Finite Length Analysis of Verification-Based Message Passing Algorithms in Compressed Sensing

by

Seyed Mohammad Ebrahim Farhangdoust, **B.Sc.**

A thesis submitted to the
Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Master of Applied Science in Electrical Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario
May, 2015

The undersigned hereby recommends to the
Faculty of Graduate and Postdoctoral Affairs
acceptance of the thesis

Finite Length Analysis of Verification-Based Message Passing Algorithms in Compressed Sensing

submitted by **Seyed Mohammad Ebrahim Farhangdoust**, **B.Sc.**

in partial fulfillment of the requirements for the degree of

Master of Applied Science in Electrical Engineering

Professor Amir H. Banihashemi, Thesis Supervisor

Professor Roshdy Hafez, Chair,
Department of Systems and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems and Computer Engineering
Carleton University
May, 2015

Abstract

Compressed sensing (CS) is a newly developed area in signal processing which generally aims at compressing sparse signals. The most challenging part of a CS problem can be the recovery process in which a signal will be reconstructed from its measurements. In this thesis, we address the finite length analysis of verification-based message-passing algorithms (VB MPAs), analytically. In contrary to asymptotic regime analysis, the finite length analysis is a fairly untouched problem. We begin the analysis by characterizing the problematic combinatorial structures, defined over the measurement graph of the CS system in which VB algorithms are trapped. Then, based on this characterization, we derive exact expressions for the average probability of error of a given ensemble of graphs, and also for a given individual graph. The analysis for the individual graph is also shown to be useful for the introduction and estimation of error floors in VB MPAs in CS.

Acknowledgments

This thesis becomes a reality with the kind support and help of many individuals. I would like to extend my sincere thanks to all of them. Foremost, I would like to express my gratitude towards my family for the encouragement which helped me in completion of this thesis, my beloved and supportive wife, Fatemeh who has always been by my side when I needed her most and helped me a lot in making this study, my dear parents and my brother.

I would also like to express my special gratitude to my adviser, Prof. Banihashemi for imparting his knowledge and expertise in this study. Moreover, I would like to thank my examiners, Dr. Dansereau, Dr. Maheshwari and Dr. El-Tanany, for their thoughtful comments and suggestions.

The last but not the least, my special thanks and appreciation also go to my colleagues and friends who have willingly helped me out with their great cooperation and support.

Table of Contents

List of Tables

List of Figures

Nomenclature

Notation	Description
A	sensing matrix
G	sensing graph
V_l	set of variable nodes
V_r	set of measurement nodes
E	set of edges of sensing graph
δ	density factor
x	input signal
y	measured signal
n	input signal dimension
ℓ	iteration number
n'	measured signal dimension
v	usually referred to as a variable node
m	usually referred to as a measurement node
μ	value of a node
$\mathcal{N}(V)$	Neighbourood of a subset of nodes V
$d^{(\ell)}(c)$	degree of measurement node c at iteration ℓ
$\mathcal{K}$	set of non-zero variable nodes

$\mathcal{C}(t)$	set of measurement nodes with the same value as $t \in \mathbb{R}$
$P(G, \delta)$	Probability of error of a given graph G given the density factor δ
$P(G\|\mathcal{K})$	Probability of error of a given graph given the number of non-zero elements of the input signal $\mathcal{K}$
$\mathcal{G} \restriction_A$	restriction of sensing matrix on a subset A of variable nodes
$\mathcal{E}_G$	set of unverified variable nodes in G
$\mathcal{E}_G^{\mathcal{K}}$	set of unverified non-zero variable nodes in a set of non-zero variable nodes $\mathcal{K}$ in a graph G
$\mathcal{E}_G^{\Delta}$	set of unverified zero variable nodes in a set of zero variable nodes Δ in a graph G
$J(S)$	Multiplicity of the smallest children of S
$I(S)$	Size of the smallest children of S
$\mathcal{P}$	A problematic set or its associated algorithm
$\mathcal{C}_{\mathcal{P}}(S)$	Collection of $\mathcal{P}$-children of S
$\mathcal{P}(G)$	Collection of all $\mathcal{P}$ problematic set of graph G
$P^{\mathcal{P}}\{S\ fails\}$	Probability that a set of variable nodes S remains unverified in algorithm $\mathcal{P}$
$P_{ef}^{\mathcal{P}}(G)$	Probability of failure of graph G in error floor region in algorithm $\mathcal{P}$
$P_s^{\mathcal{P}}\{V_l(G)\}$	Probability of success of graph G in algorithm $\mathcal{P}$ (success ratio)

Abbreviation	Description
AMP	Approximate Message Passing
BEC	Binary Erasure Channel
BP	Basis Pursuit
CS	Compressed Sensing
D1MN	Degree 1 Measurement Node
DE	Density Evolution
EEMN	Extended Equal Measurement Nodes
EIM	Equal Incoming Messages
EMN	Equal Measurement Nodes
GAMP	Generalized Approximate Message Passing
IPA	Interval Passing Algorithm
LDPC	Low-Density Parity Check
MPA	Message Passing Algorithm
nRSSS	non-binary Restricted Stopping Super Set
OMP	Orthogonal Matching Pursuit
RIP	Restricted Isometry Property
ROMP	Regularized Orthogonal Matching Pursuit
RSSS	Restricted Stopping Super Set
SE	State Evolution
SS	Stopping Set
SSS	Stopping Super Set
StOMP	Stage-wise Orthogonal Matching Pursuit
VB	Verification Based
ZMN	Zero Measurement Node

Chapter 1

Introduction

1.1 Compressed Sensing and Motivations

"Why go to so much effort to acquire **all** the data when **most** of what we get will be thrown away?", David L. Donoho, 2006 [1]. This question was the first motivation for the birth of one of the greatest and most revolutionary topics in signal processing and communication theory named compressed sensing (CS). Although the question seems trivial, the real practical approach to reach to the main goal of the question is considerably difficult. CS deals with the response to this question and tries to answer that in an effective manner where both the theory and applications can come together.

Sensing and measurement of a signal has been previously considered under the name of sampling theory in signal processing. The well known Nyquist sampling rate is a must for a signal to be reconstructed from its compressed samples [2]. However, in CS, we are interested in rates far beyond the reach of Nyquist rate [3]. In fact, it is shown that one can reconstruct a signal sometimes even exactly from a number of measurement samples much lower than Nyquist rate, provided that the underlying signal is sparse. In fact, most of the signals in reality do have this property, i.e., there is at least one domain for them in which the representation of this signal is sparse.

Application of CS includes a wide range of engineering problems that mainly deal with large signals. The most known application of CS is in the imaging industry. The well-known single-pixel imaging is one of the hundreds of applications of CS in this industry [4]. The basic idea for the applications in imaging is to reduce the number of sampled image components in order to reduce the cost of the processing and storage which will result in a considerable improvement in both of these aspects.

Moreover, it is also shown that sometime CS will perform even more accurate than conventional pixel sampling imaging [5]. In video decoding, there are also lots of applications that are even growing today. The employment of CS in compression of video frames in such a way that the compressed frames can be reconstructed fast and accurately is the most important contribution of CS in this area [6–8].
Furthermore, medical imaging can be regarded as another important application of CS which has been recently developed due to the recent high volume of demands for cost lowering approaches in this industry. The intrinsic sparsity of magnetic resonance (MR) images makes them the most suitable candidates to be used in CS. In other words, for most of the signals, we need a transformation such as wavelet transform to make them sparse; however, in case of MR images this property is given without any transformation. As a result, there are lots of applications of CS on this particular topic among them we can name rapid and dynamic MRI [9–11].
The last but not the least, we can mention researchers in communication theory as one of the most frequent users of CS. There are applications in this area, including, but not limited to, channel estimation [12–14], spectrum sensing [15–17], interference cancellation [18], ultra wide band communications [19], etc.

1.2 Background and Preliminaries

Let $\mathbb{R}$ denote the set of real numbers. Also, let $\mathbb{R}^n$ and $\mathbb{R}^{n'\times n}$ denote the set of n-dimensional column vectors and $n'\times n$ dimensional matrices on $\mathbb{R}$, respectively. Based on these simple notations, we can define the CS problems as an under-determined linear system of equations, where the number of equations are less than the number of unknowns. Let x be an n dimensional signal that we want to compress. Then this compression can be modeled as:

$$y = Ax, \tag{1.1}$$

where $A \in \mathbb{R}^{n'\times n}$ is the sensing matrix and $y \in \mathbb{R}^{n'}$ is the measured or compressed signal, where $n' << n$. Basically, there are more than one solution for this problem. However, it is shown that if x is a sparse signal, then there are possibilities to recover x accurately from y. In fact, the main challenging part of the CS problem will be the estimation of x from y, given A, which is called the recovery process.
A simple mathematics shows that (1.1) is not solvable without any further knowledge

about x. That is, there should be a constraint on the final solution which can distinguish x from other solutions of (1.1). Sparsity of x is, indeed, the key point and the constraint that we are looking for. Signal x is said to be $k-$sparse if it only has k non-zero elements. For small values of k the problem is shown to be efficiently solved. This constraint, however, can be employed for solving (1.1) only if the matrix A has been chosen carefully. In other words, sparsity of x is a necessary condition but not sufficient for the reconstruction. Moreover, the feasibility of solutions of the problem depends also on how sparse x is.[1]

In the subsequent subsections, we will consider the effects of the sensing matrix along with a review of the existing recovery algorithms in CS. As a matter of fact, the choice of the sensing matrix and the recovery algorithm are two inseparable tasks. Nevertheless, assuming a proper choice for the sensing matrix, it is shown that x will be, in fact, the sparsest possible solution of (1.1), i.e. the solution with the smallest number of non-zero elements. On the other hand, it is well-known that the number of non-zero elements of a vector is representable by its l_0 norm. Therefore, the original recovery problem in CS can be written as:

$$\begin{aligned} &\min_{x} \|x\|_0, \\ &s.t. \quad Ax = y. \end{aligned} \tag{1.2}$$

1.2.1 Recovery Problem and Algorithms

The l_0 norm problem is known to be NP-hard [20]. As a result, the solution to this problem cannot be achieved efficiently by solving (1.2), directly. The recovery algorithms in CS, in fact, try to alleviate the difficulty of this problem either by approximating the solution or relaxation of the problem.

Involved in a CS problem are two matrices called the measurement matrix and the sensing matrix. The former is referred to the matrix that only does the transformation from the original n dimensional space to the n' dimensional space, and the latter is, in fact, the multiplication of the measurement matrix and the domain transformation matrix. Note that in some cases, the signal x itself is not sparse, and one needs to change its domain using an $n \times n$ transformation matrix to work with x in a domain

[1] This is, indeed, the main concern of the thesis, as we will eventually derive expressions for the probability of a signal being recovered for a given measure of the sparsity of a signal.

in which x is sparse. Throughout the thesis, we will use the sensing matrix as a more general term, and our assumption is that this matrix is constructed in a way that it satisfies the requirement of the algorithms that will be used in this work.
Based on the properties of the sensing matrix, we will have different algorithmic approaches to deal with the recovery problem in CS, namely, geometric and combinatorial approaches [21]. The former approaches focus on dense sensing matrices and try to solve the problem, generally, using linear or convex optimization techniques. Some examples of these algorithms can be found in [1, 22–25]. The latter approaches, on the other hand, focus on sparse sensing matrices and try to solve the recovery problem in CS, using iterative methods. These methods, despite their weak ability to recover signals with high density ratio, in comparison to geometric methods, are computationally fast enough to be considered as appropriate techniques for fast signal recovery in CS. There has been lots of papers investigating these methods such as [23, 26–31].

Geometric Approaches

In geometric approaches, those features of the sensing matrix which are related to its geometric properties are considered. In fact, the main concepts of these algorithms lie in the theory of dimensionality reduction, in mathematics. Given a set of points in an n dimensional space, these techniques try to represent them in a space with $O(\log n)$ dimensions, such that the distances between the points are preserved in this transformation [32]. This preservation in CS is known as restricted isometry property (RIP) condition, which ensures that the transformation, using the sensing matrix A, preserves the Euclidean distances between the points [33]. Interestingly, it is shown that a dense matrix whose elements are chosen randomly from a continuous distribution have RIP, with high probability, and as a result, one should not be worried about the construction of such matrices [34]. This condition can be used to find the solution to (1.2), more efficiently.
It is shown in [35] and [36] that if the vectors of the sensing matrix are drawn from a Gaussian distribution, then one can find the solution of (1.2) by solving:

$$\begin{aligned} &\min_{x} \|x\|_1, \\ &S.T. \quad Ax = y, \end{aligned} \tag{1.3}$$

where $\|x\|_1$ denotes the norm-1 of the vector x. Based on this relaxation, many algorithms have been developed to solve the CS problem using linear programming based approaches. Basis pursuit (BP) seems to be among the earliest approaches for solving such a problem. In fact, BP tries to decompose a signal into a super position of dictionary elements with the smallest l_1 norm [37]. Despite its lower complexity than usual l_1 norm minimization and its branches, BP still suffers from high complexity $O(n \log n)$ and the ability to recover a signal only with minimum number of non-zero elements [37]. The other group of approaches are based on matching pursuit (MP) and its embranchments. The well-known orthogonal MP (OMP) algorithm which is based on iterative greedy pursuit can recover a signal in $O(k \log n)$ [22]. Generally, in this algorithm, we try to iteratively select vectors of the sensing matrix that contain most of the energy of the measured signal. Following this idea, there are also other algorithms such as stage-wise OMP (StOMP) [24] and regularized OMP (ROMP) [38]. The difference between StOMP and OMP is that in StOMP in each stage, many coefficients can enter the procedure instead of only one which might reduce the number of iterations, in general. Moreover, ROMP is also said to benefit from the accuracy of convex optimization and the speed of greedy algorithm simultaneously, and also can be regarded in the combinatorial group. Unfortunately, most of these algorithms make an assumption that they know the number of non-zero elements of the signal x, which is not always true in practical sense, and if we put this drawback along with their high complexity, it makes them unsuitable for real time applications of CS.

Combinatorial Approaches

The other group of algorithms in CS, usually, use sparse sensing matrices, and try to solve the problem in an iterative manner. In this group, mostly, the sensing matrix is a binary matrix whose elements are chosen from $\{0, 1\}$. In [39], it is shown that these matrices are poor with respect to RIP condition. As a result, it is also shown that they require more number of rows (or equivalently more number of measurement in CS), for signal reconstruction. Moreover, it is also shown in [21] that these matrices satisfy a generalized RIP which considers Manhattan l_1 norm rather than l_2 norm. On the other hand, this class of approaches can benefit from fast reconstruction which makes them attractive for real time applications.

Most of the algorithms in this category employ the concept of group testing for sparse

signal recovery. In group testing, in order to find the non-zero elements of the signal, we have to test some groups of the signal elements together. This testing can be efficiently achieved by employing sparse sensing matrices. And also, the result of each group test is stored as the elements of measured vector y. Then if the result of a measured vector is zero then we can simply identify all of the signal elements participated in that test as zero elements of the signal. Note that for signals contaminated with noise one should be more careful for using these approaches. There are lots of algorithms for this category that use the same idea of group testing in different manners. Count-min and count-median algorithms in [40] work under the assumption that the elements of signal x are all positive, and utilize a specific sparse matrix construction for their algorithm. These algorithms are mainly considered as data streaming algorithms.

CS problems with sparse sensing matrices are the best candidates for being used by message passing algorithms (MPAs) in which x can be recovered based on the messages and information being passed between the measurement vector y and the main signal x. It is also possible to consider such schemes over dense matrices with only a few number of zero elements. Message passing (MP) algorithm and approximate MP (AMP), introduced in [41] and continued in [42], are examples of these algorithms. These algorithms, basically, use the loopy belief propagation method to recover the signal x in an efficient time complexity. Also, authors in [42] generalized the idea of AMP under the name of GAMP which, in fact, provides an efficient approximate implementation of max-sum loopy belief propagation in AMP.

The message passing algorithms over dense graphs are beneficial in terms of the fact that they are able to recover signals with less number of measurements. However, in terms of the time complexity the sparse matrices are still the most attractive ones.

In the category of algorithms using sparse sensing matrices, a belief propagation based algorithm is introduced in [43]. This algorithm assumes that the locations of non-zero elements are known. Some other algorithms in this class assume a priori distribution for non-zero elements of x and try to maximize a posteriori distribution of those elements. As an example, in [44] authors assume a Gaussian a priori distribution.

Another algorithm using sparse matrices is called interval passing algorithm (IPA) which assumes that elements of x are all positive and tries to find upper and lower bounds on the values of each elements of x in each iteration and reducing the interval between these bounds to give the estimation of the value of the element [29]. The

main draw back of this algorithm is the assumption of non-negative signal elements. Verification based message passing algorithms (VB MPAs) are other algorithms employing sparse matrices along with message passing strategy. These algorithms include Genie which is the benchmark of this problem, XH, LM and SBB [27,28,45]. We will investigate these algorithms thoroughly in subsequent sections, as the analysis of these algorithms is the main topic of the thesis.

The low computational complexity of VB MPAs makes them appealing for real time applications in CS. Moreover, these algorithms recently have attracted more attention in the literature. The combination of IPA and VB MPAs resulted in an approach called VB IPA which has been considered in [46], and is shown to perform better than both VB algorithms and IPA. Also, some extensions of PahseCode algorithm, which is basically inherited from VB MPAs for solving phase retrieval problem, has been considered in [47] and [48]. Moreover, in [49] authors proposed graph construction approaches under VB MPAs to reach to a sub-linear measurement cost. Furthermore recently their applications in fast and accurate spectrum sensing is also addressed in [50]. As a result, we will focus on these algorithms, more precisely, and eventually we will fill the gap that exists in their performance analysis, analytically. However, before considering the VB MPAs, we need to set up the main problem that will be considered in this thesis using the perspective on the recovery algorithms provided in this section.

1.3 Problem Statement and Motivations

Throughout the thesis we will mainly use the graph representation of the problem rather than its matrix version as the properties of the problem can be seen easily in terms of their sensing graphs.

We consider a bipartite graph $G = (V_l \cup V_r, E)$ associated with each sensing matrix A, where V_l and V_r are called the set of variable nodes (sub-script l for left nodes) and measurement nodes (sub-script r for right nodes), respectively, and E is also the set of its edges. Corresponding to each element of the input signal, we consider a variable node $v \in V_l$, and similarly, for each element in the measured signal we consider measurement node, $m \in V_r$. There is an edge between a variable node i and a measurement node j if the corresponding element of the matrix A, i.e. $A_{j,i}$ is non-zero. Moreover, there are weights associated with each edge in the graph which is equal to

the value of its corresponding element in A. Furthermore, $\mathcal{N} : 2^{V_l} \cup 2^{V_r} \mapsto 2^{V_l} \cup 2^{V_r}$ is defined as the neighbourhood function, where 2^A denotes the power set of A, i.e., the set of all subsets of A. This function maps a subset of variable and measurement nodes to its corresponding neighbouring nodes in the graph. We, also, denote the ensemble of bi-regular bipartite graphs with n variable nodes, d_v as the degree of variable nodes and d_m as the degree of measurement nodes with $\mathcal{G}(n, d_v, d_m)$.
Moreover, the input signal x is assumed to be a sparse vector, and each element in this vector is also assumed to be non-zero with probability δ. Clearly, x is sparse only if the value of δ is small, otherwise x cannot be considered as a sparse vector. Furthermore, the non-zero values of the input vector are chosen randomly from a continuous distribution, and they are also i.i.d.

1.4 Focusing on VB MPAs

Briefly, VB MPAs use message passing strategies for iterative recovery of variable nodes. In each iteration, variable nodes and measurement nodes will send messages to their neighbouring nodes, and based on the responses they receive, they will update their messages for the next iteration. In each iteration, some variable nodes may be verified with a certain value that they receive from their neighbouring measurement nodes, according to some verification rules. The values that will be assigned to each variable nodes is guaranteed to be the same as the actual values of the variable nodes. This issue has been considered in detail in [26], where the probability of false verification in VB MPAs are proven to be zero. As the verified variable nodes do not have any impact on the rest of the algorithm, we may consider them to be removed from the graph along with their adjacent edges. [2] That is, in each iteration, the sensing graph will be pruned. If all of the variable nodes are verified and removed from the graph in a certain iteration, the algorithm is said to be successful and otherwise failed. In the case of the failure, there will remain a residual graph which cannot be pruned any more.
Basically, in Node Based VB[3] algorithms the messages going out from each node do not depend on the destination node, and in fact, the calculation on the incoming messages to a node will result in just one message which will be passed to all of the

[2]Note that the removal of variable nodes from the graph is equivalent to the fact that their value is subtracted from their neighbouring measurement nodes.

[3]We will only focus on NB-VB MPAs throughout this thesis.

neighbouring nodes of the receiver.[4] Furthermore, as it is mentioned previously, every variable node in each iteration should decide to be verified or not based on the rules defined for the specific VB algorithm. Using the following notations, we will be able to introduce these verification rules. Let $\mu^{(\ell)} : V_l \cup V_r \mapsto \mathbb{R}$ be the value of a node in the ℓ^{th} iteration. Also let $\mu^{(\ell)} : V_l \cup V_r \mapsto \mathbb{R}$ denote the final value of a node that is eventually achieved throughout the iterations in a VB MPA, i.e.

$$\forall v \in V_l \cup V_r,\ \mu(v) = \lim_{\ell \to \infty} \mu^{(\ell)}(v).$$

Also, assume that $d^{(\ell)} : V_r \mapsto \mathbb{Z}^+$ is the degree function that outputs the degree of each measurement node at a certain iteration ℓ, where $\mathbb{Z}^+$ is the set of non-negative integer numbers. Moreover, let $Z^{(\ell)} : V_l \mapsto \{0, 1\}$ denote the state function, where we say v is verified at iteration ℓ if $Z^{(\ell)}(v) = 1$ and unverified, otherwise.

The message passing schemes in all VB MPAs are the same. The basic idea is to pass messages between variable and measurement nodes, iteratively. In each iteration, we will have two main steps. In the first step, measurement nodes will pass their messages consisting of their value and their degree to their neighbouring variable nodes. Then, in the second step, variable nodes will pass their messages consisting of the information if they have been verified or not, and if yes, their corresponding value to their neighbouring measurement nodes. Note that if a variable node remains unverified, its value has no impact in the subsequent calculation in the graph; therefore, we can assume that the value of an unverified variable node is zero. Besides, in each step, each measurement node and variable node will perform some calculations, according to their received messages.

Moreover, in all VB MPAs, the messages from measurement nodes to variable nodes are all calculated by the same approach. In contrary, the messages from variable nodes to measurement nodes are dependent upon the specific algorithm. These messages are as follows:

- Messages from measurement to variable nodes:

$$\mathcal{O}^{(\ell)}_{m \to v}(m) = (\mu^{(\ell)}(m), d^{(\ell)}(m)).$$

[4] In IPA, as an example of message-based algorithm, the messages going out from a measurement node are not all identical and is dependent to the variable node to which the message will be passed. This makes the complexity of these algorithms a bit more than their node-based counter parts.

