZCN and ECN are responsible for verifying variable nodes not in the support set. Since, the Genie algorithm has the complete knowledge of the support set, it has no need to apply these two rules. Hence, D1CN is the only rule used by the Genie. Other VB algorithms, each uses a combination of verification rules in order to verify and resolve unverified variable nodes. Assuming zero probability for false verification, the order in which the rules are applied does not affect the overall performance of the algorithm; it will only change the order in which variable nodes are verified. Verification rules adopted by different algorithms are summarized in Table 2.1.

Table 2.1: Verification rules adopted in each VB algorithm

	ZCN	D1CN	ECN
Genie	Not Needed	✓	Not Needed
LM	✓	✓	✗
SBB	✓	✓	✓
XH	✓	✗	✓

Based on Table 2.1, SBB applies the union of all rules to verify variable nodes. Therefore, this algorithm is expected to have the highest success threshold amongst the practical VB algorithms discussed here. This is verified in Section 4.8.

The ECN rule as stated above can not be easily captured in the analysis. As indicated in [40], the ECN rule can be modified to Modified Equal Check Nodes (MECN) as follows without affecting the asymptotic behavior of the recovery algorithms:

Modified Equal Check Nodes (MECN): Suppose we have N check nodes with the same non-zero value. Then if there exists a unique variable node neighbor to all such check nodes, it is verified with the common value of the check nodes. In this case, all other variable nodes connected to those check nodes are verified as zero.

2.4 False Verification

Let $\mathcal{K}$ denote the set of non-zero variable nodes in the signal; the support set. Also, let $\mathcal{V}(c)$ denote the set of variable nodes neighbor to a check node c. Now, consider the following facts:

(1) Let $\mathcal{C}$ be an arbitrary subset of check nodes. If all the check nodes in $\mathcal{C}$ are neighbor to the same subset of nodes in $\mathcal{K}$, then all these check nodes have the same value.

(2) Any check node with no neighbor in $\mathcal{K}$ has a zero value.

Verification rules ZCN and ECN in VB algorithms are designed based on the following assumptions:

(1′) Let $\mathcal{C}'$ be any arbitrary subset of check nodes with the same value. Then all these check nodes are neighbor to the same subset of $\mathcal{K}$.

(2′) None of the variable nodes neighbor to a zero valued check node belongs to the set $\mathcal{K}$.

It is worth noting that the assumptions (1′) and (2′) are the converses of the facts (1) and (2), respectively. To any choice of distributions f and g for non-zero weights of the sensing graph and non-zero signal elements, respectively, corresponds a certain probability that the converses fail to hold. Those distributions which make the converses hold true with probability 1 (almost surely), are of interest in this thesis. In the following theorem, we give an example of distributions that make the statements (1′) and (2′) hold true almost surely.

Theorem 1. *Let c_i and c_j be two distinct check nodes and $\mathcal{V}_i$ and $\mathcal{V}_j$ be their corresponding set of neighboring variable nodes in $\mathcal{K}$; i.e., $\mathcal{V}_i = \mathcal{V}(c_i) \cap \mathcal{K}$ and $\mathcal{V}_j = \mathcal{V}(c_j) \cap \mathcal{K}$. Suppose that at least one of the distributions f or g described before is continuous. Then the statements (1′) and (2′), described above, are correct with probability one for c_i and c_j.*

The continuity of f or g is a sufficient condition to have the probability of false verification equal to zero. A similar statement can be found in [20, 22, 23, 40, 41]. In the rest of the thesis, we assume that the statements (1′) and (2′) are correct with probability one and consequently, the probability of false verification for a variable node in any iteration of the VB algorithms is zero. Using the union bound, one can see that the probability of false verification in any iteration and also in the whole recovery algorithm is zero.

Chapter 3

VB Recovery Algorithms as Message-Passing Algorithms

3.1 Introduction

The verification process in VB algorithms can be seen as a message-passing procedure. In general, a variable node sends its current state (either verified or unverified) to its neighboring check nodes along with its value (if verified). A check node processes the received messages and subsequently sends some messages to its neighboring variable nodes. Each unverified variable node decides on its next state, either verified or unverified, based on the received messages from check nodes. The process of passing messages between variable nodes and check nodes continues until all variable nodes are verified, or no variable node changes its state.

In message-passing algorithms, a node can take two approaches in order to produce a message based on the set of received messages. In the first approach, the outgoing message is a function of *all* received messages. In this case, all messages leaving a node at a certain iteration are the same. In the second approach, the message passed from node a to node b in the bigraph, is a function of all the received messages by node a except the received message from node b. Therefore, the outgoing messages of a node at a certain iteration may be different, depending on the received messages. In the context of VB algorithms, the first approach is known as *node-based (NB)*, while the second approach is called *message-based (MB)* [1,22].[1] So, for a NB-VB algorithm, the state of a variable node is reported identically by all its outgoing messages, while in an MB-VB algorithm, different states may be reported by different outgoing messages of a variable node.

As noted in [1], the authors in [39] defined the two VB algorithms using the NB representation but analyzed them using the MB representation. In [1], the authors proved that for one of the VB algorithms, the NB and MB versions perform the same, but for the other VB algorithm, the NB version outperforms the MB one. In

[1] In the context of iterative decoding algorithms, NB and MB approaches are known as non-extrinsic and extrinsic message-passing, respectively [46].

compressed sensing, this implies that NB versions, in general, have higher success thresholds; i.e., can successfully recover signals with larger density factors [23].

A well-known method to analyze iterative message-passing algorithms in coding theory is density evolution [43]. In density evolution, the distribution of messages is tracked with the iteration number. The evolution of the message distributions with iterations will then reveal important properties of the decoding algorithm such as *decoding threshold* and *convergence speed* [43]. The derivation of the message distribution however, requires the independence among the incoming messages to a node. The analysis is thus only applicable to extrinsic message-passing algorithms (MB decoders). To analyze NB algorithms, Zhang and Pfister [1] derived a system of coupled differential equations. For (d_v, d_c) regular bigraphs, the number of differential equations are $O(d_v + d_c^2)$ and $O(d_v^3 + d_c^2)$ for the LM1-NB and LM2-NB algorithms, respectively. It is difficult, if not impossible, to find a closed form solution for the system of differential equations even for small values of d_v and d_c.[2] Numerical methods were thus used in [1] to solve the system of differential equations and consequently evaluate the performance of NB algorithms. This process however is susceptible to numerical errors. In addition, it is hard to know how the obtained threshold for a given finite value of n compares with the exact threshold, which applies in the asymptotic regime of $n \to \infty$. In practice, the numerical results are highly dependent on the selected, large but still finite, value of n.

In our work, we analyze NB-VB algorithms in the regime where $n \to \infty$. Our analysis has an iterative fashion, as will be clear in Chapters 4 and 5, with low-complexity update rules, which are far less complex than the one used in [1]. Moreover, our proposed analysis is applicable to parallel versions of the algorithms and leads to $\mathcal{O}(d_v + d_c^3)$ calculations per iteration.

3.2 Definitions and Setup

Each NB-VB algorithm works in iterations through exchanging messages between the check nodes and the variable nodes along the edges in the graph. Any message sent from a variable node to its neighboring check nodes belongs to an alphabet set $\mathcal{M} : \{0, 1\} \times \mathbb{R}$. The first coordinate of such a message is a status flag, sometimes referred to as "recovery flag", taking binary values. The flag indicates the verification status of the variable node. If this flag is 0, then the variable node is not verified. If, on the other hand, the flag is 1, then the variable node has been verified. In this case, the second coordinate, which is a real number, is interpreted as the verified value of the variable node.

Similarly, any message sent from a check node to all its neighboring variable nodes belongs to an alphabet set $\mathcal{O} : \mathbb{Z}^+ \times \mathbb{R}$. The first coordinate of such a message indicates the number of unverified variable nodes neighbor to the check node. The first coordinate is in fact the *degree* of the check node in the subgraph induced by

[2]The number of differential equations for LM1-NB over a $(3, 6)$ bigraph is 30.

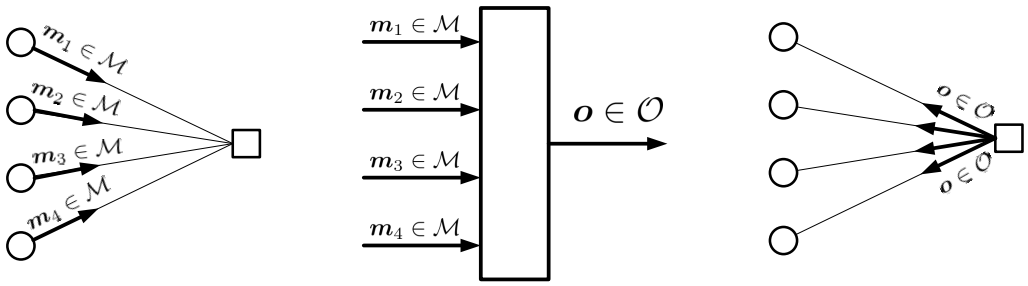
the unverified variable nodes. The second coordinate indicates the current value of the check node, i.e., the result of the linear combination of the unverified neighboring variable nodes.

The edges, in NB-MP algorithms, do not simply forward messages from check nodes to variable nodes and vice versa. Instead, based on the traveling direction of the message, edges multiply or divide the second coordinate of the message by their associated weight. More specifically, if the message is sent from a variable node to a check node, its second coordinate is multiplied by the weight. The second coordinate of the message is divided by the weight, if the message is sent from a check node to a variable node. So, although messages generated by a node (either variable node or check node) are sent identically over all adjacent edges, the fact that the edges may have different weights will result in different messages being received at the destination nodes. All such messages are independent if the weights associated with the corresponding edges are independent.

Any iteration $\ell \geq 1$ in NB-VB algorithms, consists of two rounds, denoted by R1 and R2, each with two half-rounds, denoted by HR1 and HR2. In the R1-HR1 and R2-HR1, every check node processes all its received messages from the previous round together with its associated measurement and sends out a message from the alphabet $\mathcal{O}$ to all its neighboring variable nodes. In the R1-HR2 and R2-HR2, each (unverified) variable node decides on its next state by processing all its received messages. Whatever the decision, the variable node sends back a message, from the alphabet $\mathcal{M}$, to all its neighboring check nodes. So, a round starts with check nodes processing the received messages from neighboring variable nodes, proceeds with the transmission of messages from check nodes to variable nodes, continues by variable nodes processing the received messages from neighboring check nodes, and ends with the transmission of messages from variable nodes to check nodes. The two rounds in each iteration follow the same general structure. They only differ in the processing carried out in the variable nodes.

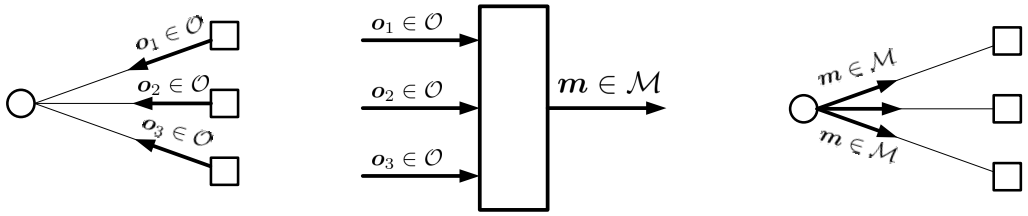
In Figures 3.1 and 3.2, we have shown the snap shots of message passing between a variable node of degree 3 and a check node of degree 4. The snap shots represent the two half-rounds (HR1 and HR2) of a generic round of recovery.

Let $\Phi_v^{(1,\ell)} : \mathcal{O}^{d_v} \to \mathcal{M}$ and $\Phi_v^{(2,\ell)} : \mathcal{O}^{d_v} \to \mathcal{M}$, $\ell \in \mathbb{N}$, represent the mappings used at any *unverified* variable node of degree d_v to map the incoming messages to the outgoing message in the first and the second round of iteration ℓ, respectively. Obviously, due to the verification-based nature of the algorithms, when a variable node becomes verified at an iteration, its outgoing message remains unchanged, irrespective of its incoming messages. In contrast to the variable nodes, the mapping function used in check nodes is identical for both the first and the second round of each iteration. Every check node has an associated received measurement; a random variable taking values in $\mathbb{R}$. So, we use the notation $\Phi_c^{(\ell)} : \mathbb{R} \times \mathcal{M}^{d_c} \to \mathcal{O}$, $\ell \in \mathbb{N}$, to denote the mapping function used in all check nodes of degree d_c at iteration ℓ. For



(a) Variable Node to Check Node (b) Check Node Processing (c) Check Node to Variable Node

Figure 3.1: Message-Passing in HR1 of a typical round



(a) Check Node to Variable Node (b) Variable Node Processing (c) Variable Node to Check Node

Figure 3.2: Message-Passing in HR2 of a typical round

the sake of completeness, let $\Phi_v^{(0)} = \Phi_v^{(2,0)} : \mathcal{O}^{d_v} \to \mathcal{M}$ and $\Phi_c^{(0)} : \mathbb{R} \to \mathcal{O}$ represent the mappings used, respectively in all variable nodes of degree d_v and all check nodes at iteration 0. This iteration consists of only one round. For the VB algorithms under consideration, the mapping functions in the variable nodes and check nodes are not a function of the iteration number. Therefore, we omit the superscript (ℓ) henceforth.

In what follows, we describe VB algorithms of Section 2.3 as message-passing algorithms with the general structure explained above.[3]

[3]It is worth mentioning that the message-passing description of SBB and XH algorithms (in particular, $\Phi_v^{(1)}$), presented in Sections 3.3.3 and 3.3.4, are only valid for the cases in which the non-zero weights of the sensing graph are drawn from an uncountable or countably infinite alphabet set. If the elements of the sensing matrix are drawn from a finite alphabet set, such as binary 0 and 1, the outgoing messages from a check node should also include the list of all unverified variable nodes neighbor to the check node. The mapping function in the variable nodes should also change in order to use the extra information in the incoming messages. Please also see Footnote 5 in Section 3.3.3.

3.3 Message-Passing Description of Recovery Algorithms

To describe the four VB recovery algorithms using the message-passing approach, we need to define the mappings $\Phi_v^{(1)}$, $\Phi_v^{(2)}$ and Φ_c. Mapping $\Phi_v^{(1)}$ embeds the verification rules D1CN and ECN, while the mapping $\Phi_v^{(2)}$ embeds the ZCN rule. To make the description of mappings Φ_v and Φ_c simpler, we introduce some notations to represent the incoming messages to variable and check nodes from the alphabet sets $\mathcal{O}$ and $\mathcal{M}$, respectively. A message $\boldsymbol{o} \in \mathcal{O}$, incoming to a variable node, is an ordered pair of elements (d, ξ), where $d \in \mathbb{Z}^+, \xi \in \mathbb{R}$. A message $\boldsymbol{m} \in \mathcal{M}$, incoming to a check node, is an ordered pair of elements (s, ω), where $s \in \{0, 1\}, \omega \in \mathbb{R}$. Moreover, we assume that there is an arbitrary numbering for edges adjacent to a node (either variable node or check node). So, we use the notations $\boldsymbol{o}_i, 1 \leq i \leq d_v$, and $\boldsymbol{m}_j, 1 \leq j \leq d_c$, to denote the incoming messages to a variable node of degree d_v and a check node of degree d_c, respectively. Henceforth, to simplify the notation, we describe the mapping rules assuming a variable node of degree d_v and a check node of degree d_c.

At iteration zero, all variable nodes are unverified and there is no received message at the check nodes. At this stage, all check nodes send their corresponding measurements along with their degree to their neighboring variable nodes. For the following iterations $\ell \geq 1$, the mapping function at any check node c_i (of degree d_c) is as follows:

$$\Phi_c(c_i, \boldsymbol{m}_1, \cdots, \boldsymbol{m}_{d_c}) = (d_c - \sum_{i=1}^{d_c} s_i, c_i - \sum_{i=1}^{d_c} s_i \omega_i),$$

where, c_i in the above equation is the measurement associated with the check node c_i, and $\boldsymbol{m}_i = (s_i, \omega_i)$ is the message received along the ith edge. The mapping functions $\Phi_v^{(1)}$, $\Phi_v^{(2)}$ are algorithm dependent and are discussed for each VB algorithm separately next.

The decoder stops at an iteration ℓ, $\ell \geq 1$, if the algorithm makes no further progress, i.e., the set of verified variable nodes are the same for the two consecutive iterations $\ell - 1$ and ℓ. Equivalently, the algorithm stops if the messages sent from variable nodes to check nodes, and also from check nodes to variable nodes, are the same for two consecutive iterations ℓ and $\ell - 1$. At this point, if the decoder is able to verify all the variable nodes, then the decoding is called successful. Otherwise, the decoder will declare a failure.

3.3.1 Genie

In this algorithm, each iteration consists of only one round, in which one verification rule (D1CN) is applied to all variable nodes. For variable nodes not in the support set, the outgoing message in all iterations is fixed and equals $\boldsymbol{m} = (1, 0)$.

For any variable node (of degree d_v) in the support set, the mapping

$\Phi_v(\boldsymbol{o}_1, \cdots, \boldsymbol{o}_{d_v})$ is defined based on the following rules.

- Rule 1: If among all received messages (from the neighboring check nodes), there exists only one message, say $\boldsymbol{o}_i, i \in [d_c]$, such that $\boldsymbol{o}_i = (1, \xi_i), \xi_i \in \mathbb{R}$, then $\Phi_v(\boldsymbol{o}_1, \cdots, \boldsymbol{o}_{d_v}) = (1, \xi_i)$. In this case, the variable node is verified with the value ξ_i.
- Rule 2: If multiple messages exist in the form $(1, \xi)$ (any $\xi \in \mathbb{R}$), then choose one at random, say $\boldsymbol{o}_i = (1, \xi_i), \xi_i \in \mathbb{R}$, and set $\Phi_v(\boldsymbol{o}_1, \cdots, \boldsymbol{o}_{d_v}) = (1, \xi_i)$. In this case, the variable node is verified with the value ξ_i.
- Rule 3: If none of the above happens, then $\Phi_v(\boldsymbol{o}_1, \cdots, \boldsymbol{o}_{d_v}) = (0, 0)$. In this case, the variable node is still unverified.

For any unverified variable node, the mappings $\Phi_v^{(1)}$ and $\Phi_v^{(2)}$ are defined according to the following sets of rules, for LM, SBB and XH, respectively.

3.3.2 LM

- $\Phi_v^{(1)}(\boldsymbol{o}_1, \cdots, \boldsymbol{o}_{d_v})$: Apply Rules 1 - 3 of Genie.
- $\Phi_v^{(2)}(\boldsymbol{o}_1, \cdots, \boldsymbol{o}_{d_v})$:
  - Rule 4: If there exists at least one message $\boldsymbol{o}_i$ such that $\boldsymbol{o}_i = (d_i, 0)$ (for any $d_i \in \mathbb{Z}^+$), then $\Phi_v^{(2)}(\boldsymbol{o}_1, \cdots, \boldsymbol{o}_{d_v}) = (1, 0)$. In this case, the variable node is verified with the value equal to 0.
  - Rule 5: If no incoming message exists in the form $(d_i, 0)$ (for any $d_i \in \mathbb{Z}^+$), then $\Phi_v^{(2)}(\boldsymbol{o}_1, \cdots, \boldsymbol{o}_{d_v}) = (0, 0)$. In this case, the variable node is still unverified.

3.3.3 SBB

- $\Phi_v^{(1)}(\boldsymbol{o}_1, \cdots, \boldsymbol{o}_{d_v})$:[4]
  - Apply Rule 1 of Genie.
  - If there exist N messages $(2 \leq N \leq d_v)$ $\boldsymbol{o}_{i_1} = (d_{i_1}, \xi_{i_1})$, $\boldsymbol{o}_{i_2} = (d_{i_2}, \xi_{i_2})$, $\cdots$, $\boldsymbol{o}_{i_N} = (d_{i_N}, \xi_{i_N})$, such that $\xi_{i_1} = \xi_{i_2} = \cdots = \xi_{i_N}$, then $\Phi_v^{(1)}(\boldsymbol{o}_1, \cdots, \boldsymbol{o}_{d_v}) =$

[4]The original recovery algorithm introduced in [20] has a computational complexity of $O(m \cdot \log m)$ (which translates to $O(n \cdot \log n)$ for biregular graphs of fixed degrees). It is easy to prove that the message-passing description provided here does not change the recovery capability of the algorithm but results in the reduction of decoding complexity from $O(n \cdot \log n)$ to $O(n)$.

$(1, \xi_{i_1})$. In this case, the variable node is verified with the common value ξ_{i_1}.[5]

- If a variable node is verified to different values according to verification rules above, then choose one at random and generate the outgoing message accordingly.
- Apply Rule 3 of Genie.

- $\Phi_v^{(2)}(\boldsymbol{o}_1, \cdots, \boldsymbol{o}_{d_v})$: Apply Rules 4 and 5 of LM.

3.3.4 XH

- $\Phi_v^{(1)}(\boldsymbol{o}_1, \cdots, \boldsymbol{o}_{d_v})$:
  - If there exist M messages ($\lceil d_v/2 \rceil \leq M \leq d_v$) $\boldsymbol{o}_{i_1} = (d_{i_1}, \xi_{i_1})$, $\boldsymbol{o}_{i_2} = (d_{i_2}, \xi_{i_2})$, $\cdots$, $\boldsymbol{o}_{i_M} = (d_{i_M}, \xi_{i_M})$, such that $\xi_{i_1} = \xi_{i_2} = \cdots = \xi_{i_M}$, then $\Phi_v^{(1)}(\boldsymbol{o}_1, \cdots, \boldsymbol{o}_{d_v}) = (1, \xi_{i_1})$. In this case, the variable node is verified with the common value ξ_{i_1}.
  - If a variable node is verified to different values according to the verification rule above, i.e., if two groups of messages both at least of size $d_v/2$ satisfy the above condition, then choose one at random and generate the outgoing message accordingly.
  - Apply Rule 3 of Genie.

- $\Phi_v^{(2)}(\boldsymbol{o}_1, \cdots, \boldsymbol{o}_{d_v})$: Apply Rules 4 and 5 of LM.

3.4 A Short Note on False Verification

In the above description of recovery algorithms, there may be cases where a variable node can be verified to different values by different rules. Using the same assumption made in Section 2.4, it is easy to see that the probability of this event is equal to zero. In such cases, we have thus assumed that the variable node is verified by one of the rules selected randomly. Clearly, the probability of false verification as a result of such selections is zero.

[5] We note that the message received by the variable node equals the message sent from the check node divided by the weight of the connecting edge. Therefore, receiving N messages with the same value would imply that, almost surely, the unverified variable node under consideration is the unique non-zero variable node neighbor to the N check nodes. Other unverified variable nodes neighbor to these N check nodes do not belong to the support set and should be verified with a value equal to zero. This, however, happens in the next round.

Chapter 4

Analysis of NB-VB Algorithms over Regular Graphs

4.1 Background

In this chapter, we first show that (i) the performance of a realization of the sensing graph, with a certain selection of the edge weights for the recovery of a realization of the input signal concentrates around the average performance of the ensemble (where the average is taken over all the elements in the ensemble $\mathcal{G}_f^n(d_v, d_c) \times \mathcal{V}_g^n(\alpha)$, for given probability distribution functions f, g, and given constant parameters d_v, d_c and α), as n tends to infinity, and (ii) the average performance of the ensemble, as n goes to infinity, converges to the performance of the cycle-free case defined as follows.

Let $\mathcal{N}_v^{2\ell}$ be the neighborhood of node v of depth 2ℓ, i.e., the subgraph consisting of the variable node v and all those nodes that are connected to v with any path of length less than or equal to 2ℓ. We say that we are working under the cycle-free assumption when for a fixed ℓ, and for every v, $\mathcal{N}_v^{2\ell}$ is tree-like.[1]

Following the concentration results, we first present the asymptotic analysis of the Genie algorithm.[2] The analysis provides us with the average ensemble performance for the asymptotic case of $n \to \infty$. We then generalize the concepts used in the analysis of Genie and analyze XH, LM and SBB algorithms.

[1] These concentration results were established as a joint work with Anoosheh Heidarzadeh, another Ph.D. student, and are presented here for the sake of completeness. The author has contributed in the derivation of the upper bound presented in Equation (4.1).

[2] An analysis of the Genie algorithm (peeling decoder) over the erasure channel is given in [45]. Unlike the approach adopted in this work, the analysis of [45] is based on modeling the progress of iterative decoding with a set of differential equations.

4.2 Concentration Results and Convergence to Cycle-Free Case

Consider a weighted graph selected at random from $\mathcal{G}_f^n(d_v, d_c)$. Also consider an input signal vector $\boldsymbol{v}$ chosen randomly from $\mathcal{V}_g^n(\alpha)$. Suppose that a VB algorithm is applied to the measurement vector $\boldsymbol{c} = \boldsymbol{G}\boldsymbol{v}$ to recover $\boldsymbol{v}$ iteratively, where $\boldsymbol{G}$ is the biadjacency matrix of the chosen weighted graph. For this scenario, let $\beta^{(\ell)}$ $(:= \beta^{(\ell)}(\boldsymbol{G}, \boldsymbol{w}, \boldsymbol{v}))$ be the fraction of unverified non-zero variable nodes at the beginning of iteration ℓ, i.e., the fraction of variable to check node messages passed along the edges of the chosen weighted graph with unverified status (sent by non-zero variable nodes); further, let $\mathbf{E}[\beta^{(\ell)}]$ denote the expected value of $\beta^{(\ell)}$, where the expectation is taken over the ensembles $\mathcal{G}_f^n(d_v, d_c)$ and $\mathcal{V}_g^n(\alpha)$. Now, consider the corresponding cycle-free case, and let $\alpha^{(\ell)}$ be the expected number of messages with unverified status passed along an edge emanating from a non-zero variable node with a tree-like neighborhood of depth at least 2ℓ at the ℓth iteration. Here, again, the expectation is taken over the ensembles of input signals and weighted graphs.

In the subsequent subsection, we will show how $\alpha^{(\ell)}$ can be calculated. It should be clear that $\alpha^{(\ell)}$, being defined as the "average" over the ensemble of weighted graphs and input vectors, is the same as the "probability" that a message from a non-zero variable node with a tree-like neighborhood of depth at least 2ℓ, at the ℓth iteration, carries an unverified status. In this section, we use the interpretation of $\alpha^{(\ell)}$ as an average. The interpretation of $\alpha^{(\ell)}$ as a probability will be used in the analysis section. In the following, we will show that over all realizations, with high probability, $\beta^{(\ell)}$ does not deviate much from $\mathbf{E}[\beta^{(\ell)}]$, and $\mathbf{E}[\beta^{(\ell)}]$, itself, is not far from $\alpha^{(\ell)}$, as n tends to infinity.

Theorem 2. *Over the probability space of all weighted graphs $\mathcal{G}_f^n(d_v, d_c)$, and all signal inputs $\mathcal{V}_g^n(\alpha)$, for a fixed ℓ, letting $\beta^{(\ell)}$ and $\alpha^{(\ell)}$ be defined as above, for each of the NB-VB algorithms discussed in this paper, there exist positive constants $\mu(d_v, d_c, \ell)$ and $\gamma(d_v, d_c, \ell)$, such that (i) for any $\epsilon > 0$,*

$$\Pr\left[\left|\beta^{(\ell)} - \mathbf{E}[\beta^{(\ell)}]\right| > \epsilon/2\right] \leq 2e^{-\epsilon^2 n/\mu}, \tag{4.1}$$

and (ii) for any $\epsilon > 0$, and $n > 2\gamma/\epsilon$,

$$\left|\mathbf{E}[\beta^{(\ell)}] - \alpha^{(\ell)}\right| < \epsilon/2. \tag{4.2}$$

Note that combining (4.1) and (4.2), the following holds: for any $\epsilon > 0$, and $n > 2\gamma/\epsilon$,

$$\Pr\left[\left|\beta^{(\ell)} - \alpha^{(\ell)}\right| > \epsilon\right] \leq 2e^{-\epsilon^2 n/\mu}.$$

Our method of proof for Theorem 2 is similar to that of [43], though due to the differences in the nature of the problems (channel coding vs. compressed sensing) and the difference in the update equations at the graph nodes, some arguments are

revised and some new components are added to the proof. We refer the reader to [47] for the details of the proof.

4.3 Analysis of the Genie

In the Genie algorithm, the support set is known. Therefore, the set of all variable nodes can be partitioned into two sets: verified $\mathcal{R}$ and unverified $\mathcal{K}$. At iteration zero, variable nodes in the support set are unverified, and the zero-valued variable nodes belong to the verified set. In future iterations, during the verification process, variable nodes are moved from set $\mathcal{K}$ to set $\mathcal{R}$. We use notations $\mathcal{K}^{(\ell)}$ and $\mathcal{R}^{(\ell)}$ to denote the set of unverified and verified variable nodes at (the beginning of) iteration ℓ, respectively. We also use the superscript ℓ to indicate the iteration number for all the other sets in this section in the same way. Our goal in the analysis is to track the evolution of the subgraph induced by the variable nodes in $\mathcal{K}^{(\ell)}$. This is the subgraph that is actively involved in the message-passing process. In the following, when we refer to the degree of a check node, we implicitly mean the degree of the check node in this subgraph. To make this explicit, we may use the term $\mathcal{K}$-degree. Similarly, the term $\mathcal{R}$-degree may be used. Clearly, the sum of the $\mathcal{K}$-degree and the $\mathcal{R}$-degree of each check node is d_c.

Each iteration of the Genie algorithm consists of only one round (two half-rounds, HR1 and HR2). At a generic iteration ℓ, in the HR1, check nodes process the received messages from variable nodes sent at iteration $\ell-1$ and generate outgoing messages to be delivered to variable nodes. We partition the check nodes based on their $\mathcal{K}$-degree. The set of check nodes with $\mathcal{K}$-degree i *after* the (processing in the) HR1 of iteration ℓ, is represented by $\mathcal{N}_i^{(\ell,1)}$. A check node with $\mathcal{K}$-degree j before HR1 may have a degree $i \leq j$ after HR1. In HR2, the variable nodes process the incoming messages and generate outgoing messages accordingly. Variable nodes, are also partitioned based on the number of neighboring check nodes of $\mathcal{K}$-degree 1. The unverified variable nodes with i neighboring check nodes of $\mathcal{K}$-degree 1 *after* the (processing in the) HR2 of iteration ℓ are represented by $\mathcal{K}_i^{(\ell,2)}$. Note that the grouping of check nodes remains unchanged during the HR2 of the same iteration. In a similar way, the grouping of variable nodes remains unchanged during the HR1 of the next iteration.

In the HR1 of iteration zero, every check node sends its corresponding measurement value along with its degree, d_c. In HR2, variable nodes in $\mathcal{R}^{(0)}$ return a verified message with a value equal to 0, while variable nodes in $\mathcal{K}^{(0)}$ return a message with the status bit equal to zero. At iteration 0, the set $\mathcal{K}_0^{(0,2)}$ includes all unverified variable nodes $\mathcal{K}^{(0)}$.

In the HR1 of iteration 1, an outgoing message of a check node has the following two properties: 1) the second coordinate of the message is still equal to the measurement value for the check node since no variable node from the support set was verified at iteration 0, and 2) the first coordinate of the message, which is the $\mathcal{K}$-degree of the check node, is less than or equal to d_c since the variable nodes not in the support

set ($\mathcal{R}^{(1)} = \mathcal{R}^{(0)}$) have been revealed at iteration 0, thus reducing the number of unverified variable nodes connected to the check nodes. We use the notation $\mathcal{N}_{i\downarrow j}^{(\ell)}$ to refer to the subset of check nodes $\mathcal{N}_i^{(\ell-1,1)}$ that are moved to $\mathcal{N}_j^{(\ell,1)}$. The arrow points downward in the notation to emphasize that $j \leq i$. Note that for iteration 1, $i = d_c$.

In the HR2 of iteration 1, after receiving the messages from check nodes, variable nodes in $\mathcal{K}_0^{(0,2)}$ are partitioned into the sets $\mathcal{K}_j^{(1,2)}, 0 \leq j \leq d_v$. We denote by $\mathcal{K}_{i\uparrow j}^{(\ell)}$ the set of variable nodes in $\mathcal{K}_i^{(\ell-1,2)}$ joining the set $\mathcal{K}_j^{(\ell,2)}$. In this case, $j \geq i$, hence the use of the arrow pointing up. Based on the verification rule for the Genie, at any iteration ℓ if an unverified variable node is neighbor to at least one check node in the set $\mathcal{N}_1^{(\ell,1)}$, it will be verified. So, variable nodes in the set $\bigcup_{j=1}^{d_v} \mathcal{K}_j^{(1,2)}$ are verified at the end of iteration 1. Therefore, the new sets $\mathcal{R}^{(2)}$ and $\mathcal{K}^{(2)}$ to be used at iteration 2 are calculated as follows.

$$\mathcal{R}^{(2)} = \bigcup_{j=1}^{d_v} \mathcal{K}_j^{(1,2)} \cup \mathcal{R}^{(1)}, \quad \mathcal{K}^{(2)} = \mathcal{K}_0^{(1,2)}.$$

The message passing and verification processes continue in next iterations in the same fashion discussed above. In summary, in a generic iteration ℓ, we have the following relationships:

$$\mathcal{N}_i^{(\ell-1,1)} = \bigcup_{j=0}^{i} \mathcal{N}_{i\downarrow j}^{(\ell)}, \quad 1 \leq i \leq, d_c, \qquad \mathcal{N}_{1\downarrow 1}^{(\ell)} = \emptyset, \qquad \mathcal{N}_j^{(\ell,1)} = \bigcup_{i=j}^{d_c} \mathcal{N}_{i\downarrow j}^{(\ell)}, \quad 0 \leq j \leq, d_c,$$

$$\mathcal{R}^{(\ell+1)} = \bigcup_{j=1}^{d_v} \mathcal{K}_j^{(\ell,2)} \cup \mathcal{R}^{(\ell)}, \quad \mathcal{K}^{(\ell+1)} = \mathcal{K}_0^{(\ell,2)} = \bigcup_{j=0}^{d_v} \mathcal{K}_{0\uparrow j}^{(\ell-1)}, \quad \mathcal{K}_j^{(\ell,2)} = \mathcal{K}_{0\uparrow j}^{(\ell)}, \quad 0 \leq j \leq d_v.$$

By tracking the set $\mathcal{K}^{(\ell)}$ with iterations, we can decide on the success or failure of the algorithm. If the size of the set $\mathcal{K}^{(\ell)}$ shrinks to zero as $\ell \to \infty$, then the algorithm is successful. On the other hand, if there exists an $\epsilon > 0$ such that $|\mathcal{K}^{(\ell)}| \geq \epsilon$, $\forall \ell \geq 1$, then the algorithm fails. The success or failure of the algorithm depends on the parameters of the graph (d_v and d_c) as well as the initial size of the support set $|\mathcal{K}^{(0)}|$.

Based on the concentration results, to analyze the Genie in the asymptotic case, we track the probability $\alpha^{(\ell)}$ that a variable node belongs to the set $\mathcal{K}^{(\ell)}$. Hence, we focus on a tree-like graph with random weights and a random input signal. Let $p_{\mathcal{N}_i}^{(\ell,1)}, 0 \leq i \leq d_c$, denote the probability that a check node belongs to the set $\mathcal{N}_i^{(\ell,1)}$. Furthermore, let $p_{\mathcal{K}_j}^{(\ell,2)}, 0 \leq j \leq d_v$, denote the probability that an unverified (non-zero) variable node belongs to the set $\mathcal{K}_j^{(\ell,2)}$. In Table 4.1, we have summarized the terminologies used in the analysis of Genie, and in Table 4.2, we have presented the step-by-step procedure to update the probabilities $p_{\mathcal{N}_i}^{(\ell,1)}$, $p_{\mathcal{K}_j}^{(\ell,2)}$, and $\alpha^{(\ell)}$, for $\ell \geq 1$, in terms of probabilities $\alpha^{(\ell-1)}$, $p_{\mathcal{N}_i}^{(\ell-1,1)}$, and $p_{\mathcal{K}_j}^{(\ell-1,2)}$. The derivation details can be found

Table 4.1: Probabilities Involved in the Analysis of Genie in Table 4.2. Each Entry of the Table is the Probability of the Corresponding Event.

Probability	**Definition**
$\alpha^{(\ell)}$	variable nodes is unverified at the beginning of iteration ℓ.
$p^{(\ell)}_{\mathcal{N}_{j\downarrow i}}$	check node has $\mathcal{K}$-degree j before and $\mathcal{K}$-degree i, $i \leq j$, after HR1 of iteration ℓ.
$p^{(\ell,1)}_{\mathcal{N}_i}$	check node has $\mathcal{K}$-degree i after the HR1 of iteration ℓ.
$p^{(\ell,2)}_{\mathcal{K}_i}$	unverified variable node has i neighboring check nodes of $\mathcal{K}$-degree 1 after HR2 of iteration ℓ.
$p^{(\ell+1)}$	edge connected to a variable node in $\mathcal{K}^{(\ell)}$ is also connected to a check node in $\mathcal{N}_1^{(\ell,1)}$.
$A^{(\ell)}$	edge connecting a check node in the set $\mathcal{N}_j^{(\ell-1,1)}$, $j \geq 2$, and an unverified variable node, carries a verified message to the check node in HR1 of iteration ℓ.

in Part A.2 of the appendix.

4.4 Framework for the Analysis of XH

The analysis of the Genie and XH algorithms are very similar and differ in a few update rules. The XH algorithm consists of two rounds. However, since the D1CN verification rule is not present, it suffices to track the evolution of the support set in the analysis. From this perspective, the analysis of the Genie and XH algorithms are the same. The only difference lies in the set of variable nodes verified in each iteration. The set of verified variable nodes at iteration ℓ for the Genie algorithm follows

$$\mathcal{R}^{(\ell+1)} = \bigcup_{j=1}^{d_v} \mathcal{K}_j^{(\ell,2)} \cup \mathcal{R}^{(\ell)}.$$

For the XH algorithm, this set is:

$$\mathcal{R}^{(\ell+1)} = \bigcup_{j=\lceil d_v/2 \rceil}^{d_v} \mathcal{K}_j^{(\ell,2)} \cup \mathcal{R}^{(\ell)}.$$

Table 4.2: Density Evolution Analysis of Genie

	Inputs: $d_v,\ d_c,\ \alpha^{(0)}$
	Initialization
	$p_{\mathcal{N}_i}^{(1,1)} = \binom{d_c}{i} \left(\alpha^{(0)}\right)^i \left(1-\alpha^{(0)}\right)^{d_c-i}, \quad 0 \le i \le d_c$
	$\alpha^{(1)} = \alpha^{(0)}$
	$p^{(2)} = \left(1-\alpha^{(1)}\right)^{d_c-1}$
	$p_{\mathcal{K}_0}^{(1,2)} = \left(1-p^{(2)}\right)^{d_v}$
	Recursive Formulas for $\ell \ge 2$
1)	$A^{(\ell)} = 1-\left(1-p^{(\ell)}\right)^{d_v-1}$
2)	$p_{\mathcal{N}_{1\downarrow 0}}^{(\ell)} = 1,\ p_{\mathcal{N}_{1\downarrow 1}}^{(\ell)} = 0,\ p_{\mathcal{N}_{0\downarrow 0}}^{(\ell)} = 1$
	$p_{\mathcal{N}_{j\downarrow i}}^{(\ell)} = \binom{j}{j-i} \left(A^{(\ell)}\right)^{j-i} \left(1-A^{(\ell)}\right)^i, \quad 2 \le j \le d_c, \quad 0 \le i \le j$
3)	$p_{\mathcal{N}_i}^{(\ell,1)} = \sum_{j=i}^{d_c} p_{\mathcal{N}_j}^{(\ell-1,1)} p_{\mathcal{N}_{j\downarrow i}}^{(\ell)}, \quad 0 \le i \le d_c$
4)	$\alpha^{(\ell)} = \alpha^{(\ell-1)} p_{\mathcal{K}_0}^{(\ell-1,2)}$
5)	$p^{(\ell+1)} = \dfrac{p_{\mathcal{N}_1}^{(\ell,1)}}{\alpha^{(\ell)} d_c}$
6)	$p_{\mathcal{K}_j}^{(\ell,2)} = \binom{d_v}{j} \left(p^{(\ell+1)}\right)^j \left(1-p^{(\ell+1)}\right)^{d_v-j}, \quad 0 \le j \le d_v$

Details of the analysis can be found in Section A.3. The summary of the update equations are listed in Table 4.3.

4.5 General Framework for the Analysis of LM and SBB

In LM and SBB, at the beginning of any iteration ℓ, the set of all variable nodes is partitioned into three sets: $\mathcal{K}^{(\ell)}$, $\mathcal{R}^{(\ell)}$, and $\Delta^{(\ell)}$. The set $\mathcal{K}^{(\ell)}$ consists of all unverified non-zero variable nodes, while the set $\Delta^{(\ell)}$ consists of all unverified zero-valued variable nodes. The set $\mathcal{R}^{(\ell)}$ includes all the verified variable nodes. Clearly, the decoder can not make the distinction between variable nodes in the sets $\mathcal{K}^{(\ell)}$ and $\Delta^{(\ell)}$. The distinction between the two sets, however, is needed for the analysis.