- Messages from variable to measurement nodes:

$$\mathcal{O}^{(\ell)}_{v\to m}(v) = (\mu^{(\ell)}(v), Z^{(\ell)}(v)).$$

The initial value of a measurement node $\mu^{(0)}(m)$ is equal to its corresponding element in the measured vector y. Also, based on the sensing graph, the initial degree of each measurement node is also known. Moreover, the initial state of each variable node is unverified, i.e., $Z^{(0)}(v) = 0,\ \forall v \in V_l$. Also, the initial value of a variable node is considered to be unknown.
In each iteration, each measurement node will determine its corresponding message using the following expressions:

$$\mu^{(\ell)}(m) = \mu^{(0)}(m) - \sum_{v\in\mathcal{N}(m)} \mu^{(\ell)}(v)\mathcal{W}_{m,v}Z^{(\ell)}(v), \tag{1.4}$$

and

$$d^{(\ell)}(m) = d^{(0)}(m) - \sum_{v\in\mathcal{N}(m)} Z^{(\ell)}(v), \tag{1.5}$$

where $\mathcal{W}_{m,v}$ is the weight of the edge between m and v. Note that in the case of a non-binary sensing matrix this weight is a real number.
Furthermore, in an iteration ℓ each variable node will also determine its own message by employing the following verification rules:[5]

1. Zero Measurement Nodes (ZMN): If a variable node is connected to at least one measurement node with value zero, then the variable node will be verified and its value will become zero.

$$\forall\ v \in V_l :\ if\ \exists\ m \in \mathcal{N}(v)\ \mid\ \mu^{(\ell)}(m) = 0\ \Rightarrow\ \mu^{(\ell)}(v) = 0,\ Z^{(\ell)}(v) = 1.$$

2. Degree 1 measurement Nodes (D1MN): If a variable node is connected to at least one measurement node whose degree is one, then the variable node will be verified and the value of the variable node will become the value of one of the

[5]Note that in each iteration, when a variable node become verified then we can empirically assume that this variable node is removed from the graph along with its connected edges. This issue implies that the graph will be pruned throughout the iterations.

neighbouring measurement nodes with degree 1 at random.

$$\forall\ v \in V_l :\ if\ \exists\ m \in \mathcal{N}(v)\ \mid\ d^{(\ell)}(m) = 1\ \Rightarrow\ \mu^{(\ell)}(v) = \frac{\mu^{(\ell)}(m)}{\mathcal{W}_{m,v}},\ Z^{(\ell)}(v) = 1.$$

3. Equal Measurement Nodes (only for binary sensing matrices) for binary sensing matrices: If there is a single variable node connected to at least two measurement nodes, with the same non-zero value, then the variable node will be verified with the common value of those measurement nodes.

$$\mathcal{C}^{(\ell)}(t) \coloneqq \{m \in V_r\ \mid\ \mu^{(\ell)}(m) = t\},$$

$$\begin{aligned}\forall\ v \in V_l :\ \exists\ t \in \mathbb{R} \mid \mathcal{C}^{(\ell)}(t) \subseteq \mathcal{N}(v),\ \left|\cap_{m\in\mathcal{C}^{(\ell)}(t)}\mathcal{N}(m)\right| = 1,\\ \Rightarrow\ \mu^{(\ell)}(v) = t,\ Z^{(\ell)}(v) = 1.\end{aligned}$$

4. Equal Incoming Messages (non-binary version of EMN) for non-binary sensing matrices: If a variable node receives two identical non-zero messages along at least two incoming edges, then the variable node will be verified with the common value of those edges.

$$\begin{aligned}&\forall v \in V_l :\ if\ \frac{\mu^{(\ell)}(m_1)}{\mathcal{W}_{m_1,v}} = \frac{\mu^{(\ell)}(m_2)}{\mathcal{W}m_2, v} \neq 0\ for\ m_1, m_2 \in \mathcal{N}(v),\\ &\mu^{(\ell)}(v) = \frac{\mu^{(\ell)}(m_1)}{\mathcal{W}m_1, v},\ Z^{(\ell)}(v) = 1.\end{aligned}$$

5. Extended Equal Measurement Nodes (EEMN): If there are two or more measurement nodes with the same non-zero value, any variable node, which is partially connected to these measurement nodes, will be verified with zero value.

$$\forall\, v \in V_l :\ if\ \exists t \in \mathbb{R} \setminus \{0\}\ \mid\ \mathcal{C}^{(\ell)}(t) \cap \mathcal{N}(v) \neq \emptyset,$$
$$\mathcal{C}^{(\ell)}(t) \nsubseteq \mathcal{N}(v),\ \Rightarrow\ \mu^{(\ell)}(v) = 0,\ Z^{(\ell)}(v) = 1.$$

Note that, the order of implementation of each verification rule, in an iteration, does not have any impact on the final result. Moreover, when a variable node becomes verified, its message remains unchanged for the rest of the algorithm [6]. Furthermore, before being verified, each variable node only sends a message indicating that it is not verified.

In each of the above rules, the value of the variable node will be estimated according to the situation, and then, the variable node will be marked as verified. Once a variable node becomes verified then we can simply remove it from the graph, along with its corresponding edges. This removal will result in another graph which could possibly be more simplified, in the subsequent iterations. At the end, the algorithm will stop until there is no change in the number of verified variable nodes, from one iteration to another. When the algorithm is terminated, if there is no variable nodes left in the graph, the algorithm is said to be successful, and it returns the estimated values of variable nodes $\hat{x}$ as $\mu(v)\ \forall v \in V_r$, otherwise, it is said to be failed. Algorithm 1 shows a brief review of VB MPAs.

Basically, each of the rules given above has a specific meaning in the original problem which should be pointed out. That is, the equivalent statements, in algebraic point of view of the problem, are also worth mentioning, and contain important information. For instance, ZMN states that if in a system of linear equations, which acting on a sparse vector, the right hand side of an equation is zero, it means that every variable, contributing in that equation, is zero with high probability which is a consequence of the fact that the non-zero elements of the signal has been chosen randomly from a continuous distribution. The proof of this statement is given in [26]. Besides, binary EMN states that if there are two or more equations, with the same right hand side,

[6] This is why we can assume that the variable node is removed from the graph.

Algorithm 1 NB-VB MPAs

procedure NB-VB MPAs IN GENERAL
  Inputs: G, y
  initialization
  for $m \in V_r$ **do**
    set $\mu^{(0)}(m)$ and $d^{(0)}(m)$
  end for
  for $v \in V_l$ **do**
    $\mu^{(0)}(v) \leftarrow 0,\ Z^{(0)}(v) \leftarrow 0$
  end for
  $\ell \leftarrow 0$
  Measurement nodes pass their messages
  while there is progress in verification **do**
    $ell \leftarrow \ell + 1$
    for $v \in V_l$ **do**
      set $\mathcal{O}^{(\ell)}_{v\to m}(v)$ based on verification rules
    end for
    variable nodes pass their messages
    for $m \in V_r$ **do**
      set $\mathcal{O}^{(\ell)}_{m\to v}(c)$ based on (1.4) and (1.5)
    end for
    measurement nodes pass their messages
  end while
  if every variable node is verified **then**
    Return Success
    Return $\hat{x}$ according to values of variable nodes
  else
    Return Failure
  end if
end procedure

and there is a single variable, common in all of these equations, then the value of this common variable equals the value of the equations. Furthermore, the extended version of EMN states that all the other variables in these equations are zero with high probability. Moreover, D1MN considers equations with just one variable. Clearly, in an equation with just one variable, the value of the variable node is the same as the value of the right hand side of the equation assuming that its coefficient is one.

Remark 1. The EIM and EMN rules are, basically, the same; however, the former can be implemented in problems with sensing matrices whose non-zero elements are chosen from a continuous distribution, and the latter can only be implemented in problems with sensing matrices consisting of zero and one elements. This will make a huge difference between the algorithms using EMN and EIM. In binary case, the algorithm should find "single" variable nodes, connected to the equal measurement nodes, to verify them as non-zero valued variable nodes (EMN), and also, the algorithm should look for variable nodes, which are partially connected to a subset of those equal measurement nodes, to verify them as zero valued variable nodes (EEMN). Obviously, in this case, a variable node cannot decide about verification, locally. Nevertheless, in the non-binary case, the most important issue is that the probability of having two measurement nodes with the same value is zero, because the weights of the edges of graphs are chosen randomly from a continuous distribution. In a variable node, a received message, in the non-binary case, should be divided by the weight of the edge through which the message has been passed. As a result, no zero variable node is capable of receiving two equal incoming messages (divided by weights) which makes EEMN rule inapplicable in the non-binary case. However, a non-zero variable node can receive two equal messages (divided by weights). In this case, it is guaranteed that this variable node is the "only" one who is completely connected to those measurement nodes; therefore, the variable node will be verified.

There are specific algorithms that we will investigate throughout this thesis, namely Genie, LM, binary SBB and non-binary SBB algorithms.[7] The difference between these algorithms is that they use different set of the rules for verification of variable nodes.

In Genie algorithm the assumption is that the recovery algorithm knows the support set of x. That is, the set of non-zero elements of the signal x is known for the algorithm. Therefore, the only task of the algorithm is to verify and find the value of

[7]The name of these algorithms are adapted from [26].

non-zero elements of the original vector x. Besides, for estimating those values, Genie only requires to exploit the D1MN rule. One might argue that Genie is not practical, due to its support set assumption. In fact, performance analysis of Genie is important not only because it is the upper bound on the performance of other algorithms in this category, but also it has a very interesting relation to LDCP decoding over Binary Erasure Channel (BEC). It will be shown that the performance of the Genie algorithm is exactly the same as the performance of the LDPC iterative message passing decoding algorithm over BEC, which will enable us to bridge the analysis of LDPC decoders in coding theory and recovery algorithms in compressed sensing.
LM algorithm however does not have any information about the support set of the signal, and it uses both D1MN and ZMN to verify variable nodes. According to the algebraic statements of these rules, we know that ZMN is responsible for the recovery of zero valued variable nodes, while D1MN recovers non-zero valued variable nodes. Furthermore, SBB algorithm also uses the same rules, along with EMN and EEMN to verify variable nodes. Clearly, SBB will be able to recover more signals than LM, and this better performance is achieved by only a small amount of extra computation. Note that in non-binary SBB algorithm, the algorithm will use the EIM rule, and as discussed in Remark 1, EEMN rule cannot be applied. Table 1.1 shows which VB MPA uses which verification rule in summary.

Table 1.1: Verification rules used by different VB MPAs

Algorithm	ZMN	D1MN	EMN	EEMN	EIM
Genie		✓			
LM	✓	✓			
SBB	✓	✓	✓	✓	
non-binary SBB	✓	✓			✓

The following example for SBB algorithm is given to clearly demonstrate the message passing verification rules and their applications within the iterations.

Example 1.4.1. Consider the following 6×8 sensing matrix:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix},$$

and assume that the sparse vector x is:

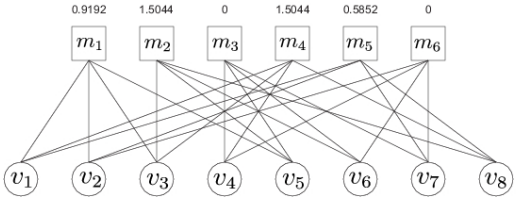
$$x = \begin{bmatrix} 0 & 0 & 0.9192 & 0 & 0 & 0 & 0 & 0.5852 \end{bmatrix}^T.$$

Thus, the measurement vector will be:

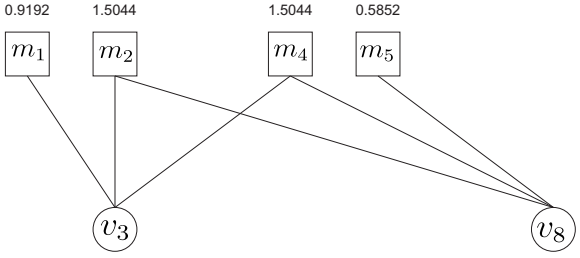
$$y = Ax = \begin{bmatrix} 0.9192 & 1.5044 & 0 & 1.5044 & 0.5852 & 0 \end{bmatrix}^T.$$

The graphical representation of matrix A is depicted in Figure 1.1a. The first step will begin by considering measurement nodes whose values are zero. According to ZMN, variable nodes in $\{v_2, v_4, v_5, v_6, v_7\}$ are all zero. Besides, there is another rule which is also applicable in the first iteration. According to EEMN since the values of m_2 and m_4 are the same, we can conclude that every variable node, which is partially connected to these measurement nodes, is zero. Thus, $\{v_1, v_2, v_4, v_5, v_6, v_7\}$ should be eliminated from the graph, in the next iteration. The new graph has been shown in Figure 1.1b. Note that although there are two measurement nodes with the same non-zero value, EMN rule cannot be applied in this situation because there is no single variable node connected to these measurement nodes; in fact, there are two variable nodes $\{v_3, v_8\}$ connected to them.

In the second iteration, since there is no zero measurement node, we have to check if there exists a measurement node with degree 1. According to D1MN rule, the



(a) The graph at the beginning of the first itereation in the SBB algorithm



(b) The graph at the beginning of the second itereation in the SBB algorithm

Figure 1.1: Iterations in the SBB algorithm

remaining variable nodes will be verified with the following values:

$$\hat{x}_3 = \mu(v_3) = \mu(c_1) = 0.9192,$$
$$\hat{x}_5 = \mu(v_5) = \mu(c_5) = 0.5852.$$

Clearly, since there is no more variable node left in the graph, the algorithm will be terminated at this point, verifying that $\hat{x}_i = \mu(v_i)\ \forall i \in \{1, 2, \ldots, 8\}$.

1.5 Analysis of Algorithms

The main goal of this thesis is to investigate the analysis of VB MPAs. The analysis, that will be provided in the thesis, is of great importance in two senses. Firstly, this analysis is, to the best of our knowledge, among the first analytic works on finite length analysis in CS, as there are only a few works covering this topic. Secondly, the analysis will finally result in the discovery of error floor phenomenon in VB MPAs, which, to the best of our knowledge, has not been observed in this category of algorithms in CS yet. Nevertheless, we will briefly review the previous works in the analysis of related algorithms in this topic.

Firstly, authors in [30] showed that the algorithm proposed in [27] is similar to verification decoding algorithm for low-density parity check (LDPC) codes proposed in [51]. This allowed the authors in [30] to use the density evolution (DE) technique to analyze the message based version of the algorithm, in the asymptotic regime. Also, authors in [52] consider the analysis of the node based version of the algorithm, in terms of LDPC codes over q-ary symmetric channel. Furthermore, authors in [26] alleviate the complexity of the DE analysis of [52] by deriving simple closed-form update equations, for the analysis in CS version of the algorithm.

Similar to DE, state evolution (SE) also plays an important role in the asymptotic analysis of MPAs, over dense graphs. SE was first, empirically, introduced in [41], and later authors in [53] proved that SE is true for Gaussian sensing matrices, and can be used for a wide range of message passing algorithms. Moreover, in [42], SE has also been proven to be useful in the asymptotic analysis of GAMP.

The above analysis can only be useful for problems in the asymptotic regime. For, finite length which is, clearly, more practical, we start with similar and useful analysis

in coding theory, and we will then show that finite length analysis in CS is nearly an untouched problem.

Thanks to the connection of CS and LDPC coding that has been considered in [54], we are, to some extent, allowed to consider some of the analysis provided in coding theory for LDPC codes in CS. One of the most well-known result in this topic, certainly, belongs to [55], where the authors discover the well-known stopping sets as the only sources of failure in LDPC decoding over the binary erasure channel (BEC) using belief propagation decoder.[8] The authors then proposed an analytic combinatorial technique for average performance analysis of graphs in a given bi-regular bipartite ensemble. To reduce the complexity of the analysis, authors also provided a simplified version of the analysis which only covers graphs with degree of variable nodes equal to 2 and 3. This simplified version has been improved to cover all variable node degrees later in [56]. In addition, to further reduce the complexity of the analysis authors in [57], empirically, showed that the performance of LDPC codes over BEC, in the waterfall region, follows a scaling law, which makes the finite length analysis in this category more practical. Furthermore, authors in [58] investigated the distribution of stopping sets in different ensembles of graphs, and showed that for almost all codes with variable degree > 2, the size of the smallest stopping sets scales linearly with the block length. The smallest stopping sets can also be useful for the error floor analysis. As a result, using the scaling law in [57] and the stopping set analysis one can perform the finite length analysis for LDPC codes over BEC with an efficient complexity [57].

Later in the subsequent chapters, we will show that review of the previous works in finite length analysis in coding literature is, in fact, much helpful in CS.

Moreover, for the case of IPA, authors in [59], improved the idea of IPA employing the concepts in LDPC decoding, and showed that stopping sets can cause failure in IPA. Later, authors in [60] completed this analysis and showed that smallest stopping sets are not the smallest configuration in which IPA might fail. Based on that, they provided sufficient conditions for signal recovery in IPA.

To the best of our knowledge, in contrary to asymptotic analysis, the finite length has only been addressed by a few number of researchers including the one considered for IPA [60]. Clearly, the finite length analysis is of great importance, in the sense that almost all practical sensing graphs are finite length, and more importantly, main

[8] Later we will show that, interestingly, stopping sets are the sources of failure in Genie algorithm, as well.

approaches for the design purposes require a deep knowledge of the situations in which an algorithm might fail, in order to avoid them. For the case of VB MPAs, to the best of our knowledge, there is no finite length analysis presented, which results in a gap in this area.
In the following subsection, we will consider the differences between finite length and asymptotic analysis, in an analytic manner. Moreover, as a side result, we will show how one can use average ensemble performance to gain a picture of the analysis of individual graphs in certain conditions.

1.5.1 Asymptotic vs Finite Length Regime

Generally speaking, one approach to analyze an MPA is to provide a general framework that calculates the probability of success of the algorithm for a given graph. However, instead of considering every graph individually, we can find the expected probability of success of a set of graphs in a given ensemble. This issue is important in two senses. In the first place, average performance analysis over an ensemble is useful since it is usually more practical than individual graph analysis, as it is less complex. Furthermore, one can compare performances of two or more ensembles using their average performances. Nevertheless, individual graph analysis is also addressed in this thesis. In an ensemble $\mathcal{G}(n, d_v, d_m)$, it can be shown that for some appropriate choices of n, we can represent the expected probability of success of the algorithm over $\mathcal{G}(n, d_v, d_m)$ as the probability of success of the algorithm for every single realization of graphs, from this ensemble. This issue is known as the concentration result, which has been mainly investigated in [61]. The following theorems will cover this result, along with another property which is the convergence to the cycle-free graph. Let $P(G, \delta)$ denote the probability of failure of a certain MPA on G.

Theorem 1.5.1. *For any $\epsilon > 0$ there exists a $f(\epsilon) > 0$ for some function f such that:*

$$Pr\left\{\left|P(G,\delta) - \mathbf{E}_{\mathcal{G}(n,d_v,d_m)}[P(G,\delta)]\right| > \epsilon\right\} \leq e^{-f(\epsilon)n}$$

This theorem states that if the number of variable nodes in the graph is chosen appropriately large, then the probability of success in each individual graph is the same as the average of the probability of the success of the graphs over $\mathcal{G}(n, d_v, d_m)$. Since the rate of the convergence is exponential in the size of the graph, then we can

expect that for moderate number of variable nodes the performance of every single graph is likely concentrated around the average performance. This issue has been quantitatively considered in Section 5.4. Furthermore, the following theorem captures another property of the graph ensembles which is quite useful in the asymptotic analysis.

Theorem 1.5.2. *There exists $a \in \mathbb{R}$ such that*

$$\lim_{n' \to \infty} \left| \mathbf{E}_{\mathcal{G}(n,d_v,d_m)}[P(G,\delta)] - \mathbf{E}_{\mathcal{G}(n',d_v,d_m)}[P(G,\delta)] \right| \leq \frac{a}{n}$$

Since the performance of the graphs with infinite size is the same as the graph without cycle [26], the above theorem is also used for convergence to the cycle-free case. Unlike Theorem 1.5.1, this theorem is not applicable in moderate size graphs. Besides if we reduce the size of the graph, none of these theorems is applicable any more.
Fortunately, for the asymptotic analysis one can apply both Theorems 1.5.1 and 1.5.2, which makes the analysis and its formulation, although cumbersome to extract, efficiently tractable [26]. One of the major results of the asymptotic analysis is that there is a threshold, called success threshold, δ^*, which determines a threshold on density factor, δ. Then based on this threshold, if $\delta < \delta^*$, the algorithm is guaranteed to succeed, and fail otherwise. Unfortunately, the situation in the finite length regime is not the same. In fact, there is no threshold in the probability of success curve. Besides, since in the finite length regime, we cannot assume the cycle-free property in a graph, the analysis is much more complicated. The DE techniques which have been used for asymptotic case can no longer be useful here; however, one can use combinatorial tools for analysis in the finite length case. We will consider this technique in this thesis.

1.6 Contributions

Throughout this thesis, we will mainly focus on the following topics: Characterization of failure patterns in sensing graphs of VB MPAs, average performance analysis of graphs in bi-regular bipartite ensembles, and individual graph performance analysis, in those algorithms. As a matter of fact, we will show that this research will result in the following contributions that will help to fill the finite length analysis of VB MPAs in CS by:

- Fully understanding of failure patterns in the underlying sensing graphs of VB MPAs, which is done only by employing graph properties. In other words, the characterizations provided only depends on the features of the underlying graph of the problem.
- Introduction of a generalized approach for average performance analysis of VB MPAs.
- Individual sensing graph analysis which provides the exact probability of failure of a given graph in a given VB MPA, using the characterizations provided.
- Introduction of the error floor phenomenon in VB MPAs in CS, which will shed some light on future aspects of this important and non trivial issue in CS.

Each of the analysis given as part of the contribution in the thesis are also supported by numerous the simulation results, that can give the reader a better understanding and insight into the problem, and also it is shown that most of the analytic results are matched to simulation results.

1.7 Organization of Thesis

The organization of the thesis is as follows: In chapter 2, we will introduce the notions of problematic sets in VB MPAs, which will be the main tool of the thesis for the rest of the thesis. Then, in the third chapter, we will consider the average performance analysis of the VB MPAs over bi-regular bipartite ensembles of graphs; in this section, we will provide analytic expressions for the expected probability of failure of graphs in an ensemble of bi-regular bipartite graphs, along with some simplified versions of expressions for lowering the complexity of the analysis. Continuing the discussion, we will address the individual graph analysis, in chapter 4. This analysis, down the road, will result in the introduction of error floor analysis in VB MPAs in CS. In addition, we provide lots of numerical results and discussions on the analysis of algorithms in order to verify the techniques provided in the thesis. Finally, we will conclude the thesis in the last chapter.

Chapter 2

Failure Characterization of MPAs

The basic step towards the finite length analysis is the identification of failure patterns in the sensing graphs in CS. This type of analysis is a combinatorial approach as it recognizes some combinatorial objects as the sources of failure. Unlike DE, the combinatorial technique is not able to pursue the fraction of verified variable nodes in each iteration; however, it gives us the information that only considers the set of variable nodes that are eventually remained unverified.[1] That is, given a graph with determined set of non-zero elements, one can predict the set of nodes which will remain unverified without running the algorithms on the graph and tracking the iterations. The knowledge of these problematic sets is not only useful for the sake of design but also for analysis of the algorithms. We will start from Genie algorithm, and we will show that the well-known stopping sets are the problematic structures in this algorithm. Furthermore, we will show that a generalization of the stopping sets known as Stopping Super Sets (SSS) will allow us to characterize the failure patterns in LM algorithm. Besides, an extension of SSS known as Restricted Stopping Super Set (RSSS) is also responsible for failures in SBB algorithm. More precisely, we will introduce RSSS for binary SBB algorithm performing on binary sensing matrices and also nRSSS for non-bianry version of the algorithm which is different from SBB according to Remark 1.

Let $G \restriction_A = (A \cup \mathcal{N}(A), E')$ denote the sub-graph induced by the set of variable nodes $A \subseteq V_l$ where E' is a subset of the set of edges of G that remains in $G \restriction_A$. Also, suppose that in a given graph G the set of non-zero variable nodes is denoted by $\mathcal{K}$. To be able to characterize the problematic structures in Genie, LM and SBB we have

[1] Note that a subset of variable nodes is called eventually unverified if the algorithm cannot verify the variable nodes in that subset after enough number of iterations.

to consider the following definitions:

Definition 2.0.1. In a given graph G and a set $S \subseteq \mathcal{K}$, for two measurement nodes $m_1, m_2 \in \mathcal{N}(S)$ we say that $m_1 \equiv_S m_2$ if:

$$\mathcal{N}(m_1) \cap S = \mathcal{N}(m_2) \cap S.$$

Lemma 2.0.1. *$\equiv_S$ is an equivalence relation between two measurement nodes in $\mathcal{N}(S)$.*

Proof. We have to show that this relation satisfies the following properties:

- Reflexivity: It is trivial that $m_1 \equiv_S m_1$.
- Symmetry: It is trivial that if $m_1 \equiv_S m_2$ then $m_2 \equiv_S m_1$.
- Transitivity: if $m_1 \equiv_S m_2$ and $m_2 \equiv_S m_3$ then

  $$\mathcal{N}(m_1) \cap S = \mathcal{N}(m_2) \cap S = \mathcal{N}(m_3) \cap S.$$

  That is $m_1 \equiv_S m_3$.

□

Definition 2.0.2. The equivalence class of a node m with respect to the set of variable nodes S are defined as follows:

$$[m]_S \coloneqq \{m' \in \mathcal{N}(S) \mid m' \equiv_S m\}.$$

Lemma 2.0.2. *For a given set of variable nodes S, the set of equivalence classes on its neighbouring measurement nodes, with respect to S, partitions $\mathcal{N}(S)$.*

Proof. Refer to [62] for equivalence relations and the partitioning theory. □

Definition 2.0.3. In a given set of variable nodes S in G, a measurement node $m \in \mathcal{N}(S)$ may satisfy one or more of the following properties. If m satisfies P_i, it is then said to satisfy P_i on S.

- $\mathbf{P_1}$: it has at least two connections to set S, i.e.

  $$|\mathcal{N}(m) \cap S| \geq 2.$$

- **$\mathbf{P_2}$**: There exists a variable node v in the neighbourhood of m not in S such that the neighbouring measurement nodes of v are all included in $\mathcal{N}(S)$, i.e.

$$\exists v \in \mathcal{N}(m) \setminus S \mid \mathcal{N}(v) \subseteq \mathcal{N}(S).$$

- **$\mathbf{P_3}$**: There exists a variable node v in the neighbourhood of m not in S, such that not only the neighbouring measurement nodes of v are all included in $\mathcal{N}(S)$ (property P_2 is satisfied), but also the equivalence classes of all measurement nodes in $\mathcal{N}(v)$, with respect to S, are included in $\mathcal{N}(v)$, i.e.

$$\exists v \in \mathcal{N}(m) \setminus S \mid [m']_S \subseteq \mathcal{N}(v) \subseteq \mathcal{N}(S),\ \forall m' \in \mathcal{N}(v),$$

  where $[m]_S$ is the equivalence class of m with respect to S defined in Definition 2.0.2.