Furthermore, at any iteration ℓ, we partition the set of all check nodes into subsets $\mathcal{N}_{i,j}^{(\ell)}$. The index i indicates the number of neighboring variable nodes in the set $\mathcal{K}^{(\ell)}$ while the index j indicates the number of neighboring variable nodes in the set $\Delta^{(\ell)}$. This is shown in Fig. 4.1. The two indices are referred to as $\mathcal{K}$- and Δ-degrees of the check nodes, respectively. Note that: i) the degree of each check node in the subgraph

Table 4.3: Density Evolution Analysis of XH

Inputs:	d_v, d_c, $\alpha^{(0)}$
Initialization	
	$p_{\mathcal{N}_i}^{(1,1)} = \binom{d_c}{i} \left(\alpha^{(0)}\right)^i \left(1-\alpha^{(0)}\right)^{d_c-i}, \quad 0 \le i \le d_c$
	$\alpha^{(1)} = \alpha^{(0)}$
	$p^{(2)} = \left(1-\alpha^{(1)}\right)^{d_c-1}$
	$p_{\mathcal{K}_0}^{(1,2)} = \left(1-p^{(2)}\right)^{d_v}$
Recursive Formulas for $\ell \ge 2$	
1)	$A^{(\ell)} = 1 - \sum_{i=0}^{\lceil d_v/2 \rceil - 1} \frac{p_{\mathcal{K}_i}^{(\ell-1,2)}}{1-p^{(\ell)}}$
2)	$p_{\mathcal{N}_{1\downarrow 0}}^{(\ell)} = \sum_{i=\lceil d_v/2 \rceil}^{d_v} \frac{i p_{\mathcal{K}_i}^{(\ell-1,2)}}{\sum_{i=0}^{d_v} i p_{\mathcal{K}_i}^{(\ell-1,2)}}$, $p_{\mathcal{N}_{1\downarrow 1}}^{(\ell)} = 1 - p_{\mathcal{N}_{1\downarrow 0}}^{(\ell)}$, $p_{\mathcal{N}_{0\downarrow 0}}^{(\ell)} = 1$
	$p_{\mathcal{N}_{j\downarrow i}}^{(\ell)} = \binom{j}{j-i} \left(A^{(\ell)}\right)^{j-i} \left(1-A^{(\ell)}\right)^i, \quad 2 \le j \le d_c, \quad 0 \le i \le j$
3)	$p_{\mathcal{N}_i}^{(\ell,1)} = \sum_{j=i}^{d_c} p_{\mathcal{N}_j}^{(\ell-1,1)} p_{\mathcal{N}_{j\downarrow i}}^{(\ell)}, \quad 0 \le i \le d_c$
4)	$\alpha^{(\ell)} = \alpha^{(\ell-1)} \left(1 - \sum_{i=\lceil d_v/2 \rceil}^{d_v} p_{\mathcal{K}_i}^{(\ell,2)}\right)$.
5)	$p^{(\ell+1)} = \frac{p_{\mathcal{N}_1}^{(\ell,1)}}{\alpha^{(\ell)} d_c}$
6)	$p_{\mathcal{K}_j}^{(\ell,2)} = \binom{d_v}{j} \left(p^{(\ell+1)}\right)^j \left(1-p^{(\ell+1)}\right)^{d_v-j}, \quad 0 \le j \le d_v$

induced by unverified variable nodes, at iteration ℓ (reflected in the outgoing message of the check node), is $i+j$, and ii) the second coordinate of messages received by a variable node in the support set from check nodes in the sets $\mathcal{N}_{1,j}^{(\ell)}, 0 \le j \le d_c - 1$, is the same.

In algorithms LM and SBB, each iteration consists of two rounds, each with two half-rounds. The configuration of the sets at the end of each half-round (HR1 or HR2), each round (R1 or R2), and each iteration (ℓ), is specified using the following 4 superscripts: $(\ell, R1, 1), (\ell, R1, 2), (\ell, R2, 1)$, and $(\ell, R2, 2)$, where the term "HR" (for half round) is dropped for a simpler notation. In the first half-rounds (any round and any iteration), messages are passed from check nodes to variable nodes, while in the second half-rounds, messages are passed from variable nodes to check nodes. Also, based on the definition of mapping functions $\Phi_v^{(1,\ell)}$ and $\Phi_v^{(2,\ell)}$, verified variable nodes in the first and the second rounds belong to the sets $\mathcal{K}^{(\ell)}$ and $\Delta^{(\ell)}$, respectively. We

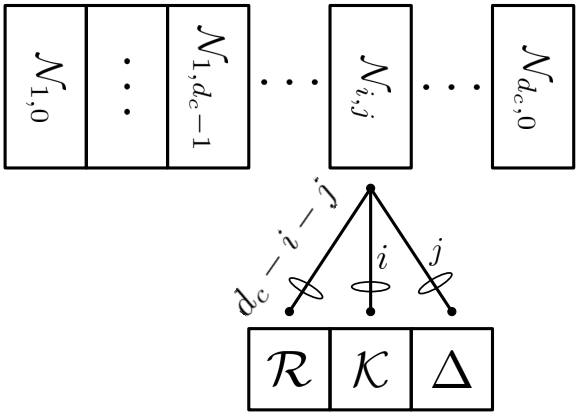


Figure 4.1: Each check node in the set $\mathcal{N}_{i,j}$ has i connections to the variable nodes in set $\mathcal{K}$ and j connections to the variable nodes in set Δ.

have summarized in Table 4.4 the sets that are affected in each half-round (HR) of each round (R) at any iteration.

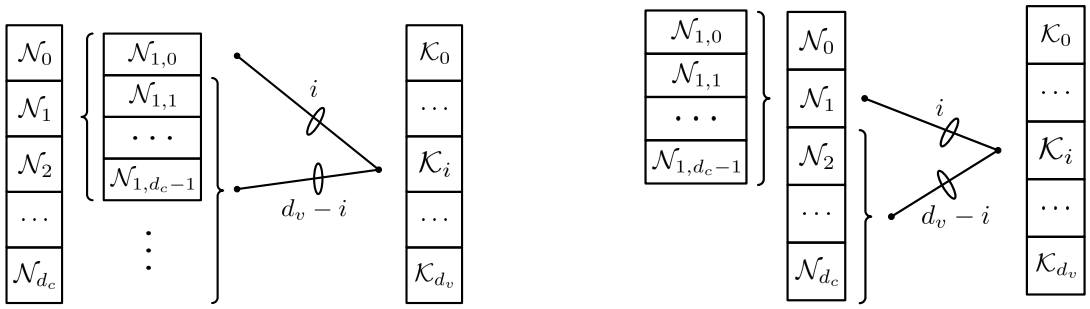
Table 4.4: Sets that are affected in each half-round of each round at any iteration of LM and SBB

R1		R2	
HR1	HR2	HR1	HR2
$\mathcal{N}_{k,i} \to \mathcal{N}_{k,j}$	$\mathcal{K}_i \to \mathcal{K}_j$	$\mathcal{N}_{i,k} \to \mathcal{N}_{j,k}$	$\Delta_i \to \Delta_j$

The set $\mathcal{K}_i^{(\ell,R1,2)}$ in the LM algorithm represents the set of unverified variable nodes in the support set with i neighboring check nodes in the set $\mathcal{N}_{1,0}^{(\ell,R1,1)}$. The definition of set $\mathcal{K}_i^{(\ell,R1,2)}$ for the SBB algorithm is different. Let $\mathcal{N}_i^{(\ell)} := \bigcup_{j=0}^{d_c-i} \mathcal{N}_{i,j}^{(\ell,R1,1)}$. With this notation, the set $\mathcal{K}_i^{(\ell,R1,2)}$ in the SBB algorithm is defined as the set of unverified variable nodes in the support set with i neighboring check nodes in the set $\mathcal{N}_1^{(\ell)}$. These sets are shown in Fig. 4.2 for the two algorithms.

In Theorems 3 and 4 below, we characterize the verification of unverified non-zero variable nodes in the set $\mathcal{K}^{(\ell)}$ in each iteration ℓ for the two algorithms LM and SBB, respectively. The proofs are rather straightforward and follow from the verification rules for LM and SBB, respectively.

Theorem 3. *In the first round of any iteration ℓ in the LM algorithm, a non-zero variable node $v \in \mathcal{K}^{(\ell)}$ is verified if and only if it belongs to the set $\bigcup_{i=1}^{d_v} \mathcal{K}_i^{(\ell,R1,2)}$.*



(a) LM (b) SBB

Figure 4.2: The Difference in the Definition of $\mathcal{K}_i$ in LM vs. SBB.

Theorem 4. *In the first round of any iteration ℓ in the SBB algorithm, a non-zero variable node $v \in \mathcal{K}^{(\ell)}$ is verified if and only if it belongs to the set $\bigcup_{i=2}^{d_v} \mathcal{K}_i^{(\ell,R1,2)} \cup \hat{\mathcal{K}}_1^{(\ell,R1,2)}$, where the set $\hat{\mathcal{K}}_1^{(\ell,R1,2)}$ consists of all variable nodes in the set $\mathcal{K}_1^{(\ell,R1,2)}$ connected to the set $\mathcal{N}_{1,0}^{(\ell,R1,1)}$.*

In LM and SBB algorithms, unverified variable nodes with zero values are verified in R2. Note that a check node is zero-valued if it belongs to the set $\mathcal{N}_{0,j}^{(\ell,R2,1)}, 0 \leq j \leq d_c$. Therefore, for the verification of zero-valued variable nodes in the second round of iteration ℓ, we partition the set of variable nodes in $\Delta^{(\ell)}$ into subsets $\Delta_i^{(\ell)}, 0 \leq i \leq d_v$, with the following definition: a variable node in the set $\Delta_i^{(\ell)}$ has i neighboring check nodes in the set $\left\{\bigcup_{j=1}^{d_c} \mathcal{N}_{0,j}^{(\ell,R2,1)} \setminus \bigcup_{j=1}^{d_c} \mathcal{N}_{0,j}^{(\ell,R1,1)}\right\}$, i.e., the set of check nodes which became zero-valued after HR1 of R2. In Theorem 5 below, we characterize the verification of unverified zero-valued variable nodes in the set $\Delta^{(\ell)}$ at R2-HR2 in each iteration ℓ of LM and SBB algorithms.

Theorem 5. *In the second half-round of the second round of any iteration ℓ in the LM and SBB algorithms a zero-valued variable node $v \in \Delta^{(\ell)}$ is verified if and only if it belongs to the set $\bigcup_{i=1}^{d_v} \Delta_i^{(\ell)}$.*

We denote by $\mathcal{N}_{i,k\downarrow j}^{(\ell,R1)}$ the set of check nodes that are moved from $\mathcal{N}_{i,k}^{(\ell-1,R2,1)}$ to $\mathcal{N}_{i,j}^{(\ell,R1,1)}$ in R1-HR1 of iteration ℓ. Similarly, the set of check nodes that are moved from $\mathcal{N}_{i,k}^{(\ell,R1,1)}$ to $\mathcal{N}_{j,k}^{(\ell,R2,1)}$ in R2-HR1 of iteration ℓ is denoted by $\mathcal{N}_{i\downarrow j,k}^{(\ell,R2)}$. Since variable nodes in $\mathcal{K}$ and Δ are verified through iterations, we always have $j \leq i$ and hence the use of notation $i \downarrow j$. Moreover, in SBB, we denote the set of variable nodes that are moved from $\mathcal{K}_i^{(\ell-1,R1,2)}$ to $\mathcal{K}_j^{(\ell,R1,2)}$ in R1-HR2 of iteration ℓ by $\mathcal{K}_{i\uparrow j}^{(\ell,R1)}$.

The sets that fully describe the state of the decoder at the beginning of iteration ℓ are: $\mathcal{K}^{(\ell)}$, $\mathcal{R}^{(\ell)}$, $\Delta^{(\ell)}$, $\mathcal{N}_{i,j}^{(\ell-1,R2,1)}$, $\mathcal{K}_i^{(\ell-1,R1,2)}$, and $\Delta_i^{(\ell-1,R2,2)}$. For the analysis, we track the probability that a node (variable node or check node) belongs to a certain set at each half-round, round, or iteration. We use the notation $\alpha^{(\ell)}$ to denote the probability that a variable node belongs to the set $\mathcal{K}^{(\ell)}$. For the rest of the sets,

we use the standard notation of probabilities that was applied in the analysis of the Genie algorithm. For instance, we denote the probability that a check node belongs to the set $\mathcal{N}_{i,j}^{(\ell,R1,1)}$ by $p_{\mathcal{N}_{i,j}}^{(\ell,R1,1)}$.

In the analysis, the goal is to find the recursive equations that relate the probabilities of different sets for consecutive iterations. As we shall see, the analysis of the decoding process for LM and SBB results in a system of coupled recursive update equations. Moreover, we show that the update equations at iteration ℓ are functions of probabilities at iteration $\ell-1$. Hence, the complexity of the analysis scales linearly with the number of iterations. In the following subsection, we present the update equations for LM and SBB algorithms. The derivation of formulas are discussed in detail in Parts A.4 and A.5 of the appendix for the two algorithms, respectively.

4.6 Update Equations for LM and SBB Algorithms

We have summarized the sets involved in the analysis of LM and SBB algorithms in Table 4.5. The step-by-step analysis of the algorithms are presented in Tables 4.7, 4.8, 4.9, 4.10 and 4.11, where the update equations are identified by the round and the half-round they correspond to. The update equations in these tables involve the probabilities of the sets described in Table 4.5, as well as some other probabilities, defined in Table 4.6.

4.7 Noisy Measurements

We adopt the following model for the case where the measurements are noisy [48]:

$$\boldsymbol{c} = \boldsymbol{G}\boldsymbol{v} + \boldsymbol{n}.$$

In this new model, $\boldsymbol{v}$ and $\boldsymbol{G}$ are the original signal and the sensing matrix, respectively. The new term, $\boldsymbol{n}$, represents the noise vector added to the noiseless measurements $\boldsymbol{G}\boldsymbol{v}$, resulting in the noisy measurement vector $\boldsymbol{c}$. Elements of the noise vector are assumed to be i.i.d. Gaussian random variables with mean 0 and variance σ^2. The addition of noise to the measurements results in the following two probabilities to be zero: 1) the probability of having a zero measurement, and 2) the probability of having two equal measurements. This will disable the ZCN and ECN rules in recovering the signal elements. Without the ZCN and ECN rules, zero-valued variable nodes

Table 4.5: Sets Involved in the Analysis of LM and SBB Algorithms ($\ell \geq 2$)

Sets Involved in Both Analyses	
$\mathcal{K}^{(\ell)}$	unverified non-zero VNs at the beginning of iteration ℓ.
$\Delta^{(\ell)}$	unverified zero-valued VNs at the beginning of iteration ℓ.
$\mathcal{K}_i^{(\ell,R1,2)}$	VNs in the support set with i neighboring CNs in the set $\mathcal{X}$ after R1-HR1. For LM, $\mathcal{X} = \mathcal{N}_{1,0}^{(\ell,R1,1)}$ and for SBB, $\mathcal{X} = \mathcal{N}_1^{(\ell)} := \bigcup_{j=0}^{d_c-1} \mathcal{N}_{1,j}^{(\ell,R1,1)}$.
$\Delta_i^{(\ell,R2,2)}$	VNs with i neighboring CNs in the set $\left\{\bigcup_{j=1}^{d_c} \mathcal{N}_{0,j}^{(\ell,R2,1)} \setminus \bigcup_{j=1}^{d_c} \mathcal{N}_{0,j}^{(\ell,R1,1)}\right\}$ i.e., the set of CNs which became zero-valued after R2-HR1.
$\mathcal{N}_{i,j}^{(\ell,R1,1)}$	CNs with $\mathcal{K}$-degree i and Δ-degree j after R1-HR1.
$\mathcal{N}_{i,j}^{(\ell,R2,1)}$	CNs with $\mathcal{K}$-degree i and Δ-degree j after R2-HR1.
$\mathcal{N}_{i,k\downarrow j}^{(\ell,R1)}$	CNs that are moved from $\mathcal{N}_{i,k}^{(\ell-1,R2,1)}$ to $\mathcal{N}_{i,j}^{(\ell,R1,1)}$ in R1-HR1.
$\mathcal{N}_{i\downarrow j,k}^{(\ell,R2)}$	CNs that are moved from $\mathcal{N}_{i,k}^{(\ell,R1,1)}$ to $\mathcal{N}_{j,k}^{(\ell,R2,1)}$ in R2-HR1.
Sets Involved in the Analysis of SBB Only	
$\mathcal{K}_{i\uparrow j}^{(\ell,R1)}$	variable nodes moved from $\mathcal{K}_i^{(\ell-1,R1,2)}$, $i = 0, 1$, into $\mathcal{K}_j^{(\ell,R1,2)}$, $i \leq j \leq d_v$.
$\mathcal{K}_1^{(\ell,R1,2,+)}$	VNs in $\mathcal{K}_{0\uparrow 1}^{(\ell,R1)}$ not verified at iteration ℓ, R1-HR2.
$\mathcal{K}_1^{(\ell,R1,2,C)}$	VNs in $\mathcal{K}_{1\uparrow 1}^{(\ell,R1)}$ not verified at iteration ℓ, R1-HR2.
$\mathcal{N}_{1,j}^{(\ell,R1,1,+)}$	CNs moved from $\mathcal{N}_{1,k}^{(\ell-1,R2,1,+)}$, as part of $\mathcal{N}_{1,k\downarrow j}^{(\ell,R1)}$ at iteration ℓ, R1-HR1.
$\mathcal{N}_{1,j}^{(\ell,R1,1,C)}$	CNs moved from $\mathcal{N}_{1,k}^{(\ell-1,R2,1,C)}$, as part of $\mathcal{N}_{1,k\downarrow j}^{(\ell,R1)}$ at iteration ℓ, R1-HR1.
$\mathcal{N}_{1,j}^{(\ell,R2,1,+)}$	CNs moved into the set $\mathcal{N}_{1,j}^{(\ell,R2,1)}$ from all the other sets $\mathcal{N}_{i,j}^{(\ell,R1,1)}$ at iteration ℓ, R2-HR1.
$\mathcal{N}_{1,i}^{(\ell,R2,1,C)}$	union of sets $\mathcal{N}_{1,i}^{(\ell,R2,C,F)}$ and $\mathcal{N}_{1,i}^{(\ell,R2,+,F)}$, $1 \leq i \leq d_c - 1$.
$\mathcal{N}_{1,i}^{(\ell,R2,+,F)}$	CNs of Δ-degree i with a neighboring VN in $\mathcal{K}_{0\uparrow 1}^{(\ell,R1)}$.
$\mathcal{N}_{1,i}^{(\ell,R2,+,O)}$	CNs of Δ-degree i with a neighboring VN in $\mathcal{K}_{0\uparrow j}^{(\ell,R1)}$.
$\mathcal{N}_{1,i}^{(\ell,R2,C,F)}$	CNs of Δ-degree $i, 0 \leq i \leq d_c - 1$, with a neighboring VN in $\mathcal{K}_{1\uparrow 1}^{(\ell,R1)}$.
$\mathcal{N}_{1,i}^{(\ell,R2,C,O)}$	CNs of Δ-degree $i, 0 \leq i \leq d_c - 1$, with a neighboring VN in $\mathcal{K}_{1\uparrow j}^{(\ell,R1)}$.

are not verified, and consequently, no check node will have a reduced degree in the subgraph induced by unverified variable nodes. Therefore, the D1CN rule will also be ineffective.

In the context of message-passing algorithms, there are generally two approaches to deal with noisy measurements. In the first approach, the original formulation of the problem is changed so that the noise is taken into consideration [26, 27, 32–

Table 4.6: Probabilities that Appear in the Update Equations of LM and SBB Algorithms for $\ell \geq 2$ (in Addition to the Probabilities of the Sets Described in Table 4.5). Each Entry of the Table is the Probability of the Corresponding Event.

Prob. Involved in Both Analyses	
$A^{(\ell)}$	message from a VN in $\Delta^{(\ell)}$ to a CN indicates an unverified status in R1-HR1 of iteration ℓ.
$B^{(\ell)}$	LM: edge adjacent to a VN in $\mathcal{K}^{(\ell)}$ is connected to a CN of degree 1 i.e., a CN in $\mathcal{N}_{1,0}^{\ell,R1,1}$, after R1-HR1 of iteration ℓ. SBB: edge emanating from a VN in $\mathcal{K}^{(\ell)}$ and not connected to a CN in $\bigcup_{k=0}^{d_c-1} \mathcal{N}_{1,k}^{(\ell,R1,1,C)}$, is adjacent to a CN in $\bigcup_{k=0}^{d_c-1} \mathcal{N}_{1,k}^{(\ell,R1,1,+)}$.
$C^{(\ell)}$	message from a VN in $\mathcal{K}^{(\ell)}$ to a CN (at iteration ℓ, R2-HR1) indicates an unverified status.
$D^{(\ell)}$	message from a CN to a VN in $\Delta^{(\ell)}$ (at iteration ℓ, R2-HR2) indicates a zero-valued check node.
Prob. Involved in the Analysis of SBB Only	
$f^{(\ell,R1,+)}$	VN in $\mathcal{K}_{0\uparrow 1}^{(\ell,R1)}$ is verified at iteration ℓ, R1-HR2.
$f^{(\ell,R1,C)}$	VN in $\mathcal{K}_{1\uparrow 1}^{(\ell,R1)}$ is verified at iteration ℓ, R1-HR2.
$N^{(\ell,R1)}$	VN in $\mathcal{K}^{(\ell)}$ remains unverified at iteration ℓ.
$p^{(\ell,R1)}$	edge is adjacent to a CN in the set $\mathcal{N}_1^{(\ell)}$ given it is adjacent to a VN in $\mathcal{K}^{(\ell)}$.

34, 49]. In the other approach, the algorithms are changed in order to cope with the presence of noise in the measurements [2]. The first approach generally results in lower reconstruction noise. In particular, it is shown in [34] that the belief propagation algorithm is asymptotically optimal in the case of sparse noisy measurements. The downside to the algorithms that are based on the first approach, however, is that they are generally more complex, may require unbounded message size and would be susceptible to approximation errors. The authors in [2] instead equipped their VB algorithm with some thresholding techniques and proved that if the original signal is sparse enough, they are able to recover the location and the sign of the non-zero signal elements successfully. In what follows, we propose a similar thresholding technique to deal with the noisy measurements.

Thresholding is a common technique in detection theory to deal with noisy measurements [50]. We apply this technique to VB algorithms by defining two thresholds ϵ_1 and ϵ_2. We use ϵ_1 to convert small noisy measurements to zero; i.e., any measurement c, such that $|c| < \epsilon_1$, is set to zero. We use ϵ_2 as the acceptable tolerance

Table 4.7: Density Evolution Analysis of LM - Initialization

Inputs: d_v, d_c, $\alpha^{(0)}$
$A^{(1)} = (1-\alpha^{(0)})\left(1-\left(1-\alpha^{(0)}\right)^{d_c-1}\right)^{d_v-1}$
$p^{(0,R2,1)}_{\mathcal{N}_{i,d_c-i}} = \binom{d_c}{i}\left(\alpha^{(0)}\right)^{i}\left(1-\alpha^{(0)}\right)^{d_c-i}, \quad 0 \le i \le d_c$
$p^{(1,R1)}_{\mathcal{N}_{i,d_c-i\downarrow j}} = \binom{d_c-i}{j}\left(A^{(1)}\right)^{j}\left(1-A^{(1)}\right)^{d_c-i-j}, \quad 1 \le i \le d_v, \quad 0 \le j \le d_c-i$
$p^{(1,R1,1)}_{\mathcal{N}_{i,j}} = p^{(0,R2,1)}_{\mathcal{N}_{i,d_c-i}} p^{(1,R1)}_{\mathcal{N}_{i,d_c-i\downarrow j}}, \quad 1 \le i \le d_v, \quad 0 \le j \le d_c-i$
$\alpha^{(1)} = \alpha^{(0)}$
Apply the update equations of R1-HR2 for $\ell = 1$
Apply the update equations of R2-HR1 for $\ell = 1$
$p^{(1)}_{\Delta} = (1-\alpha^{(0)})\left(1-\left(1-\alpha^{(0)}\right)^{d_c-1}\right)^{d_v}$
Apply the update equations of R2-HR2 for $\ell = 1$

for the equality of two noisy measurements; i.e., we consider two measurements c_1 and c_2 equal if $|c_1 - c_2| < \epsilon_2$. In this case, we assign c_1 and c_2 a new common value equal to $(c_1 + c_2)/2$. While the scope of this work is not to optimize thresholds ϵ_1 and ϵ_2, our goal is to demonstrate the potential of thresholding in desensitizing the VB algorithms to the measurement noise. We explain this through an example and by comparing the performance of the SBB algorithm equipped with thresholding and two methods based on ℓ_1 minimization in the case where the measurements are noisy.

Consider a signal of dimension $n = 1000$. We let the size of the support set, k, to increase from 10 to 150 in steps of 10. For each such support size k, we pick k out of the n elements randomly, and assign to each element an even integer uniformly distributed in the range $[-1000, 1000]$, independent of the value of the other non-zero elements. In the case of the SBB algorithm, the signal is measured through a $(3, 6)$ unweighted bigraph. In the case of ℓ_1-based algorithms, we use sensing matrices consisting of orthonormal columns with Standard Gaussian elements [48]. In all cases, the number of measurements, m, is fixed at 500. Each measurement is independently contaminated with a Gaussian noise of mean 0 and variance equal to $\sigma^2 = 0.25$.

For the SBB, we set both thresholds ϵ_1 and ϵ_2 equal to 1.99. Since the value of non-zero signal elements are drawn from a finite alphabet and since the graph is unweighted, false alarm may occur with non-zero probability in the recovery process. In our simulations, we consider a recovery algorithm "successful" if it can fully recover the support set.

The first ℓ_1-based recovery algorithm is the ℓ_1 regularization method introduced in [48]. For the second algorithm, we empower the ℓ_1 regularization algorithm with the knowledge of the size of the support set. The algorithm thus keeps the k components that are the largest in magnitude and converts the rest to zero. To simulate the two

Table 4.8: Density Evolution Analysis of LM - Recursive Formulas for $\ell \geq 2$

R1	HR1	1) $A^{(\ell)} = \dfrac{p_{\Delta}^{(\ell)}}{1 - D^{(\ell-1)}}$ 2) $p_{\mathcal{N}_{i,k\downarrow j}}^{(\ell,R1)} = \binom{k}{j} \left(A^{(\ell)}\right)^{j} \left(1 - A^{(\ell)}\right)^{k-j}, 1 \leq i \leq d_c, 0 \leq k \leq d_c - i, 0 \leq j \leq k$ 3) $p_{\mathcal{N}_{i,j}}^{(\ell,R1,1)} = \sum_{k=j}^{d_c-i} p_{\mathcal{N}_{i,k}}^{(\ell-1,R2,1)} p_{\mathcal{N}_{i,k\downarrow j}}^{(\ell,R1)}, \quad 1 \leq i \leq d_c, \quad 0 \leq j \leq d_c - i$
	HR2	1) $B^{(\ell)} = \dfrac{p_{\mathcal{N}_{1,0}}^{(\ell,R1,1)}}{\alpha^{(\ell)} d_c}$ 2) $p_{\mathcal{K}_i}^{(\ell,R1,2)} = \binom{d_v}{i} \left(B^{(\ell)}\right)^{i} \left(1 - B^{(\ell)}\right)^{d_v - i}, \quad 0 \leq i \leq d_v$ 3) $\alpha^{(\ell+1)} = \alpha^{(\ell)} \left(1 - \sum_{i=1}^{d_v} p_{\mathcal{K}_i}^{(\ell,R1,2)}\right)$
R2	HR1	1) $C^{(\ell)} = \dfrac{p_{\mathcal{K}_0}^{(\ell,R1,2)}}{1 - B^{(\ell)}}$ 2) $p_{\mathcal{N}_{1\downarrow 0,0}}^{(\ell,R2)} = 1, \quad p_{\mathcal{N}_{1\downarrow 1,0}}^{(\ell,R2)} = 0, \quad p_{\mathcal{N}_{i\downarrow j,k}}^{(\ell,R2)} = \binom{i}{j} \left(C^{(\ell)}\right)^{j} \left(1 - C^{(\ell)}\right)^{i-j},$ $2 \leq i \leq d_c, \quad 0 \leq j \leq i, \quad 0 \leq k \leq d_c - i$ 3) $p_{\mathcal{N}_{j,k}}^{(\ell,R2,1)} = \sum_{i=j}^{d_c} p_{\mathcal{N}_{i,k}}^{(\ell,R1,1)} p_{\mathcal{N}_{i\downarrow j,k}}^{(\ell,R2)}, \quad 0 \leq j \leq d_c, \quad 0 \leq k \leq d_c - i$
	HR2	1) $D^{(\ell)} = \sum_{j=1}^{d_c-1} j \dfrac{p_{\mathcal{N}_{0,j}}^{(\ell,R2,1)}}{\sum_{i=0}^{d_c} \sum_{j=1}^{d_c-1} j p_{\mathcal{N}_{i,j}}^{(\ell,R2,1)}}$ 2) $p_{\Delta_i}^{(\ell,R2,2)} = \binom{d_v}{i} \left(D^{(\ell)}\right)^{i} \left(1 - D^{(\ell)}\right)^{d_v - i}, \quad 0 \leq i \leq d_v$ 3) $p_{\Delta}^{(\ell+1)} = p_{\Delta}^{(\ell)} p_{\Delta_0}^{(\ell,R2,2)}$

ℓ_1-based decoders, we use the L1MAGIC package available in [51].

As the measure of performance, we consider the mean square error (MSE) between the original and the recovered signal. For each value of k, we perform simulations until we obtain 100 "successful" recovery instances. The results for the three algorithms are reported in Fig. 4.3, where for each algorithm, MSE averaged over all simulated cases for a given value of k is shown.

As can be seen, the SBB recovery algorithm significantly (by about two orders of magnitude) outperforms both ℓ_1-based algorithms.

Table 4.9: Density Evolution Analysis of SBB - Initialization

Inputs: d_v, d_c, $\alpha^{(0)}$

$$A^{(1)} = (1-\alpha^{(0)})\left(1-\left(1-\alpha^{(0)}\right)^{d_c-1}\right)^{d_v-1}$$

$$p_{\mathcal{N}_{i,d_c-i}}^{(0,R2,1)} = \binom{d_c}{i}\left(\alpha^{(0)}\right)^{i}\left(1-\alpha^{(0)}\right)^{d_c-i}, \qquad 0 \le i \le d_c$$

$$p_{\mathcal{N}_{i,d_c-i\downarrow j}}^{(1,R1)} = \binom{d_c-i}{j}\left(A^{(1)}\right)^{j}\left(1-A^{(1)}\right)^{d_c-i-j}, \qquad 1 \le i \le d_v, \qquad 0 \le j \le d_c-i$$

$$p_{\mathcal{N}_{i,j}}^{(1,R1,1)} = p_{\mathcal{N}_{i,d_c-i}}^{(0,R2,1)} p_{\mathcal{N}_{i,d_c-i\downarrow j}}^{(1,R1)}, \qquad 1 \le i \le d_v, \qquad 0 \le j \le d_c-i$$

$$\alpha^{(1)} = \alpha^{(0)}$$

$$f^{(1,R1)} = \frac{p_{\mathcal{N}_{1,0}}^{(1,R1,1)}}{\sum_{j=0}^{d_c-1} p_{\mathcal{N}_{1,j}}^{(1,R1,1)}}, \qquad B^{(1)} = \sum_{j=0}^{d_c-1} \frac{p_{\mathcal{N}_{1,j}}^{(1,R1,1)}}{\alpha^{(1)} d_c}$$

$$N^{(1,R1)} = \left(1-B^{(1)}\right)^{d_v} + d_v\left(1-f^{(1,R1)}\right)B^{(1)}\left(1-B^{(1)}\right)^{d_v-1}$$

$$p_{\mathcal{K}_0}^{(1,R1,2)} = \frac{1}{N^{(1,R1)}}\left(1-B^{(1)}\right)^{d_v}, \quad p_{\mathcal{K}_1}^{(1,R1,2)} = \frac{d_v}{N^{(1,R1)}}\left(1-f^{(1,R1)}\right)B^{(1)}\left(1-B^{(1)}\right)^{d_v-1}$$

$$p_{\mathcal{K}_i}^{(1,R1,2)} = 0, \quad 2 \le i \le d_v, \qquad \alpha^{(2)} = \alpha^{(1)} N^{(1,R1)}\left(p_{\mathcal{K}_0}^{(1,R1,2)} + p_{\mathcal{K}_1}^{(1,R1,2)}\right)$$

$$C^{(1)} = \left(1-B^{(1)}\right)^{d_v-1} + (d_v-1)B^{(1)}\left(1-B^{(1)}\right)^{d_v-2}\left(1-f^{(1,R1)}\right)$$

$$p_{\mathcal{N}_{0\downarrow 0,j}}^{(1,R2)} = 1, \quad 0 \le j \le d_c, \qquad p_{\mathcal{N}_{1\downarrow 0,0}}^{(1,R2)} = 1, \quad p_{\mathcal{N}_{1\downarrow 1,0}}^{(1,R2)} = 0$$

$$p_{\mathcal{N}_{1\downarrow 0,j}}^{(1,R2)} = 1-\left(1-B^{(1)}\right)^{d_v-1}, \quad p_{\mathcal{N}_{1\downarrow 1,j}}^{(1,R2)} = \left(1-B^{(1)}\right)^{d_v-1}, \quad 0 \le j \le d_c-1$$

$$p_{\mathcal{N}_{i\downarrow k,j}}^{(1,R2)} = \binom{i}{k}\left(C^{(1)}\right)^{k}\left(1-C^{(1)}\right)^{i-k}, \quad 2 \le i \le d_c, \quad 0 \le k \le i, \quad 0 \le j \le d_c-i$$

$$p_{\mathcal{N}_{1,j}}^{(1,R2,1,+)} = \sum_{i=2}^{d_c} p_{\mathcal{N}_{i,j}}^{(1,R1,1)} p_{\mathcal{N}_{i\downarrow 1,j}}^{(1,R2)}, \qquad p_{\mathcal{N}_{1,j}}^{(1,R2,1,C)} = p_{\mathcal{N}_{1,j}}^{(1,R1,1)} p_{\mathcal{N}_{1\downarrow 1,j}}^{(1,R2)}$$

$$p_{\mathcal{N}_{k,j}}^{(1,R2,1)} = \sum_{i=k}^{d_c} p_{\mathcal{N}_{i,j}}^{(1,R1,1)} p_{\mathcal{N}_{i\downarrow k,j}}^{(1,R2)}, \quad 2 \le k \le d_c, \quad 0 \le j \le d_c-i$$

$$D^{(1)} = \sum_{j=1}^{d_c-1} j \frac{p_{\mathcal{N}_{0,j}}^{(1,R2,1)}}{\sum_{i=0}^{d_c}\sum_{j=1}^{d_c-1} j p_{\mathcal{N}_{i,j}}^{(1,R2,1)}}$$

$$p_{\Delta}^{(2)} = (1-\alpha^{(0)})\left(1-\left(1-\alpha^{(0)}\right)^{d_c-1}\right)^{d_v}\left(1-D^{(1)}\right)^{d_v}$$

4.8 Simulation Results

In this section, we present simulation results obtained by running the recovery algorithms over random biregular graphs to recover sparse signals of finite length n from noiseless measurements. We also present analytical results obtained through the mathematical analysis described in this chapter for the asymptotic regime when

Table 4.10: Density Evolution Analysis of SBB - Recursive Formulas for Round 1, $\ell \geq 2$

HR1 1)
$$A^{(\ell)} = \frac{p_{\Delta}^{(\ell)}}{1 - D^{(\ell-1)}}$$

2)
$$p_{\mathcal{N}_{i,k\downarrow j}}^{(\ell,R1)} = \binom{k}{j} \left(A^{(\ell)}\right)^{j} \left(1 - A^{(\ell)}\right)^{k-j}, \; 1 \leq i \leq d_c, \; 0 \leq k \leq d_c - i, \; 0 \leq j \leq k$$

3)
$$p_{\mathcal{N}_{1,j}}^{(\ell,R1,1,+)} = \sum_{k=j}^{d_c-1} p_{\mathcal{N}_{1,k}}^{(\ell-1,R2,1,+)} p_{\mathcal{N}_{1,k\downarrow j}}^{(\ell,R1)}$$

$$p_{\mathcal{N}_{1,j}}^{(\ell,R1,1,C)} = \sum_{k=j}^{d_c-1} p_{\mathcal{N}_{1,k}}^{(\ell-1,R2,1,C)} p_{\mathcal{N}_{1,k\downarrow j}}^{(\ell,R1)}, \;\; 0 \leq j \leq d_c - 1$$

$$p_{\mathcal{N}_{i,j}}^{(\ell,R1,1)} = \sum_{k=j}^{d_c-i} p_{\mathcal{N}_{i,k}}^{(\ell-1,R2,1)} p_{\mathcal{N}_{i,k\downarrow j}}^{(\ell,R1)}, \;\; 2 \leq i \leq d_c, \quad 0 \leq j \leq d_c - i$$

HR2 1)
$$f^{(\ell,R1,+/C)} = \frac{p_{\mathcal{N}_{1,0}}^{(\ell,R1,1,+/C)}}{\sum_{j=0}^{d_c-1} p_{\mathcal{N}_{1,j}}^{(\ell,R1,1,+/C)}}, \quad B^{(\ell)} = \frac{\sum_{j=0}^{d_c-1} p_{\mathcal{N}_{1,j}}^{(\ell,R1,1,+)}}{\sum_{j=0}^{d_c-1} p_{\mathcal{N}_{1,j}}^{(\ell,R1,1,+)} + \sum_{i=2}^{d_c} \sum_{j=0}^{d_c-i} i p_{\mathcal{N}_{i,j}}^{(\ell,R1,1)}}$$

$$p_{\mathcal{K}_{j\uparrow i}}^{(\ell,R1)} = \binom{d_v - j}{i - j} \left(B^{(\ell)}\right)^{i-j} \left(1 - B^{(\ell)}\right)^{d_v - i}$$

$$N^{(\ell,R1)} = p_{\mathcal{K}_0}^{(\ell-1,R1,2)} \left(p_{\mathcal{K}_{0\uparrow 0}}^{(\ell,R1)} + p_{\mathcal{K}_{0\uparrow 1}}^{(\ell,R1)} \left(1 - f^{(\ell,R1,+)}\right) \right) + p_{\mathcal{K}_1}^{(\ell-1,R1,2)} p_{\mathcal{K}_{1\uparrow 1}}^{(\ell,R1)} \left(1 - f^{(\ell,R1,C)}\right)$$

2)
$$\alpha^{(\ell+1)} = \alpha^{(\ell)} N^{(\ell,R1)}$$

$$p_{\mathcal{K}_0}^{(\ell,R1,2)} = \frac{p_{\mathcal{K}_0}^{(\ell-1,R1,2)} p_{\mathcal{K}_{0\uparrow 0}}^{(\ell,R1)}}{N^{(\ell,R1)}}$$

$$p_{\mathcal{K}_1}^{(\ell,R1,2,+)} = \frac{1 - f^{(\ell,R1,+)}}{N^{(\ell,R1)}} p_{\mathcal{K}_0}^{(\ell-1,R1,2)} p_{\mathcal{K}_{0\uparrow 1}}^{(\ell,R1)}$$

$$p_{\mathcal{K}_1}^{(\ell,R1,2,C)} = \frac{1 - f^{(\ell,R1,C)}}{N^{(\ell,R1)}} p_{\mathcal{K}_1}^{(\ell-1,R1,2)} p_{\mathcal{K}_{1\uparrow 1}}^{(\ell,R1)}$$

$$p_{\mathcal{K}_1}^{(\ell,R1,2)} = p_{\mathcal{K}_1}^{(\ell,R1,2,+)} + p_{\mathcal{K}_1}^{(\ell,R1,2,C)}, \qquad p_{\mathcal{K}_j}^{(\ell,R1,2)} = 0, \quad 2 \leq j \leq d_v$$

$n \to \infty$. This includes the success threshold of different VB algorithms over different biregular graphs. The comparison of asymptotic and finite-length results shows that there is a good agreement between the two for moderately large block lengths ($n \geq 10^5$).

In all simulations, a signal element belongs to the support set with probability $\alpha^{(0)}$, unless otherwise specified. Also, each non-zero signal element (variable) is drawn according to a standard Gaussian distribution. The biregular graphs are constructed randomly with no parallel edges and all the edge weights are equal to one. In each set of simulations, the sensing graph is fixed and each simulation point is generated by

Table 4.11: Density Evolution Analysis of SBB - Recursive Formulas for Round 2, $\ell \geq 2$

HR1 1)
$$p^{(\ell,R1)} = \sum_{j=0}^{d_c-1} \frac{p_{\mathcal{N}_{1,j}}^{(\ell,R1,1,+)} + p_{\mathcal{N}_{1,j}}^{(\ell,R1,1,C)}}{\alpha^{(\ell)} d_c}$$

$$C^{(\ell)} = \frac{p_{\mathcal{K}_0}^{(\ell-1,R1,2)} p_{\mathcal{K}_{0\uparrow 0}}^{(\ell,R1)}}{1-p^{(\ell,R1)}} + \frac{p_{\mathcal{K}_0}^{(\ell-1,R1,2)} p_{\mathcal{K}_{0\uparrow 1}}^{(\ell,R1)} \left(\frac{d_v-1}{d_v}\right)}{1-p^{(\ell,R1)}} \left(1-f^{(\ell,R1,+)}\right)$$
$$+ \frac{p_{\mathcal{K}_1}^{(\ell-1,R1,2)} p_{\mathcal{K}_{1\uparrow 1}}^{(\ell,R1)} \left(\frac{d_v-1}{d_v}\right)}{1-p^{(\ell,R1)}} \left(1-f^{(\ell,R1,C)}\right)$$

$$p_{\mathcal{N}_{i\downarrow k,j}}^{(\ell,R2)} = \binom{i}{k} \left(C^{(\ell)}\right)^k \left(1-C^{(\ell)}\right)^{i-k}, 2 \leq i \leq d_c, 0 \leq k \leq i, 0 \leq j \leq d_c - i$$

$$p_{\mathcal{N}_{1,i}}^{(\ell,R2,+,F)} = \frac{d_v p_{\mathcal{N}_{1,i}}^{(\ell,R1,1,+)} p_{\mathcal{K}_0}^{(\ell-1,R1,2)} (1-B^{(\ell)})^{d_v-1}}{d_v p_{\mathcal{K}_0}^{(\ell-1,R1,2)} + (d_v-1) p_{\mathcal{K}_1}^{(\ell-1,R1,2)}}, \qquad 0 \leq i \leq d_c - 1$$

$$p_{\mathcal{N}_{1,i}}^{(\ell,R2,+,O)} = p_{\mathcal{N}_{1,i}}^{(\ell,R1,1,+)} - p_{\mathcal{N}_{1,i}}^{(\ell,R2,+,F)}, \qquad 0 \leq i \leq d_c - 1$$

$$p_{\mathcal{N}_{1,i}}^{(\ell,R2,C,F)} = p_{\mathcal{N}_{1,i}}^{(\ell,R1,1,C)} p_{\mathcal{K}_{1\uparrow 1}}^{(\ell,R1)}$$

$$p_{\mathcal{N}_{1,i}}^{(\ell,R2,C,O)} = p_{\mathcal{N}_{1,i}}^{(\ell,R1,1,C)} - p_{\mathcal{N}_{1,i}}^{(\ell,R2,C,F)}, 0 \leq i \leq d_c - 1$$

2)
$$p_{\mathcal{N}_{0,j}}^{(\ell,R2,1)} = p_{\mathcal{N}_{0,j}}^{(\ell,R1,1)} + p_{\mathcal{N}_{1,j}}^{(\ell,R2,C,O)} + p_{\mathcal{N}_{1,j}}^{(\ell,R2,+,O)} + \sum_{i=2}^{d_c} p_{\mathcal{N}_{i,j}}^{(\ell,R1,1)} p_{\mathcal{N}_{i\downarrow 0,j}}^{(\ell,R2)}, 0 \leq j \leq d_c - i$$

$$p_{\mathcal{N}_{1,j}}^{(\ell,R2,1,+)} = \sum_{i=2}^{d_c} p_{\mathcal{N}_{i,j}}^{(\ell,R1,1)} p_{\mathcal{N}_{i\downarrow 2,j}}^{(\ell,R2)}$$

$$p_{\mathcal{N}_{1,j}}^{(\ell,R2,1,C)} = p_{\mathcal{N}_{1,j}}^{(\ell,R2,C,F)} + p_{\mathcal{N}_{1,j}}^{(\ell,R2,+,F)}, \quad 0 \leq j \leq d_c - 1$$

$$p_{\mathcal{N}_{k,j}}^{(\ell,R2,1)} = \sum_{i=k}^{d_c} p_{\mathcal{N}_{i,j}}^{(\ell,R1,1)} p_{\mathcal{N}_{i\downarrow k,j}}^{(\ell,R2)}, \quad 2 \leq k \leq d_c, \quad 0 \leq j \leq d_c - i$$

HR2 1)
$$D^{(\ell)} = \sum_{j=1}^{d_c-1} j \frac{p_{\mathcal{N}_{0,j}}^{(\ell,R2,1)}}{\sum_{i=0}^{d_c} \sum_{j=1}^{d_c-1} j p_{\mathcal{N}_{i,j}}^{(\ell,R2,1)}}$$

$$p_{\Delta_i}^{(\ell,R2,2)} = \binom{d_v}{i} \left(D^{(\ell)}\right)^i \left(1-D^{(\ell)}\right)^{d_v-i}, \ 0 \leq i \leq d_v$$

2)
$$p_{\Delta}^{(\ell+1)} = p_{\Delta}^{(\ell)} p_{\Delta_0}^{(\ell,R2,2)}$$

averaging over 1000 random instances of the input signal, unless specified otherwise. We repeated each simulation with different randomly generated graphs (with the same variable and check node degrees), and observed that the results were almost identical for every graph. Each simulation is run until the algorithm makes no further progress. In this case, if the signal is recovered perfectly, the recovery is called successful, otherwise a failure is declared.

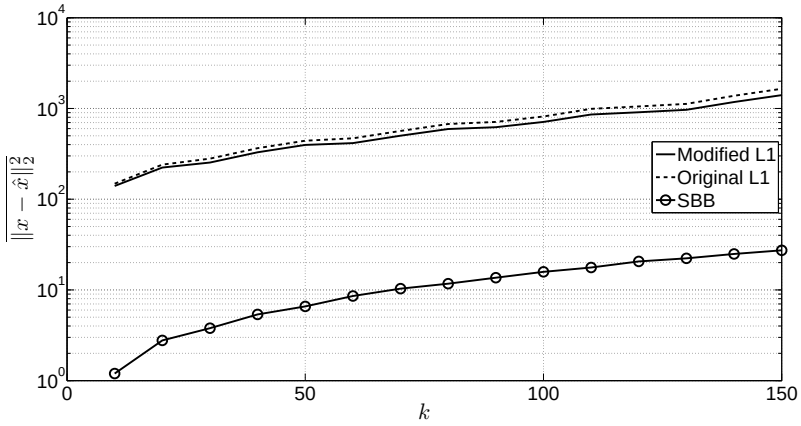


Figure 4.3: Comparison between SBB, ℓ_1 minimization, and modified ℓ_1 minimization in terms of MS reconstruction error in the presence of Gaussian measurement noise with zero mean and variance $\sigma^2 = 0.25$ $(n = 1000, m = 500)$.

For the analytical results, based on the fact that $\alpha^{(\ell)}$ is a non-increasing function of iteration number ℓ, we consider the following stopping criteria:

1. $\alpha^{(\ell)} \leq 10^{-7}$,
2. $\alpha^{(\ell)} > 10^{-7}$ and $|\alpha^{(\ell)} - \alpha^{(\ell-1)}| < 10^{-8}$.

If the analysis is stopped based on the first stopping criterion, the algorithm is considered successful. If, on the other hand, it is stopped based on the second criterion, the algorithm is considered unsuccessful and a failure is declared. To calculate the success threshold, a binary search is performed until the separation between the start and the end of the search region is less than 10^{-5}.

4.8.1 Comparison of SBB and ℓ_1-based algorithms at finite length

To motivate the use of recovery algorithms over sparse graphs, as the first set of simulation results, we present the comparison between the SBB algorithm and two benchmark ℓ_1-based algorithms, ℓ_1 minimization [5] and iterative weighted ℓ_1 minimization [52]. The setup is as follows. For SBB, we choose a random $(3, 6)$ biregular sensing graph with 1000 variable nodes and 500 check nodes. The sensing matrix used for the two ℓ_1-based algorithms consists of 500 rows and 1000 columns. The elements are initially i.i.d. standard Gaussian random variables. Then the rows are made orthonormal.

The cost functions used in ℓ_1 and weighted ℓ_1 minimization algorithms are $\|\boldsymbol{v}\|_1 := \sum_i |\boldsymbol{v}_i|$ and $\|\boldsymbol{W}\boldsymbol{v}\|_1 := \sum_i w_i|\boldsymbol{v}_i|$, respectively, where $\boldsymbol{v}$ is the original signal of interest with elements $\boldsymbol{v}_i$, and $\boldsymbol{W}$ is a diagonal matrix with positive diagonal elements w_i

representing the weights. Weighted ℓ_1 minimization is an iterative algorithm in which the weights at iteration t ($w_i^{(t)}$) are updated according to $w_i^{(t)} = 1/(|\boldsymbol{v}_i^{(t-1)}| + \epsilon)$, where $\boldsymbol{v}_i^{(t-1)}$ is the estimate of the signal element $\boldsymbol{v}_i$ at iteration $t-1$. The weighted ℓ_1 is not very sensitive to the parameter ϵ as noted in [52]. We found $\epsilon = 0.1$ is a good choice based on our simulations. Regarding the maximum number of iterations for the weighted ℓ_1 minimization algorithm, it is shown in [52] that as this parameter increases, better results are achieved, with the cost of longer running time. The improvement gained by increasing the number of iterations beyond 6 however, is negligible [52]. Therefore, in our simulations, we choose a conservative maximum number of iterations equal to 10. As ℓ_1 and weighted ℓ_1 minimization algorithms output an estimate which is very close to the original signal, but not exactly the same, we declare a success for these two algorithms if the difference between every original signal element and its corresponding estimate is less than 10^{-2}. Lastly, we use the L1MAGIC package in [51] as the optimization engine for simulating ℓ_1 and weighted ℓ_1 minimization algorithms.

To have a fair comparison, the same signal vectors are used for all the algorithms. We also choose the size of the support set deterministically, and let the size range from 10 to 300. For each support size, 100 instances of the signal vector are generated. Each signal vector is then measured according to the corresponding sensing mechanism for each class of algorithms. The success or failure of the recovery algorithms over the resulting measurements are then averaged over the 100 instances, and plotted in Figure 4.4. In Figure 4.5 the average running time, in seconds, is plotted for the three algorithms. The algorithms were implemented in MATLAB and were run on a computer with an AMD Phenom 9650 Quad-Core 2.3 GHz processor, 3 GB RAM and a Windows 7 operating system. As can be seen, the SBB algorithm recovers signals with more non-zero elements at a speed which is about 2 orders of magnitude faster compared to that of the ℓ_1 algorithms.

To demonstrate that the recovery performance of NB-VB algorithms is insensitive to the distribution of non-zero signal elements and that of non-zero elements of the sensing matrix, as long as at least one distribution is continuous, we perform the same experiments, this time by selecting the non-zero signal elements independently from the binary set $\{-1, 1\}$ and by choosing the non-zero elements of the sensing matrix independently from a standard Gaussian distribution. The results for SBB in this case are also presented in Fig. 4.4. As can be seen, they are close to the results where the non-zero input signals are Gaussian and the sensing matrix elements are binary.

4.8.2 Asymptotic and finite-length results for NB-VB algorithms

For the next experiment, we apply Genie, XH, SBB and LM algorithms to four randomly constructed $(5, 6)$ regular graphs with $n = \{3, 15, 100, 1000\} \times 10^3$. The

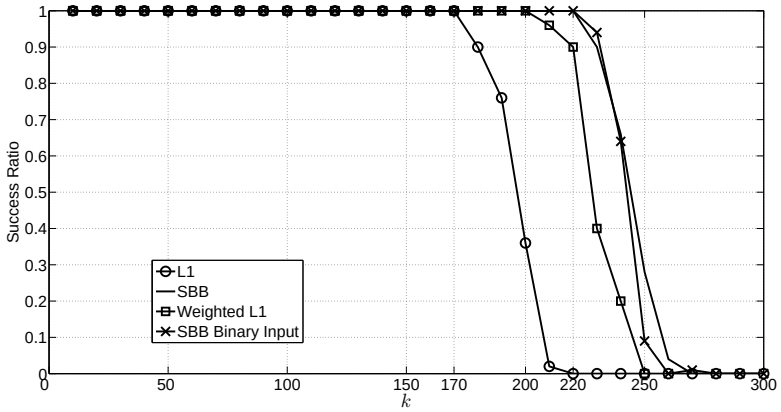


Figure 4.4: Comparison between success ratios of ℓ_1, weighted ℓ_1 and SBB (continuous-input/binary-sensing coefficients and binary-input/continuous-sensing coefficients) for $n = 1000, m = 500$.

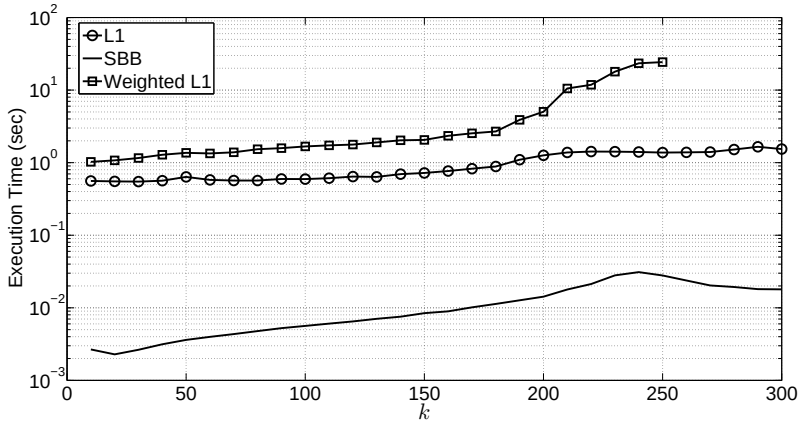


Figure 4.5: Comparison between the average running times of ℓ_1, weighted ℓ_1 and SBB for $n = 1000, m = 500$.

success ratio of the algorithms versus the initial density factor $\alpha = \alpha^{(0)}$ are shown in Figure 4.6. From the figure, we can see that, for all algorithms, by increasing n, the transition part of the curves becomes sharper such that the curves for $n = 10^6$ practically look like a step function. In the figure, we have also shown the success threshold of the algorithms for $(5, 6)$ graphs, obtained based on the proposed analysis, by arrows. As can be seen, the thresholds match very well with the waterfall region of the simulation curves.

In Table 4.12, we have listed the analytical success thresholds of the iterative recovery algorithms for graphs with different d_v and d_c values. The result for XH algorithm on $(3, 4)$ graphs, and more generally for graphs with $d_v = 3$, is missing as the

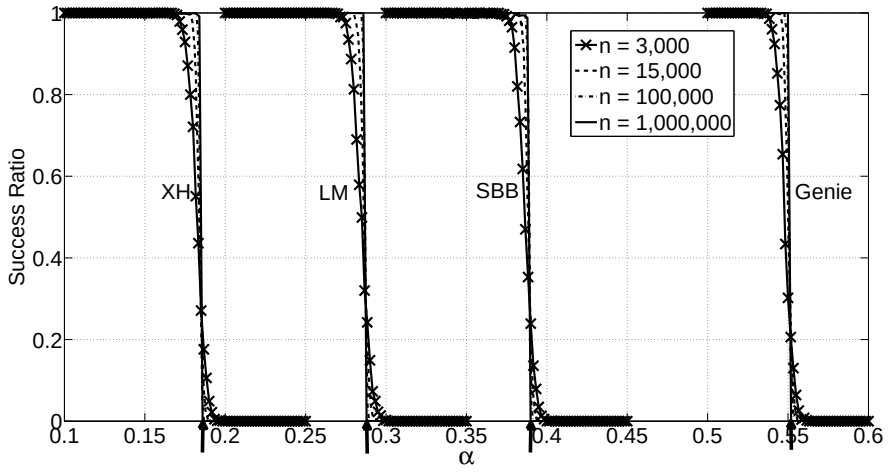


Figure 4.6: Success ratio of Genie, XH, LM and SBB algorithms vs. $\alpha = \alpha^{(0)}$ for $(5, 6)$ graphs with $n = \{3, 15, 100 \text{ and } 1000\} \times 10^3$. Analytical thresholds are shown by arrows.

algorithm performs poorly on such graphs.[3] For every graph, the Genie algorithm has the best performance. This is followed by SBB, LM and XH algorithms, respectively. Careful inspection of the results in Table 4.12 indicates that the oversampling ratio $r_o = \frac{d_v}{\alpha d_c}$ improves consistently by decreasing both d_v and d_c values. In fact, among the results presented in Table 4.12, the application of the Genie and SBB to $(3, 4)$ graphs results in the lowest oversampling ratio of ≈ 1.16 and ≈ 1.67, respectively.

Table 4.12: Success Thresholds for different graphs and algorithms

(d_v, d_c)	$(3, 4)$	$(5, 6)$	$(5, 7)$	$(5, 8)$	$(7, 8)$
Genie	0.6474	0.5509	0.4786	0.4224	0.4708
SBB	0.4488	0.3892	0.3266	0.2806	0.3335
LM	0.3440	0.2871	0.2305	0.1907	0.2385
XH	-	0.1846	0.1552	0.1339	0.1435

In Table 4.13, we have listed the analytical success thresholds of the iterative VB recovery algorithms for graphs with compression ratio $d_v/d_c = 0.5$ and different d_v and d_c values. In general, as we decrease d_v, algorithms perform better in terms

[3]The reason is that for $d_v = 3$, a variable node is verified with the common value of $\lceil d_v/2 \rceil = 2$ check nodes. However, if two non-zero variable nodes share the same two check nodes (a cycle of length 4 exists in the graph), then a false verification may occur.

of recovery capability.[4] This also implies that for a fixed compression ratio, the oversampling ratio improves by decreasing d_v and d_c.

Table 4.13: Success Thresholds for different graphs and algorithms for fixed compression ratio $r_c = 0.5$

(d_v, d_c)	$(3, 6)$	$(4, 8)$	$(5, 10)$	$(6, 12)$	$(7, 14)$
Genie	0.4294	0.3834	0.3415	0.3074	0.2797
SBB	0.2574	0.2394	0.2179	0.1992	0.1835
LM	0.1702	0.1555	0.1391	0.1253	0.1140
XH	-	0.1875	0.1050	0.1170	0.0791

We have also presented the success thresholds of NB-VB algorithms versus the compression ratio for different d_v values in Fig. 4.7. The same trends as discussed above can also be seen in this figure in addition to the expected result that the success threshold in general increases with the increase in the compression ratio. The relative rate of this increase for each algorithm in relation with the other algorithms follows the same trend as the relative performances, i.e., Genie has the highest rate followed by SBB, LM and XH, respectively. In Tables 4.14 and 4.15, we have listed the number

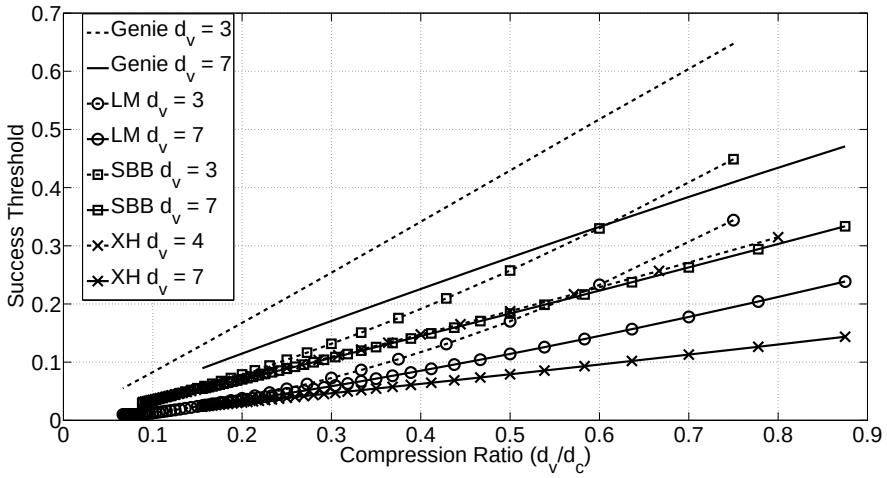


Figure 4.7: Success threshold vs. compression ratio for LM, XH, SBB, and Genie algorithms.

[4]These results are consistent with the results observed for the Belief Propagation (BP) decoding of binary LDPC codes based on biregular graphs.

of iterations required for different recovery algorithms to recover signals with density factor equal to the success thresholds reported in Tables 4.12 and 4.13 minus 0.0001, respectively. These results, which are obtained by the asymptotic analysis are in close agreement with finite-length simulation results at block lengths of about 10^5. These results indicate that with a few exceptions, the better performance comes at the expense of a larger number of iterations. In particular, among the practical recovery algorithms, SBB requires the largest number of iterations for convergence.

Table 4.14: Number of iterations required for different recovery algorithms over different graphs to recover a signal with density ratio equal to the success threshold minus 0.0001

(d_v, d_c)	$(3,4)$	$(5,6)$	$(5,7)$	$(5,8)$	$(7,8)$
Genie	106	66	66	62	55
SBB	655	178	165	200	344
LM	258	139	103	126	108
XH	-	63	58	54	41

Table 4.15: Number of iterations required for different recovery algorithms over different graphs with fixed compression ratio $r_c = 0.5$, to recover a signal with density ratio equal to the success threshold minus 0.0001

(d_v, d_c)	$(3,6)$	$(4,8)$	$(5,10)$	$(6,12)$	$(7,14)$
Genie	93	69	57	50	46
SBB	247	167	172	163	127
LM	142	94	136	97	55
XH	-	64	48	38	32

To further investigate the degree of agreement between our theoretical asymptotic analysis and finite-length simulation results, we have presented in Fig. 4.8 the evolution of $\alpha^{(\ell)}$ with iterations ℓ for Genie, LM, SBB, and XH over a $(5,6)$ graph. For each algorithm, two values of $\alpha^{(0)}$ are selected: one above and one below the success threshold presented in Table 4.12. The theoretical results are shown by solid lines

while simulations for $n = 10^5$ are presented with dotted lines. As one can see, the two sets of results are in close agreement particularly for the cases where $\alpha^{(0)}$ is above the threshold and for smaller values of ℓ.

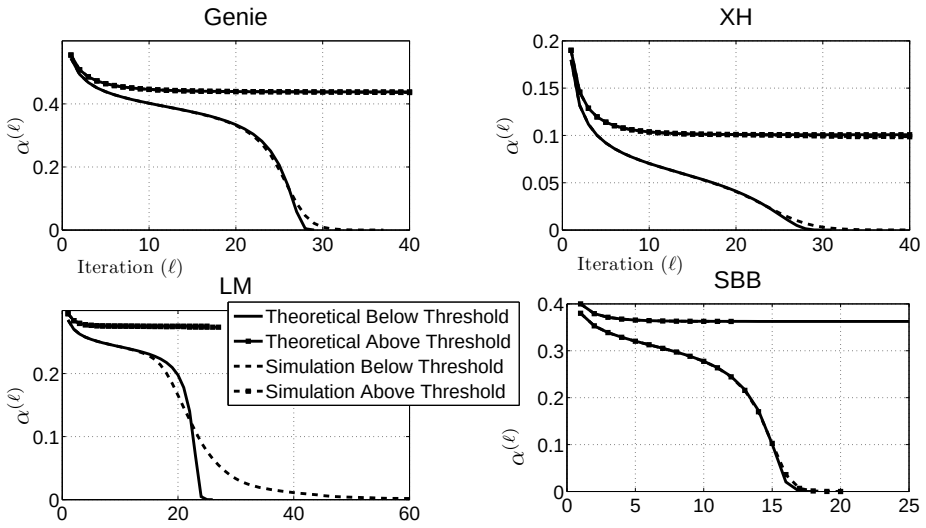


Figure 4.8: Evolution of $\alpha^{(\ell)}$ vs. iteration number ℓ for the four recovery algorithms over a $(5, 6)$ graph (finite-length simulations are for $n = 10^5$).

To demonstrate that the simulation results converge to the asymptotic analytical results as n grows, in Fig. 4.9, we have added the simulation curves for $n = 10^4$ and 10^6 to the SBB curves in Fig. 4.8. As can be seen, the larger the value of n, the closer the simulation results to the analytical ones. In particular, the curves for $n = 10^6$ practically coincide with the analytical results.

Next, for different values of $\alpha^{(0)}$, we estimate the average fraction of unverified non-zero variable nodes $\alpha^{(\ell)}$ using the analysis, and denote the value of $\alpha^{(\ell)}$ at the time that the analysis stops (because one of the stopping criteria is met) as $\alpha^{(stop)}$. These values are plotted vs. the corresponding values of $\alpha^{(0)}$ in Fig. 4.10 for the four VB recovery algorithms over the $(5, 6)$ sensing graphs. In the same figure, we have also given the corresponding simulation results for two randomly selected $(5, 6)$ sensing graphs with $n = 10^5$ and 10^6. The simulation results for both lengths closely match the analytical results, with those of $n = 10^6$ being practically identical to the analytical results. We have indicated the success threshold of the algorithms by arrows. From the figure, it can also be seen that as $\alpha^{(0)}$ increases and tends to one, the curves tend to the asymptote $\alpha^{(stop)} = \alpha^{(0)}$.

The results presented in Tables 4.12 and 4.13 are for sensing graphs with compression ratio at least 0.5. We have also investigated the application of NB-VB algorithms

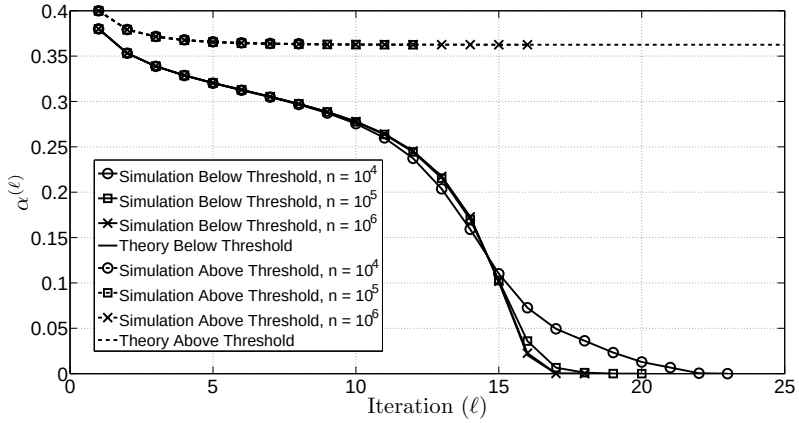


Figure 4.9: Evolution of $\alpha^{(\ell)}$ vs. iteration number for SBB over a (5,6) random graph (finite-length simulations are for $n = 10^4, 10^5$, and 10^6).

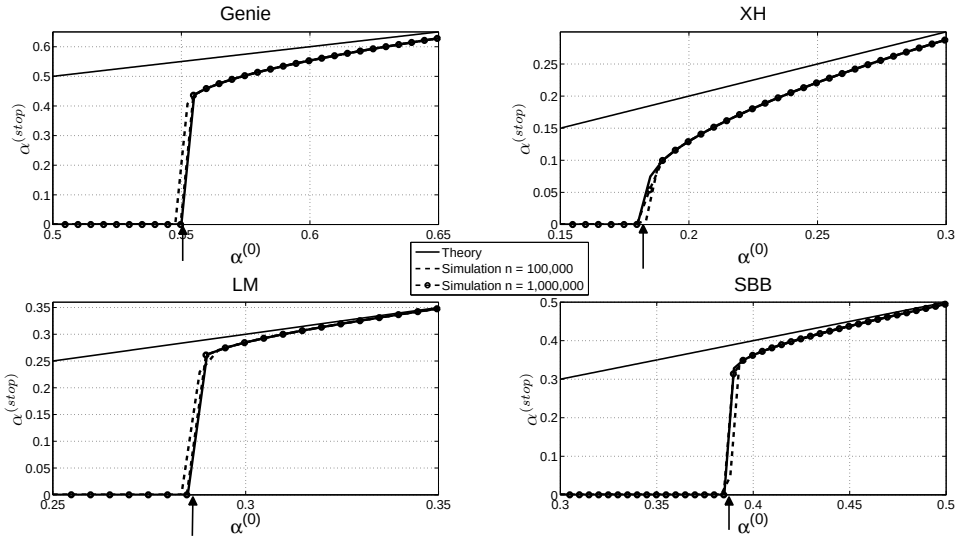


Figure 4.10: Fraction of unrecoverable variable nodes vs. the initial density factor for different recovery algorithms over random (5,6) regular bipartite graphs. The arrows represent the theoretical success thresholds. The straight line represents the function $f(x) = x$.

to graphs with lower compression ratios. For example, simulation results on the success ratio of SBB over random graphs with $n = 10^5$ and (d_v, d_c) equal to $(3, 30)$, $(4, 40)$ and $(3, 45)$ are presented in Fig. 4.11. These graphs have compression ratios equal to 1/10, 1/10 and 1/15, respectively, and their success thresholds for SBB are 0.0338, 0.0380 and 0.0209, respectively. These thresholds are shown in Fig. 4.11 by vertical dashed lines, and they each match the waterfall region of the corresponding finite-length simulation curve. As expected, the lower compression ratio corresponds

to smaller density factors for the signals that can be recovered using these graphs. In Fig. 4.7, this corresponds to the tails of the curves close to the origin.

4.8.3 Comparison of SBB and ℓ_1 recovery in the asymptotic regime of $n \to \infty$

Strong and weak thresholds [53] are important measures of the performance of the ℓ_1 recovery. In particular, the weak threshold is the largest undersampling ratio k/m for which the ℓ_1 recovery and ℓ_0 recovery are equivalent with overwhelming probability in the uniform selection of the sensing matrix for most input signals as $n \to \infty$. As another way of comparing the NB-VB recovery algorithms and ℓ_1 recovery, we compare the success threshold of the former divided by the compression ratio m/n, with the weak threshold of the latter. Consider two scenarios with compression ratios m/n equal to 0.5 and 0.75. The weak threshold of ℓ_1 recovery for these cases is 0.3848 and 0.5327, respectively [53]. The corresponding values for SBB over $(3, 6)$ and $(3, 4)$ graphs are 0.5148 and 0.5984, respectively. In both cases, the values for SBB are larger, and indicate the superiority of SBB.

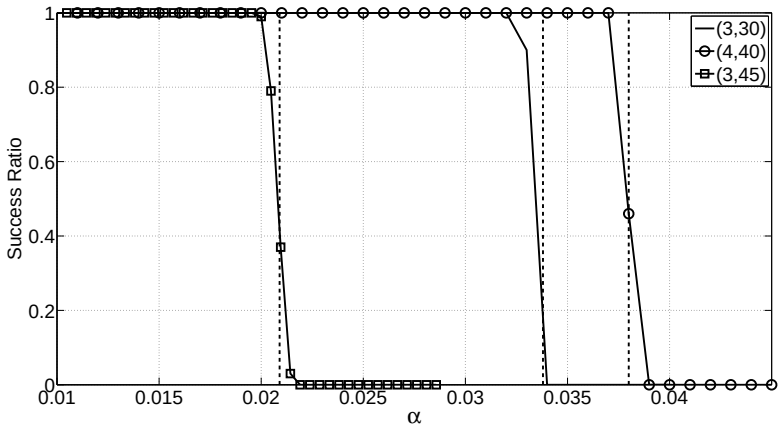


Figure 4.11: Success ratio of SBB vs. $\alpha = \alpha^{(0)}$ for graphs with $n = 10^5$ and (d_v, d_c) equal to $(3, 30)$, $(4, 40)$ and $(3, 45)$. Theoretical success thresholds are shown by vertical dashed lines.

4.8.4 Comparison with the asymptotic results of [1]

As the last experiment, we compare the running time and the accuracy of the proposed asymptotic analysis against those of the differential equation approach presented in [1]. For comparison, a biregular $(3, 6)$ graph and the SBB algorithm are chosen. The binary search for the success threshold starts with the interval $[0.2, 0.3]$ and ends

when the separation between the start and the end of the search region in less than 10^{-5}. The analysis is implemented in MATLAB and executed on the same computer described before. Using the proposed analysis, we obtain the success threshold of 0.2574 in 23.1 seconds. Table 4.16 summarizes the results of running the analysis of [1] on the same machine for different values of n. The reported thresholds increase with the increase in n. For $n = 10^5$, the running time is roughly 100 times that of our proposed method. Moreover, and more importantly, the obtained threshold of 0.2591 is only in agreement with the threshold of 0.2574, obtained by the proposed method, up to two decimal points. In fact, experiments similar to those reported in Fig. 4.8 reveal that the accuracy of the threshold obtained by the method of [1] is lower than our results. In particular, our simulations show that the SBB algorithm over $(3, 6)$ graphs with $n = 10^5$ fails for $\alpha^{(0)} = 0.259$, which would imply that the threshold 0.2591 is only accurate up to two decimal points.

Table 4.16: Success threshold and running time of the analysis of [1] for SBB over a random $(3, 6)$ regular graph.

n	100	1,000	10,000	20,000	50,000	100,000
Success Threshold	0.2465	0.2577	0.2589	0.2590	0.2590	0.2591
Running Time (seconds)	1.1	9.9	103.9	220.6	647.4	2044.1

Chapter 5

Analysis of NB-VB Algorithms over Irregular Graphs

5.1 Introduction

The main focus of this chapter is on the analysis of NB-VB recovery algorithms for compressed sensing with *irregular* sensing graphs. Our results are derived in the asymptotic regime ($n \to \infty$). In this regime, the input model is as in Chapter 4, i.e., a signal element is zero with probability $1 - \alpha$ or takes a value from a continuous distribution with probability α.

In this chapter, we extend the analysis presented in Chapter 4 to irregular graphs. Our simulations show that for a given compression ratio m/n, irregular graphs can provide up to 40% larger success thresholds compared to regular graphs. Just like the analysis in Chapter 4, the proposed analysis is developed for noiseless measurements and its computational complexity increases only linearly with the number of iterations. Moreover, the analysis is simple to perform, requiring only additions and multiplications.

The SBB algorithm performs the best among all known VB algorithms in the context of compressed sensing [1, 40, 41]. Hence, the analysis of SBB over irregular graphs is the focus in this chapter. The proposed analytical framework is, however, general and applicable to the analysis of other NB-VB algorithms over irregular graphs as well.

5.2 Asymptotic Analysis Framework

Let the probability distributions f and g, the degree distributions λ and ρ, and the density factor α be fixed. It can be shown that the fraction of unverified non-zero variable nodes at each iteration ℓ of the SBB algorithm ($\alpha^{(\ell)}$) over a realization of the sensing graph $\mathcal{G} \in \mathcal{G}_f^n(\lambda, \rho)$ and a realization of the input signal $\mathcal{V} \in \mathcal{V}_g^n(\alpha)$ concentrates around the average of $\alpha^{(\ell)}$ taken over all the elements in the ensemble

$\mathcal{G}_f^n(\lambda,\rho) \times \mathcal{V}_g^n(\alpha)$, as n tends to infinity.[1] The deterministic analysis presented here is to track the evolution of this average as n goes to infinity. In the language of coding, the analysis is similar to the density evolution analysis of iterative decoding algorithms for irregular LDPC code ensembles, with the main difference being that the NB-VB algorithms do not conform to the principle of *extrinsic* message passing which significantly simplifies the density evolution analysis in the context of coding.

The analysis of the SBB algorithm over irregular bigraphs tracks the fraction of zero and non-zero unverified variable nodes through iterations. This mathematical framework for the analysis is similar to the one presented in Chapter 4 (and previously used in [40, 41]), however with two extra variables d_v and d_c which represent the degree of a variable and a check node, respectively. These variables take different values for irregular graphs while for a regular graph they each have a fixed value. This makes the derivations more tedious. Henceforth, we denote by $d_{v,\max}$ and $d_{c,\max}$ the highest variable and check degree in the distributions $\lambda(x)$ and $\rho(x)$, respectively. Nevertheless, the same notations will be used throughout this chapter. For instance, at the beginning of each iteration ℓ, the analysis partitions the set of all variable nodes into three (disjoint) sets: unverified non-zero variable nodes ($\mathcal{K}^{(\ell)}$), unverified zero-valued variable nodes ($\Delta^{(\ell)}$), and all variable nodes recovered up to iteration ℓ ($\mathcal{R}^{(\ell)}$). Should the fraction of variable nodes in the set $\mathcal{K}^{(\ell)}$ tend to zero as iterations proceed, the fraction of variable nodes in the set $\Delta^{(\ell)}$ will also tend to zero and consequently the analysis declares a successful recovery [40].

Each iteration in the SBB algorithm is divided into two rounds (R), each consisting of two half-rounds (HR). In the first and second rounds, verified variable nodes belong to the sets $\mathcal{K}^{(\ell)}$ and $\Delta^{(\ell)}$, respectively. The configuration of the sets at the end of each processing is specified using the superscript (ℓ, Rx, y), where ℓ, $x \in \{1,2\}$ and $y \in \{1,2\}$ denote the iteration, the round and the half-round numbers, respectively.

We partition the set of all check nodes with the same degree (say d_c) into sets $\mathcal{N}_{i,j}^{(\ell)}(d_c), 0 \le i \le d_c, 0 \le j \le d_c - i$, where i and j indicate the number of neighboring variable nodes in the sets $\mathcal{K}^{(\ell)}$ and $\Delta^{(\ell)}$, respectively.

Let $\mathcal{K}^{(\ell)}(d_v)$ and $\Delta^{(\ell)}(d_v)$ denote the set of all non-zero and zero-valued unverified variable nodes with the same degree d_v, respectively. Then, the set $\mathcal{K}^{(\ell)}(d_v)$ is further divided into subsets $\mathcal{K}_i^{(\ell)}(d_v), 0 \le i \le d_v$, where i denotes the number of neighboring check nodes in the set $\mathcal{N}_1^{(\ell)} := \bigcup_{d_c} \bigcup_{j=0}^{d_c-1} \mathcal{N}_{1,j}^{(\ell,R1,1)}(d_c)$. Also, we divide the set $\Delta^{(\ell)}(d_v)$ into subsets $\Delta_i^{(\ell)}(d_v), 0 \le i \le d_v$, with the following definition: a variable node in $\Delta_i^{(\ell)}(d_v)$ has i neighboring check nodes which became zero-valued after HR1 of R2; check nodes in the set $\left\{\bigcup_{d_c} \bigcup_{j=1}^{d_c} \mathcal{N}_{0,j}^{(\ell,R2,1)}(d_c) \setminus \bigcup_{d_c} \bigcup_{j=1}^{d_c} \mathcal{N}_{0,j}^{(\ell,R1,1)}(d_c)\right\}$. Table 5.1 summarizes the sets affected in each half-round of each round at any iteration. The formulation in the table assumes a variable node of degree d_v and a check node of

Table 5.1: Sets that change in each half-round of each round at any iteration, assuming a variable node of degree d_v and a check node of degree d_c. The degrees d_v and d_c can be any valid degree according to the distributions $\lambda(x)$ and $\rho(x)$.

R1		R2	
HR1	HR2	HR1	HR2
$\mathcal{N}_{k,i}(d_c) \to \mathcal{N}_{k,j}(d_c)$	$\mathcal{K}_i(d_v) \to \mathcal{K}_j(d_v)$	$\mathcal{N}_{i,k}(d_c) \to \mathcal{N}_{j,k}(d_c)$	$\Delta_i(d_v) \to \Delta_j(d_v)$

degree d_c. This table is indeed the generalization of Table 4.4.

Theorems 6 and 7 below, characterize the verification of unverified non-zero ($\mathcal{K}^{(\ell)}$) and zero-valued ($\Delta^{(\ell)}$) variable nodes at HR2-R1 and HR2-R2 in each iteration ℓ of the SBB algorithm, respectively. The theorems are generalized version of Theorems 4 and 5. The proofs of the theorems are rather straightforward and follow from the verification rules and therefore omitted.