- **$\mathbf{P_4}$**: P_2 is valid, and also m is in equivalence relation only with itself with respect to S, i.e.

$$\exists v \in \mathcal{N}(m) \setminus S \mid \mathcal{N}(v) \subseteq \mathcal{N}(S), \quad |[m]_S| = 1.$$

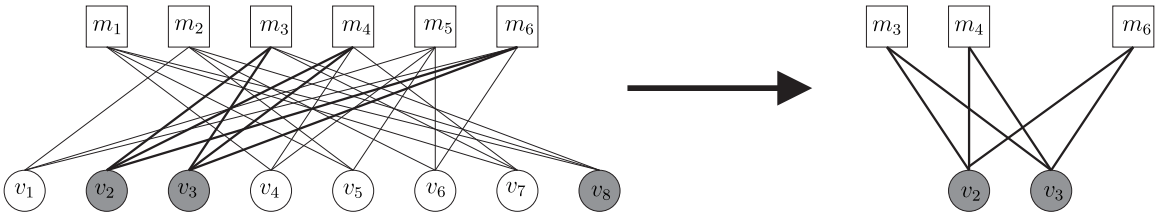
Let $\mathcal{E}_G$ denote the set of eventually unverified variable nodes in a graph G in any of the VB MPAs. Using the following definitions, we will be able to fully characterize $\mathcal{E}_G$ for Genie, LM and SBB, in the subsequent subsections.

Definition 2.0.4. [Stopping Set] A set $S \subseteq V_l$ is called a Stopping Set (SS) if every measurement node $m \in \mathcal{N}(S)$ has property P_1 on S.

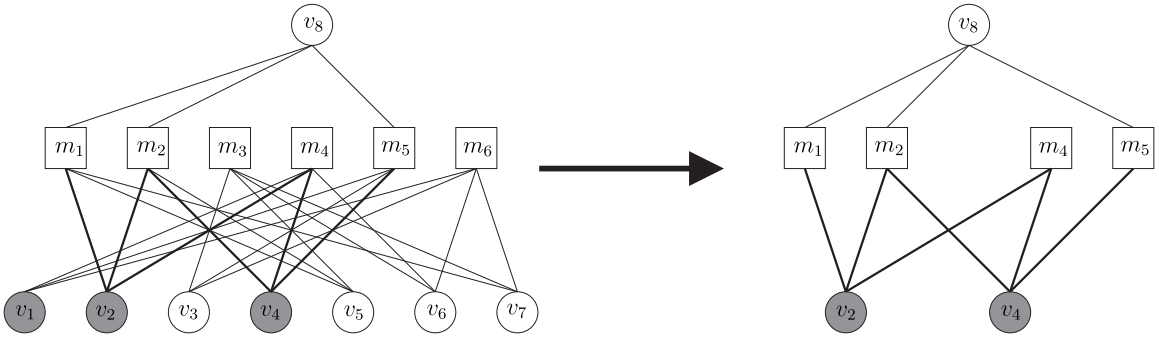
Definition 2.0.5. [Stopping Super Set] A set $S \subseteq V_l$ is called a Stopping Super Set (SSS) if every measurement node $m \in \mathcal{N}(S)$ has either property P_1 or P_2 on S.

Definition 2.0.6. [Restricted Stopping Super Set] A set $S \subseteq V_l$ is called a Restricted Stopping Super Set (RSSS) if every measurement node $m \in \mathcal{N}(S)$ has either property P_1 or P_3 on S.

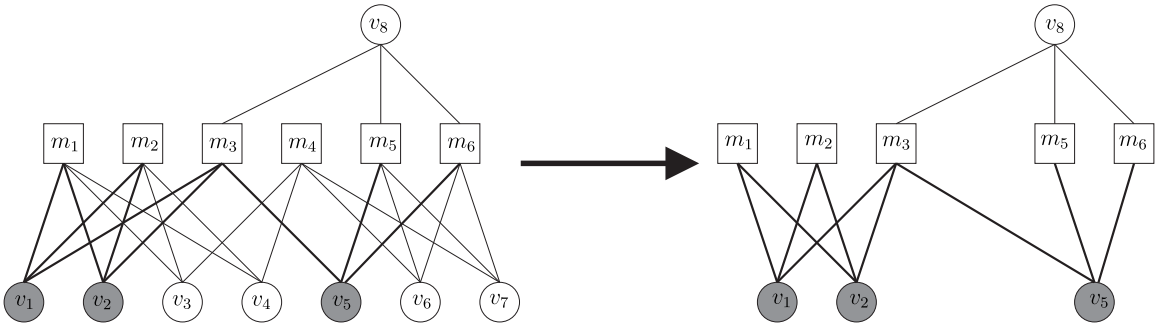
Definition 2.0.7. [Non-binary Restricted Stopping Super Set] A set $S \subseteq V_l$ is called a Non-binary Restricted Stopping Super Set (nRSSS) if every measurement node $m \in \mathcal{N}(S)$ has either property P_1 or P_4 on S.



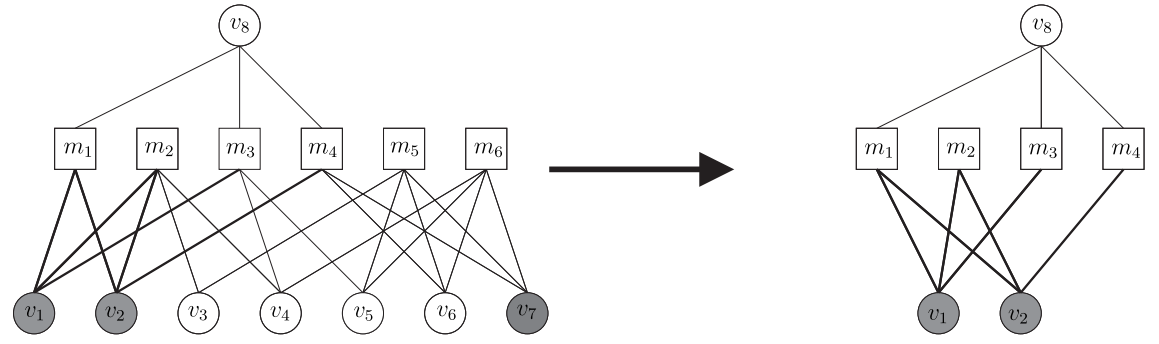
(a) $\mathcal{K} = \{v_2, v_3, v_8\}$ and the Stopping Set $S = \{v_2, v_3\}$



(b) $\mathcal{K} = \{v_1, v_2, v_4\}$ and the Stopping Super Set $S = \{v_2, v_4\}$



(c) $\mathcal{K} = \{v_1, v_2, v_5\}$ and the binary Restricted Stopping Super Set $S = \{v_1, v_2, v_5\}$



(d) $\mathcal{K} = \{v_1, v_2, v_7\}$ and the non-binary Restricted Stopping Super Set $S = \{v_1, v_2\}$

Figure 2.1: Examples of SS, SSS, RSSS and nRSSS in a simple graph

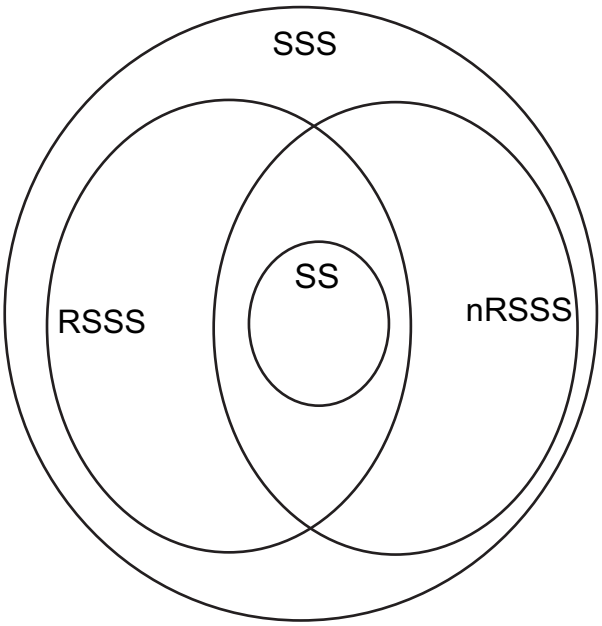


Figure 2.2: Venn diagram of the relationships between SS, SSS, RSSS and nRSSS

Figure 2.2 shows the relations of these sets in a Venn diagram. As it can be seen, any SS is also an SSS, RSSS and nRSSS. Another important issue is that, any RSSS is not necessarily an nRSSS and vice versa.

Definition 2.0.8. For any set of variable nodes $S \subseteq V_l$, the corresponding $H-$set is defined as follows:

$$H(S) \coloneqq \{v \in \mathcal{N}(\mathcal{N}(S)) \setminus S : \ \mathcal{N}(v) \subseteq \mathcal{N}(S).\}$$

Definition 2.0.9. For any set of variable nodes $S \subseteq V_l$, the corresponding $R-$set is defined as follows:

$$R(S) \coloneqq \{v \in \mathcal{N}(\mathcal{N}(S)) \setminus S \ \mid \ [m]_S \subseteq \mathcal{N}(v) \subseteq \mathcal{N}(S), \ \forall m \in \mathcal{N}(v), \}$$

where $[m]_S$ is the equivalence class of m with respect to S.

Variable nodes introduced in property P_2 and P_4 that are outside of the set S can, in fact, induce a certain H-set associated with the corresponding set S. It will be shown that the variable nodes included in an $H-$set are responsible for part of a failure set

in LM algorithm. Besides, we will also show that these sets are also responsible for part of the failures in non-binary SBB algorithm. Moreover, $R-$set of a given RSSS will also be shown to be a part of failure in binary SBB algorithm.

Lemma 2.0.3. *Union of stopping sets is also a stopping set.*

Proof. refer to [55] □

Lemma 2.0.4. *In every subset of V_l there is a unique maximal stopping set.*

Proof. refer to [55] □

Lemma 2.0.5. *The union of two SSSs S_1 and S_2 is also a SSS.*

Proof. Consider $m \in \mathcal{N}(S_1 \cup S_2)$. There are four possibilities:

- Case (A), $m \in \mathcal{N}(S_1) \setminus \mathcal{N}(S_2)$: In this case m has either P_1 or P_2 over S_1. That is:

  $$|\mathcal{N}(m) \cap S_1| \geq 2 \Rightarrow |\mathcal{N}(m) \cap (S_1 \cup S_2)| \geq 2,$$

  or

  $$\exists v \in \mathcal{N}(m) \setminus S_1 \ \mid \mathcal{N}(v) \subseteq \mathcal{N}(S_1),$$

  and since (A) hold true, thus $\mathcal{N}(m) \cap S_2 = \emptyset$, therefore,

  $$\exists v \in \mathcal{N}(m) \setminus (S_1 \cup S_2) \ \mid \mathcal{N}(v) \subseteq \mathcal{N}(S_1 \cup S_2).$$

  Therefore, in this case m has property P_1 or P_2 over $S_1 \cup S_2$.

- Case (B), $m \in \mathcal{N}(S_2) \setminus \mathcal{N}(S_1)$: The proof is exactly the same as the previous case.

- Case (C), $m \in \mathcal{N}(S_1) \cap \mathcal{N}(S_2) \setminus \mathcal{N}(S_1 \cap S_2)$: In this case, since m has at least one connection to S_1 and another connection to S_2, we can always say that property P_1 is always satisfied for m over $S_1 \cup S_2$. That is:

  $$|\mathcal{N}(m) \cap (S_1 \cup S_2)| \geq 2.$$

- Case (D), $m \in \mathcal{N}(S_1 \cap S_2)$: In this case, there are two possibilities:

1. $|\mathcal{N}(m) \cap (S_1 \cup S_2)| \geq 2$: In this case, m clearly has property P_1 on $S_1 \cup S_2$.
2. $|\mathcal{N}(m) \cap (S_1 \cup S_2)| = 1$: In this case, since S_1 and S_2 are both SSSs, then m must satisfy:

$$\exists\, v' \in \mathcal{N}(m) \notin S_1 \;\mid\; \mathcal{N}(v') \subseteq \mathcal{N}(S_1).$$

Since, m has only one connection to $S_1 \cup S_2$ then clearly, $v' \notin S_2$, otherwise m will have two connections to $(S_1 \cup S_2)$. As a result, we can rewrite the expression as:

$$\exists\, v' \in \mathcal{N}(m) \notin (S_1 \cup S_2) \;\mid\; \mathcal{N}(v') \subseteq \mathcal{N}(S_1 \cup S_2),$$

which shows that m in this case has property P_2 on $S_1 \cup S_2$.

Therefore, in all of the possible cases $\forall m \in \mathcal{N}(S_1 \cup S_2)$ either property P_1 or P_2 over $S_1 \cup S_2$ is satisfied. Therefore, $S_1 \cup S_2$ is also a SSS. □

Example 2.0.1. Examples of the problematic sets introduced are shown in Figure 2.1. Along with the problematic sets the set of non-zero elements of the signal is also depicted (shaded variable nodes). The relation between the problematic sets and the non-zero elements of the graph will be clarified later in this chapter. Figure 2.1a shows an SS consisting of nodes $S_1 = \{v_2, v_3\}$. Furthermore, an SSS consisting of nodes $S_2 = \{v_2, v_4\}$ is also depicted in Figure 2.1b. In this figure, the node v_8 is playing the role of the node which helps m_1 and m_5 to have property P_2 on S_2. As it can be seen $v_8 \in H(S)$. In fact, the definitions of the $H-$set and $R-$set is related to the variable nodes that can help measurement nodes in the neighbourhood of a problematic set to have other properties, if they cannot have property P_1 on that problematic set. Moreover, shown in Figure 2.1c and 2.1d are the binary RSSS and nRSSS sets. In the former, as we can see measurement nodes m_5 and m_6 do not have property P_1, and although they will have equal values, due to the fact that they only have one connection to non-zero variable nodes, they cannot verify any variable node, because of the presence of node v_8. The difference between the nRSSS and RSSS in these two cases is that the measurement nodes in nRSSS which do not have property P_1 on the problematic set are not permitted to be connected to a common variable node within the problematic set which is the case for m_3 and m_4, in Figure 2.1d. Also, note that the weights of the edges of the sensing graph in this case are not

binary, and they have to be chosen randomly from a continuous distribution.

2.1 Failure Characterization of The Genie Algorithm

Stopping sets, with the definition given in 2.0.4, are the only sources of failure in Genie algorithm similar to their role in the LDPC codes over BEC with belief propagation decoder. There is a strong correspondence between LDPC codes over BEC and Genie algorithm in CS. This interesting relation can be seen in the characterization of failure patterns in the Genie algorithm. A rigorous type of translation from Genie in CS to LDPC decoding over BEC is as follows:

# of variable nodes	$\rightarrow$	code length
# of measurement nodes	$\rightarrow$	# of parity checks
density factor (δ)	$\rightarrow$	erasure probability

The connection between coding and compressed sensing has been previously studied in some works such as [30] and [63]; however, the detailed connection such as the one we are introducing between LDPC codes over BEC using belief propagation algorithm as the decoder and Genie algorithm has not been addressed yet, to the best of our knowledge. In other words, we will prove that the source of failure for VB MPAs and belief propagation decoder over BEC for LDPC codes are, to some extent, the same combinatorial objects, especially in Genie algorithm. This similarity, however, will be clarified in the subsequent sections.

In LDPC codes over BEC, the belief propagation decoder knows the set of variable nodes that are received correctly, which is equivalent to the knowledge of Genie algorithm about the support set of the input signal. Besides, Theorem 2.1.2 states that the only problematic structure in the Genie algorithm is SS which is exactly the same as LDPC codes over BEC using belief propagation algorithm. In the following sequence of lemmas and theorems, we will introduce SS as the characterization of failures in Genie.

Lemma 2.1.1. *For a set $A \subseteq V_l$,*

$$\mathcal{E}_{G\restriction_A} \subseteq \mathcal{E}_G.$$

Proof. Suppose that:

$$\exists v \in \mathcal{E}_{G\restriction_A} \setminus \mathcal{E}_G.$$

Since $v \notin \mathcal{E}_G$, it means that there is a measurement node in its neighbourhood in graph G that can verify it in a VB MPA. We also know that since $v \in \mathcal{E}_{G\restriction_A}$, then every measurement node of v also exists in $G \restriction_A$, and they are not able to verify it in a VB MPA. Considering that a measurement node cannot loose its ability to verify a variable node by reducing the number of variable nodes in its neighbourhood, this situation is a contradiction, and the proof is complete. □

Theorem 2.1.2. *Let $S \subseteq \mathcal{K}$ be the unique maximal SS in the graph G, then a variable node v will eventually remain unverified in Genie algorithm if and only if $v \in S$, or equivalently, $S = \mathcal{E}_G$.*

Proof. Note that the only rule that Genie uses for verification is D1MN. Since the variable nodes in $\tilde{S} \coloneqq V_l \setminus \mathcal{K}$ are verified prior to the beginning of the algorithm we have to consider $\tilde{G} \coloneqq G \restriction_{\mathcal{K}}$ instead of G. That is, $\mathcal{E}_G = \mathcal{E}_{\tilde{G}}$
For the forward part of the proof, none of the nodes in S can be verified using D1MN rule, simply by the definition of SS.
For the reverse part, consider $\bar{S} \coloneqq \mathcal{K} \setminus S$ as the complement of S with respect to $\mathcal{K}$, and also assume that $S' \subseteq \bar{S}$ is the set of non-zero variable nodes not in S which are remained eventually unverified, that is, $S' = \mathcal{E}_{\tilde{G}} \setminus S$. This implies that:

$$\forall m \in \mathcal{N}(S'),\ |\mathcal{N}(m) \cap (S \cup S')| \geq 2,$$

which is a consequence of the fact that none of the neighbouring measurement nodes of the S' can apply D1MN on S'. Since S is an SS, we have:

$$\forall m \in \mathcal{N}(S),\ |\mathcal{N}(m) \cap S| \geq 2.$$

From the last two expressions, we can conclude that:

$$\forall m \in \mathcal{N}(S') \cup \mathcal{N}(S),\ |\mathcal{N}(m) \cap (S \cup S')| \geq 2,$$

which is equivalent to:

$$\forall m \in \mathcal{N}(S' \cup S),\ |\mathcal{N}(m) \cap (S' \cup S)| \geq 2,$$

which means that $S' \cup S$ is also an SS, and since the assumption is that S is the unique maximal SS, then it implies that $S' = \emptyset$. That is, $\mathcal{E}_G \setminus S = \emptyset$ which results in $\mathcal{E}_G \subseteq S$. □

The introduced connection between Genie and LDPC codes over BEC suggests that we can use variety of techniques used for analysis of LDPC codes over BEC to further investigate the failure characterization of the Genie algorithm. Furthermore, as it will be considered in the following sections, we can also study more general relations between coding theory and CS.

2.2 Failure Characterization of The LM Algorithm

The generalization of a stopping set as super stopping sets will result in failure characterization of LM algorithm. Unlike Genie, LM does not have the information of the support set of the input signal, and consequently, the set of eventually unverified variable nodes in LM consists of two portions. The first portion is the zero valued unverified variable nodes denoted by $\mathcal{E}_G^{\Delta}$, where the super script Δ refers to the set of zero valued variable nodes, and $\mathcal{E}_G^{\mathcal{K}}$, which denotes the set of unverified non-zero valued variable nodes. That is $\mathcal{E}_G = \mathcal{E}_G^{\Delta} \cup \mathcal{E}_G^{\mathcal{K}}$.

Lemma 2.2.1. *In a given graph G, and for the LM algorithm, we have:*

$$\mathcal{N}(\mathcal{E}_G^{\Delta}) \subseteq \mathcal{N}(\mathcal{E}_G^{\mathcal{K}}).$$

Proof. For the proof, we should focus on the eventually unverified variable nodes in the graph. In other words, we have to focused on the pruned graph on which the LM algorithm has stalled.
Assume that $E = \mathcal{N}(\mathcal{E}_G^{\Delta}) \setminus \mathcal{N}(\mathcal{E}_G^{\mathcal{K}})$. Then clearly, we have:

$$\forall m \in E,\ \mu(m) = 0,$$

as every measurement node in E is only connected to zero valued variable nodes in the pruned graph.

Therefore, according to ZMN, every $v \in \mathcal{N}(E)$ will be verified. Thus, every $v \in \mathcal{N}(E) \cap \mathcal{E}_G^{\Delta}$ is also verified. Since,

$$E \subseteq \mathcal{N}(\mathcal{E}_G^{\Delta}),$$

$\mathcal{N}(E) \cap \mathcal{E}_G^{\Delta} = \emptyset$ only if $\mathcal{N}(E) = \emptyset$, which implies that $E = \emptyset$, and consequently,

$$\mathcal{N}(\mathcal{E}_G^{\Delta}) \subseteq \mathcal{N}(\mathcal{E}_G^{\mathcal{K}}).$$

□

Lemma 2.2.2. *There is a unique maximal SSS S in a given subset $\mathcal{K}$ of V_l.*

Proof. It follows from Lemma 2.0.5 that the union of all the subsets of $\mathcal{K}$ that are SSS is also an SSS. By definition, this union is unique and is the largest SSS in $\mathcal{K}$.

□

Lemma 2.2.3. *In a given graph G, let $H(S)$ be the H-set of a given SSS S. Then $S \cup H(S)$ is an SS.*

Proof. we have to prove that:

$$\forall m \in \mathcal{N}(S \cup H(S)),\ |\mathcal{N}(m) \cap (S \cup H(S))| \geq 2,$$

According to Definition 2.0.8, we know that

$$\forall v \in H(S),\ \mathcal{N}(v) \subseteq \mathcal{N}(S),$$

therefore,

$$\mathcal{N}(H(S)) \subseteq \mathcal{N}(S).$$

That is, $\mathcal{N}(S \cup H(S)) = \mathcal{N}(\mathcal{S})$. For every $m \in \mathcal{N}(S)$, we have two cases:

- m has property P_1: therefore,

$$|\mathcal{N}(m) \cap (S \cup H(S))| \geq 2.$$

- m has property P_2:

$$\exists v \in \mathcal{N}(m) \setminus S \ \mid \mathcal{N}(v) \subseteq \mathcal{N}(S) \Rightarrow v \in H(S).$$

Therefore, m has one connection to $H(S)$ via v, besides, we know that m has another connection to S. Thus:

$$|\mathcal{N}(m) \cap (S \cup H(S))| \geq 2.$$

Therefore, $\forall m \in \mathcal{N}(S) = \mathcal{N}(S \cup H(S))$ property P_1 is satisfied. □

Lemma 2.2.4. *Let $H(S)$ be the H-set of the unique maximal SSS S in $\mathcal{K}$, the set of non-zero elements in a given graph G. Then,*

$$H(S) \cap \mathcal{K} = \emptyset.$$

Proof. Suppose that the statement is not true, then there must exists a non-zero valued variable node $v \in H(S)$. From Definition 2.0.8, we know that $\mathcal{N}(v) \subseteq \mathcal{N}(S)$. On the other hand, we know that $\forall m \in \mathcal{N}(S)$ either property P_1 or P_2 is satisfied over S. Therefore, $\forall m \in \mathcal{N}(S \cup \{v\})$, with the same reasoning similar to Lemma 2.0.5, we can conclude that either property P_1 or property P_2 is satisfied over $S \cup \{v\}$. Thus, we can say that $S \cup \{v\}$ is also an SSS which contradicts the assumption that S is the unique maximal SSS. □

Theorem 2.2.5. *Let S be the unique maximal SSS in $\mathcal{K}$ in a given graph G. Then in LM algorithm, a non-zero valued variable node v will remain eventually unverified if and only if $v \in S$. That is, $S = \mathcal{E}_G^{\mathcal{K}}$.*

Proof. Note that a non-zero valued variable node in LM algorithm can be verified only via D1MN. Moreover, we consider the situation where the LM algorithm is stalled and the graph is already pruned. Thus, the remaining graph only contains $\mathcal{E}_G$ as its set of variable nodes.
Let $S' = S \cup H(S)$, where $H(S)$ is the H-set of S. For the forward part of the proof, from Lemma 2.1.1, we know that $\mathcal{E}_{G\restriction_{S'}} \subseteq \mathcal{E}_G$, and consequently, $\mathcal{E}^{\mathcal{K}}_{G\restriction_{S'}} \subseteq \mathcal{E}^{\mathcal{K}}_G$. From Lemma 2.2.3, we know that S' is an SS. Therefore,

$$\forall m \in \mathcal{N}(S'),\ |\mathcal{N}(m) \cap S'| \geq 2. \tag{2.1}$$

Besides, since $\mathcal{N}(H(S)) \subseteq \mathcal{N}(S)$, we have:

$$\forall m \in \mathcal{N}(H(S)),\ \mu(m) \neq 0.$$

As a result, when the LM algorithm is stalled, all the measurement values in the neighbourhood of $H(S)$ are non-zero, and thus, no variable node in the set $H(S)$ can be verified. That is, $H(S) \subseteq \mathcal{E}_G^{\Delta}$. Also, all the measurement nodes in the neighbourhood of S' have at least degree two, and therefore, none of the nodes in S can be verified either, i.e. $S \subseteq \mathcal{E}_G^{\mathcal{K}}$.
For the reverse part of the theorem, assume that

$$\tilde{S} = \mathcal{E}_G^{\mathcal{K}} \setminus S.$$

Thus, every measurement node in the neighbourhood of $\tilde{S}$ should have at least two connections to $\mathcal{E}_G$ to ensure that $\tilde{S}$ remains unverified, since D1MN rule is the only responsible rule for verification of non-zero elements in LM. That is,

$$\forall m \in \mathcal{N}(\tilde{S}),\ |\mathcal{N}(m) \cap \mathcal{E}_G| \geq 2,$$

and thus,

$$\forall m \in \mathcal{N}(\tilde{S}),\ |\mathcal{N}(m) \cap (\mathcal{E}_G^{\mathcal{K}} \cup \mathcal{E}_G^{\Delta})| \geq 2.$$

For all $m \in \mathcal{N}(\tilde{S})$, we obviously have:

$$|\mathcal{N}(m) \cap \mathcal{E}_G^{\mathcal{K}}| \geq 1.$$

Thus, there are two cases:

- m has at least one more connection to $\mathcal{E}_G^{\mathcal{K}}$:

  $$|\mathcal{N}(m) \cap (S \cup \tilde{S})| \geq 2.$$

  That is m has property P_1 on $S \cup \tilde{S}$.