Theorem 6. *In the first round of any iteration ℓ, a non-zero variable node of degree d is verified if and only if it belongs to the set $\bigcup_{i=2}^{d} \mathcal{K}_i^{(\ell,R1,2)}(d) \cup \hat{\mathcal{K}}_1^{(\ell,R1,2)}(d)$, where the set $\hat{\mathcal{K}}_1^{(\ell,R1,2)}(d)$ consists of all variable nodes in the set $\mathcal{K}_1^{(\ell,R1,2)}(d)$ connected to the set $\bigcup_{d_c} \mathcal{N}_{1,0}^{(\ell,R1,1)}(d_c)$.*

Theorem 7. *In the second round of any iteration ℓ, a zero-valued variable node of degree d is verified if and only if it belongs to the set $\bigcup_{i=1}^{d} \Delta_i^{(\ell)}(d)$.*

The sets $\mathcal{K}^{(\ell)}$, $\mathcal{R}^{(\ell)}$, $\Delta^{(\ell)}$, $\mathcal{N}_{i,j}^{(\ell-1,R2,2)}(d_c)$, $\mathcal{K}_i^{(\ell-1,R1,2)}(d_v)$, and $\Delta_i^{(\ell-1,R2,2)}(d_v)$, ($1 \leq d_c \leq d_{c,\max}$ and $1 \leq d_v \leq d_{v,\max}$) fully describe the state of the algorithm at the beginning of iteration ℓ. The probability that a variable node belongs to the set $\mathcal{K}^{(\ell)}$ is $\alpha^{(\ell)}$. Following the notation introduced in Chapter 4, for any other set Z, we use the notation $p_Z^{(\ell,Rx,y)}$ to denote the probability that a node belongs to the set $Z^{(\ell,Rx,y)}$.

The asymptotic analysis tracks the probability that a node (variable node or check node) belongs to a certain set at each half-round, round, or iteration. The recovery is successful if and only if the probability $\alpha^{(\ell)}$ tends to zero, as ℓ tends to infinity. As we shall see, the analysis is based on the derivation of recursive equations that relate the probabilities described above for two consecutive iterations. The complexity of the analysis thus scales linearly with the number of iterations. In the following section, we present such recursions for the SBB algorithm. The derivation of formulas are discussed in detail in Appendix B.

[1]These *concentration results* have been proved in detail for regular graphs in [47]. Similar results apply to the irregular sensing graphs with minor changes, and are therefore not presented here.

5.3 Update Rules for the SBB Algorithm

The step-by-step analysis of the algorithm are presented in Tables 5.2, 5.3, and 5.4 at the end of this section, where the update equations are identified by the round and the half-round they correspond to. The update equations in these tables involve the probabilities of the sets described in Table 4.5, as well as some other probabilities, defined in Table 4.6, generalized so that they include the variable and check degrees as well.

It is worth comparing the update equations of Tables 5.2, 5.3, and 5.4, for irregular graphs, with their counterparts in Tables 4.9, 4.10, and 4.11, for regular graphs. Generally speaking, the update equations for irregular graphs can be seen as a generalized version of their regular counterparts in two ways: some are generalized by including the node (variable or check) degree, and some are generalized through weighted combination of their regular counterparts, where weights are functions of the degree distributions $\lambda(x)$ and $\rho(x)$.

5.4 Simulation Results

To verify the asymptotic results obtained based on the analysis of this chapter, we perform some finite-length simulations for large values of n. The input signal in all simulations follows the probabilistic model described in Section 2.2. Also, each non-zero signal element is drawn from a standard Gaussian distribution (zero-mean with variance one). The graphs are constructed randomly with no parallel edges and all edge weights are chosen to be 1. Each simulation point is generated by averaging over 100 random instances of the input signal.

The graphs used in our finite-length simulations include variable nodes of degree 2. The presence of such variable nodes, in finite length simulations, might introduce graphical structures that prohibit some variable nodes from being verified [1, 45]. Based on our extensive simulations, the *number* of such unverified variable nodes does not scale with the increase of n. So for simulation purposes, we construct the graphs randomly and report the average (over 100 simulation instances) fraction of unverified variable nodes, instead of the success/failure ratio.

For the analytical results, based on the fact that $\alpha^{(\ell)}$ is a non-increasing function of iteration number ℓ, we consider the following stopping criteria:

1. Success: $\alpha^{(\ell)} \leq 10^{-7}$.
2. Failure: $\alpha^{(\ell)} > 10^{-7}$ and $|\alpha^{(\ell)} - \alpha^{(\ell-1)}| < 10^{-8}$.

To calculate the success threshold, a binary search of initial density factors is performed within a certain range including the threshold. The search continues until the search region is smaller than 10^{-5}.

Simulation results in this section are divided into two parts. In the first part, we discuss the accuracy of our analysis compared to that of [1]. In the second part, we present simulation results confirming the accuracy of our asymptotic analysis over graphs with different compression ratio.

5.4.1 Part1: Comparison with [1]

Authors in [1] have reported the following degree distribution as an optimized degree distribution for the case where all the check nodes have the same degree (also called *right-regular graph*), the highest variable degree is 12, the average variable degree is 3.5 and the compression ratio is 0.5:

$$\lambda(x) = 0.5201x^2+0.182x^3+0.09055x^4+0.10575x^6+0.1007x^9+0.0009x^{12}, \qquad \rho(x) = x^7.$$

In [1], the threshold for this case is reported as 0.303. Based on our analysis, we obtain the threshold of 0.3025 for this ensemble. We have simulated the evolution of $\alpha^{(\ell)}$ for the SBB algorithm vs. iteration number over the reported graph ($n = 10^5$) for initial density factors $0.3024, 0.3026$, and 0.3029. As can be seen in Fig. 5.1, $\alpha^{(\ell)}$ tends to zero only for the first initial density factor and is bounded away from zero for the other two values. This means that the threshold of 0.303 is indeed larger than the success threshold of this ensemble.

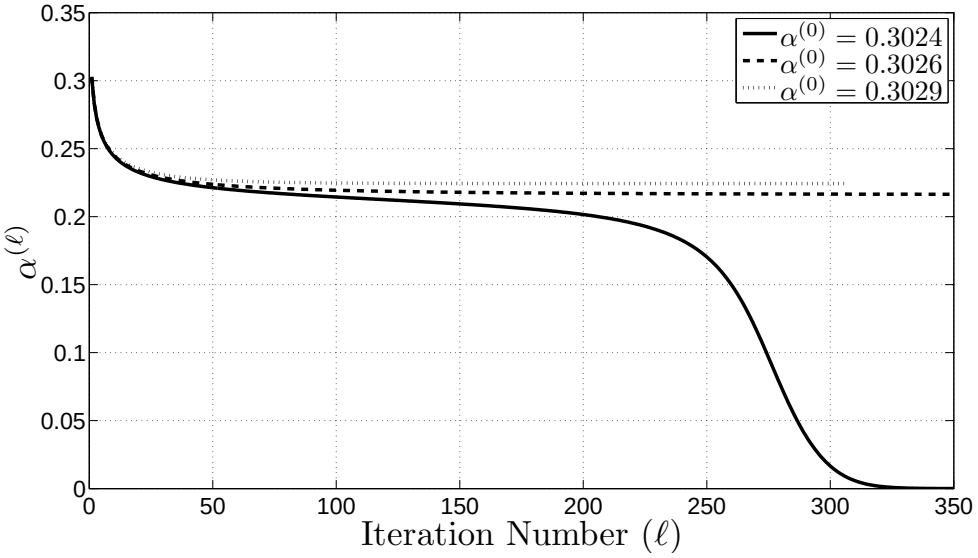


Figure 5.1: Evolution of $\alpha^{(\ell)}$ for the SBB algorithm, obtained by finite-length simulation, vs. iteration number ℓ. The evolution is reported for different initial density factors.

5.4.2 Part 2: Accuracy of the Proposed Analysis

To investigate the degree of agreement between our asymptotic analysis and finite-length simulations, we have presented in Fig. 5.2 the evolution of $\alpha^{(\ell)}$ (for the theoretical results) and the average unverified non-zero variable nodes normalized by n (for the finite-length results) with iterations ℓ for the SBB algorithm. The sensing graph in this case is randomly constructed with the degree distribution $\lambda(x) = 0.52x^2 + 0.2426x^3 + 0.0417x^4 + 0.1957x^8$ and $\rho(x) = x^7$. This degree distribution is optimized for a compression ratio of 0.5. The details of the optimization are further descrribed in Section 6.4.1. The success threshold associated with this degree distribution is 0.3066. For the simulation, two values of $\alpha^{(0)}$ are selected: one above the success threshold ($0.307 > 0.3066$) and one below it ($0.306 < 0.3066$). The theoretical results and simulations for $n = 10^5$ are shown by dashed lines and solid lines, respectively. As one can see, the sets of results are in close agreement particularly for the cases where $\alpha^{(0)}$ is above the threshold and for smaller values of ℓ.

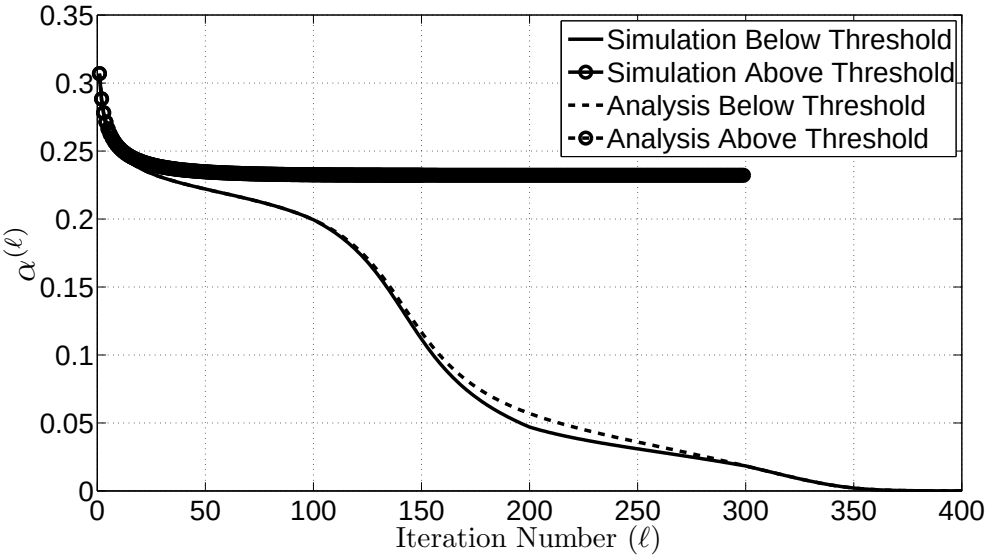


Figure 5.2: Evolution of $\alpha^{(\ell)}$, obtained by the theoretical analysis, vs. iteration number ℓ (dashed line) and that of the normalized average unverified non-zero variable nodes vs. ℓ for $n = 10^5$ (solid line).

For the last simulation, we select the following two degree distribution pairs:

$$\lambda_1(x) = 0.62x^2 + 0.2457x^3 + 0.0619x^6 + 0.0724x^9, \rho_1(x) = x^4$$
$$\lambda_2(x) = 0.52x^2 + 0.1673x^3 + 0.1054x^4 + 0.2073x^5, \rho_2(x) = x^9$$

For each degree distribution, we construct three random graphs with $n = 10^4$, $n = 5 \times 10^4$, and $n = 10^5$ variable nodes. The degree distributions yield compression ratios of 3/4 and 1/3, and have success thresholds of 0.5390 and 0.1619 under the SBB algorithm, respectively. We run the SBB algorithm over these graphs for different initial density factors. Figure 5.3, plots the average number of unverified variable nodes normalized by n vs. the initial density factor. In this simulation, we impose no limitation on the number of iterations and the unverified variable nodes are counted when the algorithm makes no further progress. Clearly, if the recovery algorithm has successfully recovered all variable nodes at this point, the number of unverified variable nodes is zero. In each case, the success threshold is demonstrated by a vertical line on the figure. The diagonal line represents the function $y(x) = x$, which serves as the asymptote for the cases where the initial density factor tends to 1. As can be seen, the finite-length results are in good agreement with the theoretical thresholds. Also, as the dimension n increases, the curves become sharper around the success threshold.

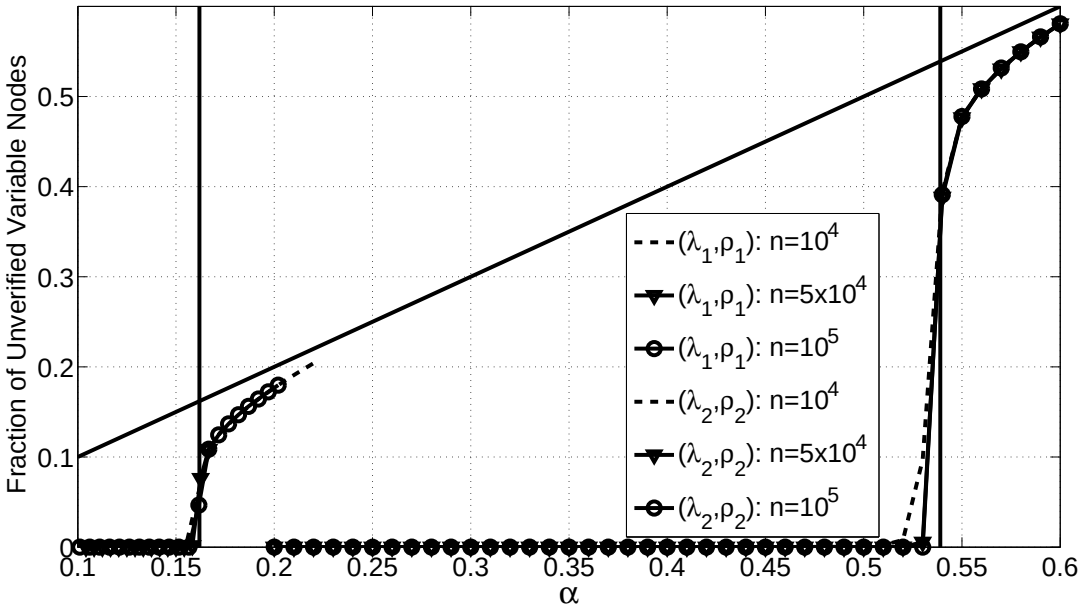


Figure 5.3: Verifying the success threshold of SBB for irregular graphs through finite-length simulations.

Table 5.2: Initialization of Parameters for the Analysis of SBB on Irregular Graphs with Inputs: $\lambda(x)$, $\rho(x)$, $\alpha^{(0)}$

$$D^{(0)} = \frac{1}{\bar{d}_c}\sum_{i=1}^{d_{c,\max}} i\rho_i\left(1-\alpha^{(0)}\right)^{i-1}, \qquad A^{(1)} = \frac{1}{\bar{d}_v}\sum_{d=1}^{d_{v,\max}} d\lambda_d(1-D^{(0)})^{d-1}.$$

$$p^{(0,R2,1)}_{\mathcal{N}_{i,d_c-i}}(d_c) = \binom{d_c}{i}\left(\alpha^{(0)}\right)^{i}\left(1-\alpha^{(0)}\right)^{d_c-i}, \qquad i = 0,\cdots,d_c.$$

$$p^{(1,R1)}_{\mathcal{N}_{i,d_c-i\downarrow j}}(d_c) = \binom{d_c-i}{j}\left(A^{(1)}\right)^{j}\left(1-A^{(1)}\right)^{d_c-i-j}, \qquad i = 1,\cdots,d_c, \qquad j = 0,\cdots,d_c-i.$$

$$p^{(1,R1,1)}_{\mathcal{N}_{i,j}}(d_c) = p^{(0,R2,1)}_{\mathcal{N}_{i,d_c-i}}(d_c)p^{(1,R1)}_{\mathcal{N}_{i,d_c-i\downarrow j}}(d_c), \qquad i = 1,\cdots,d_v, \qquad j = 0,\cdots,d_c-i.$$

$$p^{(1)}_{\Delta}(d) = (1-\alpha^{(0)})\left(1-D^{(0)}\right)^{d}, \quad d = 1,\cdots,d_{v,\max}, \quad \alpha^{(1)} = \alpha^{(0)}.$$

$$B^{(1)} = \frac{\sum_{d=1}^{d_{c,\max}}\rho_d\sum_{j=0}^{d-1}p^{(1,R1,1)}_{\mathcal{N}_{1,j}}(d)}{\alpha^{(1)}\bar{d}_c}, \qquad f^{(1,R1)} = \frac{\sum_{d=1}^{d_{c,\max}}\rho_d p^{(1,R1,1)}_{\mathcal{N}_{1,0}}(d)}{\sum_{d=1}^{d_{c,\max}}\sum_{j=0}^{d-1}\rho_d p^{(1,R1,1)}_{\mathcal{N}_{1,j}}(d)}.$$

$$N^{(1,R1)}(d) = \left(1-B^{(1)}\right)^{d} + dB^{(1)}\left(1-f^{(1,R1)}\right)\left(1-B^{(1)}\right)^{d-1}, \quad \alpha^{(2)}_d = \alpha^{(1)}N^{(1,R1)}(d).$$

$$p^{(1,R1,2)}_{\mathcal{K}_0}(d) = \frac{\left(1-B^{(1)}\right)^{d}}{N^{(1,R1)}(d)}, p^{(1,R1,2)}_{\mathcal{K}_1}(d) = \frac{dB^{(1)}\left(1-f^{(1,R1)}\right)\left(1-B^{(1)}\right)^{d-1}}{N^{(1,R1)}(d)},$$

$$p^{(1,R1,2)}_{\mathcal{K}_i}(d) = 0, 2 \le i \le d.$$

$$C^{(1)} = 1 - \frac{\sum_{i=1}^{d_{v,\max}} i\lambda_i\alpha^{(1)}_i p^{(1,R1,2)}_{\mathcal{K}_0}(i)}{\sum_{i=1}^{d_{v,\max}}\sum_{j=1}^{i}(i-j)\lambda_i\alpha^{(1)}_i p^{(1,R1,2)}_{\mathcal{K}_j}(i)} - \frac{\sum_{i=1}^{d_{v,\max}}(i-1)\lambda_i\alpha^{(1)}_i p^{(1,R1,2)}_{\mathcal{K}_1}(i)}{\sum_{i=1}^{d_{v,\max}}\sum_{j=1}^{i}(i-j)\lambda_i\alpha^{(1)}_i p^{(1,R1,2)}_{\mathcal{K}_j}(i)}\left(1-f^{(1,R1)}\right).$$

$$p^{(1,R2)}_{\mathcal{N}_{i\downarrow k,j}}(d_c) = \binom{i}{k}\left(C^{(1)}\right)^{i-k}\left(1-C^{(1)}\right)^{k}, \quad 2 \le i \le d_c, 0 \le k \le i, \quad 0 \le j \le d_c - i.$$

$$p^{(1,R2)}_{\mathcal{N}_{1\downarrow 1,j}}(d_c) = \frac{\sum_{i=1}^{d_{v,\max}}\lambda_i\alpha^{(1)}_i p^{(1,R1,2)}_{\mathcal{K}_1}(i)}{\sum_{i=1}^{d_{v,\max}}\lambda_i\alpha^{(1)}_i\sum_{j=1}^{i} j p^{(1,R1,2)}_{\mathcal{K}_j}(i)}, \; p^{(1,R2)}_{\mathcal{N}_{1\downarrow 0,j}}(d_c) = 1 - p^{(1,R2)}_{\mathcal{N}_{1\downarrow 1,j}}(d_c), \; 1 \le j \le d_c - i.$$

$$p^{(1,R2)}_{\mathcal{N}_{1\downarrow 0,0}}(d_c) = 1, \qquad p^{(1,R2)}_{\mathcal{N}_{1\downarrow 1,0}}(d_c) = 0, \qquad p^{(1,R2)}_{\mathcal{N}_{0\downarrow 0,j}}(d_c) = 1, \qquad j = 1,\cdots,d_c - i.$$

$$p^{(1,R2,1,+)}_{\mathcal{N}_{1,j}}(d_c) = \sum_{i=2}^{d_c} p^{(1,R1,1)}_{\mathcal{N}_{i,j}}(d_c)p^{(1,R2)}_{\mathcal{N}_{i\downarrow 1,j}}(d_c), \; p^{(1,R2,1,C)}_{\mathcal{N}_{1,j}}(d_c) = p^{(1,R2,1)}_{\mathcal{N}_{1,j}}(d_c) - p^{(1,R2,1,+)}_{\mathcal{N}_{1,j}}(d_c).$$

$$p^{(1,R2,1)}_{\mathcal{N}_{k,j}}(d_c) = \sum_{i=k}^{d_c} p^{(1,R1,1)}_{\mathcal{N}_{i,j}}(d_c)p^{(1,R2)}_{\mathcal{N}_{i\downarrow k,j}}(d_c), \qquad k = 0,\cdots,d_c, \qquad j = 0,\cdots,d_c - i.$$

$$D^{(1)} = \frac{\sum_{d=1}^{d_{c,\max}}\rho_d\sum_{j=1}^{d-1} j p^{(1,R2,1)}_{\mathcal{N}_{0,j}}(d)}{\sum_{d=1}^{d_{c,\max}}\rho_d\sum_{i=0}^{d}\sum_{j=1}^{d-i} j p^{(1,R2,1)}_{\mathcal{N}_{i,j}}(d)}, \quad p^{(2)}_{\Delta}(d) = p^{(1)}_{\Delta}(d)\left(1-D^{(1)}\right)^{d}.$$

Table 5.3: Recursive Formulas for the Analysis of SBB over Irregular Graphs for $\ell \geq 2$, First Round

HR1 1) $$A^{(\ell)} = \frac{1}{\bar{d}_v} \sum_{d=1}^{d_{v,\max}} d\lambda_d (1 - D^{(\ell-1)})^{d-1}.$$

2) $$p_{\mathcal{N}_{i,j\downarrow k}}^{(\ell,R1)}(d_c) = \binom{j}{k} \left(A^{(\ell)}\right)^k \left(1 - A^{(\ell)}\right)^{j-k}, \ 1 \leq i \leq d_c, \ 0 \leq j \leq d_c - i, \ 0 \leq k \leq j.$$

3) $$p_{\mathcal{N}_{1,k}}^{(\ell,R1,1,+)}(d_c) = \sum_{j=k}^{d_c-1} p_{\mathcal{N}_{1,j}}^{(\ell-1,R2,1,+)}(d_c) p_{\mathcal{N}_{1,j\downarrow k}}^{(\ell,R1)}(d_c), \ 1 \leq d_c \leq d_{c,\max}, \ 0 \leq k \leq d_c - 1.$$

$$p_{\mathcal{N}_{1,k}}^{(\ell,R1,1,C)}(d_c) = \sum_{j=k}^{d_c-1} p_{\mathcal{N}_{1,j}}^{(\ell-1,R2,1,C)}(d_c) p_{\mathcal{N}_{1,j\downarrow k}}^{(\ell,R1)}(d_c), \ 1 \leq d_c \leq d_{c,\max}, \ 0 \leq k \leq d_c - 1.$$

$$p_{\mathcal{N}_{i,k}}^{(\ell,R1,1)}(d_c) = \sum_{j=k}^{d_c-i} p_{\mathcal{N}_{i,j}}^{(\ell-1,R2,1)}(d_c) p_{\mathcal{N}_{i,j\downarrow k}}^{(\ell,R1)}(d_c), \ 1 \leq d_c \leq d_{c,\max}, \ 2 \leq i \leq d_c, \ 0 \leq k \leq d_c - i.$$

HR2 1) $$B^{(\ell)} = \frac{\sum_{d=1}^{d_{c,\max}} \sum_{k=0}^{d-1} \rho_d p_{\mathcal{N}_{1,k}}^{(\ell,R1,1,+)}(d)}{\sum_{d=1}^{d_{c,\max}} \sum_{k=0}^{d-1} \rho_d p_{\mathcal{N}_{1,k}}^{(\ell,R1,1,+)}(d) + \sum_{d=1}^{d_{c,\max}} \sum_{i=2}^{d} \sum_{j=0}^{d-i} i\rho_d p_{\mathcal{N}_{i,j}}^{(\ell,R1,1)}(d)}.$$

2) $$p_{\mathcal{K}_{i\uparrow j}}^{(\ell,R1)}(d_v) = \binom{d_v - i}{j - i} \left(B^{(\ell)}\right)^{j-i} \left(1 - B^{(\ell)}\right)^{d_v - j}, \qquad i = 0, 1, \qquad i \leq j \leq d_v.$$

3) $$p_{\mathcal{K}_j}^{(\ell,R1)}(d_v) = \sum_{i=0}^{1} p_{\mathcal{K}_i}^{(\ell-1,R1,2)}(d_v) p_{\mathcal{K}_{i\uparrow j}}^{(\ell,R1)}(d_v), \qquad 0 \leq j \leq d.$$

4) $$f^{(\ell,R1,+)} = \frac{\sum_{d=1}^{d_{c,\max}} \rho_d p_{\mathcal{N}_{1,0}}^{(\ell,R1,1,+)}(d)}{\sum_{d=1}^{d_{c,\max}} \rho_d \sum_{k=0}^{d-1} p_{\mathcal{N}_{1,k}}^{(\ell,R1,1,+)}(d)}, \qquad f^{(\ell,R1,C)} = \frac{\sum_{d=1}^{d_{c,\max}} \rho_d p_{\mathcal{N}_{1,0}}^{(\ell,R1,1,C)}(d)}{\sum_{d=1}^{d_{c,\max}} \rho_d \sum_{k=0}^{d-1} p_{\mathcal{N}_{1,k}}^{(\ell,R1,1,C)}(d)}.$$

$$N^{(\ell,R1)}(d) = p_{\mathcal{K}_0}^{(\ell-1,R1,2)}(d) p_{\mathcal{K}_{0\uparrow 0}}^{(\ell,R1)}(d) + p_{\mathcal{K}_0}^{(\ell-1,R1,2)}(d) p_{\mathcal{K}_{0\uparrow 1}}^{(\ell,R1)}(d) \left(1 - f^{(\ell,R1,+)}\right)$$
$$+ p_{\mathcal{K}_1}^{(\ell-1,R1,2)}(d) p_{\mathcal{K}_{1\uparrow 1}}^{(\ell,R1)}(d) \left(1 - f^{(\ell,R1,C)}\right).$$

5) $$p_{\mathcal{K}_0}^{(\ell,R1,2)}(d) = \frac{1}{N^{(\ell,R1)}(d)} p_{\mathcal{K}_0}^{(\ell-1,R1,2)}(d) p_{\mathcal{K}_{0\uparrow 0}}^{(\ell,R1)}(d).$$

$$p_{\mathcal{K}_1}^{(\ell,R1,2,+)}(d) = \frac{1}{N^{(\ell,R1)}(d)} p_{\mathcal{K}_0}^{(\ell-1,R1,2)}(d) p_{\mathcal{K}_{0\uparrow 1}}^{(\ell,R1)}(d) \left(1 - f^{(\ell,R1,+)}\right).$$

$$p_{\mathcal{K}_1}^{(\ell,R1,2,C)}(d) = \frac{1}{N^{(\ell,R1)}(d)} p_{\mathcal{K}_1}^{(\ell-1,R1,2)}(d) p_{\mathcal{K}_{1\uparrow 1}}^{(\ell,R1)}(d) \left(1 - f^{(\ell,R1,C)}\right).$$

$$p_{\mathcal{K}_j}^{(\ell,R1,2)}(d) = 0, \qquad j = 2, \cdots, d.$$

6) $$\alpha_d^{(\ell+1)} = \alpha_d^{(\ell)} N^{(\ell,R1)}(d).$$

Table 5.4: Recursive Formulas for the Analysis of SBB over Irregular Graphs for $\ell \geq 2$, Second Round

HR1 1) $$C^{(\ell)} = 1 - \sum_{i=1}^{d_{v,\max}} \frac{(i-1)\lambda_i \alpha_i^{(\ell)} p_{\mathcal{K}_0}^{(\ell-1,R1,2)}(i) p_{\mathcal{K}_{0\uparrow 1}}^{(\ell,R1)}(i)}{\sum_{i=1}^{d_{v,\max}} \sum_{j=1}^{i} (i-j)\lambda_i \alpha_i^{(\ell)} p_{\mathcal{K}_j}^{(\ell,R1)}(i)} \left(1 - f^{(\ell,R1,+)}\right)$$

$$- \frac{\sum_{i=1}^{d_{v,\max}} (i-1)\lambda_i \alpha_i^{(\ell)} p_{\mathcal{K}_1}^{(\ell-1,R1,2)}(i) p_{\mathcal{K}_{1\uparrow 1}}^{(\ell,R1)}(i)}{\sum_{i=1}^{d_{v,\max}} \sum_{j=1}^{i} (i-j)\lambda_i \alpha_i^{(\ell)} p_{\mathcal{K}_j}^{(\ell,R1)}(i)} \left(1 - f^{(\ell,R1,C)}\right) - \frac{\sum_{i=1}^{d_{v,\max}} i\lambda_i \alpha_i^{(\ell)} p_{\mathcal{K}_0}^{(\ell,R1)}(i)}{\sum_{i=1}^{d_{v,\max}} \sum_{j=1}^{i} (i-j)\lambda_i \alpha_i^{(\ell)} p_{\mathcal{K}_j}^{(\ell,R1)}(i)}.$$

2) $$p_{\mathcal{N}_{i\downarrow k,j}}^{(\ell,R2)}(d_c) = \binom{i}{k} \left(C^{(\ell)}\right)^{i-k} \left(1 - C^{(\ell)}\right)^{k}, \quad 1 \leq d_c \leq d_{c,\max}, \quad 2 \leq i \leq d_c,$$
$$0 \leq k \leq i, \quad 0 \leq j \leq d_c - i.$$

$$p_{\mathcal{N}_{1,i}}^{(\ell,R2,+,F)}(d_c) = \frac{p_{\mathcal{N}_{1,i}}^{(\ell,R1,1,+)}(d_c) \sum_{d=1}^{d_{v,\max}} \lambda_d \alpha_d^{(\ell)} p_{\mathcal{K}_0}^{(\ell-1,R1,2)}(d) p_{\mathcal{K}_{0\uparrow 1}}^{(\ell,R1)}(d)}{\sum_{d=1}^{d_{v,\max}} \sum_{i=1}^{d} \lambda_d \alpha_d^{(\ell)} \left(i p_{\mathcal{K}_0}^{(\ell-1,R1,2)}(d) p_{\mathcal{K}_{0\uparrow i}}^{(\ell,R1)}(d) + (i-1) p_{\mathcal{K}_1}^{(\ell-1,R1,2)}(d) p_{\mathcal{K}_{1\uparrow i}}^{(\ell,R1)}(d) \right)}.$$

$$p_{\mathcal{N}_{1,i}}^{(\ell,R2,+,O)} = p_{\mathcal{N}_{1,i}}^{(\ell,R1,1,+)} - p_{\mathcal{N}_{1,i}}^{(\ell,R2,+,F)}.$$

$$p_{\mathcal{N}_{1,i}}^{(\ell,R2,C,F)}(d_c) = p_{\mathcal{N}_{1,i}}^{(\ell,R1,1,C)}(d_c) \sum_{d=1}^{d_{v,\max}} \frac{\lambda_d \alpha_d^{(\ell)} p_{\mathcal{K}_1}^{(\ell-1,R1,2)}(d) p_{\mathcal{K}_{1\uparrow 1}}^{(\ell,R1)}(d)}{\sum_{d=1}^{d_{v,\max}} \sum_{i=1}^{d} \lambda_d \alpha_d^{(\ell)} p_{\mathcal{K}_1}^{(\ell-1,R1,2)}(d) p_{\mathcal{K}_{1\uparrow i}}^{(\ell,R1)}(d)}.$$

$$p_{\mathcal{N}_{1,i}}^{(\ell,R2,C,O)}(d_c) = p_{\mathcal{N}_{1,i}}^{(\ell,R1,1,C)}(d_c) - p_{\mathcal{N}_{1,i}}^{(\ell,R2,C,F)}(d_c).$$

3) $$p_{\mathcal{N}_{k,j}}^{(\ell,R2,1)}(d_c) = \sum_{i=k}^{d_c} p_{\mathcal{N}_{i,j}}^{(\ell,R1,1)}(d_c) p_{\mathcal{N}_{i\downarrow k,j}}^{(\ell,R2)}(d_c), \qquad 2 \leq k \leq d_c, \qquad 0 \leq j \leq d_c - i.$$

$$p_{\mathcal{N}_{1,j}}^{(\ell,R2,1,+)}(d_c) = \sum_{i=2}^{d_c} p_{\mathcal{N}_{i,j}}^{(\ell,R1,1)}(d_c) p_{\mathcal{N}_{i\downarrow 1,j}}^{(\ell,R2)}(d_c), \qquad 0 \leq j \leq d_c - 1.$$

$$p_{\mathcal{N}_{1,j}}^{(\ell,R2,1,C)}(d_c) = p_{\mathcal{N}_{1,j}}^{(\ell,R2,C,F)}(d_c) + p_{\mathcal{N}_{1,j}}^{(\ell,R2,+,F)}(d_c), \qquad 1 \leq j \leq d_c - 1.$$

$$p_{\mathcal{N}_{0,j}}^{(\ell,R2,1)}(d_c) = p_{\mathcal{N}_{0,j}}^{(\ell,R1,1)}(d_c) + p_{\mathcal{N}_{1,j}}^{(\ell,R2,C,O)}(d_c) + p_{\mathcal{N}_{1,j}}^{(\ell,R2,+,O)}(d_c)$$
$$+ \sum_{i=2}^{d_c} p_{\mathcal{N}_{i,j}}^{(\ell,R1,1)}(d_c) p_{\mathcal{N}_{i\downarrow 0,j}}^{(\ell,R2)}(d_c), \qquad 0 \leq j \leq d_c - 1.$$

HR2 1) $$D^{(\ell)} = \frac{\sum_{d=1}^{d_{c,\max}} \rho_d \sum_{j=1}^{d-1} j p_{\mathcal{N}_{0,j}}^{(\ell,R2,1)}(d)}{\sum_{d=1}^{d_{c,\max}} \rho_d \sum_{i=0}^{d} \sum_{j=1}^{d-i} j p_{\mathcal{N}_{i,j}}^{(\ell,R2,1)}(d)}.$$

2) $$p_{\Delta_i}^{(\ell,R2,2)}(d) = \binom{d}{i} \left(D^{(\ell)}\right)^{i} \left(1 - D^{(\ell)}\right)^{d-i}, \qquad 0 \leq i \leq d.$$

3) $$p_{\Delta}^{(\ell+1)}(d) = p_{\Delta}^{(\ell)}(d) p_{\Delta_0}^{(\ell,R2,2)}(d).$$

Chapter 6

Design of Irregular Graphs

6.1 Introduction

The main focus of this chapter is on the design of irregular graphs for the SBB algorithm. Since the analysis presented in the previous chapter is simple to perform, requiring only additions and multiplications, it is possible to use it at the core of an optimization loop to design degree distributions for irregular sensing graphs that perform well with NB-VB algorithms. The performance measure considered here is the success threshold. Our simulations show that for a given compression ratio m/n, irregular graphs can provide up to %40 larger success thresholds compared to regular graphs. In this comparison, since the number of edges in both graphs is the same, the recovery complexity remains almost the same.

6.2 Optimization

In the following, we fix the compression ratio m/n and the largest variable and check degrees $d_{v,\max}$ and $d_{c,\max}$. Our goal is then to find degree distributions $\lambda(x)$ and $\rho(x)$ such that the success threshold of the recovery algorithm under consideration is maximized. As discussed earlier, the SBB algorithm has the highest success threshold amongst NB-VB algorithms (under similar conditions). Hence, in this chapter, we focus on optimizing degree distributions for the SBB algorithm.

To find the optimal degree distributions, we search the entire space of degree distributions constrained by the conditions on the compression ratio and the maximum variable and check degrees. In order to reduce the complexity of such a search, we restrict the number of non-zero variable and check degrees as well. In the context of coding, it is known that the loss in terms of threshold due to such restrictions is rather small and would be justifiable by the reduction in the search complexity [54–56].

The results presented in this chapter are divided into three parts. In the first part, we present degree distributions optimized for the SBB algorithm. The obtained degree distributions outperform the one presented in [1] with a smaller number of non-zero degrees, and a smaller maximum variable degree. In the second part, we

present optimal degree distributions for graphs with compression ratios 1/3 and 3/4. We also discuss the performance improvement due to the use of irregular graphs. In the last part, we find optimal degree distributions for graphs with relatively low compression ratios of 1/10, 3/32 and 3/40.

6.3 Running Time and Computational Complexity

In this chapter, we design irregular graphs and use the success threshold as the measure of performance. It would be of interest to evaluate the designed irregular graphs on measures such as recovery complexity and running time as well. A suitable measure for the running time in the finite-length regime is the number of iterations needed for the recovery algorithm to verify all variable nodes. In the asymptotic regime, the number of iterations needed for the recovery algorithm so that the probability of unverified variable nodes is less than a threshold serves as the running time measure. As we shall see, the running time is a function of the initial density factor. In general, for density factors less than the success threshold, the higher the initial density factor, the higher the running time.

As for the computational complexity (also referred to as recovery complexity), each iteration of a NB-VB algorithm, has a computational complexity proportional to the number of edges in the subgraph induced by the unverified variable nodes, which itself is upper bounded by $|\mathcal{E}|$, where $|\mathcal{E}|$ denotes the number of edges in the initial sensing graph. One should however note that using $|\mathcal{E}|$ as an upper-bound for the computational complexity at each iteration, is a loose bound especially for higher iterations where $|\mathcal{E}|^{(\ell)} \ll |\mathcal{E}|$.

Using a similar justification, it can be shown that the computational complexity in each iteration of a NB-VB algorithm is proportional to the average variable and check degree in the asymptotic regime. In this case, however, the complexity is normalized to the signal dimension n to yield a finite value.

6.4 Simulation Results and Discussions

6.4.1 Comparison with [1]

Authors in [1] reported the following degree distribution as an optimized degree distribution for the case where the highest variable degree is 12, the average variable degree is 3.5, and the graph is right-regular with compression ratio 0.5:

$$\lambda(x) = 0.5201x^2+0.182x^3+0.09055x^4+0.10575x^6+0.1007x^9+0.0009x^{12}, \qquad \rho(x) = x^7.$$

In Section 5.4.1, we demonstrated that the threshold 0.303 reported in [1] for this degree distribution is not accurate and the correct threshold is indeed 0.3025.

Table 6.1: Two Optimized Degree Distributions with Success Thresholds Higher than 0.3025. The Degree Distributions Yield a Compression Ratio of 0.5.

$\lambda(x)$	$\rho(x)$	Success Threshold
$0.5200x^2 + 0.2426x^3 + 0.0417x^4 + 0.1957x^8$	x^7	0.3066
$0.5200x^2 + 0.2426x^3 + 0.0542x^4 + 0.1832x^8$	$0.1x^6 + 0.9x^7$	0.3056

Apart from the accuracy, we have been able to find different degree distributions that could achieve a higher success threshold. These degree distributions are found through exhaustive search and two of them are presented in Table 6.1. We would like to emphasize that the exhaustive search was possible thanks to the fast running time of our analysis compared to the one used in [1].

The first degree distribution represents a right-regular graph, with the same check degree and compression ratio as those of [1]. In this case, we were able to achieve a higher success threshold with a smaller maximum variable degree (8 compared to 12) and less number of non-zero variable degrees (4 compared to 6). To find this degree distribution we ran an exhaustive search over the whole space of degree distributions with only four variable degrees with the maximum variable degree of 12 and considering an increment of 10^{-2} for λ_2. In this setup, there are four variable degree coefficients to be selected. Of this four, two are dependent variables due to the fact that the sum of the coefficients must add up to 1 and that the average variable degree should be satisfied. Fixing the value of λ_2, only one degree of freedom remains. The remaining value is chosen using the "fminbnd" optimization function in MATLAB.

For the next degree distribution, we allowed four non-zero variable degrees between 2 and 12 and two non-zero check degrees between 3 and 7. Due to the freedom on the average check degree in this case, we searched the space of all possible check degrees so that the average check degree is from 6 to 6.9 with increments of 0.1. We first note that the optimal check degree has the highest possible average of 6.9. Also, we note that the two non-zero check degrees are the highest two degrees possible, i.e., 6 and 7. This demonstrates that fixing the compression ratio, the success threshold improves as the average check degree (and consequently the average variable degree) is increased.

According to Table 4.13, a $(3, 6)$ regular graph yields the highest success threshold (0.2574) over regular graphs with compression ratio 0.5. Degree distributions shown in Table 6.1 all have the same compression ratio and outperform the regular graph by at least 18.5%. It is worth mentioning that by increasing the average variable degree (and hence the average check degree) higher success thresholds are achievable at the cost of higher recovery complexity.

At this point, it is worth to examine the recovery complexity and running time

Table 6.2: Number of iterations ℓ^* for the SBB algorithm over different graphs and different initial density factors $\alpha^{(0)}$. The first column represents the initial density factor, while each row represents the number of iterations for various graphs of interest.

		Type of Graph		
		Regular	Right-Regular	Bi-Irregular
Initial Density Factor	0.2316	13	33	33
	0.2445	18	40	39
	0.2750	–	69	69
	0.2903	–	110	111
	0.2759	–	71	70
	0.2912	–	114	116

of the designed irregular graphs and compare them against the $(3, 6)$ regular graph. Note that degree distributions shown in Table 6.1 have the same compression ratio as the $(3, 6)$ regular graph but have higher average variable degrees. Hence, their higher success threshold comes at the cost of higher recovery complexity.