- m has at least one connection to $\mathcal{E}_G^{\Delta}$:

  $$|\mathcal{N}(m) \cap \mathcal{E}_G^{\Delta}| \geq 1.$$

  Also, from Lemma 2.2.1, we have: $\mathcal{N}(\mathcal{E}_G^{\Delta}) \subseteq \mathcal{N}(S \cup \tilde{S})$. Hence,

  $$\exists v \in \mathcal{N}(m) \setminus (S \cup \tilde{S}) \ \mid\ \mathcal{N}(v) \subseteq \mathcal{N}(S \cup \tilde{S}).$$

Thus, in this case, m has property P_2 on $S \cup \tilde{S}$.

As a result, $\forall m \in \mathcal{N}(\tilde{S})$ either property P_1 or P_2 on $S \cup \tilde{S}$ is satisfied. Besides, since S is an SSS then, we can conclude that $\forall m \in \mathcal{N}(S)$ also property P_1 or P_2 on $S \cup \tilde{S}$ is satisfied. Thus, we will have $\forall m \in \mathcal{N}(S \cup \tilde{S})$ either property P_1 or Property P_2 is satisfied on $S \cup \tilde{S}$, which results in the fact that, $S \cup \tilde{S}$ is also an SSS. Since S is assumed to be the maximal SSS then $\tilde{S} = \emptyset$, and consequently, $\mathcal{E}_G^{\mathcal{K}} \subseteq S$. □

Theorem 2.2.6. *Let $H(S)$ be the H-set associated with S, the unique maximal SSS in $\mathcal{K}$ in a given graph G. Then in LM algorithm, a zero valued variable node v will remain eventually unverified if and only if $v \in H(S)$. That is, $H(S) = \mathcal{E}_G^{\Delta}$.*

Proof. For the forward proof, refer to the proof of Theorem 2.2.5, where it has been proved that $H(S) \subseteq \mathcal{E}_G^{\Delta}$.
For the reverse part of the proof, assume that:

$$\tilde{H} = \mathcal{E}_G^{\Delta} \setminus H(S).$$

From Lemma 2.2.1 and Theorem 2.2.5, we have:

$$\mathcal{N}(\tilde{H}) \subseteq \mathcal{N}(S).$$

Hence, $\tilde{H} \subseteq \mathcal{N}(\mathcal{N}(S))$. Also since $\tilde{H} \subseteq \mathcal{E}_G^{\Delta}$, then $\tilde{H} \cap S = \emptyset$. Therefore,

$$\tilde{H} \subseteq \mathcal{N}(\mathcal{N}(S)) \setminus S,$$

and since we have:

$$\forall v \in \tilde{H},\ \mathcal{N}(v) \subseteq \mathcal{N}(S).$$

According to Definition 2.0.8, we can conclude that:

$$\tilde{H} \subseteq H(S),$$

which is true only if $\tilde{H} = \emptyset$, which results in $\mathcal{E}_G^{\Delta} \subseteq H(S)$. □

Basically, Theorems 2.2.5 and 2.2.6 characterize the failures of the LM algorithm. As it is shown in Figure 2.2, every SS is an SSS, which simply restates the fact that Genie algorithm outperforms LM, which is a result of the fact that Genie knows the support set of the input signal. Moreover, there are also interesting connections between LM

and SBB algorithms, that make their comparison worthwhile. However, one needs to consider failure characterization in SBB before comparing it to the LM algorithm. The following section will address this characterization.

2.3 Failure Characterization of The SBB Algorithm

Since there are differences between SBB algorithm implemented over binary and non-binary sensing matrices, then the failure characterization for these two scenarios will be different. Therefore, in this subsection, we will introduce two different failure characterizations. Firstly, we will focus on binary sensing matrices in SBB algorithm, and then we will turn our attention to non-binary case. Hereafter, for the sake of simplicity, we will use the term binary SBB and non-binary SBB to refer to SBB algorithm over binary sensing matrices and non-binary sensing matrices, respectively. Also, we may use the term SBB to refer to binary SBB algorithm whenever, it does not make ambiguity.

Generally speaking, SBB algorithm uses all of the rules for verification of variable nodes. Again, since SBB algorithm has no information about the support set of the signal, then the set of unverified variable nodes will be divided into two parts $\mathcal{E}_G^{\mathcal{K}}$ and $\mathcal{E}_G^{\Delta}$.

RSSSs are shown to be the sources of failure in binary SBB algorithm. These combinatorial structures are also the generalization of SSs according to the Definition of an RSSS. The addition of EMN and EEMN rules to the verification rules in binary SBB, in fact, will solve some of the problematic structures in which LM algorithm fails. These structures are going to be fully addressed via the theorems that are provided in this section.

In characterization of the SBB algorithm, we will use the equivalence relation that is defined in Definition 2.0.1. The following lemma will address the connection of the messages of the measurement nodes in the same equivalence class, being eventually passed, in binary SBB algorithm.

Lemma 2.3.1. *Consider an equivalence class $[a]_{\mathcal{E}_G^{\mathcal{K}}}$, with respect to $\mathcal{E}_G^{\mathcal{K}}$, in a given graph G. Then eventually in binary SBB algorithm we will have:*

$$\forall m \in [a]_{\mathcal{E}_G^{\mathcal{K}}},\ \mu(m) = \mu(a) \neq 0.$$

Proof. Consider a measurement node $m \in [a]_{\mathcal{E}_G^{\mathcal{K}}}$. First of all, since $m \in \mathcal{N}(\mathcal{E}_G^{\mathcal{K}})$ and $\mathcal{E}_G^{\mathcal{K}}$ is the set of unverified non-zero variable nodes, then eventually $\mu(m) \neq 0$. Besides, we know that eventually:

$$\mu(m) = \sum_{v \in \mathcal{N}(m)} \mu(v) = \sum_{v \in \mathcal{N}(m) \cap \mathcal{E}_G^{\mathcal{K}}} \mu(v).$$

Also we know that $m \equiv_{\mathcal{E}_G^{\mathcal{K}}} a.$ that is,

$$\mathcal{N}(m) \cap \mathcal{E}_G^{\mathcal{K}} = \mathcal{N}(a) \cap \mathcal{E}_G^{\mathcal{K}}.$$

Thus,

$$\sum_{v \in \mathcal{N}(m) \cap \mathcal{E}_G^{\mathcal{K}}} \mu(v) = \sum_{v \in \mathcal{N}(a) \cap \mathcal{E}_G^{\mathcal{K}}} \mu(v) = \mu(a).$$

□

Lemma 2.3.2. *For two non-empty variable node sets A and B and a measurement node m in a sensing graph. If $A \subseteq B$, then:*

$$[m]_B \subseteq [m]_A.$$

Proof. Let $Q \coloneqq [m]_B \setminus [m]_A$. For a $q \in Q$, we have:

$$\mathcal{N}(q) \cap B = \mathcal{N}(m) \cap B. \tag{2.2}$$

and

$$\mathcal{N}(q) \cap A \neq \mathcal{N}(m) \cap A. \tag{2.3}$$

Then, since $A \subseteq B$, we have:

$$\forall \alpha \in \mathcal{N}(q) \cap A,\ \alpha \in \mathcal{N}(q) \cap B.$$

Following that, from (2.2), we also have:

$$\alpha \in \mathcal{N}(c) \cap B.$$

Thus,

$$\forall \alpha \in \mathcal{N}(q) \cap A, \alpha \in \mathcal{N}(m) \cap \mathcal{N}(A).$$

As a result,

$$\mathcal{N}(q) \cap A \subseteq \mathcal{N}(m) \cap A.$$

If we start from $\mathcal{N}(m) \cap A$, with the same reasoning, we will have $\mathcal{N}(m) \cap A \subseteq \mathcal{N}(q) \cap A$. Therefore, we can prove that

$$\mathcal{N}(q) \cap A = \mathcal{N}(m) \cap A.$$

which is in a contradiction with (2.3). Therefore, there exists no such α which means that $Q = \emptyset$, i.e. $[m]_B \subseteq [m]_A$. □

Lemma 2.3.3. *The union of two RSSSs S_1 and S_2 is also an RSSS.*

Proof. we have to prove that $\forall m \in \mathcal{N}(S_1 \cup S_2)$ either property P_1 or P_3 is satisfied on $S_1 \cup S_2$. For an $m \in \mathcal{N}(S_1 \cup S_2)$, we have the following cases:

- $m \in \mathcal{N}(S_1) \setminus \mathcal{N}(S_2)$: In this case, since m satisfies either P_1 or P_3 on S_1, then it clearly satisfies one of these properties on $S_1 \cup S_2$, as well, with the same reasoning as in Lemma 2.0.5.
- $m \in \mathcal{N}(S_2) \setminus \mathcal{N}(S_1)$: The situation is exactly the same as the previous case.
- $m \in \mathcal{N}(S_1) \cap \mathcal{N}(S_2) \setminus \mathcal{N}(S_1 \cap S_2)$: therefore, it means that m has at least one connection to S_1 and at least another connection to S_2, thus

  $$|\mathcal{N}(m) \cap (S_1 \cup S_2)| \geq 2,$$

  which means that m satisfies property P_1 on $S_1 \cup S_2$.
- $m \in \mathcal{N}(S_1 \cap S_2)$: We have two possibilities:
  1. $|\mathcal{N}(m) \cap (S_1 \cup S_2)| \geq 2$: In this case, m clearly has property P_1 on $S_1 \cup S_2$.
  2. $|\mathcal{N}(m) \cap (S_1 \cup S_2)| = 1$: In this case, since S_1 is an RSSS, we have:

     $$\exists v' \in \mathcal{N}(m) \setminus S_1 \mid [m']_{S_1} \subseteq \mathcal{N}(v') \subseteq \mathcal{N}(S_1),\ \forall m' \in \mathcal{N}(v').$$

     Since m only has one connection to $S_1 \cup S_2$, v' cannot be a variable node in S_2. That is, v' also exists in $\mathcal{N}(m) \setminus (S_1 \cup S_2)$. Moreover, from Lemma 2.3.2, we have, $[m]_{S_1 \cup S_2} \subseteq [m]_{S_1}$. As a result,

     $$\exists v' \in \mathcal{N}(m) \setminus (S_1 \cup S_2 \mid [m']_{S_1 \cup S_2} \subseteq \mathcal{N}(v') \subseteq \mathcal{N}(S_1 \cup S_2),\ \forall m' \in \mathcal{N}(v')$$

Hence for all cases, for every measurement node $m \in \mathcal{N}(S_1 \cup S_2)$ either P_1 or P_3 is satisfied on $S_1 \cup S_2$. That is, $S_1 \cup S_2$ is an RSSS. □

Lemma 2.3.4. *In a given graph G with $\mathcal{K}$ as the set of non-zero variable nodes, there exists a unique maximal RSSS set $S \subseteq \mathcal{K}$.*

Proof. Suppose that there is another set $S' \neq S$, which is also a maximal RSSS, then from Lemma 2.3.3, $S \cup S'$ is also an RSSS. Since $S' \neq S$, then $|S' \cup S| > |S|$, which contradicts the maximality of S. Therefore, S as the maximal RSSS, is unique. □

Lemma 2.3.5. *For a given RSSS S in a graph G, $S \cup R(S)$ is an SS.*

Proof. The proof follows from the fact that an RSSS is a special case of an SSS, and Lemma 2.2.3. □

Lemma 2.3.6. *Let $R(S)$ be the R-set of the maximal RSSS S in $\mathcal{K}$, then:*

$$R(S) \cap \mathcal{K} = \emptyset.$$

Proof. The proof is similar to that of Lemma 2.2.4. □

Theorem 2.3.7. *In a given graph G, with $\mathcal{K}$ as the set of non-zero variable nodes, assume that $S \subseteq \mathcal{K}$ is the unique maximal RSSS in $\mathcal{K}$. Then a zero variable node $v \in \Delta$ will not be eventually verified in the binary SBB algorithm if and only if $v \in R(S)$, and also a non-zero variable node $v' \in \mathcal{K}$ will remain eventually unverified if and only if $v' \in S$. That is, $R(S) = \mathcal{E}_G^{\Delta}$, and $S = \mathcal{E}_G^{\mathcal{K}}$.*

Proof. Note that we focus on the situation where the SBB algorithm in stalled and the graph is already pruned. That is $\mathcal{E}_G$ is the set of variable nodes in the graph. For the forward part of the proof, assume that $S' = S \cup R(S)$, where $R(S)$ is the R-set of S. Consider the induced sub-graph $G \restriction_{S'}$. From Lemma 2.1.1, $\mathcal{E}_{G\restriction_{S'}} \subseteq \mathcal{E}_G$.
Besides, we know that for zero valued variable nodes there are only ZMN and EEMN rules that can verify them. From Definition 2.0.9, it is apparent that $\mathcal{N}(R(S)) \subseteq \mathcal{N}(S)$. Starting from $G \restriction_{S'}$, at the first iteration $\forall m \in \mathcal{N}(S),\ \mu(m) \neq 0$; therefore, ZMN is not applicable in the first iteration. Besides, again from Definition 2.0.9, we know that:

$$\forall v \in R(S),\ [m]_S \subseteq \mathcal{N}(v)\ \forall m \in \mathcal{N}(v). \tag{2.4}$$

Besides, from Lemma 2.3.1, we can rewrite the verification condition of EEMN rule as:

$$\exists\, m \in \mathcal{N}(v) \;\mid\; [m]_S \nsubseteq \mathcal{N}(v). \tag{2.5}$$

Therefore, from (2.4) and (2.5), we can conclude that EEMN also cannot verify a zero variable node from $R(S)$, in the first iteration in $G \upharpoonright_{S'}$. On the other hand, the only rules that can verify non-zero valued variable nodes are D1MN and EMN. In the first iteration of binary SBB on $G \upharpoonright_{S'}$, D1MN cannot be applied, because according to Lemma 2.3.5, S' is an SS, and therefore,

$$\forall m \in \mathcal{N}(S'),\ |\mathcal{N}(m) \cap S'| \geq 2.$$

Moreover, from Lemma 2.3.1, it follows that

$$\forall m \in \mathcal{N}(S),\ [m]_S = \mathcal{C}(\mu(m)).$$

Applying this fact along with EMN rule, a non-zero variable node $v' \in S$ can be verified via EMN if

$$\exists\, m \in \mathcal{N}(S) \;\mid\; \mathcal{N}(v') = [m]_S,\ |\cap_{m' \in [m]_S} \mathcal{N}(m')| = 1.$$

Since $m \in \mathcal{N}(S)$, then $|\mathcal{N}(m) \cap S| = 1$; therefore,

$$\cap_{m' \in [m]_S} (\mathcal{N}(m') \setminus S) = \emptyset.$$

If $|\mathcal{N}(m) \cap S| = 1$, then it means that m does not satisfy property P_1 on S; therefore, since S is an RSSS, then m must satisfy P_3 instead. Therefore,

$$\exists v \in \mathcal{N}(m) \setminus S \;\mid\; [m]_S \subseteq \mathcal{N}(v) \subseteq \mathcal{N}(S).$$

As a result,

$$\exists v \in \mathcal{N}(m) \setminus S \;\mid\; v \in \cap_{m' \in [m]_S} (\mathcal{N}(m') \setminus S).$$

That is,

$$\cap_{m' \in [m]_S} (\mathcal{N}(m') \setminus S) \neq \emptyset,$$

which is a contradiction, and hence such m does not exit, and therefore, EMN cannot be applied in any of the non-zero variable nodes in S. In summary, none of the

verification rules is applicable in the first iteration of SBB on $G \restriction_{S'}$, which means that the first iteration is the last one and therefore, graph $G \restriction_{S'}$ cannot be simplified more in SBB algorithm which means that $S = \mathcal{E}^{\mathcal{K}}_{G\restriction_{S'}}$, and $R(S) = \mathcal{E}^{\Delta}_{G\restriction_{S'}}$. That is, $S \subseteq \mathcal{E}^{\mathcal{K}}_{G}$ and $R(S) \subseteq \mathcal{E}^{\Delta}_{G}$.

For the reverse part of the theorem, assume that $\tilde{S} = \mathcal{E}^{\mathcal{K}}_{G} \setminus S$ and $\tilde{R} = \mathcal{E}^{\Delta}_{G} \setminus R(S)$. We will show that these two sets are empty.

Since a zero-valued variable node can be verified in SBB only via ZMN and EEMN, then no variable node in $\tilde{R}$ should be eventually verified via these rules.

Since the D1MN and EMN are responsible for a non-zero valued variable node to be verified then none of these rules should be applicable on a variable node $v \in \tilde{S}$.

Thus, from D1MN,

$$\forall m \in \mathcal{N}(\tilde{S}),\ |\mathcal{N}(m) \cap \mathcal{E}_G| \geq 2, \tag{2.6}$$

and also from EMN,

$$\nexists\, m \in \mathcal{N}(\tilde{S}) \ \mid\ [m]_{\mathcal{E}^{\mathcal{K}}_G} \subseteq \mathcal{N}(v),\ |\cap_{m' \in [m]_{\mathcal{E}^{\mathcal{K}}_G}} \mathcal{N}(m')| = 1. \tag{2.7}$$

Thus, from (2.6),

$$\forall m \in \mathcal{N}(\tilde{S}),\ |\mathcal{N}(m) \cap (\mathcal{E}^{\mathcal{K}}_{G} \cup \mathcal{E}^{\Delta}_{G})| \geq 2.$$

Since $m \in \mathcal{N}(\tilde{S})$,

$$|\mathcal{N}(m) \cap \mathcal{E}^{\mathcal{K}}_{G}| \geq 1.$$

Therefore, we have two cases:

- $|\mathcal{N}(m) \cap \mathcal{E}^{\mathcal{K}}_{G}| \geq 2$: In this case, m has property P_1 on $S \cup \tilde{S}$.
- $|\mathcal{N}(m) \cap \mathcal{E}^{\mathcal{K}}_{G}| = 1$ and $|\mathcal{N}(m) \cap \mathcal{E}^{\Delta}_{G}| \geq 1$: In this case, m has only one connection to $S \cup \tilde{S}$, besides (2.7) also should be satisfied.

  Assume that $\{v\} = \mathcal{N}(m) \cap \mathcal{E}^{\mathcal{K}}_{G}$, from Definition 2.0.1, we will have:

  $$\forall m' \in [m]_{\mathcal{E}^{\mathcal{K}}_G},\ \mathcal{N}(m') \cap \mathcal{E}^{\mathcal{K}}_{G} = \{v\}.$$

  Thus,

  $$[m]_{\mathcal{E}^{\mathcal{K}}_G} \subseteq \mathcal{N}(v).$$

Therefore, for (2.7) to be satisfied, we should have:

$$|\cap_{m'\in[m]_{\mathcal{E}_G^{\mathcal{K}}}} \mathcal{N}(m')| \geq 2,$$

ans since,

$$|\cap_{m'\in[m]_{\mathcal{E}_G^{\mathcal{K}}}} \mathcal{N}(m') \cap \mathcal{E}_G^{\mathcal{K}}| = 1,$$

then

$$|\cap_{m'\in[m]_{\mathcal{E}_G^{\mathcal{K}}}} \mathcal{N}(m') \cap \mathcal{E}_G^{\Delta}| \geq 1.$$

Thus, we can conclude that:

$$\exists v' \in \mathcal{N}(m) \cap \mathcal{E}_G^{\Delta},\ [m]_{\mathcal{E}_G^{\mathcal{K}}} \subseteq \mathcal{N}(v'). \tag{2.8}$$

Besides, using the same reasoning as in Lemma 2.2.1, and since $v' \in \mathcal{E}_G^{\Delta}$,

$$\mathcal{N}(v') \subseteq \mathcal{E}_G^{\mathcal{K}}$$

In overall, in this case from the recent equation and (2.8), we have:

$$\exists v' \in \mathcal{N}(m) \setminus (S \cup \tilde{S}),\ [m]_{S\cup\tilde{S}} \subseteq \mathcal{N}(v') \subseteq \mathcal{N}(S \cup \tilde{S}),$$

which means that in this case, m has property P_3 on $S \cup \tilde{S}$.

Hence, In all of the possible cases, $m \in \mathcal{N}(\tilde{S})$ either has property P_1 or P_3 on $S \cup \tilde{S}$. That is, $S \cup \tilde{S}$ is an RSSS. Moreover, since S is the unique maximal RSSS in $\mathcal{K}$, then $\tilde{S} = \emptyset$, which means that $\mathcal{E}_G^{\mathcal{K}} \subseteq S$.
Since ZMN and EEMN are responsible rules for verification of a zero-valued variable node, then $\forall v \in \tilde{R},\ \mathcal{N}(v) \subseteq \mathcal{N}(S)$. Also, equation (2.5) should not be satisfied $\forall v \in \tilde{R}$. Furthermore, from (2.5), we can conclude that

$$\forall m \in \mathcal{N}(v),\ [m]_S \subseteq \mathcal{N}(v).$$

Therefore, we have:

$$\forall v \in \tilde{R},\ [m]_S \subseteq \mathcal{N}(v) \subseteq \mathcal{N}(S)\ \forall m \in \mathcal{N}(v)$$

which means that $\tilde{R} \subseteq R(S)$. That is, $\tilde{R} = \emptyset$, and thus, $\mathcal{E}_G^{\Delta} \subseteq R(S)$. □

In the remainder of this subsection, we will focus on the failure characterization of the non-binary SBB algorithm. In this algorithm, the only rules applied in each iteration are ZMN, D1MN and EIM. Besides, as it is discussed in Remark 1, there is no need to apply EEMN. We will show that nRSSSs are the sources of failure in non-binary SBB algorithm. Moreover, unlike the binary SBB, the R-set of the maximal RSSS in the set of non-zero elements does not hold true in characterization of failure patterns in zero valued variable nodes, instead $H-$set of the nRSSS will be useful in this case, where H-set is defined in Definition 2.0.8. We will see that this property will make the analysis of this algorithm simpler than the analysis of binary SBB.

Remark 2. We will show in the subsequent chapters that binary SBB cannot be simply analyzed using the characterization provided. This is due to a fundamental difference between binary SBB and other three algorithms. In fact, in binary SBB a variable node cannot decide about its verification locally. In other words, according to EMN rule, a variable node which can see two or more measurement nodes in its neighborhood with the same value in SBB, should make sure that it is the unique variable node connected to those measurement nodes which requires the knowledge of the connection of the neighbouring nodes of those measurement nodes. This issue is, intrinsically, reflected in the definition of an $R-$set, and this is the main reason why we will put an exception on the analysis of SBB.

Lemma 2.3.8. *The union of two nRSSSs S_1 and S_2 is also an nRSSS.*

Proof. It is enough to show that $\forall m \in \mathcal{N}(S_1 \cup S_2)$, either property P_1 or P_4 is satisfied on $S_1 \cup S_2$. For every $m \in \mathcal{N}(S_1 \cup S_2)$, there are four possible cases:

- $m \in \mathcal{N}(S_1) \setminus \mathcal{N}(S_2)$: In this case, m should satisfy either P_1 or P_4 on S_1. Suppose that it satisfies P_1 on S_1, then it definitely satisfies P_1 on $S_1 \cup S_2$. Now suppose that m only satisfies P_4 on S_1. That is:

$$|[m]_{S_1}| = 1, \ \exists v \in \mathcal{N}(m) \setminus S_1 \ \mid \ \mathcal{N}(v) \subseteq \mathcal{N}(S_1).$$

  Since $m \notin \mathcal{N}(S_2)$, then such v certainly does not belong to S_2, i.e., $v \in \mathcal{N}(m) \setminus (S_1 \cup S_2)$. Moreover, from Lemma 2.3.2, we know that, $[m]_{S_1 \cup S_2} \subseteq [m]_{S_1}$, and since m is always a member of equivalence class of itself, then it follows that $|[m]_{S_1 \cup S_2}| = 1$. Therefore, we have:

$$|[m]_{S_1 \cup S_2}| = 1, \ \exists v \in \mathcal{N}(m) \setminus (S_1 \cup S_2) \ \mid \ \mathcal{N}(v) \subseteq \mathcal{N}(S_1 \cup S_2).$$

As a result, m should also satisfy P_4 on $S_1 \cup S_2$.

- $m \in \mathcal{N}(S_2) \setminus \mathcal{N}(S_1)$: This case is the same as the previous one.
- $m \in \mathcal{N}(S_1) \cap \mathcal{N}(S_2) \setminus \mathcal{N}(S_1 \cap S_2)$. In this case, m should have at lease one connection to S_1 and another connection to S_2. Thus,

$$|\mathcal{N}(m) \cap (S_1 \cup S_2)| \geq 2.$$

That is, m satisfies P_1 on $S_1 \cup S_2$.

- $m \in \mathcal{N}(S_1 \cap S_2)$: There are two possibilities in this case:
  1. $|\mathcal{N}(m) \cap (S_1 \cup S_2)| \geq 2$: In this case, m clearly satisfies P_1 on $S_1 \cap S_2$.
  2. $|\mathcal{N}(m) \cap (S_1 \cup S_2)| = 1$: In this case, since S_1 is an nRSSS then we have:

$$|[m]_{S_1}| = 1,\ \exists v \in \mathcal{N}(m) \setminus S_1 \ \mid\ \mathcal{N}(v) \subseteq \mathcal{N}(S_1).$$

Since m only has one connection to $S_1 \cup S_2$, then clearly, such v does not belong to S_2, and thus $v \in \mathcal{N}(m) \setminus (S_1 \cup S_2)$. Moreover, from Lemma 2.3.2, $[m]_{S_1 \cup S_2} \subseteq [m]_{S_1}$, as a result $|[m]_{S_1 \cup S_2}| \leq 1$, and since m is always in the equivalence relation with itself then, $|[m]_{S_1 \cup S_2}| \geq 1$. Therefore, $|[m]_{S_1 \cup S_2}| = 1$. Hence, we have:

$$|[m]_{S_1 \cup S_2}| = 1,\ \exists v \in \mathcal{N}(m) \setminus (S_1 \cup S_2) \ \mid\ \mathcal{N}(v) \subseteq \mathcal{N}(S_1 \cup S_2),$$

which means that m satisfies P_4 on $S_1 \cup S_2$.

Therefore, in all possible cases for every $m \in \mathcal{N}(S_1 \cup S_2)$, either property P_1 or P_4 is satisfied on $S_1 \cup S_2$. □

Lemma 2.3.9. *In a graph G, with $\mathcal{K}$ as the set of non-zero variable nodes, there exists a unique maximal nRSSS $S \subseteq \mathcal{K}$.*

Proof. The proof is similar to that of Lemma 2.3.4. □

Lemma 2.3.10. *For a given nRSSS S, in a graph G, $S \cup H(S)$ is an SS.*

Proof. The proof is similar to that of Lemma 2.2.3. □

Lemma 2.3.11. *In a given graph G, with edge weights selected from a continuous distribution and with the set of non-zero variable nodes $\mathcal{K}$, for a given variable node $v \in \mathcal{K}$ and a given measurement node $m \in \mathcal{N}(v)$, we will have the following if and only if $|\mathcal{N}(m) \cap \mathcal{K}| = 1$:*

$$\forall m' \in [m]_{\mathcal{K}} : \quad \frac{\mu(m')}{A_{m',v}} = \frac{\mu(m)}{A_{m,v}} \neq 0.$$

Proof. Suppose that $|\mathcal{N}(m) \cap \mathcal{K}| = 1$, then we have:

$$\forall m' \in [m]_{\mathcal{K}} : \quad \mathcal{N}(m') \cap \mathcal{K} = \{v\}.$$

Therefore,

$$\forall m' \in [m]_{\mathcal{K}} : \quad \mu(m') = \mu(v) A_{m',v} \rightarrow \frac{\mu(m')}{A_{m',v}} = \frac{\mu(m)}{A_{m,v}} \neq 0.$$

For the reverse part of the proof, suppose that,

$$\forall m' \in [m]_{\mathcal{K}} : \quad \frac{\mu(m')}{A_{m',v}} = \frac{\mu(m)}{A_{m,v}} \neq 0,$$

and also suppose that $|\mathcal{N}(m) \cap \mathcal{K}| \geq 2$. Therefore,

$$\exists\ v' \neq v \in \mathcal{N}(m) \cap \mathcal{K}.$$

We have:

$$\forall m' \in [m]_{\mathcal{K}} : \quad \mu(m') = \sum_{v' \in \mathcal{N}(m) \cap \mathcal{K}} \mu(v') A_{m',v'} \\ \sum_{\substack{v' \in \mathcal{N}(m) \cap \mathcal{K} \\ v' \neq v}} \mu(v') A_{m',v'} + \mu(v) A_{m',v}.$$

Then, $\forall m' \neq m \in [m]_{\mathcal{K}}$, we have:

$$\frac{\mu(m')}{A_{m',v}} = \sum_{\substack{v' \in \mathcal{N}(m) \cap \mathcal{K} \\ v' \neq v}} \mu(v') \frac{A_{m',v'}}{A_{m',v}} + \mu(v),$$

while for m, we have:

$$\frac{\mu(m)}{A_{m,v}} = \sum_{\substack{v' \in \mathcal{N}(m) \cap \mathcal{K} \\ v' \neq v}} \mu(v') \frac{A_{m,v'}}{A_{m,v}} + \mu(v).$$

Since m and m' are at least connected to $\mathcal{K}$ once, then their value is not zero. Besides, since the elements of A are chosen randomly from a continuous distribution, then with high probability, we will have:

$$\frac{\mu(m')}{A_{m',v}} \neq \frac{\mu(m)}{A_{m,v}} \neq 0,$$

which is a contradiction; thus, we have: $|\mathcal{N}(m) \cap \mathcal{K}| = 1$. □

Lemma 2.3.12. *In a given graph G, with edge weights selected from a continuous distribution and with the set of non-zero variable nodes $\mathcal{K}$ for a given variable node $v \in \mathcal{K}$ and a given measurement node $m \in \mathcal{N}(v)$, we have:*

$$\forall m_1, m_2 \in [m]_{\mathcal{K}},\ m_1 \neq m_2 : \quad \frac{\mu(m_1)}{A_{m_1,v}} \neq \frac{\mu(m_2)}{A_{m_2,v}},$$

if and only if $|\mathcal{N}(m) \cap \mathcal{K}| \geq 2$.

Proof. since $[m']_{\mathcal{K}} = [m]_{\mathcal{K}}$ for every $m' \in [m]_{\mathcal{K}}$, then we can apply Lemma 2.3.11 for different $[m']_{\mathcal{K}}$, then the proof will be completed. □

Lemma 2.3.13. *Let $H(S)$ be the $H-$set of the unique maximal nRSSS in $\mathcal{K}$. Then,*

$$H(S) \cap \mathcal{K} = \emptyset.$$

Proof. Suppose that this is not true. Then, there must exists a non-zero variable node $v \in H(S)$. Since S_1 is an nRSSS, then $\forall m \in \mathcal{N}(S)$, m should satisfy either property P_1 or P_2 on S. It also follows from Definition 2.0.8 that $\mathcal{N}(v) \subseteq \mathcal{N}(S)$. As a result $\mathcal{N}(S \cup \{v\}) = \mathcal{N}(S)$. Therefore, for every $m \in \mathcal{N}(S \cup \{v\})$, we have two possibilities:

- m satisfies P_1 on S: In this case, m clearly satisfies P_1 on $S \cup \{v\}$, as well.
- m satisfies P_4 on S: In this case, we have:

$$|[m]_S| = 1,\ \exists v' \in \mathcal{N}(m) \setminus S, \quad | \quad \mathcal{N}(v') \subseteq \mathcal{N}(S).$$

There are two more possibilities here:

1. $v' \neq v$: In this case, $v' \in \mathcal{N}(S) \setminus (S \cup \{v\})$. Also from Lemma 2.3.2, we know that $[m]_{S\cup\{v\}} \subseteq [m]_S$, and since m is always in equivalence relation with itself, then $|[m]_{S\cup\{v\}}| = 1$, as well. Therefore, we have:

$$|[m]_{S\cup\{v\}}| = 1,\ \exists v' \in \mathcal{N}(m) \setminus (S \cup \{v\}),\ \ |\ \ \mathcal{N}(v') \subseteq \mathcal{N}(S \cup \{v\}).$$

   which means that m satisfies P_4 on $S \cup \{v\}$.

2. $v = v'$: In this case m clearly has property P_1 on $S \cup \{v\}$.

As it is seen, in all of the cases m satisfies either P_1 or P_4 on $S \cup (S \cup \{v\})$. This means that $S \cup \{v\}$ is also an nRSSS, which is a contradiction to the fact that S is the unique maximal nRSSS in $\mathcal{K}$. □

Theorem 2.3.14. *In a given graph G, with edge weights chosen randomly from a continuous distribution and with $\mathcal{K}$ as the set of non-zero variable nodes, assume that $S \subseteq \mathcal{K}$ is the unique maximal nRSSS in $\mathcal{K}$. Then a zero valued variable node $v \in \Delta$ will remain eventually unverified in non-binary SBB algorithm if and only if $v \in H(S)$, where $H(S)$ is the $H-$set of S, and also a non-zero valued variable node $v' \in \mathcal{K}$ will remain eventually unverified in this algorithm if and only if $v' \in S$. That is, $H(S) = \mathcal{E}_G^{\Delta}$ and $S = \mathcal{E}_G^{\mathcal{K}}$.*

Proof. Define $S' = S \cup H(S)$. We know from Lemma 2.1.1 that $\mathcal{E}_{G\restriction_{S'}} \subseteq \mathcal{E}_G$, which results in $\mathcal{E}^{\mathcal{K}}_{G\restriction_{S'}} \subseteq \mathcal{E}^{\mathcal{K}}_G$, and $\mathcal{E}^{\Delta}_{G\restriction_{S'}} \subseteq \mathcal{E}^{\Delta}_G$.
Let us start from graph $G\restriction_{S'}$. We know that the only rules that can verify non-zero variable nodes in non-binary SBB are D1MN and EIM. From Lemma 2.3.10, we know that S' is a SS; therefore, there is no degree 1 measurement node in $G\restriction_{S'}$, and consequently, D1MN rule is not applicable on $G\restriction_{S'}$. Moreover, since S is an nRSSS, then every measurement node in its neighbourhood either satisfies property P_1 or P_4 on S. Thus, for a measurement node $m \in \mathcal{N}(v)$ where $v \in S$, we have two cases:

- m satisfies property P_1:

$$|\mathcal{N}(m) \cap S| \geq 2.$$

  In this case, according to Lemma 2.3.12, we know that

$$\forall m_1, m_2 \in [m]_{\mathcal{K}} :\quad \frac{\mu(m_1)}{A_{m_1,v}} \neq \frac{\mu(m_2)}{A_{m_2,v}}.$$

Therefore, EIM rule cannot be applied, in this case.

- m satisfies property P_4: In this case, since $|[m]_S| = 1$; therefore, there exists no two distinct measurement nodes m_1, m_2 in $[m]_S$. Therefore, EIM cannot be applied in this case either, on $G \restriction_{S'}$.

Therefore, none of the variable nodes in S can be verified in the first iteration. Besides, we know that the only rule that can verify zero variable nodes in non-binary SBB is ZMN. Since $\mathcal{N}(H(S)) \subseteq \mathcal{N}(S)$, then every measurement node in $\mathcal{N}(H(S))$ does not have zero value in the first iteration. Besides, using the same reasoning as in Lemma 2.2.4, we can conclude that $H(S) \subseteq \Delta$. Consequently, ZMN rule cannot be applied to any of the variable nodes in $H(S)$. Thus, none of the variable nodes in S' will be verified in the first iteration, and as a result, the first iteration on $G \restriction_{S'}$ is also the last iteration; thus, $S = \mathcal{E}^{\mathcal{K}}_{G\restriction_{S'}}$, and $H(S) = \mathcal{E}^{\Delta}_{G\restriction_{S'}}$, i.e. $S \subseteq \mathcal{E}^{\mathcal{K}}_G$ and $H(S) \subseteq \mathcal{E}^{\Delta}_G$.
For the reverse part of the proof, let $\tilde{S} = \mathcal{E}^{\mathcal{K}}_G \setminus S$, and $\tilde{H} = \mathcal{E}^{\Delta}_G \setminus H(S)$. None of the D1MN and EIM rules should be applied to variable nodes in $\tilde{S}$. Moreover, ZMN rule should not be applicable on variable nodes in $\tilde{H}$, as well.
Therefore, $\forall v \in \tilde{S}$, for D1MN rule not being applicable, we should have:

$$\forall m \in \mathcal{N}(v),\ |\mathcal{N}(m) \cap \mathcal{E}_G| \geq 2, \tag{2.9}$$

and for EIM rule not being applicable:

$$\forall\ m_1, m_2 \in \mathcal{N}(v),\ m_1 \neq m_2 : \quad \frac{\mu(m_1)}{A_{m_1,v}} \neq \frac{\mu(m_2)}{A_{m_2,v}}. \tag{2.10}$$

From (2.9), we have:

$$\forall m \in \mathcal{N}(v) :\ |\mathcal{N}(m) \cap (\mathcal{E}^{\mathcal{K}}_G \cup \mathcal{E}^{\Delta}_G)| \geq 2.$$

Since $\mathcal{E}^{\mathcal{K}}_G \cap \mathcal{E}^{\Delta}_G = \emptyset$, and $v \in \tilde{S}$, which means that

$$|\mathcal{N}(m) \cap \mathcal{E}^{\mathcal{K}}_G| \geq 1,$$

there are two possibilities for every $m \in \mathcal{N}(v)$:

- $|\mathcal{N}(m) \cap \mathcal{E}^{\mathcal{K}}_G| \geq 2$: In this case, m has property P_1 over $S \cup \tilde{S}$.
- $|\mathcal{N}(m) \cap \mathcal{E}^{\mathcal{K}}_G| = 1$, and $|\mathcal{N}(m) \cap \mathcal{E}^{\Delta}_G| \geq 1$: There are two cases here:

– $|[m]_{\mathcal{E}_G^{\mathcal{K}}}| \geq 2$. In this case, From (2.10) and Lemma 2.3.12, we can conclude that

$$|\mathcal{N}(m) \cap \mathcal{E}_G^{\mathcal{K}}| \geq 2,$$

which is a contradiction, i.e., this case is not possible.

– $|[m]_{\mathcal{E}_G^{\mathcal{K}}}| = 1$. In this case, since

$$|\mathcal{N}(m) \cap \mathcal{E}_G^{\Delta}| \geq 1,$$

and $\mathcal{N}(\mathcal{E}_G^{\Delta}) \subseteq \mathcal{N}(\mathcal{E}_G^{\mathcal{K}})$ from the same reasoning as in Lemma 2.2.1. thus, we can conclude that

$$\exists\ v \in \mathcal{N}(m) \setminus \mathcal{E}_G^{\mathcal{K}} : \quad \mathcal{N}(v) \subseteq \mathcal{N}(\mathcal{E}_G^{\mathcal{K}}).$$

Therefore, m satisfies property P_4 on $S \cup \tilde{S}$.

It is shown that in all of the cases m satisfies either property P_1 or P_4 on $S \cup \tilde{S}$, besides, since this fact is true for all $m \in \mathcal{N}(v)$ and also for all $v \in \tilde{S}$; therefore, we conclude that $S \cup \tilde{S}$ is also an nRSSS. Since S is the unique maximal nRSSS, then we have, $\tilde{S} = \emptyset$, which means that $\mathcal{E}_G^{\mathcal{K}} \subseteq S$.
Furthermore, for $\tilde{H}$ not being verified, we have to make sure that ZMN rule cannot be applied on the variable nodes in $\tilde{H}$. From the same reasoning as in Lemma 2.2.1, we have, $\mathcal{N}(\mathcal{E}_G^{\Delta}) \subseteq \mathcal{N}(\mathcal{E}_G^{\mathcal{K}})$. As a result, $\mathcal{N}(\tilde{H}) \subseteq \mathcal{N}(\mathcal{E}_G^{\mathcal{K}})$. We have also proved that $S = \mathcal{E}_G^{\mathcal{K}}$; thus, $\mathcal{N}(\tilde{H}) \subseteq \mathcal{N}(S)$. Therefore,

$$\tilde{H} \subseteq \mathcal{N}(\mathcal{N}(S))$$

Besides, since $\tilde{H} \subseteq \mathcal{E}_G^{\Delta}$, then $\tilde{H} \cap S = \emptyset$. Thus,

$$\tilde{H} \subseteq \mathcal{N}(\mathcal{N}(S)) \setminus S.$$

Moreover, we know that $\forall v \in \tilde{H} : \quad \mathcal{N}(v) \subseteq \mathcal{N}(S)$. Therefore, we can conclude that $\tilde{H} \subseteq H(S)$, which means that $\tilde{H} = \emptyset$. Thus, $\mathcal{E}_G^{\Delta} \subseteq H(S)$. □

Chapter 3

Average Ensemble Analysis

The aim of Chapter 2 of the thesis was the characterization of failure patterns in different VB algorithms. In this chapter, we will mainly focus on deriving analytic expressions for the average performance of graphs in an ensemble. For this sake, we will use the notion of the characterization introduced in the previous chapter. Furthermore, in order to be as general as possible in this chapter, we will use the general term *problematic set* to refer to SS, SSS, RSSS and nRSSS, and we will call their corresponding $H-$set or $R-$set as their *associated sets*. In fact, these are problematic sets that are the roots of the failure in VB MPAs. In other words, problematic sets are responsible for the non-zero part of the failure while their associated sets are usually used to highlight the zero part of the failure. Basically, using the characterization provided in Chapter 2, one can easily find the problematic sets in which a specific algorithm will be trapped, given the support set of the input signal. This characterization is worth mentioning in two points of view. Firstly, one can introduce a method of graph construction, which to some extent, can reduce the effects of the problematic sets. Secondly, one can use these characterizations to find the probability of success of a given algorithm, both for a given graph and in a given ensemble of graphs. In this chapter, we will focus mainly on the latter purpose to derive expressions to find the probability of failure or success of a given algorithm in an ensemble. Individual graph analysis will also be addressed later in the next chapter. One useful benefit of working on the average probability of failure in an ensemble is the fact that according to the concentration results, discussed in Theorem 1.5.1, we can conclude that for moderate length problems (e.g. $n = 200$), this expected value for the probability is a good representation of the actual probability for each individual graph in the ensemble. However, we should also be careful about the sparsity of the signal we are working

with. Figure 5.2, in Section 5.4, clearly shows the extent in which the concentration results are applicable to some individual random sensing graphs with different sizes. Clearly, for $n = 252$ the performance of all those sensing graphs are concentrated over their average performance. This concentration can be seen around those values of δ that are far enough from the error floor region. The complete investigation of the error floor phenomenon will be postponed to the next chapter where the individual graph analysis will be addressed.

3.1 General Analysis Framework

For better understanding of the analysis, we will give a review of the work that has been done in [55], meanwhile, we will focus on the changes that are necessary to be considered in order to be able to find the expressions for VB MPAs. The main issue is the contribution of zero variable nodes in the failure patterns as discussed in Chapter 2. Unlike, stopping sets, other problematic sets in this paper contain zero variable nodes. This issue is the main difference between the analysis in this thesis and the one presented in [55].

Consider the ensemble of bi-regular bipartite graphs $\mathcal{G}(n, d_v, d_m)$ with n variable nodes. Each graph in this ensemble is assumed to be uniquely identified using a constellation $\phi : [nd_v] \mapsto [n'd_m]$ where $[a] \coloneqq \{1, 2, \ldots, a\}$, and n' is the number of measurement nodes. In other words, we consider a labelling for each of the edges coming out from a node in the graph. For instance, we assume that the edges coming out from variable nodes are labelled from 1 to nd_v and also the edges coming out from measurement nodes are labelled from 1 to $n'd_m$. Clearly, every constellation from $[nd_v]$ to $[n'd_m]$ will result in a new graph in the ensemble. Now, assume that the probability of having a variable node non-zero is δ, then we can denote the probability of failure of one of VB MPAs over a given graph $G \in \mathcal{G}(n, d_v, d_m)$ by $P_f(G, \delta)$, and also the probability of success by $P_s(G, \delta)$. Now, if we choose a graph randomly from an ensemble $\mathcal{G}(n, d_v, d_m)$ (according to uniform distribution), given δ, then the average probability of failure will be $\mathbf{E}_{\mathcal{G}(n,d_v,d_m)}\{P_f(G, \delta)\}$, which is only a function of δ. For finding this probability, we first need to find the probability of failure as a function of $\mathcal{K}$, the support set of the input signal, and then use the total probability theorem with respect to the probability of having $\mathcal{K}$ to be the support set, to reach to the main goal.

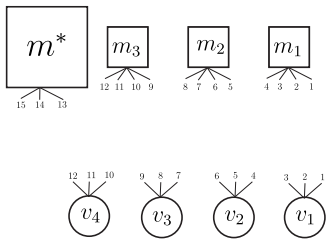


Figure 3.1: Example of a variable and measurement socket labelling in $\mathcal{T}(\{v_1, \ldots, v_4\}, \{m_1, m_2, m_3\}, 3)$.

Firstly, let us define a variable (measurement) socket as the socket of a variable (measurement) node that accepts an edge coming from a measurement (variable) node. We will use the definition of these sockets for better understanding of the constellations in an ensemble of graphs. To be consistent with the literature, we will use, to some extent, the same notations as in [55], which are shown in Table 3.1. In this table, V_l is the set of variable nodes, V_r is the set of measurement nodes and d^* is the degree of a super measurement node m^*. The super measurement node, m^*, is a measurement node in a graph which always assumed to have property P_1 to P_4 on any subset of variable nodes regardless of its connection to variable nodes. The addition of this node is necessary for the sake of the analysis. This matter will be clarified later in this section. Note that the addition of a super measurement node with degree d^* will increase the number of edge sockets in the measurement node side which will eventually affect the total number of graphs in the ensemble. Also, we will only consider the regular ensembles which make our expressions simpler. Besides, the degree of the variable nodes and measurement nodes are still assumed to be d_v and d_m, respectively. Moreover, in the analysis, we will frequently use S as a problematic set, K as the set of neighbouring measurement node of S and A as the associated set of S. Furthermore, for the cardinalities of these sets we will use $s = |S|$, $k = |K|$ and $a = |A|$.

The first step toward the calculation of the probability of failure is the enumeration of all the graphs in $\mathcal{T}(V_l, V_r, d^*)$.[1] In $\mathcal{T}(V_l, V_r, d^*)$, we consider an index for each variable and measurement socket in the graph. In total, we have nd_v variable sockets labelled from 1 to nd_v and also we have $n'd_m + d^*$ measurement socket labelled from 1 to $n'd_m + d^*$. For better understanding of the labelling see Figure 3.1.

[1] Note that $\mathcal{T}(V_l, V_r, 0) \equiv \mathcal{G}(v, d_v, d_m)$.

Table 3.1: Notations used in the analysis

$\mathcal{T}(V_l, V_r, d^*)$	Set of all graphs with V_l as the set of variable nodes all with degree d_v and V_r as the set of measurement nodes with degree at most d_m where $\|V_l\|d_v = \|V_r\|d_m$, along with a super measurement node with degree at most d^*.
$\mathcal{N}(\mathcal{K}, V_l, V_r, d^*)$	Set of all graphs $G \in \mathcal{T}(V_l, V_r, d^*)$ with $\mathcal{K} \subseteq V_l$ as the set of non-zero variable nodes, that do not have any non-zero eventually unverified variable node contained in $\mathcal{K}$.
$\mathcal{M}(\mathcal{K}, V_l, V_r, d^*)$	Set of all graphs $G \in \mathcal{T}(V_l, V_r, d^*)$ with $\mathcal{K} \subseteq V_l$ as the set of non-zero variable nodes, that have at least one non-zero eventually unverified variable node contained in $\mathcal{K}$.
$\mathcal{O}(\mathcal{K}, S, V_l, V_r, d^*)$	Set of all graphs $G \in \mathcal{T}(V_l, V_r, d^*)$ with $\mathcal{K} \subseteq V_l$ as the set of non-zero variable nodes, that have S as their maximal problematic set contained in $\mathcal{K}$.
$\mathcal{L}(\mathcal{K}, S, A, V_l, K, d^*)$	Set of all graphs $G \in \mathcal{T}(V_l, V_r, d^*)$ with $\mathcal{K} \subseteq V_l$ as the set of non-zero variable nodes, that have S as their problematic set in $\mathcal{K}$, not necessarily the maximal one. Also, K is the set of measurement node in the neighbourhood of S, and A is the associated set of S.

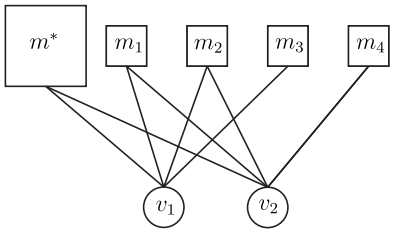


Figure 3.2: Example of a graph in $\mathcal{T}(\{v_1, v_2\}, \{m_1, \ldots, m_4\}, 2)$.

Based on the definition, $\mathcal{T}(V_l, V_r, d^*)$ contains regular variable nodes with degree d_v and irregular measurement nodes with degree at most d_m, and a super measurement node with degree at most d^*. In the real problem, we do not have any super measurement node which means that $d^* = 0$. In such case, every measurement node in a graph in the ensemble will have exactly degree d_m. In other words, $\mathcal{T}(V_l, V_r, 0)$ is equivalent to the ensemble of bi-regular bipartite graphs. Nevertheless, in order to be able to find the probability of failure in an ensemble, we need to define the super measurement node. This will be clarified later. An example of a graph in $\mathcal{T}(\{v_1, \ldots, v_4\}, \{m_1, m_2\}, 2)$ is shown in Figure 3.2.