To compare the running time of the designed irregular graphs and the $(3, 6)$ regular graph, we focus on the asymptotic regime only. The reason being, as shown in Sections 4.8.2 and 5.4.2, for a fixed initial density factor, the evolution of $\alpha^{(\ell)}$ vs. iteration number ℓ for the finite length simulations can be predicted relatively accurately using the asymptotic analysis. For this discussion the number of iterations of interest is $\ell^* : \alpha^{(\ell^*)} < 10^{-8}$. Table 6.2 summarizes the number of iterations ℓ^* for various initial density factors when running the SBB algorithm over graphs mentioned before. The initial density factors are selected as the %90 and %95 of the success thresholds associated with regular and bi-irregular sensing graphs, respectively.

First we note that the number of iterations needed for regular graph is considerably lower compared to other graphs. This means that when considering the use of regular vs. irregular graphs, one has to consider the trade-off between performance (measured by success threshold), recovery complexity (measured by the average variable and check degrees) and running time (measured by the required number of iterations). The regular graph in this example has a smaller success threshold but is computationally less complex and requires smaller number of iterations.

Second, we note that for the same initial density factor, the number of iterations needed for the bi-irregular graph is almost the same as those needed for the right-regular graph, which is expected as the two degree distributions are rather close. Third, we note that the number of iterations increases as the initial density factor

Table 6.3: Optimized Degree Distributions for Compression Ratios (r_c) 1/3 and 3/4

r_c	$(\lambda(x), \rho(x))$	Success Threshold	Δ (%)
1/3	$(0.5200x^2 + 0.1673x^3 + 0.1054x^4 + 0.2073x^5, x^9)$	0.1619	7.5
3/4	$(0.6200x^2 + 0.2457x^3 + 0.0619x^6 + 0.0724x^9, x^4)$	0.5390	20.0

approaches the success threshold. This is because increasing the initial density factor, the fraction of non-zero variable nodes increases, which directly translates to more iterations needed to recover them. Also, as the initial density factor gets closer to the success threshold, a slight increase in this parameter causes a significant increase in the number of iterations.

6.4.2 Optimal Degree Distributions for Graphs with Compression Ratios 1/3 and 3/4

In this section, we present the optimal degree distributions for compression ratios 1/3 and 3/4. According to Table 4.12, a regular $(3, 4)$ graph yields a success threshold of 0.4488. Using the analysis presented in Chapter 4, it is easy to verify the success threshold of a regular $(3, 9)$ graph to be 0.1506 for the SBB algorithm. It is also easy to verify that these graphs yield the highest success thresholds for compression ratios 3/4 and 1/3, respectively.

To find optimal degree distributions for these compression ratios, we searched over the space of all right-regular graphs with four non-zero variable degrees and the maximum variable degree of 10. The check degrees for these graphs were selected the same as the regular graphs yielding the highest success thresholds in each case. The result of optimization is reported in Table 6.3. In this table the last column shows the percentage of improvement over the success threshold of the corresponding regular graphs.

6.4.3 Optimal Degree Distributions for Graphs with Relatively Low Compression Ratios

As the last set of results, we present the optimal degree distributions yielding the highest success thresholds for graphs with low compression ratios. The results are summarized in Table 6.5. The last column of the table reports the percentage of improvement over the success threshold of regular graphs with similar compression ratio and average degrees.

All optimizations are run as an exhaustive search over the set of all right-regular graphs with three variable degrees, all less than 12. The average variable degrees for

Table 6.4: Optimized Degree Distributions for Low Compression Ratios (r_c)

r_c	$(\lambda(x), \rho(x))$	Success Threshold	Δ (%)
1/10	$(0.3527x^2 + 0.1963x^3 + 0.4510x^6, x^{40})$	0.0446	17.3
3/32	$(0.0001x^2 + 0.8123x^3 + 0.1876x^7, x^{40})$	0.0367	17.2
3/32	$(0.0001x^2 + 0.9998x^3 + 0.0001x^4, x^{32})$	0.0314	0.3
3/40	$(0.0001x^2 + 0.9998x^3 + 0.0001x^4, x^{40})$	0.0241	0.4

the rows of Table 6.5 are 4, 3.75, 3 and 3, while the check degrees are 40, 40, 32, and 40, respectively. The regular graphs used for comparison are $(4, 40)$, $(3, 32)$ and $(3, 40)$, with success thresholds 0.0380, 0.0313 and 0.0240, respectively. For compression ratio 3/32, we have included two different degree distributions, with check degrees 32 and 40. We first note that the last two degree distributions reported in the table perform essentially the same as their corresponding regular graphs. Also focusing on the results for compression ratio 3/32, we note that the higher the average variable node, the higher the success threshold.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we studied the asymptotic behavior of iterative node-based verification-based (NB-VB) recovery algorithms over random sparse graphs in the context of compressed sensing. Such algorithms are particularly interesting due to their low complexity (linear in the signal dimension n).

We proposed a mathematical framework that predicts the average fraction of unverified signal elements at each iteration ℓ in the asymptotic regime, where the average is taken over the ensembles of input signals and sensing graphs as a function of ℓ as $n \to \infty$. To perform the analysis, a non-extrinsic message-passing interpretation of NB-VB algorithms was provided. The analysis of recovery algorithms of interest were discussed over random regular and irregular sensing graphs.

We showed that the proposed analysis is simpler to implement and more accurate compared to the existing technique for the analysis of NB-VB algorithms, which is based on numerically solving a large system of coupled differential equations. We further used the proposed analysis in an optimization loop and designed irregular sensing graphs that outperformed previously reported results. We also showed that, in many cases, the success threshold associated with the designed irregular graphs exceeds that of the regular graphs substantially.

Moreover, we discussed concentration results that ensure the performance of the recovery algorithms applied to any choice of the input signal over any realization of the sensing matrix follows the deterministic results of the analysis closely. We also demonstrated that the proposed asymptotic analysis matches the performance of recovery algorithms for large but finite values of n.

7.2 Future Work

To optimize the degree distributions, two general approaches are possible. First is to search the entire space of feasible solutions. Second is to start with an initial guess for the degree distributions, find the success threshold, change the distributions according

to a strategy (e.g., differential evolution [57]) and compare the success threshold of the new pair with the previous results, and accordingly modify the solution. The method of differential evolution has been successfully applied in the context of coding to optimize degree distributions for recovery algorithms over different channels [58,59]. It would be interesting to examine the application of differential evolution to the design of irregular sensing graphs in the context of compressed sensing. As part of such study, one would need to prove that certain parameters in the recovery algorithm remain almost unchanged when the input parameters to the recovery algorithm changed slightly.

When using irregular sensing graphs, we observed that for a fixed compression ratio, increasing the average variable degree increases the success threshold as well. However, this is achieved at the expense of higher recovery complexity. Two interesting research paths emerge here: first to find the highest success threshold that can be achieved for a fixed compression ratio using irregular graphs as sensing graphs and NB-VB algorithms as the recovery algorithm when the average variable degree (and consequently the average check degree) is allowed to go to infinity. In this context, the issue of the design of the sequences of irregular graphs that can achieve the ultimate performance is also interesting. Second to look at the trade-off between the running time, recovery complexity and the success threshold in the context of irregular graph design. In this case, new measures need to be devised that capture this trade-off and allow the design of optimal degree distributions.

The verification rules in the NB-VB algorithms are devised for the noiseless measurements. Using thresholding techniques presented in this work, NB-VB algorithms can be applied to noisy measurements if the power of noise in the measurements is much less than the signal power (very high SNR regime). In order to make NB-VB algorithms more robust against the noise in the measurements one way is to design verification rules that can explicitly cope with the noisy measurements. Even in the case of high SNR, one interesting research path would be to find the optimal value for the thresholds ϵ_1 and ϵ_2 introduced in Section 4.7 as functions of density factor, SNR, and possibly other design parameters, such as the compression ratio.

It would also be interesting to formulate some practical application scenarios so that they can be solved by the application of NB-VB recovery algorithms.

List of References

[1] F. Zhang and H. D. Pfister, "Analysis of verification-based decoding on the q-ary symmetric channel for large q," *IEEE Trans. Inform. Theory*, vol. 57 (10), pp. 6754–6770, 2011.

[2] W. Xu and B. Hassibi, "Efficient compressive sensing with deterministic guarantees using expander graphs," in *Proc. Information Theory Workshop (ITW)*, pp. 414–419, September 2007.

[3] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, pp. 21–30, March 2008.

[4] D. Donoho, "Compressed sensing," *IEEE Trans. Inform. Theory*, vol. 52 (4), pp. 1289–1306, April 2006.

[5] E. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inform. Theory*, vol. 52 (2), pp. 489–509, February 2006.

[6] Y. Wu and S. Verdú, "Fundamental limits of almost lossless analog compression," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, pp. 359 – 363, 2009.

[7] R. G. Baraniuk, "Compressive sensing," *IEEE Signal Processing Magazine*, vol. 24, pp. 118–124, July 2007.

[8] J. Tropp, *Topics in Sparse Approximation*. PhD thesis, University of Texas at Austin, 2004.

[9] R. Berinde, A. Gilbert, P. Indyk, and K. Strauss, "Combining geometry and combinatorics: A unified approach to sparse signal recovery," in *46th Annual Allerton Conference on Communication, Control, and Computing*, pp. 798–805, September 2008.

[10] J. Tropp and A. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inform. Theory*, vol. 53 (12), pp. 4655–4666, December 2007.

[11] D. Needell and R. Vershynin, "Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit," *Foundations of Computational Mathematics*, vol. 9 (3), pp. 317–334, June 2009.

[12] S. Mendelson, A. Pajor, and N. Tomczak-Jaegermann, "Uniform uncertainty principle for Bernoulli and sub-Gaussian ensembles," *Constructive Approximation*, vol. 28 (3), pp. 277–289, December 2008.

[13] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A simple proof of the restricted isometry property for random matrices," *Constructive Approximation, Springer New York*, vol. 28, pp. 253–263, December 2008.

[14] E. Candès and T. Tao, "Decoding by linear programming," *IEEE Trans. Inform. Theory*, vol. 51 (12), pp. 4203–4215, Dec. 2005.

[15] V. Chandar, "A negative result concerning explicit matrices with the restricted isometry property," tech. rep., 2008.

[16] G. Cormode and M. Muthukrishnan, "Combinatorial algorithms for compressed sensing," in *Proc. Structural Information and Communication Complexity (SIROCCO)*, pp. 280–294, 2006.

[17] P. Indyk, "Explicit constructions for compressed sensing of sparse signals," in *Proc. Symp. on Discrete Algorithms (SODA)*, pp. 30–33, 2008.

[18] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin, "Algorithmic linear dimension reduction in the ℓ_1 norm for sparse vectors," in *44th Annual Allerton Conference on Communication, Control, and Computing*, 2006.

[19] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin, "One sketch for all: Fast algorithms for compressed sensing," in *Proc. 39th ACM Symposium on Theory of Computing (STOC)*, pp. 237–246, 2007.

[20] S. Sarvotham, D. Baron, and R. Baraniuk, "Sudocodes - fast measurement and reconstruction of sparse signals," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, pp. 2804–2808, July 2006.

[21] V. Chandar, D. Shah, and G. W. Wornell, "A simple message-passing algorithm for compressed sensing," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, pp. 1968–1972, June 2010.

[22] F. Zhang and H. D. Pfister, "On the iterative decoding of high rate LDPC codes with applications in compressed sensing," *Submitted to IEEE Trans. Inform. Theory*.

[23] F. Zhang and H. D. Pfister, "Compressed sensing and linear codes over real numbers," in *Proc. Information Theory and Applications Workshop*, pp. 558–561, February 2008.

[24] F. Zhang and H. D. Pfister, "List-message passing achieves capacity on the q-ary symmetric channel for large q," in *IEEE Global Telecom. Conf. (GLOBECOM)*, pp. 283–287, November 2007.

[25] Y. Lu, A. Montanari, B. Prabhakar, S. Dharmapurikar, and A. Kabbani, "Counter braids: A novel counter architecture for per-flow measurement," in *Proc. Int. Conf. on Measurement and Modeling of Computer Sys. ACM SIGMETRICS*, pp. 121–132, June 2008.

[26] M. Akcakaya, J. Park, and V. Tarokh, "Low density frames for compressive sensing," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 3642–3645, March 2010.

[27] D. Baron, S. Savotham, and R. G. Baraniuk, "Bayesian compressive sensing via belief propagation," *IEEE Transactions on Signal Processing*, vol. 58 (1), pp. 269–280, January 2010.

[28] A. Abdelkefi, Y. Jiang, W. Wang, and A. Kvittem, "Robust traffic anomaly detection with principal component pursuit," in *Proc. 6th ACM Int. Conf. on emerging Networking EXperiments and Technologies (CoNEXT) Student Workshop*, 2010.

[29] Y. Lu, A. Montanari, and B. Prabhakar, "Counter braids: Asymptotic optimality of the message passing decoding algorithm," in *46th Annual Allerton Conference on Communication, Control, and Computing*, pp. 209 – 216, September 2008.

[30] J. Meng, W. Yin, H. Li, E. Houssain, and Z. Han, "Collaborative spectrum sensing from sparse observations using matrix completion for cognitive radio networks," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Proc. (ICASSP)*, pp. 3114 – 3117, March 2010.

[31] R. Heckel and H. Bolcskei, "Compressive identification of linear operators," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, pp. 1412–1416, August 2011.

[32] A. Montanari and D. Tse, "Analysis of belief propagation for non-linear problems: The example of CDMA (or: How to prove tanaka's formula)," in *Proc. Information Theory Workshop (ITW)*, pp. 160–164, March 2006.

[33] D. Guo and C.-C. Wang, "Multiuser detection of sparsely spread CDMA," *IEEE Journal on Selected Areas in Communications*, vol. 26 (3), pp. 421–431, April 2008.

[34] D. Guo, D. Baron, and S. Shamai, "A single-letter characterization of optimal noisy compressed sensing," in *47th Annual Allerton Conference on Communication, Control, and Computing*, pp. 52–59, October 2009.

[35] C. P. Robert, *The Bayesian Choise: A Decision Theoretic Motivation*. Springer-Verlag, 1994.

[36] C. C. Paige and M. A. Saunders, "LSQR: Sparse linear equations and least squares problems," *ACM Transactions on Mathematical Software (TOMS)*, vol. 8, pp. 195–209, June 1982.

[37] D. Donoho, A. Javanmard, and A. Montanari, "Information-theoretically optimal compressed sensing via spatial coupling and approximate message passing," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, 2012.

[38] D. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing," *Proc. of the National Academy of Sciences (PNAS)*, vol. 106 (45), pp. 18914–18919, 2009.

[39] M. Luby and M. Mitzenmacher, "Verification-based decoding for packet-based low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 51 (1), pp. 120–127, January 2005.

[40] Y. Eftekhari, A. Heidarzadeh, A. H. Banihashemi, and I. Lambadaris, "Density evolution analysis of node-based verification-based algorithms in compressed sensing," *IEEE Trans. Inform. Theory*, vol. 58 (10), pp. 6616–6645, October 2012.

[41] Y. Eftekhari, A. Heidarzadeh, A. H. Banihashemi, and I. Lambadaris, "An efficient approach toward the asymptotic analysis of node-based verification-based algorithms in compressed sensing," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, 2011.

[42] Y. Eftekhari, A. H. Banihashemi, and I. Lambadaris, "Analysis and design of irregular graphs for node-based verification-based algorithms in compressed sensing," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, 2012.

[43] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47 (2), pp. 599–618, February 2001.

[44] D. J. MacKay, *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003.

[45] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Inform. Theory*, vol. 47 (2), pp. 569–584, February 2001.

[46] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Trans. Comm.*, vol. 44 (10), pp. 1261–1271, October 1996.

[47] 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, 2011.

[48] E. Candès, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure Mathematics*, vol. 59, pp. 1207–1223, August 2006.

[49] M. Bayati and A. Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing," *IEEE Trans. Inform. Theory*, vol. 57 (2), pp. 764–785, February 2011.

[50] H. L. V. Trees, *Detection, Estimation, and Modulation Theory, Part I.* John Wiley & Sons, 2001.

[51] E. Candès and J. Romberg, "L1magic," in *Available online at: http://www-stat.stanford.edu/ candes/l1magic/*, October 2005.

[52] E. Candès, M. Wakin, and S. Boyd, "Enhancing sparsity by reweighted l1 minimization," *The Journal of Fourier Analysis and Applications*, vol. 14, pp. 877–905, December 2008.

[53] D. Donoho and J. Tanner, "Thresholds for the recovery of sparse solutions via l1 minimization," in *Proc. 40th Annual Conference on Information Sciences and Systems (CISS)*, pp. 202–206, March 2006.

[54] H. Saeedi and A. H. Banihashemi, "Systematic design of low-density parity-check code ensembles for binary erasure channels," *IEEE Trans. Comm.*, vol. 58 (1), pp. 118–127, 2010.

[55] H. Saeedi and A. H. Banihashemi, "On the design of irregular LDPC code ensembles for BIAWGN channels," *IEEE Trans. Comm.*, vol. 58 (5), pp. 1376–1382, 2010.

[56] H. Saeedi and A. H. Banihashemi, "New sequences of capacity achieving LDPC code ensembles over the binary erasure channel," *IEEE Trans. Inform. Theory*, vol. 56 (12), pp. 6332–6346, 2010.

[57] K. Price and R. Storn, "Differential evolution a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, p. 341359, 1997.

[58] A. Shokrollahi and R. Storn, "Design of efficient erasure codes with differential evolution," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, pp. 5–5, June 2000.

[59] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 619 – 637, 2001.

Appendix A

Analysis of NB-VB Algorithms for Regular Graphs

A.1 Assumptions

In our analysis, we often use two assumptions: "uniformity assumption" and "independence assumption". Both assumptions are natural consequences of the randomness of the graph and the input as well as the asymptotic nature of the analysis. The uniformity assumption states that all the possible choices within the constraints imposed by the event under consideration are equally likely. As an example, consider the Genie algorithm and the case, where an edge is connected to a variable node with i connections to check nodes with $\mathcal{K}$-degree 1, and thus $d_v - i$ connections to the check nodes with other values of $\mathcal{K}$-degree. Now, suppose that we are interested in calculating the probability that the other end of such an edge is connected to a check node of $\mathcal{K}$-degree 1. The uniformity assumption implies that this probability is equal to i/d_v.

The independence assumption, on the other hand, states that all the edges of a certain type, carry independent messages. In our analysis such messages are often identically distributed as well. As an example, consider a check node with $\mathcal{K}$-degree j and suppose that the probability that this node receives a verified message along each of the j edges is p. Then the independence assumption implies that the number of verified messages that this node receives has a binomial distribution. One should note that the independence assumption here is different from the similar assumption made for the density evolution analysis of MB algorithms. The latter requires the *extrinsic* nature of message passing and implies that all the messages, each originating from non-overlapping leaves on a tree-like neighborhood of an edge are independent. The former however does not require the message passing to be extrinsic and is a consequence of the random permutation of edges between the variable nodes and the check nodes.

A.2 Genie

For the analysis, we assume to have the probabilities $\alpha^{(\ell)}$, $p_{\mathcal{N}_i}^{(\ell-1,1)}$, and $p_{\mathcal{K}_j}^{(\ell-1,2)}$, and are interested in deriving the same probabilities for iteration ℓ.

A.2.1 Update equations for ℓ-HR1, $\ell \geq 2$ (Calculation of $p_{\mathcal{N}_i}^{(\ell,1)}$)

When a variable node is verified in HR2 of an iteration, the d_v edges adjacent to the variable node carry the recovery message to the neighboring check nodes. These check nodes thus face a reduction in their $\mathcal{K}$-degree. We denote by $p_{\mathcal{N}_{j\downarrow i}}^{(\ell)}$ the probability that the $\mathcal{K}$-degree of a check node is reduced from j to $i \leq j$ after HR1 of iteration ℓ. This happens if out of j edges emanating from the check node and incident to the set of unverified variable nodes $\mathcal{K}^{(\ell)}$, $j-i$ of them now carry a message from their variable nodes indicating that they have been verified.

On the other side of the graph, when a variable node in $\mathcal{K}_i^{(\ell-1,2)}$ $(1 \leq i \leq d_v)$ is verified, by definition, out of d_v check nodes receiving the recovery message, i have $\mathcal{K}$-degree 1 and $d_v - i$ have $\mathcal{K}$-degrees larger than 1. For each verified variable node, the set of i check nodes of degree 1 are distributed uniformly with respect to the set of all check nodes of degree 1. Based on the operation of the Genie algorithm, it is easy to see that

$$p_{\mathcal{N}_{1\downarrow 1}}^{(\ell)} = 0 \text{ and } p_{\mathcal{N}_{1\downarrow 0}}^{(\ell)} = 1\,.$$

Furthermore, one may assume that the remaining $d_v - i$ edges carrying a verified message are uniformly connected to the check nodes with $\mathcal{K}$-degrees larger than 1. Based on this assumption, we derive $p_{\mathcal{N}_{j\downarrow i}}^{(\ell)}$ for $j > 1$. Once these probabilities are found, the new distribution of check node degrees $p_{\mathcal{N}_i}^{(\ell,1)}$ can be derived using the total probability law:

$$p_{\mathcal{N}_i}^{(\ell,1)} = \sum_{j=i}^{d_c} p_{\mathcal{N}_j}^{(\ell-1,1)} p_{\mathcal{N}_{j\downarrow i}}^{(\ell)}, \qquad 0 \leq i \leq d_c.$$

To find the probability $p_{\mathcal{N}_{j\downarrow i}}^{(\ell)}$, $2 \leq j \leq d_c$, we need the conditional probability that an edge connecting a check node in the set $\mathcal{N}_j^{(\ell-1,1)}$, $2 \leq j \leq d_c$, and an unverified variable node, carries a verified message to the check node in the first half-round of iteration ℓ. We denote this conditional probability by $p_{d>1}^{(\ell)}$. Assuming this probability is known, for $2 \leq j \leq d_c$ and $0 \leq i \leq j$ we have:

$$p_{\mathcal{N}_{j\downarrow i}}^{(\ell)} = \binom{j}{j-i} \left(p_{d>1}^{(\ell)}\right)^{j-i} \left(1 - p_{d>1}^{(\ell)}\right)^{i}.$$

The probability $p_{d>1}^{(\ell)}$ can be computed as follows. In the following, s_e is the status bit of the message sent from v to c over the edge $e = (v, c)$, and $p^{(\ell)}$ is defined as the

probability of an edge e being adjacent to a check node $c \in \mathcal{N}_1^{(\ell-1,1)}$ conditioned on the fact that it is adjacent to a variable node $v \in \mathcal{K}^{(\ell-1)}$.

$$\begin{aligned}
p_{d>1}^{(\ell)} &= \Pr[s_e = 1 | c \in \bigcup_{j=2}^{d_c} \mathcal{N}_j^{(\ell-1,1)}, v \in \mathcal{K}^{(\ell-1)}] \\
&= \sum_{i=1}^{d_v} \Pr[v \in \mathcal{K}_i^{(\ell-1,2)} | c \in \bigcup_{j=2}^{d_c} \mathcal{N}_j^{(\ell-1,1)}, v \in \mathcal{K}^{(\ell-1)}] \\
&= \sum_{i=1}^{d_v} \frac{\Pr[c \in \bigcup_{j=2}^{d_c} \mathcal{N}_j^{(\ell-1,1)} | v \in \mathcal{K}_i^{(\ell-1,2)} \cup \mathcal{K}^{(\ell-1)}]}{\Pr[c \in \bigcup_{j=2}^{d_c} \mathcal{N}_j^{(\ell-1,1)} | v \in \mathcal{K}^{(\ell-1)}]} \times \Pr[v \in \mathcal{K}_i^{(\ell-1,2)} | v \in \mathcal{K}^{(\ell-1)}] \\
&= \sum_{i=1}^{d_v} \frac{\left(1 - \Pr[c \in \mathcal{N}_1^{(\ell-1,1)} | v \in \mathcal{K}_i^{(\ell-1,2)}]\right) p_{\mathcal{K}_i}^{(\ell-1,2)}}{\left(1 - \Pr[c \in \mathcal{N}_1^{(\ell-1,1)} | v \in \mathcal{K}^{(\ell-1)}]\right)} = \frac{\sum_{i=1}^{d_v} p_{\mathcal{K}_i}^{(\ell-1,2)} - \sum_{i=1}^{d_v} \frac{i}{d_v} p_{\mathcal{K}_i}^{(\ell-1,2)}}{1 - p^{(\ell)}} \\
&= \frac{1 - p_{\mathcal{K}_0}^{(\ell-1,2)} - p^{(\ell)}}{1 - p^{(\ell)}} = 1 - \frac{p_{\mathcal{K}_0}^{(\ell-1,2)}}{1 - p^{(\ell)}}.
\end{aligned} \tag{A.1}$$

where, $p^{(\ell)}$ can be calculated as:

$$\begin{aligned}
p^{(\ell)} &:= \Pr[c \in \mathcal{N}_1^{(\ell-1,1)} | v \in \mathcal{K}^{(\ell-1)}] \\
&= \frac{\Pr[v \in \mathcal{K}^{(\ell-1)} | c \in \mathcal{N}_1^{(\ell-1,1)}] \Pr[c \in \mathcal{N}_1^{(\ell-1,1)}]}{\Pr[v \in \mathcal{K}^{(\ell-1)}]} \\
&= \frac{\frac{1}{d_c} \times p_{\mathcal{N}_1}^{(\ell-1,1)}}{\alpha^{(\ell-1)}} = \frac{p_{\mathcal{N}_1}^{(\ell-1,1)}}{\alpha^{(\ell-1)} d_c}.
\end{aligned} \tag{A.2}$$

A.2.2 Update equations for ℓ-HR2, $\ell \geq 2$ (Calculation of $p_{\mathcal{K}_j}^{(\ell,2)}$ and $\alpha^{(\ell+1)}$)

In the second half-round, variable nodes receive messages from their neighboring check nodes. According to the verification rule in the Genie algorithm, the only unverified variable nodes are those in the set $\mathcal{K}_0^{(\ell-1,2)}$. Variable nodes in this set have no connection to check nodes in the set $\mathcal{N}_1^{(\ell-1,1)}$ but d_v connections to the sets $\{\mathcal{N}_2^{(\ell-1,1)}, \cdots, \mathcal{N}_{d_c}^{(\ell-1,1)}\}$. Suppose $v \in \mathcal{K}_0^{(\ell-1,2)}$. In this case, if j out of d_v adjacent check nodes of v in $\{\mathcal{N}_2^{(\ell-1,1)}, \cdots, \mathcal{N}_{d_c}^{(\ell-1,1)}\}$ move to $\mathcal{N}_1^{(\ell,1)}$, v will move from $\mathcal{K}_0^{(\ell-1,2)}$ to $\mathcal{K}_j^{(\ell,2)}$. This is shown in Figure A.1.

We define the probability $p_{\mathcal{K}_{0\uparrow j}}^{(\ell)}$ as the probability of a variable node $v \in \mathcal{K}_0^{(\ell-1,2)}$

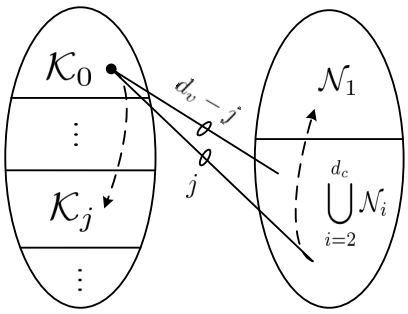


Figure A.1: A variable node in $\mathcal{K}_0$ moves to $\mathcal{K}_j$.

moving to $\mathcal{K}_j^{(\ell,2)}$. We thus have:

$$\begin{aligned} p_{\mathcal{K}_j}^{(\ell,2)} &= p_{\mathcal{K}_{0\uparrow j}}^{(\ell)} = \Pr[v \in \mathcal{K}_j^{(\ell,2)} | v \in \mathcal{K}_0^{(\ell-1,2)}] \\ &= \binom{d_v}{j} \left(p_x^{(\ell)}\right)^j \left(1 - p_x^{(\ell)}\right)^{d_v - j}, \ 0 \le j \le d_v, \end{aligned} \tag{A.3}$$

where $p_x^{(\ell)}$ is defined as the probability that an edge adjacent to a variable node $v \in \mathcal{K}_0^{(\ell-1,2)}$ carries a message indicating that the adjacent check node c has a degree equal to 1, i.e., $c \in \mathcal{N}_1^{(\ell,1)}$. The probability $p_x^{(\ell)}$ is calculated as follows:

$$\begin{aligned} p_x^{(\ell)} &= \Pr[c \in \mathcal{N}_1^{(\ell,1)} | v \in \mathcal{K}_0^{(\ell-1,2)}] \\ &= \frac{\Pr[c \in \mathcal{N}_1^{(\ell,1)}] \Pr[v \in \mathcal{K}_0^{(\ell-1,2)} | c \in \mathcal{N}_1^{(\ell,1)}]}{\Pr[v \in \mathcal{K}_0^{(\ell-1,2)}]} \\ &= \frac{p_{\mathcal{N}_1}^{(\ell,1)} \times \frac{1}{d_c}}{\alpha^{(\ell)}} = \frac{p_{\mathcal{N}_1}^{(\ell,1)}}{\alpha^{(\ell)} d_c} = p^{(\ell+1)}, \end{aligned} \tag{A.4}$$

where $p^{(\ell+1)}$ is given in (A.2).

In the Genie algorithm, the probability of a variable node in the set $\mathcal{K}^{(\ell)}$ being verified is calculated as $\sum_{i=1}^{d_v} p_{\mathcal{K}_i}^{(\ell,2)}$. Therefore, the probability of a variable node v remaining unverified, i.e., $v \in \mathcal{K}^{(\ell+1)}$, is:

$$\alpha^{(\ell+1)} = \alpha^{(\ell)} \left(1 - \sum_{i=1}^{d_v} p_{\mathcal{K}_i}^{(\ell,2)}\right) = \alpha^{(\ell)} p_{\mathcal{K}_0}^{(\ell,2)}.$$

A.2.3 Initial probabilities for the Genie algorithm ($\ell = 0$ and $\ell = 1$)

Assuming an initial density factor of $\alpha^{(0)}$, $1 - \alpha^{(0)}$ fraction of the edges from variable nodes in the first iteration carry a recovery message to check nodes. Since at iteration

zero no variable node in the support set is verified, we have $\alpha^{(1)} = \alpha^{(0)}$. Moreover, $p^{(1)}_{d>1} = 1 - \alpha^{(1)}$ and $p^{(0,1)}_{\mathcal{N}_{d_c}} = 1$. The set of probabilities $p^{(1,1)}_{\mathcal{N}_i}$ is thus given by the following.

$$\begin{aligned} p^{(1,1)}_{\mathcal{N}_i} &= p^{(0,1)}_{\mathcal{N}_{d_c}} p^{(1)}_{\mathcal{N}_{d_c \downarrow i}} = p^{(1)}_{\mathcal{N}_{d_c \downarrow i}} \\ &= \binom{d_c}{d_c - i} \left(p^{(1)}_{d>1}\right)^{d_c - i} \left(1 - p^{(1)}_{d>1}\right)^{i} \\ &= \binom{d_c}{i} \left(\alpha^{(1)}\right)^{i} \left(1 - \alpha^{(1)}\right)^{d_c - i}, \ 0 \le i \le d_c. \end{aligned}$$

To find the probability $p^{(1,2)}_{\mathcal{K}_i}$, we first find the probability $p^{(1)}_x$ from (A.4) and then replace it in (A.3).

A.3 XH

As before, we assume to have the probabilities $\alpha^{(\ell)}$, $p^{(\ell-1,1)}_{\mathcal{N}_i}$, and $p^{(\ell-1,2)}_{\mathcal{K}_j}$, and are interested in deriving the same probabilities for iteration ℓ. Since in the analysis of XH we only track the probability of unverified non-zero variable nodes, the analysis will include only one round with two half-rounds just like the analysis for the Genie.

A.3.1 Update equations for ℓ-HR1, $\ell \ge 2$ (Calculation of $p^{(\ell,1)}_{\mathcal{N}_i}$)

When a variable node in $\mathcal{K}^{(\ell-1,2)}_i$ ($\lceil d_v/2 \rceil \le i \le d_v$) is verified, by definition, out of d_v check nodes receiving the recovery message, i have $\mathcal{K}$-degree 1 and $d_v - i$ have $\mathcal{K}$-degrees larger than 1. For each verified variable node, the set of i check nodes of degree 1 are distributed uniformly with respect to the set of all check nodes of degree 1. It is easy to see that

$$p^{(\ell)}_{\mathcal{N}_{1\downarrow 1}} = \sum_{i=\lceil d_v/2 \rceil}^{d_v} \frac{i p^{(\ell-1,2)}_{\mathcal{K}_i}}{\sum_{i=0}^{d_v} i p^{(\ell-1,2)}_{\mathcal{K}_i}} \qquad p^{(\ell)}_{\mathcal{N}_{1\downarrow 0}} = 1 - p^{(\ell)}_{\mathcal{N}_{1\downarrow 1}}.$$

Furthermore, one may assume that the remaining $d_v - i$ edges carrying a verified message are uniformly connected to the check nodes with $\mathcal{K}$-degrees larger than 1.

Following the same approach as the one presented for the Genie, we have:

$$p^{(\ell)}_{\mathcal{N}_{j\downarrow i}} = \binom{j}{j-i}\left(p^{(\ell)}_{d>1}\right)^{j-i}\left(1-p^{(\ell)}_{d>1}\right)^{i}, \quad 2 \le j \le d_c, \quad 0 \le i \le j.$$

$$p^{(\ell,1)}_{\mathcal{N}_i} = \sum_{j=i}^{d_c} p^{(\ell-1,1)}_{\mathcal{N}_j} p^{(\ell)}_{\mathcal{N}_{j\downarrow i}}, \qquad 0 \le i \le d_c,$$

where, $p^{(\ell)}_{d>1}$ denotes the conditional probability that an edge connecting a check node in the set $\mathcal{N}^{(\ell-1,1)}_j$, $2 \le j \le d_c$, and an unverified variable node, carries a verified message to the check node in the first half-round of iteration ℓ.

Following the same path for the derivation of Equation A.1, we have:

$$p^{(\ell)}_{d>1} = 1 - \sum_{i=0}^{\lceil d_v/2\rceil - 1} \frac{p^{(\ell-1,2)}_{\mathcal{K}_i}}{1-p^{(\ell)}}.$$

where, $p^{(\ell)}$ is calculated as in A.2.

A.3.2 Update equations for ℓ-HR2, $\ell \ge 2$ (Calculation of $p^{(\ell,2)}_{\mathcal{K}_j}$ and $\alpha^{(\ell+1)}$)

In the second half-round, variable nodes receive messages from their neighboring check nodes. According to the verification rule in the XH algorithm, the unverified variable nodes are those in the set $\bigcup_{i=0}^{\lceil d_v/2\rceil-1} \mathcal{K}^{(\ell-1,2)}_i$. A variable node $v \in \mathcal{K}^{(\ell-1,2)}_j$ in this set has j connections to check nodes in the set $\mathcal{N}^{(\ell-1,1)}_1$ and $d_v - j$ connections to the sets $\{\mathcal{N}^{(\ell-1,1)}_2, \cdots, \mathcal{N}^{(\ell-1,1)}_{d_c}\}$. We define the probability $p^{(\ell)}_{\mathcal{K}_{i\uparrow j}}$ as the probability of a variable node $v \in \mathcal{K}^{(\ell-1,2)}_i$ moving to $\mathcal{K}^{(\ell,2)}_j$. We thus have:

$$p^{(\ell,2)}_{\mathcal{K}_j} = \sum_{i=0}^{\lceil d_v/2\rceil-1} p^{(\ell-1,2)}_{\mathcal{K}_i} p^{(\ell)}_{\mathcal{K}_{i\uparrow j}}, \quad 0 \le j \le d_v$$

$$p^{(\ell)}_{\mathcal{K}_{i\uparrow j}} = \binom{d_v - i}{j-i}\left(p^{(\ell)}_x\right)^{j-i}\left(1-p^{(\ell)}_x\right)^{d_v-j}, \tag{A.5}$$

where $p^{(\ell)}_x$ is the probability that an edge adjacent to an unverified variable node carries a message indicating that the adjacent check node c has a degree equal to 1. This probability is calculated according to A.4.

In the XH algorithm, the probability of a variable node in the set $\mathcal{K}^{(\ell)}$ being verified is calculated as $\sum_{i=\lceil d_v/2\rceil}^{d_v} p^{(\ell,2)}_{\mathcal{K}_i}$. Therefore, the probability of a variable node

v remaining unverified, i.e., $v \in \mathcal{K}^{(\ell+1)}$, is:

$$\alpha^{(\ell+1)} = \alpha^{(\ell)} \left(1 - \sum_{i=\lceil d_v/2 \rceil}^{d_v} p_{\mathcal{K}_i}^{(\ell,2)} \right).$$

A.3.3 Initial probabilities for the XH algorithm ($\ell = 0$ and $\ell = 1$)

The initialization of the parameters in the XH algorithm are the same as those in the Genie algorithm, and hence not repeated here.

A.4 LM

A.4.1 Iteration zero, R1 (Verification of variable nodes based on D1CN)

In R1-HR1, each check node sends its degree (equal to d_c) and its received measurement to all its neighboring variable nodes. In R1-HR2, variable nodes receive such messages. Since the verification of (non-zero) variable nodes at this stage is based solely on D1CN, no variable node is verified at this stage. Therefore, we have $\mathcal{K}^{(0)} = \mathcal{K}^{(1)}$, and hence,

$$\alpha^{(1)} = \Pr[v \in \mathcal{K}^{(1)}] = \alpha^{(0)}.$$

A.4.2 Iteration zero, R2 (Verification of variable nodes based on ZCN)

Since no variable node was verified in R1-HR2, the messages sent from check nodes to variable nodes in R2-HR1 are the same as those sent in R1-HR1. All check nodes at this stage have degree d_c in the subgraph induced by the unverified variable nodes. Thus, the set of all check nodes can be partitioned into subsets $\mathcal{N}_{i,d_c-i}^{(0,R2,1)}$, where i, $0 \leq i \leq d_c$, denotes the $\mathcal{K}$-degree of the check node. The edges adjacent to a check node are also partitioned into two sets: $\mathcal{K}$-edges and Δ-edges. $\mathcal{K}$-edges are connected to variable nodes in the support set, while Δ-edges are connected to zero-valued variable nodes. Therefore, a check node in the set $\mathcal{N}_{i,d_c-i}^{(0,R2,1)}$ $(0 \leq i \leq d_c)$ has i, $\mathcal{K}$-edges and $d_c - i$, Δ-edges.

In R2-HR2, variable nodes process their incoming messages. At this stage, a variable node is verified based on the ZCN rule, if it receives at least one message with a value equal to zero.

Let $\mathcal{N}_i^{(0)}$, $0 \leq i \leq d_c$, denote the set of check nodes with $\mathcal{K}$-degree equal to i. The probability $p_{\mathcal{N}_i}^{(0)}$ defined as the probability that a check node belongs to the set $\mathcal{N}_i^{(0)}$

is calculated as follows:

$$p_{\mathcal{N}_i}^{(0)} := \Pr(c \in \mathcal{N}_i^{(0)}) = \binom{d_c}{i} \left(\alpha^{(0)}\right)^i \left(1-\alpha^{(0)}\right)^{d_c-i}.$$

Hence, the probability that a check node c has a value equal to zero ($c=0$) is:

$$\Pr[c=0] = p_{\mathcal{N}_0}^{(0)} = \left(1-\alpha^{(0)}\right)^{d_c}.$$

Let $\Delta_j^{(0,R2,2)}$ denote the set of zero-valued variable nodes that receive j zero-valued messages. The probability $p_{\Delta_j}^{(0,R2,2)}$ defined as the probability that a zero-valued variable node belongs to the set $\Delta_j^{(0,R2,2)}$ is calculated as follows:

$$\begin{aligned} p_{\Delta_j}^{(0,R2,2)} &= \Pr[v \in \Delta_j^{(0,R2,2)} | v \in \Delta^{(0)}] \\ &= \binom{d_v}{j} \left(p_\delta^{(0)}\right)^j \left(1-p_\delta^{(0)}\right)^{d_v-j}, 0 \le j \le d_v, \end{aligned} \tag{A.6}$$

where $p_\delta^{(0)}$ is the probability that an edge adjacent to a zero-valued variable node is also connected to a check node with value equal to zero. This happens if the other d_c-1 variable nodes adjacent to the check node are all zero-valued. Therefore, $p_\delta^{(0)}$ is calculated as follows:

$$p_\delta^{(0)} := \Pr[c_e \in \mathcal{N}_0^{(1)} | v_e \in \Delta^{(1)}] = \left(1-\alpha^{(0)}\right)^{d_c-1}. \tag{A.7}$$

Let $p_\Delta^{(1)} := \Pr[v \in \Delta^{(1)}]$ denote the probability that a variable node has a zero value but is not verified in R2-HR2 of iteration zero. Based on (A.6) and (A.7), we have:

$$\begin{aligned} p_\Delta^{(1)} &= \Pr[v \in \Delta^{(0)}] \Pr[v \in \Delta_0^{(0,R2,2)} | v \in \Delta^{(0)}] \\ &= (1-\alpha^{(0)}) \left(1-\left(1-\alpha^{(0)}\right)^{d_c-1}\right)^{d_v}. \end{aligned} \tag{A.8}$$

A.4.3 Iteration one, R1-HR1 (Regrouping of check nodes in sets $\mathcal{N}_{i,j}$ based on the index j)

Based on the recovery process at iteration zero, all verified variable nodes are in the sets Δ_j, $1 \le j \le d_v$. The processing of verified messages sent from these variable nodes to check nodes at iteration 1, R1-HR1 results in some check nodes to move from $\mathcal{N}_{i,d_c-i}^{(0,R2,1)}$ to $\mathcal{N}_{i,j}^{(1,R1,1)}$, $1 \le i \le d_c$, $0 \le j \le d_c - i$. The set of such check nodes is denoted by $\mathcal{N}_{i,d_c-i\downarrow j}^{(1,R1)}$ and we are interested in the probability that a check node belongs to this set.