The next general step in the analysis is enumerating the graphs inside $\mathcal{M}(\mathcal{K}, V_l, V_r, d^*)$, as the graphs in this ensemble contain at least one problematic set inside the support set of the input signal, which is enough for the failure of the graph according to characterization provided in Chapter 2. Then having $\mathcal{M}(\mathcal{K}, V_l, V_r, d^*)$ and $\mathcal{T}(V_l, V_r, d^*)$ available, the probability that a VB MPA fails on a randomly chosen graph from the ensemble under consideration will be:

$$\mathbf{E}_{\mathcal{G}(n,d_v,d_m)}\{P_f(G,\delta)\} = \sum_{\mathcal{K}\subseteq V} \delta^{|\mathcal{K}|}(1-\delta)^{v-|\mathcal{K}|}\mathbf{E}_{\mathcal{G}(n,d_v,d_m)}\{P_f(G|\mathcal{K})\},$$

where

$$\mathbf{E}_{\mathcal{G}(n,d_v,d_m)}\{P_f(G|\mathcal{K})\} = \frac{|\mathcal{M}(\mathcal{K}, V_l, V_r, 0)|}{|\mathcal{T}(V_l, V_r, 0)|}. \tag{3.1}$$

Clearly, the value of $|\mathcal{M}(\mathcal{K}, V_l, V_r, 0)|$ does not depend on the specific choice of $\mathcal{K}$, but it depends on the size of $\mathcal{K}$. As a result we have:

$$\mathbf{E}_{\mathcal{G}(n,d_v,d_m)}\{P_f(G||\mathcal{K}|)\} = \frac{|\mathcal{M}'(|\mathcal{K}|, V_l, V_r, 0)|}{|\mathcal{T}(V_l, V_r, 0)|},$$

where, informally speaking, $|\mathcal{M}'(|\mathcal{K}|, V_l, V_r, 0)| = |\mathcal{M}(\mathcal{K}, V_l, V_r, 0)|$. Therefore, we have:

$$\mathbf{E}_{\mathcal{G}(n,d_v,d_m)}\{P_f(G,\delta)\} = \sum_{|\mathcal{K}|=1}^{|V_l|} \binom{|V_l|}{|\mathcal{K}|} \delta^{|\mathcal{K}|}(1-\delta)^{v-|\mathcal{K}|}\mathbf{E}_{\mathcal{G}(n,d_v,d_m)}\{P_f(G||\mathcal{K})|\}. \tag{3.2}$$

As mentioned before, corresponding to each graph in the ensemble there exists a constellation $\phi : [nd_v] \mapsto [n'd_m + d^*]$, which maps the labelled variable sockets to labelled measurement sockets. In this mapping, the range is clearly larger than the the domain. Therefore, in the enumeration process a nd_v variable sockets can choose between $n'd_m + d^*$ measurement sockets. Obviously, we have:

$$|\mathcal{T}(V_l, V_r, d^*)| = \binom{n'd_m + d^*}{nd_v} (nd_v)!$$

as the total number of constellations, or equivalently, the total number of graphs in the ensemble. Hereafter, we will provide required expressions for calculating

$|\mathcal{M}(\mathcal{K}, V_l, V_r, d^*)|$.

Lemma 3.1.1.

$$\mathcal{O}(\mathcal{K}, S_1, V_l, V_r, d^*) \cap \mathcal{O}(\mathcal{K}, S_2, V_l, V_r, d^*) = \emptyset, \ \forall S_1 \neq S_2.$$

Proof. By contradiction, assume that the intersection is not empty. That is, there exists a graph G in the ensemble that has both S_1 and S_2 as its maximal problematic set in $\mathcal{K}$. Since for every problematic set, the maximal one is proven to be unique, this requires $S_1 = S_2$ which is a contradiction. □

Based on the definition in Table 3.1, we have:

$$\mathcal{M}(\mathcal{K}, V_l, V_r, d^*) = \bigcup_{\substack{S \subseteq \mathcal{K} \\ S \neq \emptyset}} \mathcal{O}(\mathcal{K}, S, V_l, V_r, d^*).$$

From Lemma 3.1.1, we have:

$$|\mathcal{M}(\mathcal{K}, V_l, V_r, d^*)| = \sum_{\substack{S \subseteq \mathcal{K} \\ S \neq \emptyset}} |\mathcal{O}(\mathcal{K}, S, V_l, V_r, d^*)|. \tag{3.3}$$

Therefore, for calculation of $|\mathcal{M}(\mathcal{K}, V_l, V_r, d^*)|$, we need to find $|\mathcal{O}(\mathcal{K}, S, V_l, V_r, d^*)|$. This part is the most challenging part of the analysis.

Definition 3.1.1 (Left sub-graph (LSG) and right sub-graph (RSG))**.** Given a partition of the set of variable nodes $P = (V_l^{(L)}, V_l^{(R)})$, i.e. $V_l = V_l^{(L)} \dot{\cup} V_l^{(R)}$, a graph $G \in \mathcal{T}(V_l, V_r, d^*)$ can be partitioned into two sub-graphs with respect to P, namely left sub-graph $G^{(L)}$ and right sub-graph $G^{(R)}$, as follows:

$$G^{(L)} = \left(V_l^{(L)} \cup (\mathcal{N}(V_l^{(L)} \cup \{m^*\})), E^{(L)}\right),$$
$$G^{(R)} = \left(V_l^{(R)} \cup ((\mathcal{N}(V_l^{(R)}) \setminus \{m^*\}) \cup \{m_r^*\}), E^{(R)}\right).$$

where $E^{(L)}$ is the set of all edges emanating from $V_l^{(L)}$, and $E^{(R)}$ is also the set of all edges emanating from $V_l^{(R)}$. Also, the node m^* is the super measurement node of the original graph G and m_r^* is a newly added super measurement node for $G^{(R)}$, which has not existed in the original graph. In fact, every edges which emanate from $V_l^{(R)}$ and are connected to $G^{(L)}$ will be connected to m_r^* inside $G^{(R)}$. An example of a partitioning is shown in Figure 3.3.

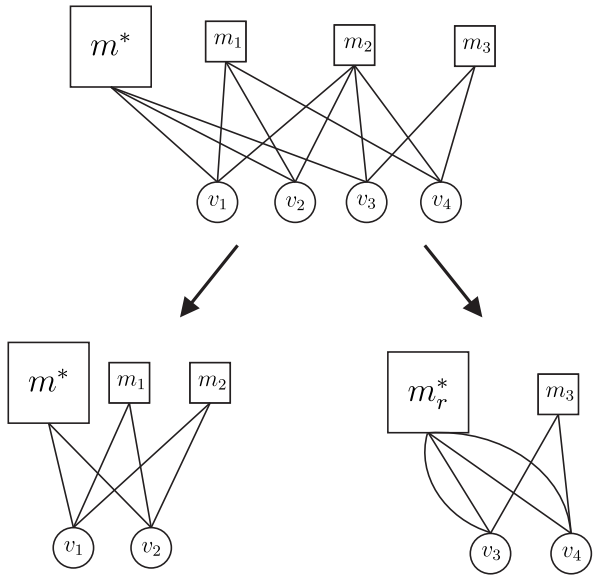


Figure 3.3: Example of partitioning of a graph in $\mathcal{T}(\{v_1, \ldots, v_4\}, \{m_1, m_2, m_3\}, 3)$.

Note that each of these sub-graphs is a member of another ensemble which is smaller than the original ensemble $\mathcal{T}(V_l, V_r, d^*)$. That is,

$$G^{(L)} \in \mathcal{T}(V_l^{(L)}, \mathcal{N}(V_l^{(L)}), d^*),$$
$$G^{(R)} \in \mathcal{T}(V_l^{(R)}, V_r \setminus \mathcal{N}(V_l^{(L)}), d_r^*),$$

where $d_r^* = |\mathcal{N}(V_l^{(L)})|d_m + d^* - |V_l^{(L)}|d_v$. Based on the definition of m_r^*, the recent equation is obvious. The value of d_r^* is the maximum possible degree of m_r^* and it is, in fact, the total number of measurement sockets in $G^{(L)}$ which has not been connected to any variable nodes in $V_l^{(L)}$.

Lemma 3.1.2. *In a given graph G with the set of non-zero variable nodes $\mathcal{K}$, consider a partitioning with respect to $(S \cup A, V_l \setminus (S \cup A))$, where $S \subseteq \mathcal{K}$ is a problematic set and A is its associated set. Then for any $Q \subseteq \mathcal{N}(V_l^{(R)})$, if a measurement node $m \in \mathcal{N}(S) \setminus \mathcal{N}(Q)$ has property P_i, $i \in \{1, 2, 3, 4\}$ on S then it has the same property on $S \cup Q$, where $V_l^{(R)}$ is the set of variable nodes of the RSG, $G^{(R)}$, in the partition.*

Proof. We have 4 possibilities:

1. m **has** P_1 **on** S**:** In this case, m has at least two connections to S, which means that m also has at least two connections to $S \cup Q$.

2. m **has** P_2 **on** S**:** In this case, we have:

$$\exists v \in \mathcal{N}(m) \setminus S \ \mid \ \mathcal{N}(v) \subseteq \mathcal{N}(S).$$

From Chapter 2, we know that such v belongs to the associated set of S, i.e. A. Since $A \cap Q = \emptyset$, according to the partitioning, then

$$\exists v \in \mathcal{N}(m) \setminus (S \cup Q) \ \mid \ \mathcal{N}(v) \subseteq \mathcal{N}(S \cup Q),$$

which means that m has P_2 on the $S \cup Q$.

3. m **has** P_3 **on** S**:** Similar to case 2.

4. m **has** P_4 **on** S**:** Similar to case 2.

□

Theorem 3.1.3. *In a given graph $G \in \mathcal{T}(V_l, V_r, d^*)$, with $\mathcal{K} \subseteq V_l$ as the set of non-zero variable nodes, let $S \subseteq \mathcal{K}$ be a problematic set and A be its associated set. Consider a partitioning of G with respect to $(S \cup A, V_l \setminus (S \cup A))$, as a partition on V_l, which results in two sub-graphs $G^{(L)}$ with the set of variable nodes $V_l^{(L)} = S \cup A$ and $G^{(R)}$ with the set of variable nodes $V_l^{(R)} = V_l \setminus (S \cup A)$. In such a case, S is the unique maximal problematic set in G if and only if $G^{(R)}$ does not contain any problematic set in $V_l^{(R)} \cap \mathcal{K}$.*

Proof. For the forward part of the proof, let us assume that S is the problematic set in G but not the maximal one in $\mathcal{K}$. In $G^{(L)}$, since S is the set of non-zero elements, there cannot exist a problematic set greater than S. Thus, at least a subset of the maximal problematic set, denoted by S_M, should be contained in the set of non-zero variable nodes of $G^{(R)}$, i.e. $V_l^{(R)} \cap \mathcal{K}$. Let us call this subset S_R, i.e. $S_M = S_R \cup S$. For S_R to be a part of the S_M, every measurement node connected to S_R should either have property P_1 or one of the properties between P_2 to P_4 on S. A measurement nodes m' in neighbourhood of S_R in G can be in one of the following cases:

1. $m' \in \mathcal{N}(S_R) \setminus \mathcal{N}(S)$**:** In this case, since m' is not connected to S, then it should have the appropriate properties on the set S_R, which means that even in $G^{(R)}$ this measurement node satisfies the appropriate properties on S_R.

2. $m' \in \mathcal{N}(S_R) \cap \mathcal{N}(S)$**:** In this case, m' has one or more connections to S and one or more connections to S_R. Based on Definition 3.1.1, m' does not exist in the measurement nodes of $G^{(R)}$. Note that in this case, in $G^{(R)}$, those edges of S_R that are connected to m' in G, are now connected to m_r^* in $G^{(R)}$. In fact, every measurement node similar to m' in this case, produce the new super measurement node m_r^* in $G^{(R)}$. This is why we assume that the super measurement node has all of the properties on any given set of variable nodes in a graph.

Therefore, we can conclude that S_R is a problematic set in $G^{(R)}$. Note that, all of the measurement nodes in the neighbourhood of S_R have appropriate properties on S_R, according to case 1. m_r^* is also assumed to have every properties on every variable set based on the definition. Having S_R as a problematic set in $G^{(R)}$ is a contradiction, which proves the forward part of the theorem.
For the reverse part of the proof, suppose that S_R is a problematic set in $G^{(R)}$. Then all we need to show is that $S \cup S_R$ is a problematic set in G which proves that S is not the maximal problematic set in G. In order to show that $S \cup S_R$ is also a problematic set in G, we need to show that every measurement node in $\mathcal{N}(S \cup S_R)$ satisfies appropriate properties on $S \cup S_R$. For a measurement node $m'' \in \mathcal{N}(S \cup S_R)$, we have:

1. $m'' \in \mathcal{N}(S) \setminus \mathcal{N}(S_R)$**:** In this case, if m'' has property P_i on S then it has the same property on $S \cup S_R$, according to Lemma 3.1.2.

2. $m'' \in \mathcal{N}(S_R) \setminus \mathcal{N}(S)$**:** With the same reasoning as in Lemma 3.1.2, we can conclude that m'' also has the appropriate properties on $S \cup S_R$.

3. $m'' \in \mathcal{N}(S_R) \cap \mathcal{N}(S)$**:** In this case, since $S \cap S_R = \emptyset$ then, m'' certainly has at least one connection to S and another to S_R. That is, m'' has property P_1 on $S \cup S_R$.

Therefore, in all of the cases it has been shown that m'' satisfies the appropriate property on $S \cup S_R$, which means that $S \cup S_R$ is a problematic set, which is a contradiction to the maximality of S.

□

Lemma 3.1.4. *In an ensemble $\mathcal{T}(V_l, V_r, d^*)$, let $\mathcal{K} \subseteq V_l$ be the set of non-zero variable nodes. Moreover, consider $S \subseteq \mathcal{K}$ and $A \subseteq V_l \setminus \mathcal{K}$, and $K \subseteq V_r$. The number of*

Table 3.2: Information we know about $G^{(L)}$ and $G^{(R)}$ based on the partitioning.

Description of the set	$G^{(L)}$	$G^{(R)}$
set of variable nodes	$S \cup A$	$V_l \setminus (S \cup A)$
set of non-zero variable nodes	S	$\mathcal{K} \setminus (S \cup A)$
set of measurement nodes	K	$V_r \setminus K$
problematic set	S	-
problematic associated set	A	-
degree of super measurement node	d^*	d_r

graphs in this ensemble that has S as their unique maximal problematic set and A as its associated set and $K = \mathcal{N}(S)$ is as follows:

$$|\mathcal{L}(S, S, A, S \cup A, K, d^*)||\mathcal{N}(\mathcal{K} \setminus S, V_l \setminus (S \cup A), V_r \setminus K, d_r)|,$$

where $d_r = d^* + |K|d_m - |S \cup A|d_v$.

Proof. For a graph $G \in \mathcal{T}(V_l, V_r, d^*)$, to have S as its maximal problematic set according to Lemma 3.1.3, the only way is that the two sub-graphs of G produced by partitioning of G with respect to $(S \cup A, V_l \setminus (S \cup A))$ has the following properties. The graph $G^{(L)}$ should have S as one of its problematic set, and $G^{(R)}$ should contain no problematic set. In other words, according to the information that we know from the partitioning (see Table 3.2), the only way that G can have S as its maximal problematic set with A as its associated set and K as its neighbourhood, is that $G^{(L)}$ belongs to $\mathcal{L}(S, S, A, S \cup A, K, d^*)$ and $G^{(R)}$ belongs to $\mathcal{N}(\mathcal{K} \setminus S, V_l \setminus (S \cup A), V_r \setminus K, d_r)$. As a result, the total number of graphs in $\mathcal{T}(V_l, V_r, d^*)$ with S as the maximal problematic set and A as its associated set and K as its neighbourhood, clearly, equals:

$$|\mathcal{L}(S, S, A, S \cup A, K, d^*)||\mathcal{N}(\mathcal{K} \setminus S, V_l \setminus (S \cup A), V_r \setminus K, d_r)|.$$

□

From Lemma 3.1.4, we can immediately conclude the following:

$$|\mathcal{O}(\mathcal{K}, S, V_l, V_r, d^*)| = \sum_{K \subseteq V_r} \sum_{A \subseteq (V_l \setminus \mathcal{K})} |\mathcal{L}(S, S, A, S \cup A, K, d^*)||\mathcal{N}(\mathcal{K} \setminus S, V_l \setminus (S \cup A), V_r \setminus K, d_r)|. \tag{3.4}$$

Moreover, from Table 3.1, it is clear, that we have:

$$|\mathcal{N}(\mathcal{K}, V_l, V_r, d^*)| = |\mathcal{T}(V_l, V_r, d^*)| - |\mathcal{M}(\mathcal{K}, V_l, V_r, d^*)|. \tag{3.5}$$

As it can be seen, (3.3), (3.4) and (3.5) make a recursive formulation for computing the value of $|\mathcal{M}(\mathcal{K}, V_l, V_r, d^*)|$, whose boundary condition is given in (3.6), which states that the number of edges emanating from variable nodes should not exceed the number of edges of the measurement nodes.

$$|\mathcal{O}(\mathcal{K}, S, V_l, V_r, d^*)| = 0, \ if \ nd_v > n'd_m + d^*. \tag{3.6}$$

What remains from the analysis is to find an expression for $|\mathcal{L}(S, S, A, S \cup A, K, d^*)|$. As it is expected, this expression is dependent upon the specific algorithm being used for the problem, or in other words, it depends on the specific problematic set. Moreover, evaluation of this expression is equivalent to the enumeration of problematic sets. In fact, the general framework that has been introduced in this section so far only prepares the road for the main part of the analysis which is the enumeration of SSs, SSSs, RSSSs and nRSSSs.

As a summary of the general framework of the analysis, Algorithm 2 demonstrates a general overview of the steps that are required for the analysis of VB algorithms. The procedure from which one should start the analysis is the calculation of $\mathcal{M}$-Fun$(\mathcal{K}, V_l, V_r, d^*)$. If one follows the instruction in the procedure the recursions will be cleared within the functions. Moreover, the contribution of zero variable nodes which is the main difference between the analysis in this thesis and the one in [55] is mainly in the computation of $|\mathcal{L}(S, S, A, S \cup A, K, d^*)|$.

Algorithm 2 Analysis of VB Algorithms

procedure $\mathcal{M}$-FUN($\mathcal{K}, V_l, V_r, d^*$) ▷ see equation (3.3)
  Input: $\mathcal{K}, V_l, V_r, d^*$
  Output: $|\mathcal{M}(\mathcal{K}, V_l, V_r, d^*)|$
  $Sum \leftarrow 0$
  for $S \subseteq \mathcal{K}$ **do**
    $Sum \leftarrow Sum+$ $\mathcal{O}$-Fun($\mathcal{K}, S, V_l, V_r, d^*$)
  end for
  return Sum
end procedure

procedure $\mathcal{O}$-FUN($\mathcal{K}, S, V_l, V_r, d^*$) ▷ see equation (3.4)
  Input: $\mathcal{K}, S, V_l, V_r, d^*$
  Output: $|\mathcal{O}(\mathcal{K}, S, V_l, V_r, d^*)|$
  $Sum \leftarrow 0$
  if $|V_l|d_v > |V_r|d_m + d^*$ **then** ▷ see condition (3.6)
    $|\mathcal{O}(\mathcal{K}, S, V_l, V_r, d^*)| \leftarrow 0$
    return 0
  else
    for $K \subseteq V_r$ **do**
      for $A \subseteq V_l \setminus \mathcal{K}$ **do**
        $g_l \leftarrow |\mathcal{L}(S, S, A, S \cup A, K, d^*)|$
        if $g_l \neq 0$ **then**
          $d_r \leftarrow |K|d_m + d^* - |S \cup A|d_v$
          $\bar{V} \leftarrow V_l \setminus (S \cup A)$
          $g_r \leftarrow \mathcal{N}$-Fun($\mathcal{K} \setminus S, \bar{V}, V_r \setminus K, d_r$)
          $Sum \leftarrow Sum + g_l \times g_r$
        end if
      end for
    end for
    return Sum
  end if
end procedure

procedure $\mathcal{N}$-FUN($\mathcal{K}, V_l, V_r, d^*$) ▷ see equation (3.5)
  Input: $\mathcal{K}, V_l, V_r, d^*$
  Output: $|\mathcal{N}(\mathcal{K}, V_l, V_r, d^*)|$
  $T \leftarrow \binom{|V_r|d_m+d^*}{|V_l|d_v}(|V_l|d_v)!$
  return $T -$ $\mathcal{M}$-Fun($\mathcal{K}, V_l, V_r, d^*$)
end procedure

3.2 Problematic Set Enumeration

In this section, we will use the polynomial function method to enumerate problematic sets in $\mathcal{L}(S, S, A, S \cup A, K, d^*)$. In general, polynomial function method will use the coefficients of a polynomial function as the enumerator of the objects under consideration. This polynomial function is obtained using the properties of the enumerative objects in the problem.

We will first introduce the general approach of the enumeration, and then, based on this approach, we will also show that the problem of enumeration of a specific problematic set, e.g. SS, SSS, RSSS and nRSSS, is reduced to determination of a single parameter (which will be defined later in this section) related to that problematic set. Theorem 3.2.1, given in the following, states the general approach of the enumeration which can be applied to SS, SSS and nRSSS. Note that the case for RSSS is excluded from this approach. The reason is given in Remark 2.

We start with the fact that every measurement node in the neighbourhood of a problematic set should either have property P_1 on S or another property based on a specific problematic set. Since the existence of property P_1 is enough for a measurement node to be involved in a given problematic set, we can use this common property to simplify the enumeration. Firstly, we will divide the set of measurement nodes into two different sets, that is $K = \Gamma \dot{\cup} \bar{\Gamma}$, where Γ will be considered as the set of measurement nodes which does not have property P_1 on S, and $\dot{\cup}$ indicates the disjoint union operator which emphasize that the two operands are disjoint sets. In other words, Γ is the set of measurement nodes that have only one connection to S. Then, we can separately study the connection of measurement nodes in Γ and $\bar{\Gamma}$ to variable nodes in S and A. Theorem 3.2.1 will use this fact to derive a general equation for problematic set enumeration.

Theorem 3.2.1.

$$|\mathcal{L}(S, S, A, S \cup A, K, d^*)| = \sum_{\Gamma \subseteq K} F A_{\mathcal{P}}(\Gamma) d_m^{\gamma} (sd_v - \gamma)!(\gamma)!(ad_v)!,$$

where

$$F = coef(f(x, y)g(y)(1 + x + y)^{d^*}, x^{sd_v - \gamma} y^{ad_v}),$$

in which

$$f(x,y) = \left((1+x+y)^{d_m} - (1+y)^{d_m} - d_m x(1+y)^{d_m-1})\right)^{k-\gamma},$$
$$g(y) = \left((1+y)^{d_m-1} - 1\right)^{\gamma},$$

and $coef(f(x), i)$ *denotes the coefficient of* x^i *in the polynomial expansion of* $f(x)$. *Besides,* $A_{\mathcal{P}}$ *is the problematic set multiplier, where* $\mathcal{P} \in \{SS, SSS, nRSSS\}$. *Also note that* $\gamma = |\Gamma|$, $k = |K|$, $a = |A|$, *and* $s = |S|$.

Sketch of the proof. In order to be as general as possible, we have to take common properties of SS, SSS and nRSSS into consideration. The most important common property is clearly property P_1. As discussed, in the first step, we separate the neighbouring measurement nodes into two groups, namely Γ as the set of measurement nodes that have one connection to the problematic set and its complement $\bar{\Gamma}$.
We will, then, count the number of graphs with the desired property in which a measurement node m has one connection to the problematic set when $m \in \Gamma$. Using this unique separation, we can now take the summation of all of these enumerations to reach to the main goal.
At this point we enumerate connections of measurement nodes in Γ or $\bar{\Gamma}$ to S or A, separately.
Let us start with F which is the main core of the enumeration. The function which is responsible for the enumeration of connections between $\bar{\Gamma}$ and $S \cup A$ is $f(x,y)$. The main enumerator of this function is

$$(1+x+y)^{d_m}. \tag{3.7}$$

The coefficient of $x^i y^j$ in this function represents the number of connections that a measurement node can make to two different sets of different objects (informally two different sets of variable nodes) X and Y with cardinalities i and j. Note that the existence of 1 in the enumerator indicates that it is also possible that the measurement sockets of the measurement node can be empty and not connected to anything. In other words, it shows that the measurement node can have at most d_m connections and there is no requirement for them to have exactly d_m connections to X and Y. [2] Assume that X is the set of edges in S, and Y is the set of edges in A, that

[2] In the case when we have $(x+y)^{d_m}$, the measurement node is assumed to have exactly d_m connections.

are available to be connected to that measurement node. Available edges are those that have not been connected to any other nodes. Since we are considering $m \in \bar{\Gamma}$ then this measurement node should have at least two connections to X, moreover we are indifferent about its connection to Y. Based on this, we have to ensure that we deduct those terms from this function for which the measurement node has less than two connections to X. In other words, we have to make sure that there is no term in this function with degree of x being less than two. The following two functions are subtracted from the main enumerator to ensure this condition:

$$(1+y)^{d_m},$$

and

$$d_m(1+y)^{d_m-1}x.$$

The same issue is applicable to all of the measurement nodes in $\bar{\Gamma}$, and that is why $f(x,y)$ is appeared with the power of $k-\gamma$.
Furthermore, $g(y)$ is responsible for the connection of measurement nodes in Γ to A. Since all of the measurement nodes in Γ should have at least one connection to A, then using the same approach we can interpret $g(y)$, as well. Moreover, since there is no condition for the super measurement node, we only have the main enumerator for it as it is appeared in (3.7). Since all of these measurement nodes share the same set of X and Y as defined above, the final polynomial function will be their product, and the appropriate coefficient that we are looking for will be corresponding to $|X| = sd_v - \gamma$ and $|Y| = ad_v$. Recall that X was defined as the set of available edges in S which means that we already reserved those γ edges of S that should be connected to Γ.
The only part remains is the number of ways that we can reserve edges for being connected to Γ and S, i.e. $A_{\mathcal{P}}(\Gamma)$. Clearly, for each measurement node in Γ, we have d_m choices, this justifies the term d_m^{γ}, however, for the variable node side, we have to consider each problematic set, separately. This will be considered shortly, in the current section.
Note that after selection of appropriate connection to variable nodes, every permutation of the grouped variable edges is also allowed. This justifies the existence of the factorial terms in the main equation. □

$A_{\mathcal{P}}(\Gamma)$ **Computation:**

1. **SS**: In case of an SS, since each measurement node should have property P_1 on S then there should not exist any measurement node in Γ. Therefore, when $\Gamma \neq \emptyset$ the value of $A_{SS}(\Gamma)$ should be zero.

$$A_{SS}(\Gamma) = \begin{cases} 1 & \Gamma = \emptyset \\ 0 & \Gamma \neq \emptyset \end{cases}.$$

   We can also simplify the equations in Theorem 3.2.1, for this case, as follows:

$$|\mathcal{L}(S, S, A, S \cup A, K, d^*)| = \\ coef(f(x,y)(1+x+y)^{d^*}, x^{sd_v} y^{ad_v})(sd_v)!(ad_v)!. \tag{3.8}$$

2. **SSS**: There is no restriction on connection of measurement nodes in Γ to S in SSS. Thus, those measurement nodes can choose their connected edges among all of the sd_v edges available in S. Therefore, the number of ways that measurement nodes in Γ can be connected to S is:

$$A_{SSS}(\Gamma) = \binom{sd_v}{\gamma}.$$

3. **nRSSS**: In this case, the measurement nodes in Γ should be connected to S so that none of them has the same connection in S, otherwise $|[m]|$ will not be equal to 1 which contradicts the condition for property P_4. Therefore, there are $\binom{s}{\gamma}$ ways for measurement nodes to choose different variable nodes in S, and also there are d_v additional ways for each measurement node to choose between the edges emanating from its selected variable node. Thus,

$$A_{nRSSS}(\Gamma) = \binom{s}{\gamma}(d_v)^{\gamma}.$$

The reason why we cannot model the enumeration of binary-RSSS using this approach is that since the connection of measurement nodes, in Γ, in binary-RSSS is important to both A and S, then if two or more measurement nodes in Γ are connected to the same variable node in S, they should not be connected to another variable node

in A. As a result $g(y)$ will be affected. In fact, addition of binary-RSSS to the general method will make it so complicated, and it is beyond the scope of this thesis. Nevertheless, the analysis of performance of non-binary SBB will give us a good insight about the performance of its binary counter part.
In the rest of this chapter, we will consider some facts about these analyses, and we will try to simplify the equations by employing the connections between LDPC coding and CS and also by restricting the analysis to a specific group of ensembles.

3.3 A note on Genie vs LDPC decoding over BEC

The connection between LDPC decoding over BEC and Genie algorithm has been previously addressed in Chapter 2. This relation is basically inherited from the existence of stopping sets. Since the analysis of the LDPC decoding over BEC is completely done in [56], then we can use this analysis for the Genie, as well. In fact, the probability of failure of Genie in a given graph versus δ in Genie is exactly the same as the block error probability of a given code versus the erasure probability of the channel in LDPC decoding. In other words, being non-zero in Genie is equivalent to being erased in LDPC decoding over BEC using belief propagation decoder.
As an alternative to the complex method which is discussed here, we can use the simplified version of the analysis provided by [55] for cycle codes and completed by [56] for arbitrary d_v. We will not go through the details of the simplified expressions. However, for the case of the LM algorithm, we will consider some simplified expressions which are basically inherited from the simplified expressions in [55], but with a bit more complexity.
It is also worth noting that although the coding problem and CS problem are, to some extent, the same problems, there are also some fundamental differences between these two that requires the researchers to be extremely careful about using these two areas together. If we focus on the VB algorithms in CS, the most important difference is the probability of false verification. In CS, since we are only working on infinite fields, the probability of false verification is basically small enough to be neglected, however, in coding since we are concerned about finite fields, the probability of false verification will not be the same. It should also be pointed out that in case of Genie and LDPC decoding over BEC using belief propagation algorithm, there exists no difference.

3.4 Simplified Expressions for LM Analysis

The complexity of the analysis provided in the Section 3.1 does not allow us to find the probability of failure for large graphs. In fact, the algorithm provided for analysis is intractable. In this section, we will consider another approach for analysis of LM algorithm when $d_v = 2$ which makes the analysis simpler, however, we will show that for $d_v \geq 3$ even this simplified version of the algorithm is not efficiently tractable. Instead of considering the structures in which LM algorithm will fail, this time, we will enumerate structures in which LM will succeed. In other words, given the set of non-zero elements $\mathcal{K}$, we are interested to enumerate all the possible graphs that do not contain any SSS. The following theorem which is also inspired form [55], will enumerate the number of graphs without any SSS when $d_v = 2$.

Theorem 3.4.1. *In an ensemble $\mathcal{T}(V_l, V_r, 0)$ with $d_v = 2$, we have:*

$$|\mathcal{N}(\mathcal{K}, V_l, V_r, 0)| = \sum_{x=1}^{m} \sum_{y=0}^{x} \sum_{z=0}^{v-|\mathcal{K}|} T(|\mathcal{K}|, x, y, z),$$

where

$$T(v, u, h, s) = a_1 + a_2 + a_3 + a_4,$$

where a_1, a_2, a_3 and a_4 are defined in Table 3.3. with initial condition as:

$$T(0, x, y, z) = \begin{cases} 1 & x = y = z = 0 \\ 0 & everywhere\ else \end{cases}.$$

Sketch of the proof. The main approach in this theorem is to count the number of SSS-free structures in an ensemble $\mathcal{T}(V_l, V_r, 0)$. For simplicity let (α, β, γ) SSS-free structure denote an SSS-free structure with α non-zero variable nodes, β zero variable nodes and γ measurement nodes. Also, assume that the set of non-zero variable nodes, zero variable nodes and measurement nodes are indexed from 1 to $|\mathcal{K}|$, 1 to $|V_l \setminus \mathcal{K}|$ and from 1 to $|V_r|$, respectively. Clearly, an (α, β, γ) SSS-free structure, as a graph, belongs to $\mathcal{N}([\alpha], [\alpha + \beta], [\gamma], 0)$. Also, an (α, β, γ) SSS-free structure has a property that allows us to construct it using another $(\alpha', \beta', \gamma')$ SSS-free structure where $\alpha' = \alpha - 1$ and $\gamma' < \gamma$.

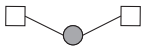


Figure 3.4: The basic SSS-free structure

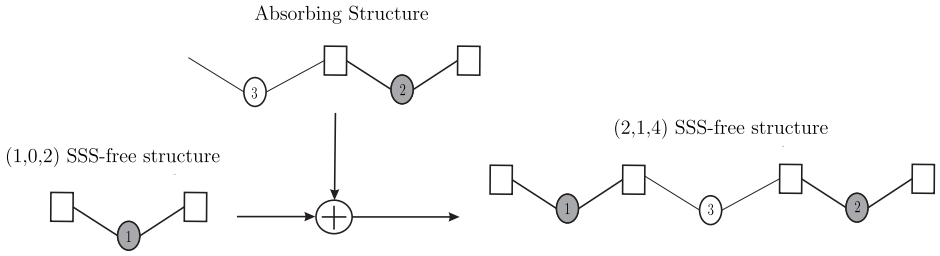


Figure 3.5: Concatenation procedure for constructing larger SSS-free structures. The shaded nodes are assumed to be non-zero variable nodes.

By starting from the smallest possible SSS-free structures, we can construct all of the SSS-free Structures in $\mathcal{N}(\mathcal{K}, V_l, V_r, 0)$. Let us start from the most basic non-trivial SSS-free structures. Clearly a graph with no nodes can be considered as an SSS-free structure. Suppose that we want to count the number of graphs that has 1 non-zero variable node and 2 measurement nodes, i.e. $(1, 0, 2)$ SSS-free structures. Figure 3.4 shows the only possible structure of this kind.

Using a simple calculation, we realize that there are $2d_m^2$ $(1, 0, 2)$ SSS-free structures in $\mathcal{N}([1], [1], [2], 0)$. Starting from each of these SSS-free structures one can construct SSS-free structures in $\mathcal{N}([2], [\beta'], [\gamma'], 0)$, where $\gamma' > 2$. The procedure of this construction includes the concatenation of some structures, called absorbing structures, to the smaller SSS-free structures. An example of this concatenation is shown in Figure 3.5. Furthermore, some of the absorbing structures are also shown in Figure 3.6. As can be seen, addition of all of these absorbing structures to the available and appropriate measurement node sockets of Figure 3.4 will construct different SSS-free structures with larger α. If we continue this procedure, and enumerate every possible

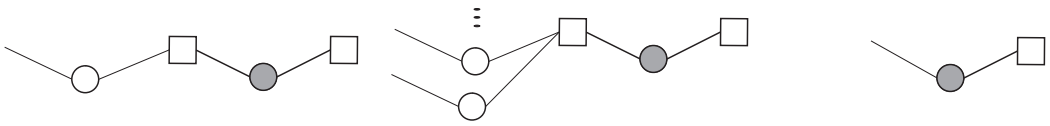


Figure 3.6: Some absorbing structures. The shaded nodes are assumed to be non-zero variable nodes.

Table 3.3: Recursions required for Theorem 3.4.1.

$a_1 = (ud_m + (1-d_m)s - 2v + 2 - 1 - hd_v)uvd_m d_v! T(v-1, u-1, h, s-1)$
$a_2 = uvsd_m(d_m-1)d_v!\, T(v-1, u-1, h, s)$
$a_3 = 2d_m^2 v\binom{u}{2} d_v!\, T(v-1, u-2, h, s-2)$
$a_4 = 2vr^2\binom{u}{2}\sum_{i=1}^{d_m-1}\sum_{j=1-i}^{1}\binom{h}{i}(d_v!)^{i+1}(i)! p_{ij} T(v-1, u-2, h-i, s-j)$
$P_{ij} = \sum_{q=0}^{i}\binom{d_m-1}{d_v q}\binom{d_m-1-d_v q}{i-q} W(i,j,q)$
$W(i,j,q) = \sum_{t=0}^{1-j}(-1)^t\binom{1-j}{t}\binom{Q}{i-q}$
$Q = (u-2)d_m - (s-j)(d_m-1) + (1-j-t)(d_m-1) - (v-1)d_v - (h-i)d_v$

structure, carefully, we will end up with the number of possible SSS-free structures in $\mathcal{N}(\mathcal{K}, V_l, V_r, 0)$. The recursions in this theorem do this careful enumeration. basically, the expressions a_1, a_2, a_3 and a_4 try to enumerate different SSS-free structures in different cases, for instance, a_1 enumerate the number of SSS-free structures that can be constructed from SSS-free structures with one less number of variable nodes and one less number of measurement nodes. As another example, a_4 enumerates the SSS-free structures constructed from SSS-free structures with two less measurement nodes and some other features. □

Corollary. *From Theorem 3.4.1, we can find the average probability of failure in a given ensemble with $d_v = 2$ as follows:*

$$\mathbf{E}_{\mathcal{G}(n,d_v,d_m)}\{P_f(G|\mathcal{K})\} = 1 - \frac{|\mathcal{N}(\mathcal{K}, V_l, V_r, 0)|}{|\mathcal{T}(V_l, V_r, 0)|}.$$

It is worth mentioning that the recursive construction of SSS-free structures is inherited from the simplified analysis method provided for LDPC coding over BEC using belief propagation algorithm as the decoder. Therefore, one more time, we can see that the nature of these problematic sets are all the same.

The reason behind the fact that enumeration of the SSS-free structures with $d_v \geq 3$ is so complex is that the number of absorbing structures in these cases will be dramatically increased, so that one cannot efficiently extract mathematical expressions or computational algorithms to enumerate all of those structures. Some of the absorbing structures in $d_v = 3$ are shown in Figure 3.7.

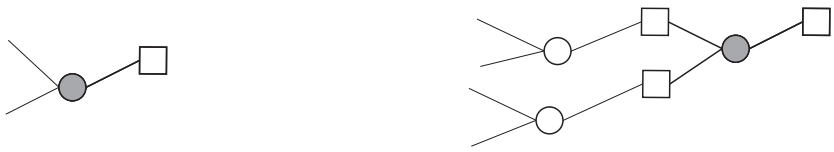


Figure 3.7: Some absorbing structures for $d_v = 3$. The shaded nodes are assumed to be non-zero variable nodes.

Chapter 4

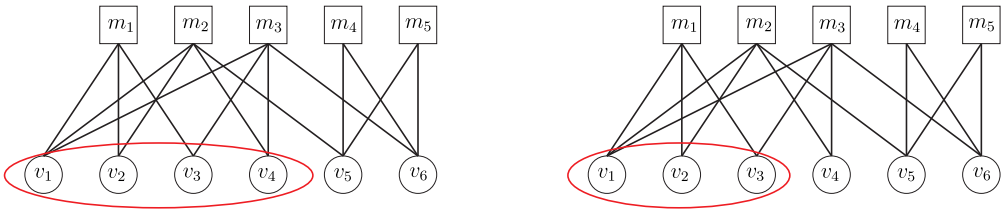
Individual Graph Analysis

Based on the characterization presented for VB MPAs in Chapter 2, we know that there are certain combinatorial structures that cause failure in these algorithms. This issue will help us to completely analyze the failure patterns in a given graph. In other words, we are capable of evaluating the probability of failure of a given graph in a given VB algorithm by knowing the collection of all of its problematic sets.

Some useful applications of this combinatorial characterization can be inherited from the coding theory. In LDPC codes, so far, we are able to analyze the error patterns of codes for BEC using the combinatorial characterization that has been introduced in [55]. Using this characterization, one can find many properties of the underlying graph such, as error floor, in LDPC codes over BEC using belief propagation decoder [64]. Likewise, In VB algorithms in CS, we are now able to analytically evaluate the errors in a given graph, thanks to the characterization of SSSs, RSSSs, and nRSSSs. Moreover, we do not have to be concerned about the fundamental problems that arise for trapping sets in coding.[1] We will tackle the problem of finite length analysis of a given graph by considering a new technique which is based on finding some certain SSs rather than focusing on every problematic set itself. This approach will help us alleviate the complexity involved in computing problematic sets and also benefit from the previous works on finding SSs [65–68]. Thanks to the relation between SSs and other problematic sets discussed in this thesis, we will be able to analytically find the probability of error of a given graph and VB MPA in CS.

The probability of failure of an individual graph is usually more complicated than finding the average probability of error in a given ensemble, and as a result it is

[1] Given the fact that the characterization of dominant trapping sets for a given decoding algorithm and a given graph is not in general available, one would not be able to accurately estimate the error floor with confidence or design codes with provably low error floors.



(a) The set $\{v_1, v_2, v_3, v_4\}$ is an LMSS as there is no larger SS over $\{m_1, m_2, m_3\}$.

(b) The set $\{v_1, v_2, v_3\}$ is not an LMSS, since $\{v_1, v_2, v_3, v_4\}$ is a larger SS over the same set of measurement nodes.

Figure 4.1: Example of an LMSS and an SS which is not LMSS. Note that the neighbouring measurement node of both sets are the measurement nodes in $\{m_1, m_2, m_3\}$

sometimes avoided in the literature. Nevertheless, both of these analyses give us different insight about the problem. The average graph analysis usually provides us with the general performance of a code within an ensemble. This result cannot be employed in error floor region. Therefore, the individual graph analysis will gives us the insight on the performance of the algorithm and graph in the error floor, and also the contribution of problematic sets in the error of a graph. The latter will be much useful in the design procedure in the sense that one can use the information acquired from the analysis to avoid harmful structures inside the sensing graph to improve the performance of the graph.

4.1 Problematic Set Analysis

Definition 4.1.1 (**Locally Maximal Stopping Set (LMSS)**)**.** We say that a stopping set S is locally maximal or LMSS if S is the maximal possible SS on $\mathcal{N}(S)$. In other words, the maximum possible SS which has all of its neighbours inside $\mathcal{N}(S)$ is called LMSS. Figure 4.1 shows two different SSs in a graph, where one of them, i.e. $\{v_1, v_2, v_3, v_4\}$ is an LMSS (Figure 4.1a) and the other, i.e. $\{v_1, v_2, v_3\}$ is not (Figure 4.1b).

From now on, for the sake of simplicity and generality of expressions, we use the notation $\mathcal{P}$ to indicate one type of the problematic sets considered in this work, and this $\mathcal{P}$ can be replaced by appropriate set types such as SS, SSS, RSSS, and nRSSS, whenever it is necessary. Moreover, we may also use the same notation to indicate

the algorithm associated with each of these problematic sets, e.g. $\mathcal{P}$ can refer to a set type such as SSS, or its corresponding algorithm which is LM, wherever it is appropriate.

Definition 4.1.2. For an LMSS S:
A problematic set $S' \subseteq S$ is called a child problematic set of S if $\mathcal{N}(S') = \mathcal{N}(S)$. That is:

1. An SS $S_1 \subseteq S$ is called a child SS of S if $\mathcal{N}(S_1) = \mathcal{N}(S)$.
2. An SSS $S_2 \subseteq S$ is called a child SSS of S if $\mathcal{N}(S_2) = \mathcal{N}(S)$.
3. An RSSS $S_3 \subseteq S$ is called a child RSSS of S if $\mathcal{N}(S_3) = \mathcal{N}(S)$.
4. An nRSSS $S_4 \subseteq S$ is called a child nRSSS of S if $\mathcal{N}(S_4) = \mathcal{N}(S)$.

Accordingly, S is also called the mother of those problematic sets, as well. For a problematic set type $\mathcal{P}$, we also refer to $\mathcal{C}_{\mathcal{P}}(S)$ as the set of all type-$\mathcal{P}$ children of S.

Lemma 4.1.1. *For any problematic set S of type $\mathcal{P}$, $S \cup H(S)$ is an LMSS, where $H(S)$ is the $H-$set of S, which has been defined in Definition 2.0.8.*

Proof. First, we have to show that $S \cup H(S)$ is an SS. In order to prove that, we have to show that every measurement node in the neighbourhood of $\mathcal{N}(S \cup H(S))$ has at least two connection to $S \cup H(S)$. According to Definition 2.0.8, it is obvious that $\mathcal{N}(S \cup H(S)) = \mathcal{N}(S)$. Therefore, it is enough to show that every measurement node in the neighbourhood of S has at least two connection to $S \cup H(S)$. For any measurement node $m \in \mathcal{N}(S)$, there are two possibilities:

- $|\mathcal{N}(m) \cap \mathcal{N}(S)| \geq 2$: In this case, m definitely has two connection to $S \cup H(S)$.
- $|\mathcal{N}(m) \cap \mathcal{N}(S)| = 1$: This case happens only when $\mathcal{P} \in \{SSS, RSSS, nRSSS\}$. In this case, in order for S to be a problematic set, m should be connected to one variable node $v' \mathcal{N}(m) \setminus S$. The variable node v' in SSS and nRSSS, is proven to belong to $H(S)$, and also for the case of RSSS, it is shown that $v' \in R(S)$ in Chapter 2. It is also obvious from the definition of $R-$set and $H-$set that $R(S) \subseteq H(S)$. As a result, v' in any of the possible problematic set types in this case belongs to $H(S)$. Furthermore, from the definition, we can also conclude that $H(S) \cap S = \emptyset$. Therefore, m has one connection to S and the other one to $H(S)$, which means that m has at least two connections to $S \cup H(S)$.

All that remains is to prove that $S \cup H(S)$ is also an LMSS. Suppose that it is not an LMSS, which means:

$$\exists\, v \notin S \cup H(S) :\ \mathcal{N}(v) \subseteq \mathcal{N}(S).$$

Since $\mathcal{N}(v) \subseteq \mathcal{N}(S)$, then $v \in \mathcal{N}(\mathcal{N}(S))$. Based on Definition 2.0.8, v either belongs to S or $H(S)$, which is a contradiction. □

Lemma 4.1.2. *In LM and non-binary SBB algorithms the final variable nodes on which these algorithms fail are all LMSS.*

Proof. From Chapter 2, we know that these two algorithms always fail in an $S \cup H(S)$, which is an LMSS from Lemma 4.1.1. □

Remark 3. Note that the converse of this lemma is not true, as any LMSS will fail only if one of its children is non-zero.

Lemma 4.1.3. *Suppose that S_1 and S_2 are two different LMSSs of a given graph. Then for any problematic set type $\mathcal{P}$, we have:*

$$\mathcal{C}_{\mathcal{P}}(S_1) \cap \mathcal{C}_{\mathcal{P}}(S_2) = \emptyset.$$

Proof. Since $S_1 \neq S_2$, then $\mathcal{N}(S_1) \neq \mathcal{N}(S_2)$. On the other hand, from Definition 4.1.2, we know that for every problematic set $S' \in \mathcal{C}_{\mathcal{P}}(S)$, we have: $\mathcal{N}(S') = \mathcal{N}(S)$. Therefore, every problematic sets in $\mathcal{C}_{\mathcal{P}}(S_1)$ has different neighbouring measurement nodes than those in $\mathcal{C}_{\mathcal{P}}(S_2)$, and consequently, they are different problematic sets, and there is no intersection between $\mathcal{C}_{\mathcal{P}}(S_1)$ and $\mathcal{C}_{\mathcal{P}}(S_2)$. □

Remark 4. Note that this Lemma 4.1.3, does not meant that there are no common variable node between different LMSSs and their non-zero failure children.

Lemma 4.1.4. *The set of all problematic sets of type $\mathcal{P}$ of a graph is given by:*

$$\mathcal{P}(G) = \dot{\bigcup}_{S \in \mathcal{LMSS}} \mathcal{C}_{\mathcal{P}}(S),$$

where $\mathcal{LMSS}$ is the set of all LMSSs of the graph.

Proof. Since for all $S \in \mathcal{LMSS}$ every elements of $\mathcal{C}_{\mathcal{P}}(S)$ is a problematic set of type $\mathcal{P}$ in G then we have:

$$\dot{\bigcup}_{S \in \mathcal{LMSS}} \mathcal{C}_{\mathcal{P}}(S) \subseteq \mathcal{P}(G).$$

Furthermore, clearly, for every problematic set $S' \in \mathcal{P}(G)$, there exists an LMSS $\tilde{S} = S \cup H(S)$ in G, where $S' \in \mathcal{C}_{\mathcal{P}}(\tilde{S})$. Therefore,

$$\mathcal{P}(G) \subseteq \dot{\bigcup}_{S \in \mathcal{LMSS}} \mathcal{C}_{\mathcal{P}}(S).$$

Moreover, from Lemma 4.1.3, we know that there is no intersection between any pair of $\mathcal{C}_{\mathcal{P}}(S_1)$ and $\mathcal{C}_{\mathcal{P}}(S_2)$, which justifies the disjoint union. Thus the proof is complete. □

In the following theorem, we will show how harmful an LMSS is for a given graph. Then, this harmfulness will be the only measure that we will use to find the exact probability of failure for a given algorithm. Moreover, defining the harmfulness in terms of LMSS rather than the problematic sets themselves will let us to compare the harmfulness of a given LMSS for different algorithms and to see how much contribution a specific LMSS might have in those algorithms. This will, eventually, lead us to design appropriate graphs for different algorithms. Furthermore, based on this analysis it can be shown that sometimes the amount of improvement that one can gain from using one algorithm instead of another is not high enough in a given graph, and thus it is recommended to use the one with the lower computational complexity. Also, one can measure the amount of improvement for a given algorithm by removing a specific LMSS from a given graph.

Definition 4.1.3. Failure of an LMSS is defined as the situation in which all of the measurement nodes in the LMSS have, eventually, a non-zero value in a given graph.

To bridge this definition to the failure of variable nodes, consider that an algorithm stalls in a graph, then there will be some non-zero measurement nodes in the graph. In this case, it will be shown that at least one of the children of the LMSS corresponding to the set of non-zero measurement nodes will remain unverified in the graph. It will also be shown that failure of an LMSS, the way it is defined in Definition 4.1.3, is equivalent to have at least one of the children of that LMSS being non-zero.
Also, note that failure of an LMSS may result in the failure of other LMSSs, as well. For example, consider the set of all variable nodes in a given graph. This set is, clearly, an LMSS. As it is expected, the failure of this LMSS will result in the failure of every other LMSS in the graph.