Let $p_{\mathcal{E}_R}$ denote the probability of an edge in the set of Δ-edges carrying a verified message. Such edges are not connected to the set Δ_0. Let v_e and c_e denote the

variable node and the check node connected by edge e. We thus have:

$$\begin{aligned} p_{\mathcal{E}_R}^{(1,R1)} &:= 1 - \Pr[v_e \in \Delta_0^{(0,R2,2)} | v_e \in \Delta^{(1)}, c_e \notin \mathcal{N}_0^{(1)}] \\ &= 1 - \frac{\Pr[v_e \in \Delta_0 | v_e \in \Delta^{(1)}] \Pr[c_e \notin \mathcal{N}_0^{(1)} | v_e \in \Delta_0]}{\Pr[c_e \notin \mathcal{N}_0^{(1)} | v_e \in \Delta^{(1)}]} \\ &= 1 - \frac{p_{\Delta_0}^{(0,R2,2)} \times 1}{1 - p_\delta^{(0)}} = 1 - \frac{p_\Delta^{(1)}}{1 - p_\delta^{(0)}}, \end{aligned}$$

where $p_\Delta^{(1)}$ and $p_\delta^{(0)}$ are given in (A.8) and (A.7), respectively. Moreover, for $1 \leq i \leq d_v$ and $0 \leq j \leq d_c - i$,

$$p_{\mathcal{N}_{i,d_c-i\downarrow j}}^{(1,R1)} = \binom{d_c - i}{j} \left(1 - p_{\mathcal{E}_R}^{(1,R1)}\right)^j \left(p_{\mathcal{E}_R}^{(1,R1)}\right)^{d_c-i-j}.$$

Hence,

$$p_{\mathcal{N}_{i,j}}^{(1,R1,1)} = p_{\mathcal{N}_{i,d_c-i}}^{(0,R2,1)} p_{\mathcal{N}_{i,d_c-i\downarrow j}}^{(1,R1)}. \tag{A.9}$$

A.4.4 Iteration one, R1-HR2 (Verification of variable nodes based on D1CN)

In R1-HR2, received messages from check nodes are processed at variable nodes. At this stage, a check node in the set $\mathcal{N}_{1,0}^{(1,R1,1)}$ transmits a message with its first coordinate equal to 1, making it possible for variable nodes in the support set to be verified according to D1CN rule. We partition the set of all variable nodes in $\mathcal{K}^{(1)}$ into subsets $\mathcal{K}_i^{(1,R1,2)}$, $0 \leq i \leq d_v$, where i denotes the number of received messages with the first coordinate equal to 1 (refer to Fig. 4.2a for more information). We denote the set of such variable nodes by $\mathcal{K}_{0\uparrow i}^{(1,R1)}$. Let $p^{(1,R1)}$ denote the probability that an edge adjacent to a variable node in the support set carries a message with the first coordinate equal to 1. Using the same notations v_e and c_e defined above, we have:

$$\begin{aligned} p^{(1,R1)} &:= \Pr[c_e \in \mathcal{N}_{1,0}^{(1,R1,1)} | v_e \in \mathcal{K}^{(1)}] \\ &= \frac{\Pr[c_e \in \mathcal{N}_{1,0}^{(1,R1,1)}] \Pr[v_e \in \mathcal{K}^{(1)} | c_e \in \mathcal{N}_{1,0}^{(1,R1,1)}]}{\Pr[v_e \in \mathcal{K}^{(1)}]} \\ &= \frac{p_{\mathcal{N}_{1,0}}^{(1,R1,1)} \times 1/d_c}{\alpha^{(1)}} = \frac{p_{\mathcal{N}_{1,0}}^{(1,R1,1)}}{\alpha^{(1)} d_c}. \end{aligned} \tag{A.10}$$

Hence, the probability $p^{(1,R1)}_{\mathcal{K}_{0\uparrow i}}$, $0 \le i \le d_v$, that a variable node $v \in \mathcal{K}^{(1)}$ belongs to the set $\mathcal{K}^{(1,R1,2)}_i$ is calculated as follows:

$$\begin{aligned} p^{(1,R1,2)}_{\mathcal{K}_i} &= p^{(1,R1)}_{\mathcal{K}_{0\uparrow i}} := \Pr(v \in \mathcal{K}^{(1,R1,2)}_i | v \in \mathcal{K}^{(1)}) \\ &= \binom{d_v}{i} \left(p^{(1,R1)}\right)^i \left(1 - p^{(1,R1)}\right)^{d_v - i}. \end{aligned} \tag{A.11}$$

Based on the D1CN rule in the LM algorithm, variable nodes in the set $\bigcup_{i=1}^{d_v} \mathcal{K}^{(1,R1,2)}_i$ are verified. Therefore, the probability that a variable node in the support set remains unverified for iteration 2 is as follows:

$$\alpha^{(2)} = \alpha^{(1)} \left(1 - \sum_{i=1}^{d_v} p^{(1,R1,2)}_{\mathcal{K}_i}\right).$$

A.4.5 Iteration one, R2-HR1 (Regrouping of check nodes in sets $\mathcal{N}_{i,j}$ based on the index i)

Since some variable nodes in the support set have been verified in R1-HR2, a check node in the set $\mathcal{N}^{(1,R1,1)}_{i,k}$ might move into the set $\mathcal{N}^{(1,R2,1)}_{j,k}$ in R2-HR1. This happens if from i edges in the set of $\mathcal{K}$-edges adjacent to the check node, $i - j$ of them carry a verified message. The set of such check nodes are denoted by $\mathcal{N}^{(1,R2)}_{i\downarrow j,k}$ and we are interested in the probability that a check node belongs to this set.

Let $p^{(1,R2)}_d$ denote the probability that a $\mathcal{K}$-edge carries a verified message. We have:

$$\begin{aligned} p^{(1,R2)}_d &:= \Pr[v_e \notin \mathcal{K}^{(1,R1,2)}_0 | v_e \in \mathcal{K}^{(1)}, c_e \notin \mathcal{N}^{(1,R1,1)}_{1,0}] \\ &= 1 - \frac{\Pr[v_e \in \mathcal{K}_0 | v_e \in \mathcal{K}^{(1)}] \Pr[c_e \notin \mathcal{N}_{1,0} | v_e \in \mathcal{K}_0]}{\Pr[c_e \notin \mathcal{N}_{1,0} | v_e \in \mathcal{K}^{(1)}]} \\ &= 1 - \frac{p^{(1,R1,2)}_{\mathcal{K}_0} \times 1}{1 - p^{(1,R1)}} = 1 - \frac{p^{(1,R1,2)}_{\mathcal{K}_0}}{1 - p^{(1,R1)}}, \end{aligned}$$

where $p^{(1,R1,2)}_{\mathcal{K}_0}$ and $p^{(1,R1)}$ are given by (A.11) and (A.10), respectively. Hence, the probability $p^{(1,R2)}_{\mathcal{N}_{i\downarrow j,k}}$, $2 \le i \le d_c, 0 \le j \le i, 0 \le k \le d_c - i$, that a check node belongs to the set $\mathcal{N}^{(1,R2)}_{i\downarrow j,k}$ is calculated as follows:

$$p^{(1,R2)}_{\mathcal{N}_{i\downarrow j,k}} = \binom{i}{j} \left(p^{(1,R2)}_d\right)^{i-j} \left(1 - p^{(1,R2)}_d\right)^j.$$

Note that we have $p^{(1,R2)}_{\mathcal{N}_{1\downarrow 0,0}} = 1$ and $p^{(1,R2)}_{\mathcal{N}_{1\downarrow 1,0}} = 0$. After the regrouping, the probability $p^{(1,R2,1)}_{\mathcal{N}_{j,k}}$, $0 \le j \le d_c, 0 \le k \le d_c - i$, that a check node belongs to the set $\mathcal{N}^{(1,R2,1)}_{j,k}$ is

calculated by:

$$p_{\mathcal{N}_{j,k}}^{(1,R2,1)} = \sum_{i=j}^{d_c} p_{\mathcal{N}_{i,k}}^{(1,R1,1)} p_{\mathcal{N}_{i\downarrow j,k}}^{(1,R2)}.$$

A.4.6 Iteration one, R2-HR2 (Verification of variable nodes based on ZCN)

The measurement corresponding to check nodes in the set $\mathcal{N}_{0,k}^{(1,R2,1)}$, $1 \leq k \leq d_c - 1$, changes to zero, as such check nodes are no longer connected to unverified variable nodes in the support set. Hence, the messages transmitted by such check nodes have their second coordinate equal to zero, which in turn verifies some variable nodes (in the set Δ) at R2-HR2 of iteration 1.

To find the probability that a zero-valued variable node is verified at this stage, we need the probability $p_{\delta}^{(1,R2)}$ that an edge adjacent to a zero-valued variable node carries a message with value zero. This probability is calculated as follows:

$$\begin{aligned}
p_{\delta}^{(1,R2)} &= \sum_{j=1}^{d_c-1} \Pr[c_e \in \mathcal{N}_{0,j}^{(1,R2,1)} | v_e \in \Delta^{(1)}] \\
&= \frac{\sum_{j=1}^{d_c-1} \Pr[c_e \in \mathcal{N}_{0,j}^{(1,R2,1)}] \Pr[v_e \in \Delta^{(1)} | c_e \in \mathcal{N}_{0,j}^{(1,R2,1)}]}{\sum_{i=0}^{d_c} \sum_{j=1}^{d_c-1} \Pr[c_e \in \mathcal{N}_{i,j}^{(1,R2,1)}] \Pr[v_e \in \Delta^{(1)} | c_e \in \mathcal{N}_{i,j}^{(1,R2,1)}]} \\
&= \frac{\sum_{j=1}^{d_c-1} p_{\mathcal{N}_{0,j}}^{(1,R2,1)} \left(\frac{j}{d_c}\right)}{\sum_{i=0}^{d_c} \sum_{j=1}^{d_c-1} p_{\mathcal{N}_{i,j}}^{(1,R2,1)} \left(\frac{j}{d_c}\right)} = \frac{\sum_{j=1}^{d_c-1} j p_{\mathcal{N}_{0,j}}^{(1,R2,1)}}{\sum_{i=0}^{d_c} \sum_{j=1}^{d_c-1} j p_{\mathcal{N}_{i,j}}^{(1,R2,1)}}.
\end{aligned} \tag{A.12}$$

We then, for $0 \leq i \leq d_v$, have:

$$p_{\Delta_i}^{(1,R2,2)} = \binom{d_v}{i} \left(p_{\delta}^{(1,R2)}\right)^i \left(1 - p_{\delta}^{(1,R2)}\right)^{d_v - i}.$$

Lastly, the probability that a variable node is zero-valued and remains unverified for iteration 2 is calculated by:

$$p_{\Delta}^{(2)} = p_{\Delta}^{(1)} p_{\Delta_0}^{(1,R2,2)}.$$

A.4.7 Iteration two and beyond

The analysis of the second iteration and beyond is similar to that of iteration one, and hence omitted. The summary of the formulas can be found in Table 4.8.

A.5 SBB

In this section, we adopt the notation $\mathcal{N}_i$, with any superscript, to denote the set $\bigcup_{j=0}^{d_c-i} \mathcal{N}_{i,j}$.

A.5.1 Iteration zero

The analysis for iteration zero is the same as that of the LM algorithm discussed in Part A.4.1 of the appendix, and is thus not repeated here.

A.5.2 Iteration one, R1-HR1 (Regrouping of check nodes in sets $\mathcal{N}_{i,j}$ based on the index j)

The analysis of R1-HR1 for SBB and LM are the same. (See Part A.4.3 of the appendix.)

A.5.3 Iteration one, R1-HR2 (Verification of variable nodes based on D1CN and ECN)

In R1-HR2, the variable nodes in the support set are verified based on D1CN and ECN rules. Therefore, based on received messages from check nodes, we partition the support set $\mathcal{K}^{(1)}$ into subsets $\mathcal{K}_i^{(1,R1)}$, $0 \leq i \leq d_v$, where i denotes the number of neighboring check nodes in the set $\mathcal{N}_1^{(1,R1,1)}$ (refer to Fig. 4.2b for more information). We denote the set of such variable nodes by $\mathcal{K}_{0\uparrow i}^{(1,R1)}$. This set shall serve as an intermediate set that reflects the processing of messages from check nodes but does not reflect the verification of support set elements.

Let $p^{(1,R1)}$ denote the conditional probability that an edge e is adjacent to a check node in the set $\mathcal{N}_1^{(1,R1,1)}$ given that it is adjacent to a variable node in the support set. We then have:

$$\begin{aligned} p^{(1,R1)} &:= \Pr[c_e \in \mathcal{N}_1^{(1,R1,1)} | v_e \in \mathcal{K}^{(1)}] \\ &= \frac{\Pr[c_e \in \mathcal{N}_1^{(1,R1,1)}] \Pr[v_e \in \mathcal{K}^{(1)} | c_e \in \mathcal{N}_1^{(1,R1,1)}]}{\Pr[v_e \in \mathcal{K}^{(1)}]} \\ &= \sum_{j=0}^{d_c-1} \frac{p_{\mathcal{N}_{1,j}}^{(1,R1,1)} \times 1/d_c}{\alpha^{(1)}} = \sum_{j=0}^{d_c-1} \frac{p_{\mathcal{N}_{1,j}}^{(1,R1,1)}}{\alpha^{(1)} d_c}. \end{aligned} \tag{A.13}$$

Hence,

$$\begin{aligned} p_{\mathcal{K}_i}^{(1,R1)} &= p_{\mathcal{K}_{0\uparrow i}}^{(1,R1)} := \Pr(v \in \mathcal{K}_i^{(1,R1)} | v \in \mathcal{K}^{(1)}), \\ &= \binom{d_v}{i} \left(p^{(1,R1)}\right)^i \left(1 - p^{(1,R1)}\right)^{d_v - i}, \quad 0 \le i \le d_v. \end{aligned} \tag{A.14}$$

Based on the ECN rule, variable nodes in the set $\bigcup_{i=2}^{d_v} \mathcal{K}_i^{(1,R1)}$ are verified. A fraction $f^{(1,R1)}$ of variable nodes in the set $\mathcal{K}_1^{(1,R1)}$ that receive a message with the first coordinate equal to 1 are also verified based on the D1CN rule. This fraction is equal to:

$$f^{(1,R1)} = \Pr[c_e \in \mathcal{N}_{1,0}^{(1,R1,1)} | v_e \in \mathcal{K}_1^{(1,R1)}, c_e \in \mathcal{N}_1^{(0,R2,2)}].$$

By omitting the superscripts for simplicity, and noting $\Pr[c_e \in \mathcal{N}_1 | c_e \in \mathcal{N}_{1,0}, v_e \in \mathcal{K}_1] = 1$, we have:

$$\begin{aligned} f^{(1,R1)} &= \frac{\Pr[c_e \in \mathcal{N}_{1,0}] \Pr[v_e \in \mathcal{K}_1 | c_e \in \mathcal{N}_{1,0}]}{\Pr[c_e \in \mathcal{N}_1] \Pr[v_e \in \mathcal{K}_1 | c_e \in \mathcal{N}_1]} \\ &= \frac{p_{\mathcal{N}_{1,0}}^{(1,R1,1)} \times \frac{1}{d_c}}{p_{\mathcal{N}_1}^{(1,R1,1)} \times \frac{1}{d_c}} = \frac{p_{\mathcal{N}_{1,0}}^{(1,R1,1)}}{p_{\mathcal{N}_1}^{(1,R1,1)}}. \end{aligned}$$

Therefore, the probability that a variable node in the support set remains unverified for iteration 2 is calculated as follows:

$$\begin{aligned} \alpha^{(2)} &= \alpha^{(1)} \left(1 - f^{(1,R1)} p_{\mathcal{K}_1}^{(1,R1)} - \sum_{i=2}^{d_v} p_{\mathcal{K}_i}^{(1,R1)}\right) \\ &= \alpha^{(1)} \left(p_{\mathcal{K}_0}^{(1,R1)} + \left(1 - f^{(1,R1)}\right) p_{\mathcal{K}_1}^{(1,R1)}\right). \end{aligned}$$

Correspondingly, the unverified non-zero variable nodes are partitioned based on the following probabilities:

$$\begin{aligned} p_{\mathcal{K}_0}^{(1,R1,2)} &= \frac{1}{N^{(1,R1)}} p_{\mathcal{K}_0}^{(1,R1)}, \\ p_{\mathcal{K}_1}^{(1,R1,2)} &= \frac{1}{N^{(1,R1)}} \left(1 - f^{(1,R1)}\right) p_{\mathcal{K}_1}^{(1,R1)}, \\ p_{\mathcal{K}_i}^{(1,R1,2)} &= 0, \qquad 2 \le i \le d_v. \end{aligned}$$

The normalization factor $N^{(1,R1)}$ is used to make the set of parameters $p_{\mathcal{K}_i}^{(1,R1,2)}$ a valid probability measure, and is calculated as follows:

$$N^{(1,R1)} = p_{\mathcal{K}_0}^{(1,R1)} + \left(1 - f^{(1,R1)}\right) p_{\mathcal{K}_1}^{(1,R1)} = \frac{\alpha^{(2)}}{\alpha^{(1)}}.$$

A.5.4 Iteration one, R2-HR1 (Regrouping of check nodes in sets $\mathcal{N}_{i,j}$ based on the index i)

At this point, since some variable nodes in the support set have been verified at R1-HR2, check nodes should be regrouped based on their $\mathcal{K}$-degree. Since the variable nodes in the set $\mathcal{K}_1^{(1,R1,2)}$ are left unverified, not all check nodes in the set $\mathcal{N}_{1,j}^{(1,R1,1)}$ $(1 \leq j \leq d_c - 1)$ are moved into the set $\mathcal{N}_{0,j}^{(1,R2,1)}$; some will stay in the same set $\mathcal{N}_{1,j}^{(1,R2,1)}$. Hence, in addition to analyzing the set of check nodes $\mathcal{N}_{i\downarrow k,j}^{(1,R2)}$ that are moved from $\mathcal{N}_{i,j}^{(1,R1,1)}$ to $\mathcal{N}_{k,j}^{(1,R2,1)}$, we also have to analyze the set of check nodes $\mathcal{N}_{1\downarrow 0,j}^{(1,R2)}$ that are moved from $\mathcal{N}_{1,j}^{(1,R1,1)}$ to $\mathcal{N}_{0,j}^{(1,R2,1)}$.

We partition the edges adjacent to a check node into two sets: $\mathcal{K}$-edges and Δ-edges. $\mathcal{K}$-edges are connected to variable nodes in the set $\mathcal{K}^{(1)}$, while Δ-edges are connected to unverified zero-valued variable nodes. To analyze the regrouping of check nodes in the sets $\mathcal{N}_1^{(1,R1,1)}$ and $\mathcal{N}_i^{(1,R1,1)}$, $2 \leq i \leq d_c$, we need the probabilities $p_{d=1}^{(1,R2)}$ and $p_{d\neq 1}^{(1,R2)}$ defined as follows. The probability $p_{d=1}^{(1,R2)}$ is the conditional probability of an edge carrying a verified message given that it is a $\mathcal{K}$-edge adjacent to a check node in the set $\mathcal{N}_1^{(1,R1,1)}$. The probability $p_{d\neq 1}^{(1,R2)}$ is defined similarly with respect to the set of check nodes in $\bigcup_{i=2}^{d_c} \mathcal{N}_i^{(1,R1,1)}$.

Since the verified messages involved in the calculation of $p_{d=1}^{(1,R2)}$ originate from the set $\bigcup_{i=2}^{d_v} \mathcal{K}_i^{(1,R1)}$, we have:

$$\begin{aligned}
p_{d=1}^{(1,R2)} &:= \Pr[v_e \in \bigcup_{i=2}^{d_v} \mathcal{K}_i^{(1,R1)} | v_e \in \mathcal{K}^{(1)}, c_e \in \mathcal{N}_1^{(1,R1,1)}] \\
&= 1 - \Pr[v_e \in \mathcal{K}_1^{(1,R1)} | v_e \in \mathcal{K}^{(1)}, c_e \in \mathcal{N}_1^{(1,R1,1)}] \\
&= 1 - \frac{\Pr[v_e \in \mathcal{K}_1 | v_e \in \mathcal{K}^{(1)}] \Pr[c_e \in \mathcal{N}_1^{(1,R1,1)} | v_e \in \mathcal{K}_1]}{\Pr[c_e \in \mathcal{N}_1^{(1,R1,1)} | v_e \in \mathcal{K}^{(1)}]} \\
&= 1 - \frac{p_{\mathcal{K}_1}^{(1,R1)} \times 1/d_v}{p^{(1,R1)}} = 1 - \frac{p_{\mathcal{K}_1}^{(1,R1)}}{d_v p^{(1,R1)}},
\end{aligned}$$

where $p_{\mathcal{K}_1}^{(1,R1)}$ and $p^{(1,R1)}$ are given by (A.14) and (A.13), respectively. Similarly, $p_{d\neq 1}^{(1,R2)}$ is derived as:

$$p_{d\neq 1}^{(1,R2)} = 1 - \frac{p_{\mathcal{K}_0}^{(1,R1)} + p_{\mathcal{K}_1}^{(1,R1)} \left(\frac{d_v - 1}{d_v}\right) \left(1 - f^{(1,R1)}\right)}{1 - p^{(1,R1)}}.$$

Hence, the set of probabilities $p_{\mathcal{N}_{i\downarrow k,j}}^{(1,R2)}$, $2 \leq i \leq d_c, 0 \leq k \leq i, 0 \leq j \leq d_c - i$, that a

check node belongs to the set of check nodes $\mathcal{N}_{i\downarrow k,j}^{(1,R2)}$ are calculated as follows:

$$p_{\mathcal{N}_{i\downarrow k,j}}^{(1,R2)} = \binom{i}{k} \left(p_{d\neq 1}^{(1,R2)}\right)^{i-k} \left(1-p_{d\neq 1}^{(1,R2)}\right)^{k},$$

$$p_{\mathcal{N}_{1\downarrow 0,j}}^{(1,R2)} = p_{d=1}^{(1,R2)}, \qquad p_{\mathcal{N}_{1\downarrow 1,j}}^{(1,R2)} = 1-p_{d=1}^{(1,R2)}, \quad 1\leq j\leq d_c-1,$$
$$p_{\mathcal{N}_{1\downarrow 0,0}}^{(1,R2)} = 1, \qquad p_{\mathcal{N}_{1\downarrow 1,0}}^{(1,R2)} = 0, \qquad p_{\mathcal{N}_{0\downarrow 0,j}}^{(1,R2)} = 1, \quad 1\leq j\leq d_c.$$

Consequently, the probability $p_{\mathcal{N}_{k,j}}^{(1,R2,1)}$, $0\leq k\leq d_c, 0\leq j\leq d_c-i$, that a check node belongs to the set $\mathcal{N}_{k,j}^{(1,R2,1)}$ is calculated as follows:

$$p_{\mathcal{N}_{k,j}}^{(1,R2,1)} = \sum_{i=k}^{d_c} p_{\mathcal{N}_{i,j}}^{(1,R1,1)} p_{\mathcal{N}_{i\downarrow k,j}}^{(1,R2)}.$$

For the analysis in Part A.5.7, we need to partition the set of check nodes in $\mathcal{N}_{1,j}^{(1,R2,1)}$ into two sets: $\mathcal{N}_{1,j}^{(1,R2,1,+)}$ and $\mathcal{N}_{1,j}^{(1,R2,1,C)}$. Check nodes in the set $\mathcal{N}_{1,j}^{(1,R2,1,+)}$ were moved into the set $\mathcal{N}_{1,j}^{(1,R2,1)}$ from all the other sets $\mathcal{N}_{i,j}^{(1,R1,1)}$, $2\leq i\leq d_c$. Check nodes in the set $\mathcal{N}_{1,j}^{(1,R2,1,C)}$, however, are those that stayed in the set from $\mathcal{N}_{1,j}^{(1,R1,1)}$. Figure A.2 shows the relationship between the sets.

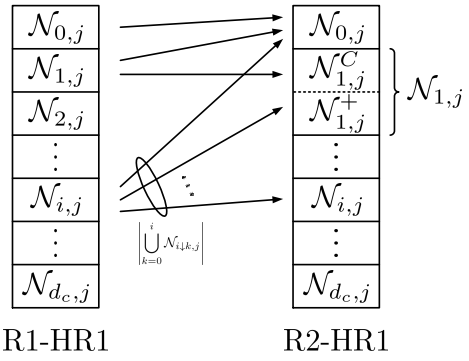


Figure A.2: Relationship between the sets $\mathcal{N}_{i,j}^{(1,R1,1)}, 1\leq i\leq d_c, 0\leq j\leq d_c-i$, on the left, and the sets $\mathcal{N}_{1,j}^{(1,R2,1,+)}$ and $\mathcal{N}_{1,j}^{(1,R2,1,C)}$, on the right.

Let $p_{\mathcal{N}_{1,j}}^{(1,R2,1,+)}$ and $p_{\mathcal{N}_{1,j}}^{(1,R2,1,C)}$ denote the probabilities that a check node belongs to the sets $\mathcal{N}_{1,j}^{(1,R2,1,+)}$ and $\mathcal{N}_{1,j}^{(1,R2,1,C)}$, respectively. We have:

$$p_{\mathcal{N}_{1,j}}^{(1,R2,1,+)} = \sum_{i=2}^{d_c} p_{\mathcal{N}_{i,j}}^{(1,R1,1)} p_{\mathcal{N}_{i\downarrow 1,j}}^{(1,R2)},$$
$$p_{\mathcal{N}_{1,j}}^{(1,R2,1,C)} = p_{\mathcal{N}_{1,j}}^{(1,R1,1)} p_{\mathcal{N}_{1\downarrow 1,j}}^{(1,R2)} = p_{\mathcal{N}_{1,j}}^{(1,R2,1)} - p_{\mathcal{N}_{1,j}}^{(1,R2,1,+)}.$$

A.5.5 Iteration one, R2-HR2 (Verification of variable nodes based on ZCN)

Let $p_{\delta}^{(1,R2)}$ denote the probability that an unverified zero-valued variable node is verified at this stage. This probability is derived similar to (A.12) and is given by:

$$p_{\delta}^{(1,R2)} := \sum_{j=1}^{d_c-1} \Pr[c_e \in \mathcal{N}_{0,j}^{(1,R2,1)} | v_e \in \Delta^{(1)}] = \sum_{j=1}^{d_c-1} j \frac{p_{\mathcal{N}_{0,j}}^{(1,R2,1)}}{\sum_{i=0}^{d_c} \sum_{j=1}^{d_c-1} j p_{\mathcal{N}_{i,j}}^{(1,R2,1)}}. \tag{A.15}$$

Hence, the probability $p_{\Delta_i}^{(1,R2,2)}$, $0 \leq i \leq d_v$, defined as the probability that an unverified zero-valued variable node belongs to the set $\Delta_i^{(1,R2,2)}$, is calculated as follows:

$$p_{\Delta_i}^{(1,R2,2)} = \binom{d_v}{i} \left(p_{\delta}^{(1,R2)}\right)^i \left(1 - p_{\delta}^{(1,R2)}\right)^{d_v - i}.$$

Lastly, the probability that a variable node is zero-valued and remains unverified for iteration 2 is given by:

$$p_{\Delta}^{(2)} = p_{\Delta}^{(1)} p_{\Delta_0}^{(1,R2,2)}.$$

A.5.6 Iteration two, R1-HR1 (Regrouping of check nodes in $\mathcal{N}_{i,j}$ based on the index j)

Similar to the analysis of R1-HR1 at iteration 1, for $1 \leq i \leq d_c, 0 \leq j \leq d_c - i, 0 \leq k \leq j$, we have:

$$p_{\mathcal{N}_{i,j\downarrow k}}^{(2,R1)} = \binom{j}{k} \left(1 - p_{\mathcal{E}_R}^{(2,R1)}\right)^k \left(p_{\mathcal{E}_R}^{(2,R1)}\right)^{j-k},$$

where,

$$p_{\mathcal{E}_R}^{(2,R1)} = 1 - \frac{p_{\Delta}^{(2)}}{1 - p_{\delta}^{(1,R2)}}.$$

Hence, for $0 \leq k \leq d_c - 1$,

$$p_{\mathcal{N}_{i,k}}^{(2,R1,1)} = \sum_{j=k}^{d_c-i} p_{\mathcal{N}_{i,j}}^{(1,R2,1)} p_{\mathcal{N}_{i,j\downarrow k}}^{(2,R1)}, \qquad 2 \leq i \leq d_c,$$

$$p_{\mathcal{N}_{1,k}}^{(2,R1,1,+)} = \sum_{j=k}^{d_c-1} p_{\mathcal{N}_{1,j}}^{(1,R2,1,+)} p_{\mathcal{N}_{1,j\downarrow k}}^{(2,R1)}, \qquad p_{\mathcal{N}_{1,k}}^{(2,R1,1,C)} = \sum_{j=k}^{d_c-1} p_{\mathcal{N}_{1,j}}^{(1,R2,1,C)} p_{\mathcal{N}_{1,j\downarrow k}}^{(2,R1)}.$$

A.5.7 Iteration two, R1-HR2 (Verification of variable nodes based on D1CN and ECN)

Edges emanating from the set $\mathcal{N}_1^{(2,R1,+)} := \bigcup_{k=0}^{d_c-1} \mathcal{N}_{1,k}^{(2,R1,1,+)}$, are responsible for the regrouping of variable nodes at iteration 2, R1-HR2. Let $p_k^{(2,R1)}$ be the conditional probability that an edge is adjacent to a check node in $\mathcal{N}_1^{(2,R1,+)}$ given that 1) it emanates from an unverified non-zero variable node, and 2) it is not adjacent to a check node in the set $\mathcal{N}_1^{(2,R1,C)} := \bigcup_{k=0}^{d_c-1} \mathcal{N}_{1,k}^{(2,R1,1,C)}$. This is indeed the probability that a variable node has an edge that increases its index. This probability is calculated as follows ($\Pr[c_e \notin \mathcal{N}_1^{(2,R1,C)} | v_e \in \mathcal{K}^{(2)}, c_e \in \mathcal{N}_1^{(2,R1,+)}] = 1$):

$$\begin{aligned} p_k^{(2,R1)} &= \Pr[c_e \in \mathcal{N}_1^{(2,R1,+)} | v_e \in \mathcal{K}^{(2)}, c_e \notin \mathcal{N}_1^{(2,R1,C)}] \\ &= \frac{\Pr[c_e \in \mathcal{N}_1^{(2,R1,+)}] \Pr[v_e \in \mathcal{K}^{(2)} | c_e \in \mathcal{N}_1^{(2,R1,+)}]}{\Pr[v_e \in \mathcal{K}^{(2)}, c_e \notin \mathcal{N}_1^{(2,R1,C)}]}. \end{aligned}$$

The two probabilities in the numerator are equal to $p_{\mathcal{N}_1}^{(2,R1,+)}$ and $1/d_c$, respectively. The denominator can be further processed as:

$$\begin{aligned} \Pr[v_e \in \mathcal{K}^{(2)}, c_e \notin \mathcal{N}_1^{(2,R1,C)}] &= \Pr[c_e \in \mathcal{N}_1^{(2,R1,+)}] \Pr[v_e \in \mathcal{K}^{(2)} | c_e \in \mathcal{N}_1^{(2,R1,+)}] \\ &\quad + \sum_{i=2}^{d_c} \Pr[\mathcal{N}_i^{(2,R1,1)}] \Pr[v_e \in \mathcal{K}^{(2)} | \mathcal{N}_i^{(2,R1,1)}]. \end{aligned}$$

We thus have:

$$p_k^{(2,R1)} = \frac{p_{\mathcal{N}_1}^{(2,R1,+)}}{p_{\mathcal{N}_1}^{(2,R1,+)} + \sum_{i=2}^{d_c} i p_{\mathcal{N}_i}^{(2,R1,1)}},$$

where,

$$p_{\mathcal{N}_1}^{(2,R1,+)} = \sum_{k=0}^{d_c-1} p_{\mathcal{N}_{1,k}}^{(2,R1,1,+)}.$$

Hence, the probability $p_{\mathcal{K}_{i\uparrow j}}^{(2,R1)}$, $i \in \{0,1\}$, that a variable node from $\mathcal{K}_i^{(1,R1,2)}$ is moved into $\mathcal{K}_j^{(2,R1,2)}$, $i \leq j \leq d_v$, is calculated as follows:

$$p_{\mathcal{K}_{i\uparrow j}}^{(2,R1)} = \binom{d_v - i}{j - i} \left(p_k^{(2,R1)}\right)^{j-i} \left(1 - p_k^{(2,R1)}\right)^{d_v - j}. \tag{A.16}$$

Finally, the probability $p_{\mathcal{K}_j}^{(2,R1)}$ is calculated by:

$$p_{\mathcal{K}_j}^{(2,R1)} = \sum_{i=0}^{1} p_{\mathcal{K}_i}^{(1,R1,2)} p_{\mathcal{K}_{i\uparrow j}}^{(2,R1)}, \qquad 0 \leq j \leq d_v.$$

The probability $p_{\mathcal{K}_j}^{(2,R1,2)}$ that a variable node in the support set belongs to the set $\mathcal{K}_j^{(2,R1,2)}$, is calculated based on the set of verified variable nodes at this stage. Variable nodes in the set $\mathcal{K}_j^{(2,R1)}$, $2 \leq j \leq d_v$, are all verified. Variable nodes in the set $\mathcal{K}_0^{(2,R1)}$ are all left unverified, and a fraction of the variable nodes in the set $\mathcal{K}_1^{(2,R1)}$ are verified. The set $\mathcal{K}_1^{(2,R1)}$ consists of two sets of variable nodes: $\mathcal{K}_{0\uparrow 1}^{(2,R1)}$ and $\mathcal{K}_{1\uparrow 1}^{(2,R1)}$. A variable node in $\mathcal{K}_{0\uparrow 1}^{(2,R1)}$ has a neighbor in $\mathcal{N}_1^{(2,R1,+)}$, while a variable node in $\mathcal{K}_{1\uparrow 1}^{(2,R1)}$ has a neighbor in $\mathcal{N}_1^{(2,R1,C)}$. Variable nodes in the two sets $\mathcal{K}_{0\uparrow 1}^{(2,R1)}$ and $\mathcal{K}_{1\uparrow 1}^{(2,R1)}$ are verified if and only if they are neighbor to check nodes in the sets $\mathcal{N}_{1,0}^{(2,R1,1,+)}$ and $\mathcal{N}_{1,0}^{(2,R1,1,C)}$, respectively. Let $f^{(2,R1,+)}$ and $f^{(2,R1,C)}$ be the probabilities that a variable node in $\mathcal{K}_{0\uparrow 1}^{(2,R1)}$ and $\mathcal{K}_{1\uparrow 1}^{(2,R1)}$ is verified at iteration 2, R1-HR2, respectively. We have:

$$f^{(2,R1,+)} = \frac{p_{\mathcal{N}_{1,0}}^{(2,R1,1,+)}}{p_{\mathcal{N}_1}^{(2,R1,+)}}, \qquad f^{(2,R1,C)} = \frac{p_{\mathcal{N}_{1,0}}^{(2,R1,1,C)}}{p_{\mathcal{N}_1}^{(2,R1,C)}}.$$

Finally, for the set of probabilities $p_{\mathcal{K}_j}^{(2,R1,2)}$, we have:

$$\begin{aligned}
p_{\mathcal{K}_0}^{(2,R1,2)} &= \frac{1}{N^{(2,R1)}} p_{\mathcal{K}_0}^{(2,R1)} = \frac{1}{N^{(2,R1)}} p_{\mathcal{K}_0}^{(1,R1,2)} p_{\mathcal{K}_{0\uparrow 0}}^{(2,R1)}, \\
p_{\mathcal{K}_1}^{(2,R1,2,+)} &= \frac{1}{N^{(2,R1)}} p_{\mathcal{K}_0}^{(1,R1,2)} p_{\mathcal{K}_{0\uparrow 1}}^{(2,R1)} \left(1 - f^{(2,R1,+)}\right), \\
p_{\mathcal{K}_1}^{(2,R1,2,C)} &= \frac{1}{N^{(2,R1)}} p_{\mathcal{K}_1}^{(1,R1,2)} p_{\mathcal{K}_{1\uparrow 1}}^{(2,R1)} \left(1 - f^{(2,R1,C)}\right), \\
p_{\mathcal{K}_j}^{(2,R1,2)} &= 0, \qquad j = 2, \cdots, d_v.
\end{aligned}$$

The normalization factor $N^{(2,R1)}$ is calculated by:

$$N^{(2,R1)} = p_{\mathcal{K}_0}^{(1,R1,2)} p_{\mathcal{K}_{0\uparrow 1}}^{(2,R1)} \left(1 - f^{(2,R1,+)}\right) + p_{\mathcal{K}_1}^{(1,R1,2)} p_{\mathcal{K}_{1\uparrow 1}}^{(2,R1)} \left(1 - f^{(2,R1,C)}\right) + p_{\mathcal{K}_0}^{(1,R1,2)} p_{\mathcal{K}_{0\uparrow 0}}^{(2,R1)}.$$

The probability $\alpha^{(3)}$ that a variable node in $\mathcal{K}^{(2)}$ remains unverified after iteration 2 is calculated as follows:

$$\alpha^{(3)} = \alpha^{(2)} N^{(2,R1)}.$$

A.5.8 Iteration two, R2-HR1 (Regrouping of check nodes in sets $\mathcal{N}_{i,j}$ based on the index i)

In what follows, we find the probability $p_{d\neq 1}^{(2,R2)}$ defined in Part A.5.4. In the derivation, we use the notation $s_e = 1$ (or $s_e = 0$) to denote a verified (or unverified) message passing over edge e from a variable node to a check node. (For simplicity, some superscripts are omitted. They appear when there is a risk of ambiguity.)

$$
\begin{aligned}
p_{d\neq 1}^{(2,R2)} &= \Pr[s_e = 1 | v_e \in \mathcal{K}^{(2)}, c_e \in \bigcup_{j=2}^{d_c} \mathcal{N}_j^{(2,R1,1)}] \\
&= 1 - \Pr[v_e \in \mathcal{K}_0^{(2,R1)}, s_e = 0 | v_e \in \mathcal{K}^{(2)}, c_e \in \bigcup_{j=2}^{d_c} \mathcal{N}_j] \\
&\quad - \Pr[v_e \in \mathcal{K}_1^{(2,R1)}, s_e = 0 | v_e \in \mathcal{K}^{(2)}, c_e \in \bigcup_{j=2}^{d_c} \mathcal{N}_j] \\
&= 1 - \Pr[v_e \in \mathcal{K}_0 | v_e \in \mathcal{K}, c_e \in \bigcup_{j=2}^{d_c} \mathcal{N}_j] \\
&\quad - \Pr[v_e \in \mathcal{K}_{0\uparrow 1} | v_e \in \mathcal{K}, c_e \in \bigcup_{j=2}^{d_c} \mathcal{N}_j] \Pr[s_e = 0 | v_e \in \mathcal{K}_{0\uparrow 1}, v_e \in \mathcal{K}, c_e \in \bigcup_{j=2}^{d_c} \mathcal{N}_j] \\
&\quad - \Pr[v_e \in \mathcal{K}_{1\uparrow 1} | v_e \in \mathcal{K}, c_e \in \bigcup_{j=2}^{d_c} \mathcal{N}_j] \Pr[s_e = 0 | v_e \in \mathcal{K}_{1\uparrow 1}, v_e \in \mathcal{K}, c_e \in \bigcup_{j=2}^{d_c} \mathcal{N}_j] \\
&= 1 - \frac{\Pr[v_e \in \mathcal{K}_0 | v_e \in \mathcal{K}] \Pr[c_e \in \bigcup_{j=2}^{d_c} \mathcal{N}_j | v_e \in \mathcal{K}_0, v_e \in \mathcal{K}]}{\Pr[c_e \in \bigcup_{j=2}^{d_c} \mathcal{N}_j | v_e \in \mathcal{K}]} \\
&\quad - \frac{\Pr[v_e \in \mathcal{K}_{0\uparrow 1} | v_e \in \mathcal{K}] \Pr[c_e \in \bigcup_{j=2}^{d_c} \mathcal{N}_j | v_e \in \mathcal{K}_{0\uparrow 1}, v_e \in \mathcal{K}]}{\Pr[c_e \in \bigcup_{j=2}^{d_c} \mathcal{N}_j | v_e \in \mathcal{K}]} \left(1 - f^{(2,R1,+)}\right) \\
&\quad - \frac{\Pr[v_e \in \mathcal{K}_{1\uparrow 1} | v_e \in \mathcal{K}] \Pr[c_e \in \bigcup_{j=2}^{d_c} \mathcal{N}_j | v_e \in \mathcal{K}_{1\uparrow 1}, v_e \in \mathcal{K}]}{\Pr[c_e \in \bigcup_{j=2}^{d_c} \mathcal{N}_j | v_e \in \mathcal{K}]} \left(1 - f^{(2,R1,C)}\right) \\
&= 1 - \frac{p_{\mathcal{K}_0}^{(1,R1,2)} p_{\mathcal{K}_{0\uparrow 0}}^{(2,R1)}}{1 - p^{(2,R1)}} \\
&\quad - \frac{\left(\frac{d_v - 1}{d_v}\right)}{1 - p^{(2,R1)}} \left(p_{\mathcal{K}_0}^{(1,R1,2)} p_{\mathcal{K}_{0\uparrow 1}}^{(2,R1)} \left(1 - f^{(2,R1,+)}\right) + p_{\mathcal{K}_1}^{(1,R1,2)} p_{\mathcal{K}_{1\uparrow 1}}^{(2,R1)} \left(1 - f^{(2,R1,C)}\right) \right),
\end{aligned}
$$

where,

$$
p^{(2,R1)} = \frac{p_{\mathcal{N}_1}^{(2,R1,+)} + p_{\mathcal{N}_1}^{(2,R1,C)}}{\alpha^{(2)} d_c}.
$$

Hence, the probability $p^{(2,R2)}_{\mathcal{N}_{i\downarrow k,j}}$, $2 \le i \le d_c$, $0 \le k \le i$, $0 \le j \le d_c - i$, is calculated as follows:

$$p^{(2,R2)}_{\mathcal{N}_{i\downarrow k,j}} = \binom{i}{k} \left(p^{(2,R2)}_{d\neq 1}\right)^{i-k} \left(1 - p^{(2,R2)}_{d\neq 1}\right)^{k}.$$

The evolution of sets $\mathcal{N}_1^{(2,R1,+)}$ and $\mathcal{N}_1^{(2,R1,C)}$ is a bit more involved. Let a check node $c \in \mathcal{N}_1^{(2,R1,+)}$. Suppose, c is neighbor to a variable node $v \in \mathcal{K}^{(2,R1)}_{0\uparrow 1}$. Variable node v is verified if and only if c belongs to the subset $\mathcal{N}^{(2,R1,1,+)}_{1,0}$. Hence, c moves to the set of zero-valued check nodes if it belongs to the set $\mathcal{N}^{(2,R1,1,+)}_{1,0}$. Now, suppose c is neighbor to a variable node $v' \in \bigcup_{i=2}^{d_v} \mathcal{K}^{(2,R1)}_{0\uparrow i}$, or $v' \in \bigcup_{i=2}^{d_v} \mathcal{K}^{(2,R1)}_{1\uparrow i}$. Since the variable node v' is verified with probability 1, check node c becomes a zero-valued check node with probability 1 as well. A similar argument holds true for the set of check nodes in $\mathcal{N}_1^{(2,R1,C)}$ and variable nodes in $\mathcal{K}^{(2,R1)}_{1\uparrow 1}$. Therefore, we need to further divide check nodes in the sets $\mathcal{N}_1^{(2,R1,+)}$ and $\mathcal{N}_1^{(2,R1,C)}$ based on whether or not they are neighbor to variable nodes in the sets $\mathcal{K}^{(2,R1)}_{0\uparrow 1}$ and $\mathcal{K}^{(2,R1)}_{1\uparrow 1}$, respectively.

We partition the set $\mathcal{N}_1^{(2,R1,+)}$ into subsets $\mathcal{N}_1^{(2,R2,+,O)}$ and $\mathcal{N}_1^{(2,R2,+,F)}$. The set $\mathcal{N}_1^{(2,R2,+,F)}$ consists of check nodes with a neighboring variable node in $\mathcal{K}^{(\ell,R1)}_{0\uparrow 1}$. The rest of the check nodes in $\mathcal{N}_1^{(2,R1,+)}$ are all collected in the set $\mathcal{N}_1^{(2,R2,+,O)}$.

Similarly, the set $\mathcal{N}_1^{(2,R1,C)}$ is partitioned into subsets $\mathcal{N}_1^{(2,R2,C,O)}$ and $\mathcal{N}_1^{(2,R2,C,F)}$, where $\mathcal{N}_1^{(2,R2,C,F)}$ consists of check nodes with a neighboring variable node in $\mathcal{K}^{(\ell,R1)}_{1\uparrow 1}$. The rest of the check nodes in $\mathcal{N}_1^{(2,R1,C)}$ are collected in the set $\mathcal{N}_1^{(2,R2,C,O)}$. Figure A.3 depicts the relationship between the sets.

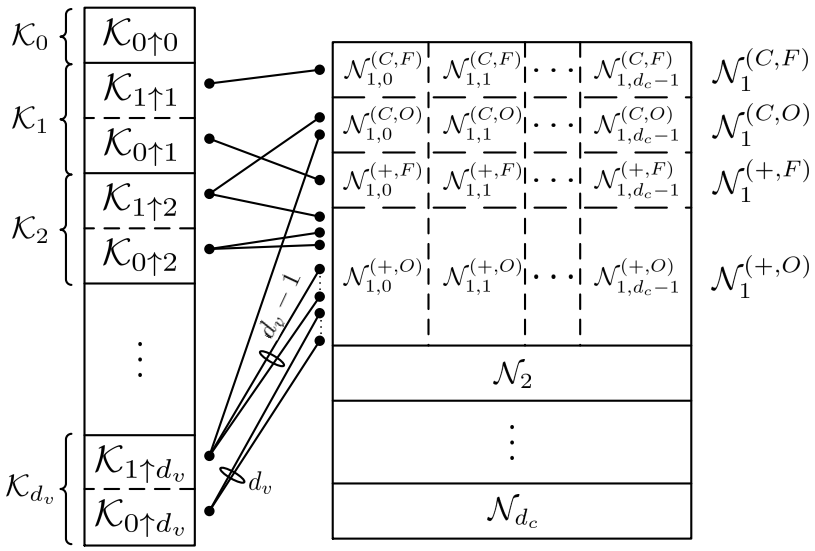


Figure A.3: Graphical description of sets $\mathcal{N}^{(\ell,R2,+,F)}_{1,i}$, $\mathcal{N}^{(\ell,R2,+,O)}_{1,i}$, $\mathcal{N}^{(\ell,R2,C,F)}_{1,i}$, $\mathcal{N}^{(\ell,R2,C,O)}_{1,i}$, and $\mathcal{K}_{i\uparrow j}$.

Let $p_{\mathcal{N}_{1,i}}^{(2,R2,+,O)}$ and $p_{\mathcal{N}_{1,i}}^{(2,R2,+,F)}$, for $0 \leq i \leq d_c - 1$, denote the probabilities that a check node belongs to sets $\mathcal{N}_{1,i}^{(2,R2,+,O)}$ and $\mathcal{N}_{1,i}^{(2,R2,+,F)}$, respectively. Probabilities $p_{\mathcal{N}_{1,i}}^{(2,R2,C,O)}$ and $p_{\mathcal{N}_{1,i}}^{(2,R2,C,F)}$ are defined similarly. The calculation of the probability $p_{\mathcal{N}_{1,i}}^{(2,R2,+,F)}$, $0 \leq i \leq d_c - 1$, follows:

$$
\begin{aligned}
p_{\mathcal{N}_{1,i}}^{(2,R2,+,F)} &:= \Pr[c \in \mathcal{N}_{1,i}^{(2,R2,+,F)}] = \Pr[c \in \mathcal{N}_{1,i}^{(2,R1,1,+)}] \Pr[c \in \mathcal{N}_{1,i}^{(2,R2,+,F)} | c \in \mathcal{N}_{1,i}^{(2,R1,1,+)}] \\
&= \Pr[c \in \mathcal{N}_{1,i}^{(2,R1,1,+)}] \Pr[v_e \in \mathcal{K}_{0\uparrow 1}^{(2,R2)} | c_e \in \mathcal{N}_{1,i}^{(2,R1,1,+)}, v_e \in \mathcal{K}^{(2)}] \\
&= \Pr[c \in \mathcal{N}_{1,i}^{(2,R1,1,+)}] \frac{\Pr[v_e \in \mathcal{K}_{0\uparrow 1}^{(2,R2)} | v_e \in \mathcal{K}^{(2)}] \Pr[c_e \in \mathcal{N}_{1,i}^{(2,R1,1,+)} | v_e \in \mathcal{K}_{0\uparrow 1}^{(2,R2)}, v_e \in \mathcal{K}^{(2)}]}{\Pr[c_e \in \mathcal{N}_{1,i}^{(2,R1,1,+)} | v_e \in \mathcal{K}^{(2)}]} \\
&= \Pr[c \in \mathcal{N}_{1,i}^{(2,R1,1,+)}] \frac{\Pr[v_e \in \mathcal{K}_{0\uparrow 1}^{(2,R2)} | v_e \in \mathcal{K}^{(2)}] \Pr[c_e \in \mathcal{N}_{1,i}^{(2,R1,1,+)} | v_e \in \mathcal{K}_{0\uparrow 1}^{(2,R2)}, v_e \in \mathcal{K}^{(2)}]}{A + B} \\
&= \frac{p_{\mathcal{N}_{1,i}}^{(2,R1,1,+)} p_{\mathcal{K}_0}^{(1,R1,2)} p_{\mathcal{K}_{0\uparrow 1}}^{(2,R1)}}{d_v p_{\mathcal{K}_0}^{(1,R1,2)} p_k^{(2,R1)} + (d_v - 1)\, p_{\mathcal{K}_1}^{(1,R1,2)} p_k^{(2,R1)}}.
\end{aligned} \tag{A.17}
$$

where (using (A.16)),

$$
\begin{aligned}
A &= \sum_{j=1}^{d_v} \Pr[v_e \in \mathcal{K}_{0\uparrow j}^{(2,R2)}, c_e \in \mathcal{N}_{1,i}^{(2,R1,1,+)} | v_e \in \mathcal{K}^{(2)}] = \sum_{j=1}^{d_v} j p_{\mathcal{K}_0}^{(1,R1,2)} p_{\mathcal{K}_{0\uparrow j}}^{(2,R1)} = d_v p_{\mathcal{K}_0}^{(1,R1,2)} p_k^{(2,R1)}, \\
B &= \sum_{j=2}^{d_v} \Pr[v_e \in \mathcal{K}_{1\uparrow j}^{(2,R2)}, c_e \in \mathcal{N}_{1,i}^{(2,R1,1,+)} | v_e \in \mathcal{K}^{(2)}] = \sum_{j=2}^{d_v} (j-1) p_{\mathcal{K}_1}^{(1,R1,2)} p_{\mathcal{K}_{1\uparrow j}}^{(2,R1)} \\
&= (d_v - 1)\, p_{\mathcal{K}_1}^{(1,R1,2)} p_k^{(2,R1)}.
\end{aligned}
$$

Also,

$$
p_{\mathcal{N}_{1,i}}^{(2,R2,+,O)} = p_{\mathcal{N}_{1,i}}^{(2,R1,1,+)} - p_{\mathcal{N}_{1,i}}^{(2,R2,+,F)}.
$$

Following a similar approach, we have:

$$
p_{\mathcal{N}_{1,i}}^{(2,R2,C,F)} = p_{\mathcal{N}_{1,i}}^{(2,R1,1,C)} p_{\mathcal{K}_{1\uparrow 1}}^{(2,R1)}, \qquad p_{\mathcal{N}_{1,i}}^{(2,R2,C,O)} = p_{\mathcal{N}_{1,i}}^{(2,R1,1,C)} - p_{\mathcal{N}_{1,i}}^{(2,R2,C,F)}.
$$

All check nodes in the sets $\mathcal{N}_1^{(2,R2,+,O)}$, $\mathcal{N}_1^{(2,R2,C,O)}$, $\mathcal{N}_{1,0}^{(2,R2,+,F)}$, and $\mathcal{N}_{1,0}^{(2,R2,C,F)}$ are moved into the set $\mathcal{N}_0^{(2,R2,1)}$ after R2-HR1. On the other hand, check nodes in the sets $\mathcal{N}_{1,i}^{(2,R2,+,F)}$, and $\mathcal{N}_{1,i}^{(2,R2,C,F)}$, $1 \leq i \leq d_c - 1$, are moved into the sets $\mathcal{N}_{1,i}^{(2,R2,1,C)}$. Consequently, we have ($2 \leq k \leq d_c$):

$$
p_{\mathcal{N}_{k,j}}^{(2,R2,1)} = \sum_{i=k}^{d_c} p_{\mathcal{N}_{i,j}}^{(2,R1,1)} p_{\mathcal{N}_{i\downarrow k,j}}^{(2,R2)}, \quad 0 \leq j \leq d_c - i,
$$

$$p_{\mathcal{N}_{1,j}}^{(2,R2,1,+)} = \sum_{i=2}^{d_c} p_{\mathcal{N}_{i,j}}^{(2,R1,1)} p_{\mathcal{N}_{i\downarrow 2,j}}^{(2,R2)}, \quad 1 \leq j \leq d_c - 1,$$

$$p_{\mathcal{N}_{1,j}}^{(2,R2,1,C)} = p_{\mathcal{N}_{1,j}}^{(2,R2,C,F)} + p_{\mathcal{N}_{1,j}}^{(2,R2,+,F)}, \quad 1 \leq j \leq d_c - 1,$$

$$p_{\mathcal{N}_{0,j}}^{(2,R2,1)} = p_{\mathcal{N}_{0,j}}^{(2,R1,1)} + p_{\mathcal{N}_{1,j}}^{(2,R2,C,O)} + p_{\mathcal{N}_{1,j}}^{(2,R2,+,O)} + \sum_{i=2}^{d_c} p_{\mathcal{N}_{i,j}}^{(2,R1,1)} p_{\mathcal{N}_{i\downarrow 0,j}}^{(2,R2)}, \quad 0 \leq j \leq d_c - 1.$$

A.5.9 Iteration two, R2-HR2 (Verification of variable nodes based on ZCN)

At this stage, the probability $p_{\Delta_i}^{(2,R2,2)}$, $0 \leq i \leq d_v$, is calculated as follows:

$$p_{\Delta_i}^{(2,R2,2)} = \binom{d_v}{i} \left(p_{\delta}^{(2,R2)}\right)^i \left(1 - p_{\delta}^{(2,R2)}\right)^{d_v - i},$$

where $p_{\delta}^{(2,R2)}$ is given by (A.15) with the appropriate change of probabilities to reflect the values corresponding to iteration 2. Lastly, the probability that a variable node is zero-valued and remains unverified for iteration 3 is given by:

$$p_{\Delta}^{(3)} = p_{\Delta}^{(2)} p_{\Delta_0}^{(2,R2,2)}.$$

A.5.10 Iterations three and beyond

The analysis of an iteration ℓ, $\ell \geq 3$, is similar to that of iteration 2. The summary of update equations for SBB is given in Tables 4.10 and 4.11.

Appendix B

Analysis of SBB over Irregular Graphs

B.1 Iteration Zero

In this section we assume that for a variable degree d_v we have $1 \leq d_v \leq d_{v,\max}$ and for a check degree d_c we have $1 \leq d_c \leq d_{c,\max}$. The constraints on typical d_v and d_c values will be omitted for ease of presentation henceforth. They appear when there is a chance of ambiguity. Throughout the analysis we use the following notations.

- Let v_e and c_e denote the edge sockets at variable and check side connected by edge e, respectively.
- Let $\bar{d}_v := \sum_{i=1}^{d_{v,\max}} i\lambda_i$ be the average variable degree. Similarly, let $\bar{d}_c := \sum_{i=1}^{d_{c,\max}} i\rho_i$ be the average check degree.
- Let $\mathcal{E}_i^v$ and $\mathcal{E}_i^c$ denote the set of edges emanating from variable nodes and check nodes of degree i, respectively.
- Let $\mathcal{V}_i$ and $\mathcal{C}_i$ denote the set of variable nodes and check nodes of degree i, respectively.

Let $\eta(x)$ denote the variable *edge* degree distribution; η_i denotes the fraction of *edges* emanating from variable nodes of degree i. In the analysis sometimes we need to find the elements η_i in terms of λ_i. Using combinatorial arguments, it can easily be shown that:

$$\eta_i := \Pr[e \in \mathcal{E}_i^v] = \frac{i\lambda_i}{\sum_{i=1}^{d_{v,\max}} i\lambda_i} = \frac{i\lambda_i}{\bar{d}_v} \qquad \text{and similarly} \qquad \Pr[e \in \mathcal{E}_i^c] = \frac{i\rho_i}{\bar{d}_c}. \tag{B.1}$$

Iteration zero consists of only one round and hence, two half-rounds. In the first half-round, check nodes pass their measurement values along with their degrees to their neighboring variable nodes. In the second half-round, variable nodes process

the incoming messages. A variable node is verified if it receives at least one message with a value equal to zero. In this case, the variable node is verified with a value equal to zero according to the ZCN rule. The set of all variable nodes verified in this half-round make the set $\mathcal{R}^{(1)}$. Since no variable node in the support set is verified, we have: $\alpha^{(1)} = \alpha^{(0)}$.

Let $\mathcal{N}^{(0,R2,1)}_{i,d_c-i}(d_c)$ denote the set of check nodes of degree d_c with i neighboring variable nodes in the support set, and therefore $d_c - i$ neighboring zero-valued variable nodes. The probability $p^{(0)}_{\mathcal{N}_{i,d_c-i}}(d_c)$ defined as the probability that a check node of degree d_c belongs to the set $\mathcal{N}^{(0)}_{i,d_c-i}(d_c)$ is calculated as follows:

$$p^{(0)}_{\mathcal{N}_{i,d_c-i}}(d_c) = \binom{d_c}{i} \left(\alpha^{(0)}\right)^i \left(1-\alpha^{(0)}\right)^{d_c-i}, \qquad 0 \le i \le d_c.$$

Let $\Delta^{(0,R2,2)}_j(d_v)$ denote the set of zero-valued variable nodes of degree d_v so that each variable node in this set receives j messages with value equal to zero from its neighboring check nodes. The probability $p^{(0,R2,2)}_{\Delta_j}(d_v)$ defined as the probability that a zero-valued variable node of degree d_v belongs to the set $\Delta^{(0,R2,2)}_j(d_v)$ is calculated as follows:

$$\begin{aligned} p^{(0,R2,2)}_{\Delta_j}(d_v) &= \Pr[v \in \Delta^{(0,R2,2)}_j(d_v) | v \in \Delta^{(0)}, v \in \mathcal{V}^n_{d_v}] \\ &= \binom{d_v}{j} \left(p^{(0,R2)}_\delta\right)^j \left(1-p^{(0,R2)}_\delta\right)^{d_v-j}, \quad 0 \le j \le d_v, \end{aligned}$$

where $p^{(0,R2)}_\delta$ is the probability that an edge adjacent to a zero-valued variable node carries a message with value equal to zero. This probability is independent of the degree of the variable nodes and is calculated as follows:

$$\begin{aligned} p^{(0,R2)}_\delta &= \Pr[c_e \in \bigcup_i \mathcal{N}_{0,i}(i) | v_e \in \Delta] \\ &= \sum_i \Pr[c_e \in \mathcal{N}_{0,i}(i) | v_e \in \Delta] = \sum_i \Pr[c_e \in \mathcal{N}_{0,i}(i), e \in \mathcal{E}^c_i | v_e \in \Delta] \\ &= \sum_i \frac{\Pr[e \in \mathcal{E}^c_i] \Pr[c_e \in \mathcal{N}_{0,i}(i) | e \in \mathcal{E}^c_i] \Pr[v_e \in \Delta | c_e \in \mathcal{N}_{0,i}(i), e \in \mathcal{E}^c_i]}{\Pr[v_e \in \Delta]} \\ &= \sum_i \frac{\frac{i\rho_i}{\bar{d}_c} \times p^{(0)}_{\mathcal{N}_{0,i}}(i) \times 1}{1-\alpha^{(0)}} = \frac{1}{\bar{d}_c} \sum_{i=1}^{d_{c,\max}} i\rho_i \left(1-\alpha^{(0)}\right)^{i-1}. \end{aligned} \tag{B.2}$$

Let $p^{(1)}_\Delta(d)$ denote the probability that a variable node of degree d has a zero value and is not verified according to ZCN at iteration 0; does not receive any message with

value equal to zero in the second half-round of iteration zero. We have:

$$\begin{aligned} p_{\Delta}^{(1)}(d) &= \Pr[v \in \Delta^{(1)} | v \in \mathcal{V}_d^n] = \Pr[v \in \Delta^{(0)}, v \in \Delta_0^{(0,R2,2)}(d) | v \in \mathcal{V}_d^n] \\ &= \Pr[v \in \Delta^{(0)} | v \in \mathcal{V}_d^n] \Pr[v \in \Delta_0^{(0,R2,2)}(d) | v \in \Delta^{(0)}, v \in \mathcal{V}_d^n] \\ &= (1 - \alpha^{(0)}) \left(1 - p_{\delta}^{(0,R2)}\right)^d, \quad 1 \leq d \leq d_{v,\max}. \end{aligned}$$

Since no element of the support set is verified at iteration zero, we have $\mathcal{K}^{(0)} = \mathcal{K}^{(1)}$, and hence,

$$\alpha^{(1)} = \Pr[v \in \mathcal{K}^{(1)}] = \alpha^{(0)}.$$

The edges adjacent to a check node are partitioned into two sets: $\mathcal{K}$-edges and Δ-edges. $\mathcal{K}$-edges are connected to variable nodes in the support set, while Δ-edges are connected to zero-valued variable nodes. Therefore, a check node in the set $\mathcal{N}_{i,j}^{(1,R1,1)}(d_c)$ $(0 \leq i \leq d_c, 0 \leq j \leq d_c - i)$ has i, $\mathcal{K}$-edges and j, Δ-edges. The regrouping of check nodes is discussed in the next section.

B.2 Iteration One, R1-HR1 (Regrouping check nodes in sets $\mathcal{N}_{i,j}(d_c)$ based on the index j)

In this section, we adopt the notation $\mathcal{N}_i(d_c)$, $0 \leq i \leq d_c$, with any superscript, to denote the set $\bigcup_{j=0}^{d_c-i} \mathcal{N}_{i,j}(d_c)$. The verified messages sent from zero-valued variable nodes to check nodes at the end of iteration zero, are processed at check nodes at iteration 1, R1-HR1. We partition the set of edges adjacent to a variable node in the set $\Delta_j(d)$, $0 \leq j \leq d$, into $\mathcal{N}_{=0}$-edges and $\mathcal{N}_{\neq 0}$-edges. $\mathcal{N}_{=0}$-edges are connected to zero-valued check nodes (check nodes in the set $\bigcup_i \mathcal{N}_0(i)$), while $\mathcal{N}_{\neq 0}$-edges are connected to non-zero check nodes.

Let $\mathcal{N}_{i,d_c-i\downarrow j}^{(1,R1)}(d_c)$ denote the set of check nodes of degree d_c that are regrouped from $\mathcal{N}_{i,d_c-i}^{(0,R2,1)}(d_c)$ to $\mathcal{N}_{i,j}^{(1,R1,1)}(d_c)$, $1 \leq i \leq d_c$ and $0 \leq j \leq d_c - i$. To analyze the regrouping, we need to find the probability $p_{\mathcal{E}_R}$ of an edge in the set of Δ-edges to carry a non-verified message. Such edges are connected to the set $\Delta_0(d), 1 \leq d \leq d_{v,\max}$.

We have:

$$
\begin{aligned}
p_{\mathcal{E}_R}^{(1,R1)} &= \sum_{d=1}^{d_{v,\max}} \Pr[v_e \in \Delta_0^{(0,R2,2)}(d), e \in \mathcal{V}_d^e | v_e \in \Delta^{(0)}, c_e \notin \bigcup_i \mathcal{N}_0^{(1)}(i)] \\
&= \sum_{d=1}^{d_{v,\max}} \frac{\Pr[e \in \mathcal{V}_d^e, v_e \in \Delta^{(0)}, v_e \in \Delta_0^{(0,R2,2)}(d), c_e \notin \bigcup_i \mathcal{N}_0^{(1)}(i)]}{\Pr[v_e \in \Delta^{(0)}, c_e \notin \bigcup_i \mathcal{N}_0^{(1)}(i)]} \\
&= \sum_{d=1}^{d_{v,\max}} \frac{\Pr[e \in \mathcal{V}_d^e] \Pr[v_e \in \Delta^{(0)} | e \in \mathcal{V}_d^e] \Pr[v_e \in \Delta_0^{(0,R2,2)}(d) | v_e \in \Delta^{(0)}, e \in \mathcal{V}_d^e]}{\sum_{d=1}^{d_{v,\max}} \sum_{i=0}^{d} \Pr[e \in \mathcal{V}_d^e, v_e \in \Delta^{(0)}, v_e \in \Delta_i^{(0,R2,2)}(d), c_e \notin \bigcup_i \mathcal{N}_0^{(1)}(i)]} \\
&\quad \times \Pr[c_e \notin \bigcup_i \mathcal{N}_0^{(1)}(i) | v_e \in \Delta_0^{(0,R2,2)}(d), v_e \in \Delta^{(0)}, e \in \mathcal{V}_d^e] \\
&= \sum_{d=1}^{d_{v,\max}} \frac{\frac{d\lambda_d}{\bar{d}_v}(1-\alpha^{(0)}) p_{\Delta_0}^{(0,R2,2)}(d) \times 1}{\sum_{d=1}^{d_{v,\max}} \sum_{i=0}^{d} \frac{d\lambda_d}{\bar{d}_v}(1-\alpha^{(0)}) p_{\Delta_i}^{(0,R2,2)}(d) \frac{d-i}{d}} = \sum_{d=1}^{d_{v,\max}} \frac{d\lambda_d p_{\Delta_0}^{(0,R2,2)}(d)}{\sum_{d=1}^{d_{v,\max}} \sum_{i=0}^{d} (d-i)\lambda_d p_{\Delta_i}^{(0,R2,2)}(d)} \\
&= \frac{\sum_{d=1}^{d_{v,\max}} d\lambda_d p_{\Delta_0}^{(0,R2,2)}(d)}{\bar{d}_v (1 - p_\delta^{(0,R2)})} = \frac{1}{\bar{d}_v} \sum_{d=1}^{d_{v,\max}} d\lambda_d (1 - p_\delta^{(0,R2)})^{d-1}.
\end{aligned}
$$

where $p_\delta^{(0,R2)}$ is given in (B.2). We thus have the following regrouping of check nodes based on the second index:

$$
p_{\mathcal{N}_{i,d_c-i\downarrow j}}^{(1,R1)}(d_c) = \binom{d_c - i}{j} \left(p_{\mathcal{E}_R}^{(1,R1)}\right)^j \left(1 - p_{\mathcal{E}_R}^{(1,R1)}\right)^{d_c-i-j}, 1 \le i \le d_c, 0 \le j \le d_c - i.
$$

Hence, the probability $p_{\mathcal{N}_{i,j}}^{(1,R1,1)}(d_c)$ is calculated as:

$$
p_{\mathcal{N}_{i,j}}^{(1,R1,1)}(d_c) = p_{\mathcal{N}_{i,d_c-i}}^{(0,R2,1)}(d_c) p_{\mathcal{N}_{i,d_c-i\downarrow j}}^{(1,R1)}(d_c), \qquad 1 \le i \le d_v, \qquad 0 \le j \le d_c - i. \tag{B.3}
$$

B.3 Iteration One, R1-HR2 (Verification of variable nodes based on ECN and D1CN)

In R1-HR2, for each variable degree $d, 1 \le d \le d_{v,\max}$, we divide the set of all unverified variable nodes in the support set $\mathcal{K}^{(1)}(d)$ into subsets $\mathcal{K}_i^{(1,R1)}(d)$, $0 \le i \le d$, according to the received messages from check nodes. The index i denotes the number of neighboring check nodes in the set $\bigcup_{d_c} \mathcal{N}_1^{(1,R1,1)}(d_c)$.

Let $p^{(1,R1)}$ denote the conditional probability that an edge is adjacent to a check node in the set $\bigcup_{d_c} \mathcal{N}_1^{(1,R1,1)}(d_c)$ given that it is adjacent to a variable node in the support set. This probability is independent of the degree of the variable node. We have:

$$\begin{aligned} p^{(1,R1)} &= \sum_{d=1}^{d_{c,\max}} \Pr[e \in \mathcal{C}_d^e, c_e \in \mathcal{N}_1^{(1,R1,1)}(d) | v_e \in \mathcal{K}^{(1)}] \\ &= \sum_{d=1}^{d_{c,\max}} \frac{\Pr[e \in \mathcal{C}_d^e] \Pr[c_e \in \mathcal{N}_1^{(1,R1,1)}(d) | e \in \mathcal{C}_d^e] \Pr[v_e \in \mathcal{K}^{(1)} | c_e \in \mathcal{N}_1^{(1,R1,1)}(d), e \in \mathcal{C}_d^e]}{\Pr[v_e \in \mathcal{K}^{(1)}]} \\ &= \sum_{d=1}^{d_{c,\max}} \frac{\frac{d\rho_d}{\bar{d}_c} \left(\sum_{j=0}^{d-1} p_{\mathcal{N}_{1,j}(d)}^{(1,R1,1)} \right) \times 1/d}{\alpha^{(1)}} = \frac{\sum_{d=1}^{d_{c,\max}} \rho_d \sum_{j=0}^{d-1} p_{\mathcal{N}_{1,j}}^{(1,R1,1)}(d)}{\alpha^{(1)} \bar{d}_c}. \end{aligned} \tag{B.4}$$

Hence, the probability $p_{\mathcal{K}_i}^{(1,R1)}(d)$,$1 \leq d \leq d_{v,\max}, 0 \leq i \leq d$, that a variable node $v \in \mathcal{K}^{(1)}(d)$ of degree d belongs to the set $\mathcal{K}_i^{(1,R1)}(d)$ is calculated as follows:

$$p_{\mathcal{K}_i}^{(1,R1)}(d) = \Pr[v \in \mathcal{K}_i^{(1,R1)}(d) | v \in \mathcal{K}^{(1)}(d)] = \binom{d}{i} \left(p^{(1,R1)}\right)^i \left(1 - p^{(1,R1)}\right)^{d-i}.$$

The verification of variable nodes of degree $d, 1 \leq d \leq d_{v,\max}$, in the support set is as follows:

1. variable nodes in the set $\bigcup_{i=2}^{d} \mathcal{K}_i^{(1,R1)}(d)$ are verified based on the ECN rule.
2. variable nodes in the set $\mathcal{K}_1^{(1,R1)}(d)$ are resolved based on D1CN only if they are neighbor to a check node in the set $\bigcup_{d_c} \mathcal{N}_{1,0}^{(1,R1,1)}(d_c)$.
3. variable nodes in the set $\mathcal{K}_0^{(1,R1)}(d)$ are not resolved in this iteration.

It is worth mentioning that some variable nodes in the set $\bigcup_{i=2}^{d} \mathcal{K}_i^{(1,R1)}(d)$ can also be verified according to D1CN. However, since the probability of false verification is zero and since all such variable nodes are verified, the source of verification is not important in the analysis.

Let $f^{(1,R1)}$ denote the fraction of variable nodes in the set $\bigcup_{d_v=1}^{d_{v,\max}} \mathcal{K}_1^{(1,R1)}(d_v)$ that are verified according to the D1CN rule. This fraction is independent of the degree of the variable nodes and is calculated using the Bayes' rule as follows:

$$f^{(1,R1)} = \Pr[\bigcup_{d=1}^{d_{c,\max}} c_e \in \mathcal{N}_{1,0}^{(1,R1,1)}(d) | v_e \in \bigcup_{d_v=1}^{d_{v,\max}} \mathcal{K}_1^{(1,R1)}(d_v), c_e \in \bigcup_{d_c=1}^{d_{c,\max}} \bigcup_{j=0}^{d_c-1} \mathcal{N}_{1,j}^{(1,R1,1)}(d_c)].$$

Since the formulation is independent of the variable degree, for the sake of presentation

we remove the union notation on d_v and all the superscripts. Hence, we obtain:

$$
\begin{aligned}
f^{(1,R1)} &= \frac{\sum_{d=1}^{d_{c,\max}} \Pr[c_e \in \mathcal{N}_{1,0}(d), e \in \mathcal{C}_d^e] \Pr[v_e \in \mathcal{K}_1(d_v) | c_e \in \mathcal{N}_{1,0}(d)]}{\sum_{d=1}^{d_{c,\max}} \sum_{j=0}^{d-1} \Pr[c_e \in \mathcal{N}_{1,j}(d), e \in \mathcal{C}_d^e] \Pr[v_e \in \mathcal{K}_1(d_v) | c_e \in \mathcal{N}_{1,j}(d)]} \\
&\times \Pr[c_e \in \bigcup_j \mathcal{N}_{1,j}(d) | c_e \in \mathcal{N}_{1,0}(d), v_e \in \bigcup_{d_v=1}^{d_{v,\max}} \mathcal{K}_1(d_v)] \\
&= \frac{\sum_{d=1}^{d_{c,\max}} \frac{d\rho_d}{\bar{d}_c} p_{\mathcal{N}_{1,0}}^{(1,R1,1)}(d) \times \frac{1}{d} \times 1}{\sum_{d=1}^{d_{c,\max}} \sum_{j=0}^{d-1} \frac{d\rho_d}{\bar{d}_c} p_{\mathcal{N}_{1,j}}^{(1,R1,1)}(d) \times \frac{1}{d}} = \frac{\sum_{d=1}^{d_{c,\max}} \rho_d p_{\mathcal{N}_{1,0}}^{(1,R1,1)}(d)}{\sum_{d=1}^{d_{c,\max}} \sum_{j=0}^{d-1} \rho_d p_{\mathcal{N}_{1,j}}^{(1,R1,1)}(d)}.
\end{aligned} \tag{B.5}
$$

Therefore, the probability $\alpha_d^{(2)}$ that a variable node of degree $d, 1 \leq d \leq d_{v,\max}$, in the support set remains unverified at iteration 1 is as follows:

$$
\begin{aligned}
\alpha_d^{(2)} &:= \Pr[v \in \mathcal{K}^{(2)}(d)] = \Pr[v \in \mathcal{K}^{(2)} | v \in \mathcal{D}_d] \\
&= \alpha^{(1)} \left(1 - f^{(1,R1)} p_{\mathcal{K}_1}^{(1,R1)}(d) - \sum_{i=2}^{d} p_{\mathcal{K}_i}^{(1,R1)}(d) \right) \\
&= \alpha^{(1)} \left(p_{\mathcal{K}_0}^{(1,R1)}(d) + \left(1 - f^{(1,R1)}\right) p_{\mathcal{K}_1}^{(1,R1)}(d) \right).
\end{aligned}
$$

The final regrouping of variable nodes of degree $d, 1 \leq d \leq d_{v,\max}$, into sets $\mathcal{K}_i^{(1,R1,2)}(d)$, $0 \leq i \leq d$, is performed by taking into account the set of verified variable nodes of degree d. We have:

$$
\begin{aligned}
p_{\mathcal{K}_0}^{(1,R1,2)}(d) &= \frac{1}{N^{(1,R1)}(d)} p_{\mathcal{K}_0}^{(1,R1)}(d). \\
p_{\mathcal{K}_1}^{(1,R1,2)}(d) &= \frac{1}{N^{(1,R1)}(d)} \left(1 - f^{(1,R1)}\right) p_{\mathcal{K}_1}^{(1,R1)}(d). \\
p_{\mathcal{K}_i}^{(1,R1,2)}(d) &= 0, \qquad 2 \leq i \leq d.
\end{aligned} \tag{B.6}
$$

The normalization factor $N^{(1,R1)}(d)$ is used to make the set of parameters $p_{\mathcal{K}_i}^{(1,R1,2)}(d)$ a valid probability measure, and is calculated as follows:

$$
N^{(1,R1)}(d) = p_{\mathcal{K}_0}^{(1,R1)}(d) + \left(1 - f^{(1,R1)}\right) p_{\mathcal{K}_1}^{(1,R1)}(d) = \frac{\alpha_d^{(2)}}{\alpha^{(1)}}.
$$

B.4 Iteration One, R2-HR1 (Regrouping check nodes in sets $\mathcal{N}_{i,j}(d_c)$ based on the index i)

Let $\mathcal{N}_{i\downarrow k,j}^{(1,R2)}(d)$, $1 \leq d \leq d_{c,\max}, 1 \leq i \leq d, 0 \leq j \leq d-i, 0 \leq k \leq i$, denote the set of check nodes of degree d that are regrouped from $\mathcal{N}_{i,j}^{(1,R1,1)}(d)$ to $\mathcal{N}_{k,j}^{(1,R2,1)}(d)$. We need to consider two separate cases: $i = 1$ and $i \geq 2$, and find the probabilities $p_{d=1}^{(1,R2)}$ and $p_{d\neq 1}^{(1,R2)}$ defined as follows. The probability $p_{d=1}^{(1,R2)}$ is defined as the conditional probability of an edge carrying a verified message (due to the ECN) given that it is a $\mathcal{K}$-edge adjacent to a check node in the set $\bigcup_{d_c=1}^{d_{c,\max}} \mathcal{N}_1^{(1,R1,1)}(d_c)$. The probability $p_{d\neq 1}^{(1,R2)}$ is defined as the conditional probability of an edge carrying a verified message (due to the ECN or D1CN) given that it is a $\mathcal{K}$-edge adjacent to a check node in the set $\bigcup_{d_c=1}^{d_{c,\max}} \bigcup_{i=2}^{d_c} \mathcal{N}_i^{(1,R1,1)}(d_c)$. To find the probability $p_{d=1}^{(1,R2)}$, we shall consider only the variable nodes in the set $\bigcup_{d_v=2}^{d_{v,\max}} \bigcup_{i=2}^{d_v} \mathcal{K}_i^{(1,R1,2)}(d_v)$. This is because variable nodes in the set $\bigcup_{d_v=2}^{d_{v,\max}} \mathcal{K}_1^{(1,R1,2)}(d_v)$ are verified based on D1CN. We proceed as follows:

$$
\begin{aligned}
p_{d=1}^{(1,R2)} &= \Pr[v_e \in \bigcup_{d_v=2}^{d_{v,\max}} \bigcup_{i=2}^{d_v} \mathcal{K}_i^{(1,R1,2)}(d_v) | v_e \in \mathcal{K}^{(1)}, c_e \in \bigcup_{d_c=1}^{d_{c,\max}} \mathcal{N}_1^{(1,R1,1)}(d_c)] \\
&= 1 - \Pr[v_e \in \bigcup_{d_v=1}^{d_{v,\max}} \mathcal{K}_1^{(1,R1,2)}(d_v) | v_e \in \mathcal{K}^{(1)}, c_e \in \bigcup_{d_c=1}^{d_{c,\max}} \mathcal{N}_1^{(1,R1,1)}(d_c)] \\
&= 1 - \sum_{d_v=1}^{d_{v,\max}} \Pr[v_e \in \mathcal{K}_1^{(1,R1,2)}(d_v), e \in \mathcal{V}_{d_v}^e | v_e \in \mathcal{K}^{(1)}, c_e \in \bigcup_{d_c=1}^{d_{c,\max}} \mathcal{N}_1^{(1,R1,1)}(d_c)] \\
&= 1 - \frac{\sum_{d_v=1}^{d_{v,\max}} \Pr[e \in \mathcal{V}_{d_v}^e, v_e \in \mathcal{K}^{(1)}, v_e \in \mathcal{K}_1^{(1,R1,2)}(d_v), c_e \in \bigcup_{d_c=1}^{d_{c,\max}} \mathcal{N}_1^{(1,R1,1)}(d_c)]}{\sum_{d_v=1}^{d_{v,\max}} \sum_{i=0}^{d_v} \Pr[e \in \mathcal{V}_{d_v}^e, v_e \in \mathcal{K}^{(1)}, v_e \in \mathcal{K}_i^{(1,R1,2)}(d_v), c_e \in \bigcup_{d_c=1}^{d_{c,\max}} \mathcal{N}_1^{(1,R1,1)}(d_c)]} \\
&= 1 - \frac{\sum_{d_v=1}^{d_{v,\max}} \Pr[e \in \mathcal{V}_{d_v}^e] \Pr[v_e \in \mathcal{K} | e \in \mathcal{V}_{d_v}^e] \Pr[v_e \in \mathcal{K}_1(d_v) | v_e \in \mathcal{K}(d_v)]}{\sum_{d_v=1}^{d_{v,\max}} \sum_{j=0}^{d_v} \Pr[e \in \mathcal{V}_{d_v}^e, v_e \in \mathcal{K}^{(1)}, v_e \in \mathcal{K}_j^{(1,R1,2)}(d_v), c_e \in \bigcup_{d_c=1}^{d_{c,\max}} \mathcal{N}_1^{(1,R1,1)}(d_c)]} \\
&\quad \times \Pr[c_e \in \bigcup_{d_c=1}^{d_{c,\max}} \mathcal{N}_1(d_c) | v_e \in \mathcal{K}_1(d_v)] \\
&= 1 - \frac{\sum_{i=1}^{d_{v,\max}} \frac{i\lambda_i}{\bar{d}_v} \alpha_i^{(1)} p_{\mathcal{K}_1}^{(1,R1,2)}(i) \frac{1}{i}}{\sum_{i=1}^{d_{v,\max}} \sum_{j=1}^{i} \frac{i\lambda_i}{\bar{d}_v} \alpha_i^{(1)} p_{\mathcal{K}_j}^{(1,R1,2)}(i) \frac{j}{i}} = 1 - \frac{\sum_{i=1}^{d_{v,\max}} \lambda_i \alpha_i^{(1)} p_{\mathcal{K}_1}^{(1,R1,2)}(i)}{\sum_{i=1}^{d_{v,\max}} \lambda_i \alpha_i^{(1)} \sum_{j=1}^{i} j p_{\mathcal{K}_j}^{(1,R1,2)}(i)},
\end{aligned}
$$

where the set of probabilities $p_{\mathcal{K}_{j,i}}^{(1,R1,2)}$ are given by (B.6). Similarly, the probability $p_{d\neq 1}^{(1,R2)}$ is derived as:

$$
p_{d\neq 1}^{(1,R2)} = 1 - \frac{\sum_{i=1}^{d_{v,\max}} i\lambda_i \alpha_i^{(1)} p_{\mathcal{K}_0}^{(1,R1,2)}(i)}{\sum_{i=1}^{d_{v,\max}} \sum_{j=1}^{i} (i-j)\lambda_i \alpha_i^{(1)} p_{\mathcal{K}_j}^{(1,R1,2)}(i)} - \frac{\sum_{i=1}^{d_{v,\max}} (i-1)\lambda_i \alpha_i^{(1)} p_{\mathcal{K}_1}^{(1,R1,2)}(i)}{\sum_{i=1}^{d_{v,\max}} \sum_{j=1}^{i} (i-j)\lambda_i \alpha_i^{(1)} p_{\mathcal{K}_j}^{(1,R1,2)}(i)} \left(1 - f^{(1,R1)}\right). \tag{B.7}
$$

Hence, the probabilities $p^{(1,R2)}_{\mathcal{N}_{i\downarrow k,j}}(d_c)$ and $p^{(1,R2)}_{\mathcal{N}_{1\downarrow 0,j}}(d_c)$ that a check node belongs respectively to the set of check nodes $\mathcal{N}^{(1,R2)}_{i\downarrow k,j}(d_c)$ and $\mathcal{N}^{(1,R2)}_{1\downarrow 0,j}(d_c)$ are calculated as follows:

$$p^{(1,R2)}_{\mathcal{N}_{i\downarrow k,j}}(d_c) = \binom{i}{k}\left(p^{(1,R2)}_{d\neq 1}\right)^{i-k}\left(1-p^{(1,R2)}_{d\neq 1}\right)^{k},\ 2\leq i\leq d_c,\ 0\leq k\leq i,\ 0\leq j\leq d_c-i.$$
$$p^{(1,R2)}_{\mathcal{N}_{1\downarrow 0,j}}(d_c) = p^{(1,R2)}_{d=1},\qquad p^{(1,R2)}_{\mathcal{N}_{1\downarrow 1,j}}(d_c) = 1-p^{(1,R2)}_{d=1},\qquad 1\leq j\leq d_c-i.$$
$$p^{(1,R2)}_{\mathcal{N}_{1\downarrow 0,0}}(d_c) = 1,\qquad p^{(1,R2)}_{\mathcal{N}_{1\downarrow 1,0}}(d_c) = 0,\qquad p^{(1,R2)}_{\mathcal{N}_{0\downarrow 0,j}}(d_c) = 1,\qquad 1\leq j\leq d_c-i.$$

After the regrouping, the probability $p^{(1,R2,1)}_{\mathcal{N}_{k,j}}(d_c)$ that a check node belongs to the set $\mathcal{N}^{(1,R2,1)}_{k,j}(d_c)$ is calculated as follows:

$$p^{(1,R2,1)}_{\mathcal{N}_{k,j}}(d_c) = \sum_{i=k}^{d_c} p^{(1,R1,1)}_{\mathcal{N}_{i,j}}(d_c)\,p^{(1,R2)}_{\mathcal{N}_{i\downarrow k,j}}(d_c),\qquad 0\leq k\leq d_c,\qquad 0\leq j\leq d_c-i.$$

We need to partition the set of check nodes in $\mathcal{N}^{(1,R2,1)}_{1,j}(d_c), 1\leq d_c\leq d_{c,\max}$, into two sets: $\mathcal{N}^{(1,R2,1,+)}_{1,j}(d_c)$ and $\mathcal{N}^{(1,R2,1,C)}_{1,j}(d_c)$. Check nodes in the set $\mathcal{N}^{(1,R2,1,+)}_{1,j}(d_c)$ were moved into the set $\mathcal{N}^{(1,R2,1)}_{1,j}(d_c)$ from all the other sets $\mathcal{N}^{(1,R1,1)}_{i,j}(d_c)$, $2\leq i\leq d_c$. Check nodes in the set $\mathcal{N}^{(1,R2,1,C)}_{1,j}(d_c)$, however, are those that stayed in the set from $\mathcal{N}^{(1,R1,1)}_{1,j}(d_c)$.

Let $p^{(1,R2,1,+)}_{\mathcal{N}_{1,j}}(d_c)$ and $p^{(1,R2,1,C)}_{\mathcal{N}_{1,j}}(d_c)$ denote the probabilities that a check node belongs to the sets $\mathcal{N}^{(1,R2,1,+)}_{1,j}(d_c)$ and $\mathcal{N}^{(1,R2,1,C)}_{1,j}(d_c)$, respectively. We have:

$$p^{(1,R2,1,+)}_{\mathcal{N}_{1,j}}(d_c) = \sum_{i=2}^{d_c} p^{(1,R1,1)}_{\mathcal{N}_{i,j}}(d_c)\,p^{(1,R2)}_{\mathcal{N}_{i\downarrow 1,j}}(d_c),$$
$$p^{(1,R2,1,C)}_{\mathcal{N}_{1,j}}(d_c) = p^{(1,R1,1)}_{\mathcal{N}_{1,j}}(d_c)\,p^{(1,R2)}_{\mathcal{N}_{1\downarrow 1,j}}(d_c) = p^{(1,R2,1)}_{\mathcal{N}_{1,j}}(d_c) - p^{(1,R2,1,+)}_{\mathcal{N}_{1,j}}(d_c).$$

B.5 Iteration One, R2-HR2 (Verification of variable nodes based on ZCN)

The measurement corresponding to check nodes in the set $\bigcup_{d=1}^{d_{c,\max}}\mathcal{N}^{(1,R2,1)}_{0,k}(d)$, $1\leq k\leq d-1$, changes to zero, as the check nodes are no longer connected to an unverified variable node in the support set. Hence, the messages transmitted by such check nodes have a value equal to zero, which in turn verifies some variable nodes (in the set Δ) at R2-HR2 of iteration 1.

The probability $p^{(1,R2)}_{\delta}$ (defined in (B.2)) is calculated as follows. This parameter is needed to find the probability that a zero-valued variable node is verified at this

stage.

$$p_{\delta}^{(1,R2)} = \sum_{d=1}^{d_{c,\max}} \sum_{j=1}^{d-1} \Pr[e \in \mathcal{C}_d^e, c_e \in \mathcal{N}_{0,j}^{(1,R2,1)}(d) | v_e \in \Delta^{(1)}]$$

$$= \frac{\sum_{d=1}^{d_{c,\max}} \sum_{j=1}^{d-1} \Pr[e \in \mathcal{C}_d^e] \Pr[c_e \in \mathcal{N}_{0,j}^{(1,R2,1)}(d) | e \in \mathcal{C}_d^e] \Pr[v_e \in \Delta^{(1)} | e \in \mathcal{C}_d^e, c_e \in \mathcal{N}_{0,j}^{(1,R2,1)}(d)]}{\sum_{d=1}^{d_{c,\max}} \sum_{i=0}^{d} \sum_{j=1}^{d-i} \Pr[e \in \mathcal{C}_d^e] \Pr[c_e \in \mathcal{N}_{i,j}^{(1,R2,1)}(d) | e \in \mathcal{C}_d^e] \Pr[v_e \in \Delta^{(1)} | e \in \mathcal{C}_d^e, c_e \in \mathcal{N}_{i,j}^{(1,R2,1)}(d)]}$$

$$= \frac{\sum_{d=1}^{d_{c,\max}} \sum_{j=1}^{d-1} \frac{d\rho_d}{\bar{d}_c} p_{\mathcal{N}_{0,j}}^{(1,R2,1)}(d) \left(\frac{j}{d}\right)}{\sum_{d=1}^{d_{c,\max}} \sum_{i=0}^{d} \sum_{j=1}^{d-i} \frac{d\rho_d}{\bar{d}_c} p_{\mathcal{N}_{i,j}}^{(1,R2,1)}(d) \left(\frac{j}{d}\right)} = \frac{\sum_{d=1}^{d_{c,\max}} \rho_d \sum_{j=1}^{d-1} j p_{\mathcal{N}_{0,j}}^{(1,R2,1)}(d)}{\sum_{d=1}^{d_{c,\max}} \rho_d \sum_{i=0}^{d} \sum_{j=1}^{d-i} j p_{\mathcal{N}_{i,j}}^{(1,R2,1)}(d)}. \tag{B.8}$$

Hence, the probabilities $p_{\Delta_i}^{(1,R2,2)}(d_v)$ and $p_{\Delta}^{(2)}(d_v)$ are calculated as follows:

$$p_{\Delta_i}^{(1,R2,2)}(d_v) = \binom{d_v}{i} \left(p_{\delta}^{(1,R2)}\right)^i \left(1 - p_{\delta}^{(1,R2)}\right)^{d_v - i}, \qquad 1 \le d_v \le d_{v,\max}, \qquad 0 \le i \le d,$$

$$p_{\Delta}^{(2)}(d_v) = p_{\Delta}^{(1)}(d_v) p_{\Delta_0}^{(1,R2,2)}(d_v).$$

B.6 Iteration Two, R1-HR1 (Regrouping check nodes in sets $\mathcal{N}_{i,j}(d_c)$ based on the index j)

The regrouping of check nodes based on their second index is similar to the process in R1-HR1 at iteration 1. We have:

$$p_{\mathcal{E}_R}^{(2,R1)} = \frac{\sum_{d=1}^{d_{v,\max}} d\lambda_d p_{\Delta_0}^{(1,R2,2)}(d)}{\bar{d}_v (1 - p_{\delta}^{(1,R2)})} = \frac{1}{\bar{d}_v} \sum_{d=1}^{d_{v,\max}} d\lambda_d (1 - p_{\delta}^{(1,R2)})^{d-1}.$$

For the regrouping of check nodes based on the second index we thus have:

$$p_{\mathcal{N}_{i,j\downarrow k}}^{(2,R1)}(d_c) = \binom{j}{k} \left(p_{\mathcal{E}_R}^{(2,R1)}\right)^k \left(1 - p_{\mathcal{E}_R}^{(2,R1)}\right)^{j-k}, \; 1 \le i \le d_c, \; 0 \le j \le d_c - i, \; 0 \le k \le j. \tag{B.9}$$

Hence,

$$p_{\mathcal{N}_{i,k}}^{(2,R1,1)}(d_c) = \sum_{j=k}^{d_c-i} p_{\mathcal{N}_{i,j}}^{(1,R2,1)}(d_c) p_{\mathcal{N}_{i,j\downarrow k}}^{(2,R1)}(d_c),\ 1 \le d_c \le d_{c,\max},\ 2 \le i \le d_c,\ 0 \le k \le d_c - i,$$

$$p_{\mathcal{N}_{1,k}}^{(2,R1,1,+)}(d_c) = \sum_{j=k}^{d_c-1} p_{\mathcal{N}_{1,j}}^{(1,R2,1,+)}(d_c) p_{\mathcal{N}_{1,j\downarrow k}}^{(2,R1)}(d_c),\ 1 \le d_c \le d_{c,\max},\ 0 \le k \le d_c - 1, \quad \text{(B.10)}$$

$$p_{\mathcal{N}_{1,k}}^{(2,R1,1,C)}(d_c) = \sum_{j=k}^{d_c-1} p_{\mathcal{N}_{1,j}}^{(1,R2,1,C)}(d_c) p_{\mathcal{N}_{1,j\downarrow k}}^{(2,R1)}(d_c),\ 1 \le d_c \le d_{c,\max},\ 0 \le k \le d_c - 1. \quad \text{(B.11)}$$

B.7 Iteration Two, R1-HR2 (Verification of variable nodes based on ECN and D1CN)

Edges in the set $\bigcup_{d_c=1}^{d_{c,\max}} \mathcal{N}_1^{(2,R1,1,+)}(d_c)$ are responsible for the regrouping of variable nodes at iteration 2, R1-HR2. We denote by $p_k^{(2,R1)}$ the conditional probability that an edge is adjacent to a check node in $\bigcup_{d_c=1}^{d_{c,\max}} \mathcal{N}_1^{(2,R1,1,+)}(d_c)$ given that 1) it emanates from an unverified non-zero variable node, and 2) it is not adjacent to a check node in the set $\bigcup_{d_c=1}^{d_{c,\max}} \mathcal{N}_1^{(2,R1,1,C)}(d_c)$; i.e. it is connected to a check node in the set $\bigcup_{d_c=1}^{d_{c,\max}} \left(\bigcup_{i=2}^{d_c} \mathcal{N}_i^{(2,R1,1)}(d_c) \cup \mathcal{N}_1^{(2,R1,1,+)}(d_c) \right)$. This probability is calculated as follows:

$$
\begin{aligned}
p_k^{(2,R1)} &= \sum_{d=1}^{d_{c,\max}} \Pr[e \in \mathcal{C}_d^e, c_e \in \mathcal{N}_1^{(2,R1,1,+)}(d) | v_e \in \mathcal{K}^{(2)}, c_e \notin \bigcup_{d_c=1}^{d_{c,\max}} \mathcal{N}_1^{(2,R1,C)}(d_c)] \\
&= \sum_{d=1}^{d_{c,\max}} \frac{\Pr[e \in \mathcal{C}_d^e] \Pr[c_e \in \mathcal{N}_1^{(+)}(d) | e \in \mathcal{C}_d^e] \Pr[v_e \in \mathcal{K}^{(2)} | c_e \in \mathcal{N}_1^{(+)}(d)]}{\Pr[v_e \in \mathcal{K}^{(2)}, c_e \in \left\{\bigcup_{i=2}^{d_c} \mathcal{N}_i^{(2,R1,1)}(d_c) \cup \mathcal{N}_1^{(2,R1,1,+)}(d_c)\right\}]} \\
&\times \Pr[c_e \notin \bigcup_{d_c=1}^{d_{c,\max}} \mathcal{N}_1^{(C)}(d_c) | c_e \in \mathcal{N}_1^{(+)}(d)] \\
&= \frac{\displaystyle\sum_{d=1}^{d_{c,\max}} \sum_{k=0}^{d-1} \frac{d\rho_d}{\bar{d}_c} p_{\mathcal{N}_{1,k}}^{(2,R1,1,+)}(d) \times \frac{1}{d} \times 1}{\displaystyle\sum_{d=1}^{d_{c,\max}} \sum_{k=0}^{d-1} \frac{d\rho_d}{\bar{d}_c} p_{\mathcal{N}_{1,k}}^{(2,R1,1,+)}(d) \times \frac{1}{d} + \sum_{d=1}^{d_{c,\max}} \sum_{i=2}^{d} \sum_{j=0}^{d-i} \frac{d\rho_d}{\bar{d}_c} p_{\mathcal{N}_{i,j}}^{(2,R1,1)}(d) \frac{i}{d}} \\
&= \frac{\displaystyle\sum_{d=1}^{d_{c,\max}} \sum_{k=0}^{d-1} \rho_d p_{\mathcal{N}_{1,k}}^{(2,R1,1,+)}(d)}{\displaystyle\sum_{d=1}^{d_{c,\max}} \sum_{k=0}^{d-1} \rho_d p_{\mathcal{N}_{1,k}}^{(2,R1,1,+)}(d) + \sum_{d=1}^{d_{c,\max}} \sum_{i=2}^{d} \sum_{j=0}^{d-i} i\rho_d p_{\mathcal{N}_{i,j}}^{(2,R1,1)}(d)}.
\end{aligned}
$$

Hence, the probability $p_{\mathcal{K}_{i\uparrow j}}^{(2,R1)}(d_v)$, $1 \le d_v \le d_{v,\max}$, $i \in \{0,1\}$, is calculated as follows:

$$
p_{\mathcal{K}_{i\uparrow j}}^{(2,R1)}(d_v) = \binom{d_v - i}{j - i} \left(p_k^{(2,R1)}\right)^{j-i} \left(1 - p_k^{(2,R1)}\right)^{d_v - j}, \qquad i = 0, 1, \qquad i \le j \le d_v.
$$

Finally, the probability $p_{\mathcal{K}_j}^{(2,R1)}(d_v)$ that a variable node of degree d_v in the support set belongs to the set $\mathcal{K}_j^{(2,R1)}(d_v)$ is calculated by:

$$
p_{\mathcal{K}_j}^{(2,R1)}(d_v) = \sum_{i=0}^{1} p_{\mathcal{K}_i}^{(1,R1,2)}(d_v) p_{\mathcal{K}_{i\uparrow j}}^{(2,R1)}(d_v), \qquad 0 \le j \le d.
$$

The probability $p_{\mathcal{K}_j}^{(2,R1,2)}(d_v)$ is calculated based on the set of verified variable nodes at this stage. Variable nodes in the set $\bigcup_{d_v=1}^{d_{v,\max}} \bigcup_{j=2}^{d_v} \mathcal{K}_j^{(2,R1)}(d_v)$ are all verified. Variable nodes in the set $\bigcup_{d_v=1}^{d_{v,\max}} \mathcal{K}_0^{(2,R1)}(d_v)$ are left intact, and a fraction of the variable nodes in the set $\bigcup_{d_v=1}^{d_{v,\max}} \mathcal{K}_1^{(2,R1)}(d_v)$ are verified. The procedure to find this fraction is as follows.

For any degree d, the set $\mathcal{K}_1^{(2,R1)}(d)$ consists of two sets of variable nodes: $\mathcal{K}_{0\uparrow 1}^{(2,R1)}(d)$ and $\mathcal{K}_{1\uparrow 1}^{(2,R1)}(d)$. Variable nodes in the set $\mathcal{K}_{0\uparrow 1}^{(2,R1)}(d)$ are neighbor to check nodes in the set $\mathcal{N}_1^{(2,R1,1,+)}$, while variable nodes in the set $\mathcal{K}_{1\uparrow 1}^{(2,R1)}(d)$ are neighbor to check nodes in the set $\mathcal{N}_1^{(2,R1,1,C)}$. Variable nodes in $\mathcal{K}_{0\uparrow 1}^{(2,R1)}(d)$ and $\mathcal{K}_{1\uparrow 1}^{(2,R1)}(d)$ are verified at iteration 2, R1-HR2, if they are neighbor to check nodes in the sets $\mathcal{N}_{1,0}^{(2,R1,1,+)}$ and $\mathcal{N}_{1,0}^{(2,R1,1,C)}$, respectively. The parameters $f^{(2,R1,+)}$ and $f^{(2,R1,C)}$, defined as the respective verification probability of variable nodes in $\mathcal{K}_{0\uparrow 1}^{(2,R1)}(d)$ and $\mathcal{K}_{1\uparrow 1}^{(2,R1)}(d)$ at iteration 2, R1-HR2, are calculated similar to (B.5) (independent of the variable node degree):

$$f^{(2,R1,+)} = \frac{\sum_{d=1}^{d_{c,\max}} \rho_d p_{\mathcal{N}_{1,0}}^{(2,R1,1,+)}(d)}{\sum_{d=1}^{d_{c,\max}} \rho_d \sum_{k=0}^{d-1} p_{\mathcal{N}_{1,k}}^{(2,R1,1,+)}(d)}, \qquad f^{(2,R1,C)} = \frac{\sum_{d=1}^{d_{c,\max}} \rho_d p_{\mathcal{N}_{1,0}}^{(2,R1,1,C)}(d)}{\sum_{d=1}^{d_{c,\max}} \rho_d \sum_{k=0}^{d-1} p_{\mathcal{N}_{1,k}}^{(2,R1,1,C)}(d)}.$$

Finally, for the set of probabilities $p_{\mathcal{K}_j}^{(2,R1,2)}(d)$, $1 \le d \le d_{v,\max}$, we have:

$$\begin{aligned}
p_{\mathcal{K}_0}^{(2,R1,2)}(d) &= \frac{1}{N^{(2,R1)}(d)} p_{\mathcal{K}_0}^{(2,R1)}(d) = \frac{1}{N^{(2,R1)}(d)} p_{\mathcal{K}_0}^{(1,R1,2)}(d) p_{\mathcal{K}_{0\uparrow 0}}^{(2,R1)}(d), \\
p_{\mathcal{K}_1}^{(2,R1,2,+)}(d) &= \frac{1}{N^{(2,R1)}(d)} p_{\mathcal{K}_0}^{(1,R1,2)}(d) p_{\mathcal{K}_{0\uparrow 1}}^{(2,R1)}(d) \left(1 - f^{(2,R1,+)}\right), \\
p_{\mathcal{K}_1}^{(2,R1,2,C)}(d) &= \frac{1}{N^{(2,R1)}(d)} p_{\mathcal{K}_1}^{(1,R1,2)}(d) p_{\mathcal{K}_{1\uparrow 1}}^{(2,R1)}(d) \left(1 - f^{(2,R1,C)}\right), \\
p_{\mathcal{K}_j}^{(2,R1,2)}(d) &= 0, \qquad 2 \le j \le d.
\end{aligned}$$

The normalization factor $N^{(2,R1)}(d)$ is calculated by:

$$\begin{aligned}
N^{(2,R1)}(d) = {} & p_{\mathcal{K}_0}^{(1,R1,2)}(d) p_{\mathcal{K}_{0\uparrow 0}}^{(2,R1)}(d) + p_{\mathcal{K}_0}^{(1,R1,2)}(d) p_{\mathcal{K}_{0\uparrow 1}}^{(2,R1)}(d) \left(1 - f^{(2,R1,+)}\right) \\
& + p_{\mathcal{K}_1}^{(1,R1,2)}(d) p_{\mathcal{K}_{1\uparrow 1}}^{(2,R1)}(d) \left(1 - f^{(2,R1,C)}\right).
\end{aligned}$$

The probability that a variable node of degree d belongs to the support set and remains unverified after iteration 2, $\alpha_d^{(3)}$, is calculated as follows:

$$\alpha_d^{(3)} = \alpha_d^{(2)} N^{(2,R1)}(d).$$

B.8 Iteration Two, R2-HR1 (Regrouping check nodes in sets $\mathcal{N}_{i,j}(d_c)$ based on the index i)

The calculation of $p_{d\neq 1}^{(1,R2)}$ is very similar to that in (B.7) and hence the details are removed for brevity.

$$
\begin{aligned}
p_{d\neq 1}^{(2,R2)} = 1 &- \sum_{i=1}^{d_{v,\max}} \frac{i\lambda_i \alpha_i^{(2)} p_{\mathcal{K}_0}^{(2,R1)}(i)}{\sum_{i=1}^{d_{v,\max}} \sum_{j=1}^{i} (i-j)\lambda_i \alpha_i^{(2)} p_{\mathcal{K}_j}^{(2,R1)}(i)} \\
&- \sum_{i=1}^{d_{v,\max}} \frac{(i-1)\lambda_i \alpha_i^{(2)} p_{\mathcal{K}_0}^{(1,R1,2)}(i) p_{\mathcal{K}_{0\uparrow 1}}^{(2,R1)}(i)}{\sum_{i=1}^{d_{v,\max}} \sum_{j=1}^{i} (i-j)\lambda_i \alpha_i^{(2)} p_{\mathcal{K}_j}^{(2,R1)}(i)} \left(1 - f^{(2,R1,+)}\right) \\
&- \sum_{i=1}^{d_{v,\max}} \frac{(i-1)\lambda_i \alpha_i^{(2)} p_{\mathcal{K}_1}^{(1,R1,2)}(i) p_{\mathcal{K}_{1\uparrow 1}}^{(2,R1)}(i)}{\sum_{i=1}^{d_{v,\max}} \sum_{j=1}^{i} (i-j)\lambda_i \alpha_i^{(2)} p_{\mathcal{K}_j}^{(2,R1)}(i)} \left(1 - f^{(2,R1,C)}\right).
\end{aligned}
$$

For $1 \leq d_c \leq d_{c,\max}$, $\quad 2 \leq i \leq d_c$, $\quad 0 \leq k \leq i$, $\quad 0 \leq j \leq d_c - i$, we then have:

$$
p_{\mathcal{N}_{i\downarrow k,j}}^{(2,R2)}(d_c) = \binom{i}{k} \left(p_{d\neq 1}^{(2,R2)}\right)^{i-k} \left(1 - p_{d\neq 1}^{(2,R2)}\right)^{k}.
$$

We explain the evolution of the sets $\mathcal{N}_1^{(2,R1,1,+)}(d_c)$ and $\mathcal{N}_1^{(2,R1,1,C)}(d_c), 1 \leq d_c \leq d_{c,\max}$, in the following. A variable node in the set $\mathcal{K}_{0\uparrow i}^{(2,R1)}(d)$, $1 \leq d \leq d_{v,\max}, 1 \leq i \leq d$, has i neighboring check nodes in the set $\bigcup_{d_c} \mathcal{N}_1^{(2,R1,1,+)}(d_c)$. On the other hand, a variable node in the set $\mathcal{K}_{1\uparrow i}^{(2,R1)}(d)$, $1 \leq d \leq d_{v,\max}, 1 \leq i \leq d$, is neighbor to 1 check node in the set $\bigcup_{d_c} \mathcal{N}_1^{(2,R1,1,C)}(d_c)$ and $i-1$ check nodes in the set $\bigcup_{d_c} \mathcal{N}_1^{(2,R1,1,+)}(d_c)$. Without loss of generality, let $c \in \mathcal{N}_1^{(2,R1,1,+)}(d_c)$. Further, suppose c is neighbor to a variable node $v \in \mathcal{K}_{0\uparrow 1}^{(2,R1)}(d)$. Variable node v is verified if and only if c belongs to the subset $\mathcal{N}_{1,0}^{(2,R1,1,+)}(d_c)$. Hence, c is regrouped as a zero-valued check node if it belongs to the set $\mathcal{N}_{1,0}^{(2,R1,1,+)}(d_c)$. Now, suppose c is neighbor to a variable node $v' \in \bigcup_{i=2}^{d} \mathcal{K}_{0\uparrow i}^{(2,R1)}(d)$, or $v' \in \bigcup_{i=2}^{d} \mathcal{K}_{1\uparrow i}^{(2,R1)}(d)$. Since the variable node v' is verified with probability 1, check node c is regrouped into the set of zero-valued check nodes with probability 1 as well. This argument is independent of the check degree d_c and similarly holds true for check nodes in $\mathcal{N}_1^{(2,R1,1,C)}(d_c)$. Therefore, to regroup check nodes in the sets $\mathcal{N}_1^{(2,R1,1,+)}(d_c)$ and $\mathcal{N}_1^{(2,R1,1,C)}(d_c)$, we need to divide them further based on their neighbors; i.e., whether or not they are neighbor to variable nodes in

the sets $\mathcal{K}_{0\uparrow 1}^{(2,R1)}(d)$ and $\mathcal{K}_{1\uparrow 1}^{(2,R1)}(d)$, respectively.

For any check degree d, we partition the set $\mathcal{N}_1^{(2,R1,1,+)}(d)$ into subsets $\mathcal{N}_1^{(2,R2,+,O)}(d)$ and $\mathcal{N}_1^{(2,R2,+,F)}(d)$. We also partition the set $\mathcal{N}_1^{(2,R1,1,C)}(d)$ into subsets $\mathcal{N}_1^{(2,R2,C,O)}(d)$ and $\mathcal{N}_1^{(2,R2,C,F)}(d)$. Check nodes in sets $\mathcal{N}_1^{(2,R2,+,F)}(d)$ and $\mathcal{N}_1^{(2,R2,C,F)}(d)$ are neighbor to variable nodes in sets $\bigcup_{i=1}^{d_{v,\max}} \mathcal{K}_{0\uparrow 1}^{(2,R1)}(i)$ and $\bigcup_{i=1}^{d_{v,\max}} \mathcal{K}_{1\uparrow 1}^{(2,R1)}(i)$, respectively. Any other check node in the set $\mathcal{N}_1^{(2,R1,1,+)}(d)$, not being part of the set $\mathcal{N}_1^{(2,R2,+,F)}(d)$ is grouped into the set $\mathcal{N}_1^{(2,R2,+,O)}(d)$. Similarly, any other check node in the set $\mathcal{N}_1^{(2,R1,1,C)}(d)$, not being part of the set $\mathcal{N}_1^{(2,R2,C,F)}(d)$ is grouped into the set $\mathcal{N}_1^{(2,R2,C,O)}(d)$.

Let $p_{\mathcal{N}_1}^{(2,R2,+,O)}(d_c)$ and $p_{\mathcal{N}_1}^{(2,R2,+,F)}(d_c), 1 \leq d_c \leq d_{c,\max}$, denote the probabilities that a check node of degree d_c belongs to sets $\mathcal{N}_1^{(2,R2,+,O)}(d_c)$ and $\mathcal{N}_1^{(2,R2,+,F)}(d_c)$, respectively. Probabilities $p_{\mathcal{N}_1}^{(2,R2,C,O)}(d_c)$ and $p_{\mathcal{N}_1}^{(2,R2,C,F)}(d_c)$ are defined similarly. The calculation of the probability $p_{\mathcal{N}_{1,i}}^{(2,R2,+,F)}(d_c)$, $0 \leq i \leq d_c - 1$, follows:

$$\begin{aligned} p_{\mathcal{N}_{1,i}}^{(2,R2,+,F)}(d_c) &:= \Pr[c \in \mathcal{N}_{1,i}^{(2,R2,+,F)}(d_c)] \\ &= \Pr[c \in \mathcal{N}_{1,i}^{(2,R1,1,+)}(d_c)] \Pr[c \in \mathcal{N}_{1,i}^{(2,R2,+,F)}(d_c) | c \in \mathcal{N}_{1,i}^{(2,R1,1,+)}(d_c)]. \end{aligned}$$

The second probability on the right hand side, referred to as P_{d_c} in the following derivation for simplicity, is independent of the check degree d_c and is calculated as follows:

$$\begin{aligned} P_{d_c} &:= \Pr[c \in \mathcal{N}_{1,i}^{(2,R2,+,F)}(d_c) | c \in \mathcal{N}_{1,i}^{(2,R1,1,+)}(d_c)] \\ &= \sum_{d=1}^{d_{v,\max}} \Pr[e \in \mathcal{V}_d^e, v_e \in \mathcal{K}_{0\uparrow 1}^{(2,R2)}(d) | c_e \in \mathcal{N}_{1,i}^{(2,R1,1,+)}(d_c), v_e \in \mathcal{K}^{(2)}] \\ &= \sum_{d=1}^{d_{v,\max}} \frac{\Pr[e \in \mathcal{V}_d^e] \Pr[v_e \in \mathcal{K}^{(2)} | e \in \mathcal{V}_d^e] \Pr[v_e \in \mathcal{K}_{0\uparrow 1}^{(2,R2)}(d) | v_e \in \mathcal{K}_d^{(2)}]}{A + B} \\ &\times \Pr[c_e \in \mathcal{N}_{1,i}^{(2,+)}(d_c) | v_e \in \mathcal{K}_{0\uparrow 1}^{(2,R2)}(d)], \end{aligned}$$

where,

$$\begin{aligned} A &= \sum_{d=1}^{d_{v,\max}} \sum_{i=1}^{d} \Pr[e \in \mathcal{V}_d^e, v_e \in \mathcal{K}^{(2)}, v_e \in \mathcal{K}_{0\uparrow i}^{(2,R2)}(d), c_e \in \mathcal{N}_{1,i}^{(2,R1,1,+)}(d_c)], \\ B &= \sum_{d=1}^{d_{v,\max}} \sum_{i=1}^{d} \Pr[e \in \mathcal{V}_d^e, v_e \in \mathcal{K}^{(2)}, v_e \in \mathcal{K}_{1\uparrow i}^{(2,R2)}(d), c_e \in \mathcal{N}_{1,i}^{(2,R1,1,+)}(d_c)]. \end{aligned}$$

Hence,

$$p_{\mathcal{N}_{1,i}}^{(2,R2,+,F)}(d_c) = \frac{p_{\mathcal{N}_{1,i}}^{(2,R1,1,+)}(d_c) \sum_{d=1}^{d_{v,\max}} \frac{d\lambda_d}{\bar{d}_v} \alpha_d^{(2)} p_{\mathcal{K}_0}^{(1,R1,2)}(d) p_{\mathcal{K}_{0\uparrow 1}}^{(2,R1)}(d) \frac{1}{d}}{\sum_{d=1}^{d_{v,\max}} \sum_{i=1}^{d} \frac{d\lambda_d}{\bar{d}_v} \alpha_d^{(2)} \left\{ p_{\mathcal{K}_0}^{(1,R1,2)}(d) p_{\mathcal{K}_{0\uparrow i}}^{(2,R1)}(d) \frac{i}{d} + p_{\mathcal{K}_1}^{(1,R1,2)}(d) p_{\mathcal{K}_{1\uparrow i}}^{(2,R1)}(d) \frac{i-1}{d} \right\}}$$

$$= \frac{p_{\mathcal{N}_{1,i}}^{(2,R1,1,+)}(d_c) \sum_{d=1}^{d_{v,\max}} \lambda_d \alpha_d^{(2)} p_{\mathcal{K}_0}^{(1,R1,2)}(d) p_{\mathcal{K}_{0\uparrow 1}}^{(2,R1)}(d)}{\sum_{d=1}^{d_{v,\max}} \sum_{i=1}^{d} \lambda_d \alpha_d^{(2)} \left\{ i p_{\mathcal{K}_0}^{(1,R1,2)}(d) p_{\mathcal{K}_{0\uparrow i}}^{(2,R1)}(d) + (i-1) p_{\mathcal{K}_1}^{(1,R1,2)}(d) p_{\mathcal{K}_{1\uparrow i}}^{(2,R1)}(d) \right\}},$$

where the probability $p_{\mathcal{N}_{1,i}}^{(2,R1,1,+)}(d_c)$ can be calculated using (B.10). We also have:

$$p_{\mathcal{N}_{1,i}}^{(2,R2,+,O)} = p_{\mathcal{N}_{1,i}}^{(2,R1,1,+)} - p_{\mathcal{N}_{1,i}}^{(2,R2,+,F)}.$$

Following a similar approach, we have:

$$p_{\mathcal{N}_{1,i}}^{(2,R2,C,F)}(d_c) = p_{\mathcal{N}_{1,i}}^{(2,R1,1,C)}(d_c) \sum_{d=1}^{d_{v,\max}} \frac{\lambda_d \alpha_d^{(2)} p_{\mathcal{K}_1}^{(1,R1,2)}(d) p_{\mathcal{K}_{1\uparrow 1}}^{(2,R1)}(d)}{\sum_{d=1}^{d_{v,\max}} \sum_{i=1}^{d} \lambda_d \alpha_d^{(2)} p_{\mathcal{K}_1}^{(1,R1,2)}(d) p_{\mathcal{K}_{1\uparrow i}}^{(2,R1)}(d)},$$

$$p_{\mathcal{N}_{1,i}}^{(2,R2,C,O)}(d_c) = p_{\mathcal{N}_{1,i}}^{(2,R1,1,C)}(d_c) - p_{\mathcal{N}_{1,i}}^{(2,R2,C,F)}(d_c).$$

At the end of iteration 2, R2-HR1, check nodes in sets $\mathcal{N}_1^{(2,R2,+,O)}(d_c)$, $\mathcal{N}_1^{(2,R2,C,O)}(d_c)$, $\mathcal{N}_{1,0}^{(2,R2,+,F)}(d_c)$, and $\mathcal{N}_{1,0}^{(2,R2,C,F)}(d_c)$ are all grouped into the set $\mathcal{N}_0^{(2,R2,1)}(d_c)$. On the other hand, check nodes in sets $\mathcal{N}_{1,i}^{(2,R2,+,F)}(d_c)$, and $\mathcal{N}_{1,i}^{(2,R2,C,F)}(d_c)$, $1 \leq i \leq d_c - 1$, are all grouped into $\mathcal{N}_{1,i}^{(2,R2,1,C)}(d_c)$. Ultimately, we have:

$$p^{(2,R2,1)}_{\mathcal{N}_{k,j}}(d_c) = \sum_{i=k}^{d_c} p^{(2,R1,1)}_{\mathcal{N}_{i,j}}(d_c) p^{(2,R2)}_{\mathcal{N}_{i\downarrow k,j}}(d_c), \qquad 2 \le k \le d_c, \qquad 0 \le j \le d_c - i,$$

$$p^{(2,R2,1,+)}_{\mathcal{N}_{1,j}}(d_c) = \sum_{i=2}^{d_c} p^{(2,R1,1)}_{\mathcal{N}_{i,j}}(d_c) p^{(2,R2)}_{\mathcal{N}_{i\downarrow 1,j}}(d_c), \qquad 0 \le j \le d_c - 1,$$

$$p^{(2,R2,1,C)}_{\mathcal{N}_{1,j}}(d_c) = p^{(2,R2,C,F)}_{\mathcal{N}_{1,j}}(d_c) + p^{(2,R2,+,F)}_{\mathcal{N}_{1,j}}(d_c), \qquad 1 \le j \le d_c - 1,$$

$$\begin{aligned} p^{(2,R2,1)}_{\mathcal{N}_{0,j}}(d_c) &= p^{(2,R1,1)}_{\mathcal{N}_{0,j}}(d_c) + p^{(2,R2,C,O)}_{\mathcal{N}_{1,j}}(d_c) + p^{(2,R2,+,O)}_{\mathcal{N}_{1,j}}(d_c) \\ &\quad + \sum_{i=2}^{d_c} p^{(2,R1,1)}_{\mathcal{N}_{i,j}}(d_c) p^{(2,R2)}_{\mathcal{N}_{i\downarrow 0,j}}(d_c), \qquad 0 \le j \le d_c - 1. \end{aligned}$$

B.9 Iteration Two, R2-HR2 (Verification of variable nodes based on ZCN)

The probability $p^{(2,R2,2)}_{\Delta_i}(d), 1 \le d \le d_{v,\max}$ is calculated as follows:

$$p^{(2,R2,2)}_{\Delta_i}(d) = \binom{d}{i} \left(p^{(2,R2)}_{\delta}\right)^i \left(1 - p^{(2,R2)}_{\delta}\right)^{d-i}, \qquad 0 \le i \le d,$$

where $p^{(2,R2)}_{\delta}$ is given by (B.8) with the appropriate change of probabilities to reflect the values corresponding to iteration 2. Lastly, the probability that a variable node of degree d is zero-valued and remains unverified for iteration 3 is given by:

$$p^{(3)}_{\Delta}(d) = p^{(2)}_{\Delta}(d) p^{(2,R2,2)}_{\Delta_0}(d).$$

B.10 Iterations Three and Beyond

The analysis of an iteration ℓ, $\ell \ge 3$, is similar to that of iteration 2. The update rules for a generic iteration ℓ, $\ell \ge 2$ are given in Tables 5.3 and 5.4.