Lemma 4.1.5. *In a given graph G, with the set of non-zero variable nodes $\mathcal{K} \subseteq V_l$, for a given LMSS S, every variable node in a type-$\mathcal{P}$ children $S' \in \mathcal{C}_{\mathcal{P}}(S)$ will remain eventually unverified in algorithm $\mathcal{P}$ if $S' \subseteq \mathcal{K}$.*

Proof. From Chapter 2, we know that in algorithm $\mathcal{P}$, the unique maximal problematic set inside $\mathcal{K}$ will remain eventually unverified. Suppose that $S' \subseteq \mathcal{K}$, and S_m is the unique maximal problematic set in $\mathcal{K}$. Since the union of every problematic set is proven to be another problematic set in Chapter 2, then $S' \subseteq S_m$ since otherwise, $S' \cup S_m$ would be another larger set inside $\mathcal{K}$, which contradicts the maximality of S_m. As a result, every variable node in S' will remain eventually unverified, and the proof is complete. □

Theorem 4.1.6 (Harmfulness of an LMSS)**.** *In a given graph G, the probability that an LMSS S fails in algorithm $\mathcal{P}$ is:*

$$P^{\mathcal{P}}\{S \ fails\} = \sum_{S' \in \mathcal{C}_{\mathcal{P}}(S)} \delta^{|S'|}(1-\delta)^{|S|-|S'|}.$$

Proof. From Definition 4.1.3, we have to find the probability that all of the measurement nodes in $\mathcal{N}(S)$ have eventually a non-zero value in algorithm $\mathcal{P}$. We know that

$$\sum_{S' \in \mathcal{C}_{\mathcal{P}}(S)} \delta^{|S'|}(1-\delta)^{|S|-|S'|},$$

is the probability that only one of the type-$\mathcal{P}$ children of S becomes non-zero, while all the other variable nodes in S are zero. If at least one of the type-$\mathcal{P}$ children of S, say S', is non-zero then from Lemma 4.1.5, it will remain eventually unverified in algorithm $\mathcal{P}$. Furthermore, since $\mathcal{N}(S') = \mathcal{N}(S)$, then every measurement node in S will also remain non-zero in algorithm $\mathcal{P}$. Therefore, we have:

$$\sum_{S' \in \mathcal{C}_{\mathcal{P}}(S)} \delta^{|S'|}(1-\delta)^{|S|-|S'|} \leq P^{\mathcal{P}}\{S \ fails\}.$$

Moreover, assume that there is a situation in which all of the measurement nodes in $\mathcal{N}(S)$ are eventually non-zero and at the same time we have a non-zero $S'' \subseteq S$ that

is not a type-$\mathcal{P}$ child of S and also has the following property:[2]

$$\mathcal{N}(S'') = \mathcal{N}(S).$$

Since $S'' \notin \mathcal{C}_{\mathcal{P}}(S)$, then it cannot be a problematic set. As a result, as the iterations go on one or some of the variable nodes in S'' will be verified since otherwise S' would be considered as a problematic set. Suppose that $Q \subseteq S''$ is the set of variable nodes that becomes eventually verified. We know that $S'' \setminus Q$, as the set of variable nodes which remain eventually unverified, is a problematic set. Then there are two cases:

- $\mathcal{N}(S'' \setminus Q) = \mathcal{N}(S)$: In this case clearly, $S'' \setminus Q \in \mathcal{C}_{\mathcal{P}}(S)$. From a reasoning the same as the one that will be introduced in Lemma 4.1.7 for a more general case, we can conclude that S'' it self is a type-$\mathcal{P}$ child of S, which is a contradiction.[3].
- $\mathcal{N}(S'' \setminus Q) \subset \mathcal{N}(S)$: In this case, those measurement nodes in $\mathcal{N}(S)$ that are only neighbours of Q will become zero, since there will be no unverified non-zero variable node in their neighbourhood, and this contradicts the fact that all of the neighbouring measurement nodes of $\mathcal{N}(S)$ will remain eventually non-zero.

Therefore, we will not have such a situation, and thus:

$$P^{\mathcal{P}}\{S \ fails\} \leq \sum_{S' \in \mathcal{C}_{\mathcal{P}}(S)} \delta^{|S'|}(1-\delta)^{|S|-|S'|}.$$

Therefore, the proof is complete. □

In any of VB algorithms, the variable nodes in the graph can only be verified by the means of their connected measurement nodes. In fact, in verification process these are measurement nodes that set a neighbouring variable node to be verified or unverified. In this part, in order to be able to complete the exact analysis, we need to define a new set of measurement nodes in a graph that are not able to verify a variable node regardless of the algorithm itself.

Remark 5 (disabled measurement nodes). From now on, we partition the set of measurement nodes in a graph into two groups. The set of disabled measurement

[2]Note that the cases in which two or more children of S are non-zero and all other variable nodes are zero are, in fact, the cases where only one child of S is non-zero and other variable nodes are zero, because the the union of every two problematic set is proven to be another problematic set.

[3]A more general case is proved in Lemma 4.1.8. Please refer to that proof for further details.

nodes denoted by $\tilde{M}$, and the set of regular measurement nodes, denoted by M, i.e. $V_r = M \cup \tilde{M}$. A disabled measurement node cannot verify any of the variable nodes in the graph during the algorithm.

As an example, in LM algorithm, we know that if a measurement node's degree is one, then this measurement node can verify its single connected variable node. However, if this measurement node is disabled, the connected variable node will remain unverified until some other regular measurement node can verify it, otherwise it will remain unverified until the end of the algorithm.
This labelling of measurement nodes is useful for finding the probability of the failure of a given graph. It will be shown that without having this labelling, one would not be able to find the recursive expressions needed for the analysis. Although at the end, we will assume that the set of disabled measurement nodes of a real graph is the empty set, we still need this labelling, since in any recursion of the analysis, there will be some measurement nodes added to the set of disabled measurement nodes of the graph in the previous recursion. Therefore, to keep the generality of the formulation, we need to consider a graph with a given set of measurement nodes as the set of disabled measurement nodes, which is not necessarily the empty set.
This recursive analysis of the probability of error of a given graph is interesting in the sense that although the nature of the two problems of average ensemble performance and individual graph performance are different, the approach that we can take to solve both of these problems are, to some extent, the same. In each recursion, we need to define a set of disabled measurement nodes to avoid over-counting of some problematic sets which is the role that the super measurement node plays in the average performance analysis in Chapter 3. In each recursion, the set of the measurement nodes in an LMSS will become the set of disabled measurement nodes in the next recursion, and the size of the underlying graph will be reduced. This procedure continues until we encounter a graph with no nodes. In such case, the empty graph does not have any problematic set and as a result the recursion terminates.

Definition 4.1.4 (Generalized LMSS)**.** An SS S is called a GLMSS if it is the maximal possible SS on $\mathcal{N}(S) \cup \tilde{M}$ in a graph, where $\tilde{M}$ is the set of disabled measurement nodes in the graph. Note that the measurement nodes in $\tilde{M}$ are always assumed to have all of the properties P_1 to P_4 on any set of variable nodes, since they have no impact on verification of any variable node.

Remark 6. Note that in general, LMSS is only defined over graphs with no disabled measurement nodes. On the other hand, GLMSS is defined on graphs with and without disabled measurement nodes. That is why they are called *Genralized* LMSS. In the case that a graph has an empty set as the set of disabled measurement nodes, both GLMSS and LMSS are the same.

Definition 4.1.5. For a GLMSS S in graph G, with $\tilde{M}$ as its set of disabled measurement nodes, a type-$\mathcal{P}$ problematic set $S' \subseteq S$ is said to be a child of S if $\mathcal{N}(S) \setminus \tilde{M} \subseteq \mathcal{N}(S')$. Moreover, $\mathcal{C}_{\mathcal{P}}(S|\tilde{M})$ denotes the collection of all those children sets.

Let $P_s^{\mathcal{P}}\{V_l(G)\}$ denote the probability that every variable node in $V_l(G)$ is verified after enough number of iterations in algorithm $\mathcal{P}$ over graph G, where $V_l(G)$ is defined as the set of variable nodes of G. Also, for a graph G and a set of measurement nodes $\tilde{M}$ in G, the notation $G^{\tilde{M}}$ is used to denote the graph G in which all the measurement nodes in $\tilde{M}$ are disabled. Note that in general $\tilde{M}$ is not necessarily required to be a subset of measurement nodes of G. In other words, if we run a VB algorithm over $G^{\tilde{M}}$ none of the measurement nodes in $\tilde{M}$ can contribute in the verification process during the algorithm.

Lemma 4.1.7. *In a GLMSS S with the set of disabled measurement nodes $\tilde{M}$, suppose that $S' \in \mathcal{C}_{\mathcal{P}}(S|\tilde{M})$. Therefore, we have:*

$$\forall Q \subseteq S \setminus S',\ S' \cup Q \in \mathcal{C}_{\mathcal{P}}(S|\tilde{M}).$$

Proof. We have to show that $\mathcal{N}(S) \setminus \tilde{M} \subseteq \mathcal{N}(S' \cup Q)$, and that every measurement node in the neighbourhood of $Q \cup S'$ satisfies the appropriate properties. We know that

$$\mathcal{N}(S) \setminus \tilde{M} \subseteq \mathcal{N}(S') \subseteq \mathcal{N}(S),$$

which means that

$$\mathcal{N}(S) \setminus \tilde{M} \subseteq \mathcal{N}(S' \cup Q) \subseteq \mathcal{N}(S).$$

□

Therefore, any measurement node $m \in \mathcal{N}(S' \cup Q)$ can be considered in one of the following situations:

- $m \in \mathcal{N}(S')$: In this case, we know that m satisfies the appropriate properties on S'. If m satisfies property P_1 on S' it definitely satisfies property P_1 on $S' \cup Q$, as well. If m satisfies another property on S', then it follows that it has only one connection to S', and it should also have at least another connection to the associated set of S'. In such a case, either Q is part of the associated set to which m is connected or not. In the former case, m' will have at least two connections to $S' \cup Q$, one to S' and the other ones to Q, i.e. it has property P_1 on $S' \cup Q$, in the latter case, m will preserve the property that it had on S', on $S' \cup Q$, as well. Thus, in all of the cases, m will have the appropriate property on $S' \cup Q$.

- $m \in \mathcal{N}(S' \cup Q) \setminus \mathcal{N}(S')$: In this case, we prove that $m \in \tilde{M}$.

  suppose that $m \notin \tilde{M}$. We know that $m \in \mathcal{N}(S)$, and also in this specific case, we know that $m \notin \mathcal{N}(S')$. Therefore, m is, in fact, a measurement node that belongs to $\mathcal{N}(S)$, but not to $\mathcal{N}(S')$. Moreover, since $m \notin \tilde{M}$, it belongs to $\mathcal{N}(S) \setminus \tilde{M}$. Also, Since S' is a child problematic set, we have:

  $$\mathcal{N}(S) \setminus \tilde{M} \subseteq \mathcal{N}(S'),$$

  which means that m also belongs to $\mathcal{N}(S')$. This is in fact, a contradiction. And therefore, in this case $m \in \tilde{M}$. As a result, m satisfies all of the properties on every set of variable nodes including $S' \cup Q$.

Thus, in all of the cases we have shown that m satisfies the appropriate properties on $S' \cup Q$, which means that $S' \cup Q$ is also a type-$\mathcal{P}$ child of S.

Definition 4.1.6. Failure of a GLMSS S with the set of disabled measurement nodes $\tilde{M}$ is defined as the situation in which all the measurement nodes inside $\mathcal{N}(S) \setminus \tilde{M}$ remain eventually non-zero in a VB MPA.

Lemma 4.1.8. *The probability that a given GLMSS S fails in a graph $G^{\tilde{M}}$ is:*

$$P^{\mathcal{P}}\{S \ fails \ in \ G^{\tilde{M}}\} = \sum_{S' \in \mathcal{C}_{\mathcal{P}}(S|\tilde{M})} \delta^{|S'|}(1-\delta)^{|S|-|S'|}.$$

Proof. We know that a GLMSS S may fail if all the measurement nodes inside $\mathcal{N}(S) \setminus \tilde{M}$ remain eventually non-zero. Let us define this event as event A. We also know that

the following expression is the probability that only one type-$\mathcal{P}$ child of S become non-zero while all the other variable nodes of S are zero. Therefore, we have to show that event A happens, if and only if only one of type-$\mathcal{P}$ children of S become non-zero while other variable nodes in S are zero.

Now, assume that $S' \in \mathcal{C}_{\mathcal{P}}(S|\tilde{M})$, is non-zero. Since S' is a problematic set itself then it will not be verified throughout the corresponding VB MPA. As a result all of the measurement nodes in $\mathcal{N}(S')$ will remain eventually non-zero. That is, event A happens. Thus, we conclude that having S' as one type-$\mathcal{P}$ child of S non-zero, event A will certainly happen. Therefore,

$$P^{\mathcal{P}}\{S \ fails \ in \ G^{\tilde{M}}\} \geq \sum_{S' \in \mathcal{C}_{\mathcal{P}}(S|\tilde{M})} \delta^{|S'|}(1-\delta)^{|S|-|S'|}.$$

Now, we have to show that the reverse is also true. We will show this using contradiction. Suppose that event A has happened, and $S' \subseteq S$, as the set of non-zero elements of S, is not a type-$\mathcal{P}$ child of S.[4] Since event A has happened, then all measurement nodes in $\mathcal{N}(S) \setminus \tilde{M}$ remain eventually non-zero. In order to have all those measurement nodes non-zero, we should have $\mathcal{N}(S) \setminus \tilde{M} \subseteq \mathcal{N}(S')$. As a result, S' cannot be a type-$\mathcal{P}$ problematic set since otherwise, it would be a type-$\mathcal{P}$ child of S, which is not the case. Now, since S' is not a problematic set, there exists at least one non-empty subset of S' that should be verified throughout the algorithm. Let us call this subset Q. We have two cases:

- $\mathcal{N}(S) \setminus \tilde{M} \subseteq \mathcal{N}(S' \setminus Q)$: In this case, since $S' \cup Q$ is the set in which the algorithm stalls, then we can conclude that it is a problematic set, and as a result $S' \setminus Q$ is a type-$\mathcal{P}$ child of S. From Lemma 4.1.7, we know that $(S' \setminus Q) \cup Q$ is also a type-$\mathcal{P}$ child of S, i.e. S' is a type-$\mathcal{P}$ child of S which is a contradiction.
- $\mathcal{N}(S) \setminus \tilde{M} \nsubseteq \mathcal{N}(S' \setminus Q)$: In this case,

$$\exists m \in (\mathcal{N}(S) \setminus \tilde{M}) \setminus (\mathcal{N}(S' \setminus Q)).$$

  That is,

$$\exists m \in \mathcal{N}(S) \setminus (\tilde{M} \cup \mathcal{N}(S' \setminus Q)).$$

[4] Note that the case in which two or more type-$\mathcal{P}$ children of S are non-zero is, in fact, among the situations in which only one type-$\mathcal{P}$ child of S is non-zero. This is because that any union of problematic sets is another problematic set, and as a result, any union of type-$\mathcal{P}$ children of a GLMSS is also a type-$\mathcal{P}$ child of the same GLMSS.

Therefore, we can conclude that m will get zero value throughout the iterations of the algorithm. This is in contradiction with occurrence of event A.

Thus, in all of the cases we reach to a contradiction, and as a result, we can conclude that occurrence of event A guarantees that a type-$\mathcal{P}$ child of S being non-zero while other variable nodes are all zero. Therefore, we have:

$$P^{\mathcal{P}}\{S \ fails \ in \ G^{\tilde{M}}\} \leq \sum_{S' \in \mathcal{C}_{\mathcal{P}}(S|\tilde{M})} \delta^{|S'|}(1-\delta)^{|S|-|S'|}.$$

In overall, we conclude that

$$P^{\mathcal{P}}\{S \ fails \ in \ G^{\tilde{M}}\} = \sum_{S' \in \mathcal{C}_{\mathcal{P}}(S|\tilde{M})} \delta^{|S'|}(1-\delta)^{|S|-|S'|}.$$

□

Lemma 4.1.9. *For a given GLMSS S in a given graph $G^{\tilde{M}}$, we have:*

$$P_s^{\mathcal{P}}\{V_l(G^{\tilde{M}}) \setminus S| \ S \ fails\} = P_s^{\mathcal{P}}\{V_l((G \restriction_{\bar{S}})^{\tilde{M} \cup \mathcal{N}(S)})\},$$

where $\bar{S} = V_l \setminus S$, and $\mathcal{P} \in \{LM, nSBB, Genie\}$.

Proof. Generally speaking, we should show that when S fails in the graph, then the probability of the success of the rest of the graph can be achieved by considering the effects of S on the sub-graph induced by $\bar{S}$. Those effects will completely be reflected in the neighbourhood of S. Therefore, in the first step, we have to show that no measurement node in $\mathcal{N}(S)$ is able to verify any variable nodes in $V_l(G^{\tilde{M}}) \setminus S$, given that S is failed. Since by the definition $\tilde{M}$ is the set of disabled measurement nodes, then we should only be concerned with $\mathcal{N}(S) \setminus \tilde{M}$. We have to show that the following rules cannot be applied to variable nodes by measurement nodes in $\mathcal{N}(S) \setminus \tilde{M}$.

1. **ZMN**: Since S fails, then we know that every measurement node connected to S has a non-zero value. ZMN rule is thus not applicable.

2. **D1MN**: Since S is a GLMSS it is also an SS, which means that every regular [5] measurement node in $\mathcal{N}(S)$ should have at least two connections to S, and also because S is not verified in algorithm $\mathcal{P}$, then every regular measurement node

[5] Recall that regular measurement nodes are the ones that are not disabled.

in $\mathcal{N}(S)$ will have at least two unverified variable nodes in its neighbourhood, i.e. D1MN cannot be applied.

3. **EIM**: For EIM to be applied, there should exists a single variable node $v' \in V_l(G^{\tilde{M}}) \setminus S$ which is connected to at least two measurement nodes whose values are not affected by other variable nodes. In other words, the value of those measurement nodes, excluding the value coming from v', should be zero. Since any regular measurement node in $\mathcal{N}(S)$ is already non-zero by variable nodes in S, this situation cannot happen for those measurement nodes. Thus, regular measurement nodes in $\mathcal{N}(S)$ cannot verify any variable node by EIM.

So far, we have proven that regular measurement nodes in $\mathcal{N}(S)$ cannot verify any variable node in $V_l(G^{\tilde{M}}) \setminus S$. As a result, we can add those measurement nodes to the set of disabled measurement nodes in the graph. That is:

$$P_s^{\mathcal{P}}\{V_l(G^{\tilde{M}}) \setminus S |\ S\ fails\} = P_s^{\mathcal{P}}\{V_l(G^{\tilde{M}\cup\mathcal{N}(S)}) \setminus S |\ S\ fails\}.$$

At this point, since every measurement node in the neighbourhood of S is disabled, then no measurement node can verify any variable node in S, and as a result, S will certainly remain unverified; thus, the conditional term in the probability is not necessary, and we have:

$$P_s^{\mathcal{P}}\{V_l(G^{\tilde{M}\cup\mathcal{N}(S)}) \setminus S |\ S\ fails\} = P_s^{\mathcal{P}}\{V_l(G^{\tilde{M}\cup\mathcal{N}(S)}) \setminus S\}.$$

Furthermore, since S does not affect this probability any more, we can simply remove it from $G^{\tilde{M}\cup\mathcal{N}(S)}$, that is:

$$P_s^{\mathcal{P}}\{V_l(G^{\tilde{M}\cup\mathcal{N}(S)}) \setminus S\} = P_s^{\mathcal{P}}\{V_l((G\restriction_{\bar{S}})^{\tilde{M}\cup\mathcal{N}(S)})\}.$$

Thus, the proof is complete. □

Theorem 4.1.10. *In a given graph G, with $\tilde{M}$ as its set of disabled measurement nodes, the probability of having every variable node in V_l eventually verified in algorithm $\mathcal{P}$ is:*

$$P_s^{\mathcal{P}}\{V_l(G^{\tilde{M}})\} = 1 - \sum_{S\in G\mathcal{LMSS}} P^{\mathcal{P}}\{S\ fails\} P_s^{\mathcal{P}}\{V_l((G\restriction_{\bar{S}})^{\tilde{M}\cup\mathcal{N}(S)})\}, \qquad (4.1)$$

where $\bar{S} = V_l \setminus S$, and $\mathcal{P} \in \{LM, nSBB, Genie\}$

Proof. From Lemma 4.1.5, we know that a graph $G^{\tilde{M}}$ will fail in algorithm $\mathcal{P}$ if there exists a non-empty type-$\mathcal{P}$ problematic set inside the set of non-zero elements of the graph. In other words, the failure happens if the set of variable nodes in at least one of the child problematic sets become non-zero. It has been proven that variable nodes in the maximal problematic set inside the non-zero elements of the graph along with its associated set, Namely, $H-$set and $R-$set, will remain eventually unverified. All the other variable nodes will be verified.

From the same reasoning as in Lemma 4.1.4, we also know that every problematic set in a graph is a child of exactly one GLMSS. Also, if a child of a GLMSS become non-zero, then that GLMSS will fail in the algorithm which causes the failure of the whole graph, and conversely, failure of a graph requires the failure of at least one problematic set in the graph which results in the failure of its specific mother GLMSS in the graph.

From Lemma 4.1.4, we know that the set of children of any two GLMSS in the graph are disjoint, which means that the probability of having a GLMSS failed is independent from other GLMSSs, i.e. the failure of a GLMSS is a function of the GLMSS itself not other GLMSSs. Therefore the probability that algorithm $\mathcal{P}$ fails on $G^{\tilde{M}}$ is:

$$\sum_{S \in G\mathcal{LMSS}} P^{\mathcal{P}}\{only\ S\ fails\}.$$

To have the probability of failure of *only* S, we also have to consider the probability that any other variable nodes in V_l are verified given that S is failed. Therefore, we have:

$$P^{\mathcal{P}}\{only\ S\ fails\} = P^{\mathcal{P}}\{S\ fails\} P_s^{\mathcal{P}}\{V_l(G^{\tilde{M}}) \setminus S | S\ fails\}.$$

From Lemma 4.1.9, we have:

$$P_s^{\mathcal{P}}\{V_l(G^{\tilde{M}}) \setminus S | S\ fails\} = P_s^{\mathcal{P}}\{V_l((G \restriction_{\bar{S}})^{\tilde{M} \cup \mathcal{N}(S)})\}.$$

Therefore, the probability of failure is:

$$\sum_{S \in G\mathcal{LMSS}} P^{\mathcal{P}}\{S\ fails\} P_s^{\mathcal{P}}\{V_l((G \restriction_{\bar{S}})^{\tilde{M} \cup \mathcal{N}(S)})\},$$

which proves the theorem. □

The heart of the proof of this theorem is, in fact, Lemma 4.1.9, which employs the

features of LMSSs and GLMSSs to find a recursion for the probability of error.

Corollary. *The probability that an algorithm $\mathcal{P}$ is successful over a graph G is:*

$$P_s^{\mathcal{P}}\{V_l(G)\} = P_s^{\mathcal{P}}\{V_l(G^{\emptyset})\}.$$

The expression presented for the probability of error is general for all of the algorithms excluding SBB. As a matter of fact, the exclusion of SBB is due to the existence of EEMN rule. This rule is the only rule that cannot be employed in Lemma 4.1.9, since it is not independent of the values of the measurement nodes in the neighbourhood of a GLMSS[6]; however, we will see that since we will ignore the recursion for some reasons later, the expression will become valid for SBB algorithm, as well, in certain situations. This issue will be clarified in the next section.
The analysis provided requires finding the children of a GLMSS and finding the set of GLMSSs of a given graph. Algorithms 3 and 4 describe two possible approaches for the two problems, respectively.

Algorithm 3 Finding the failure children of a GLMSS S in a graph G with the set of disabled measurement nodes $\tilde{M}$

```
1: procedure ExtractNon-zeroFailureSet
2:     Input: G, S, M̃
3:     Output: C_P(S|M̃)
4:     O ← ∅
5:     for Each S' ⊆ S do
6:         if N(S') = N(S) ∪ M̃ then
7:             if ∀c ∈ N(S) \ M̃ : c has appropriate properties P_1 to P_4 on S' then
8:                 add S' to O
9:             end if
10:        end if
11:    end for
12:    return O
13: end procedure
```

The complexity involved in Algorithms 4 and 3 clearly makes it impossible to run them for large graphs. In particular, Algorithm 3 can be efficiently run for GLMSSs of small size. In fact, the main complexity of this algorithm is due to the fact that

[6]See Remark 2 for further details on difference of SBB and other VB MPAs.

Algorithm 4 Finding the set of all GLMSSs of a graph G with the set of disabled measurement nodes $\tilde{M}$

```
1: procedure ExtractGLMSS
2:     Input: G, M̃
3:     Output: GℒℳSS
4:     O ← ∅
5:     for Each C ⊆ V_r \ M̃ do
6:         V ← 𝒩(C ∪ M̃)
7:         C′ ← 𝒩(V)
8:         V′ ← 𝒩(C′ \ (C ∪ M̃))
9:         V ← V \ V′
10:        if V is a stopping set then
11:            Add V to O
12:        end if
13:    end for
14:    return O
15: end procedure
```

one needs to search over all of the possible subsets of S, as the GLMSS, to find the corresponding children. This exhaustive search becomes impractical for large GLMSSs. Furthermore, Algorithm 4 also, has to perform as many times as the number of the subsets of the regular measurement nodes of the graph. The improvement in the complexity of this algorithm can be done by employing the approaches that are used for finding the stopping sets of a graph in the literature, and will be remained as a future work.

In terms of the correctness of these algorithms, one can see that Algorithm 3 is basically the implementation of the definition of a child set. However, the procedure for Algorithm 4 seems not quite straight forward. In this algorithm, in order to find the GLMSS corresponding to a given subset of measurement nodes C, first of all, one have to find all of variable nodes in the neighbourhood of $C \cup \tilde{M}$, that have all their neighbours inside $C \cup \tilde{M}$. This is done through lines 6 to 9 in the algorithm. In fact, V' in the algorithm stores the set of variable nodes which are in the neighbourhood of $C \cup \tilde{M}$, but they do not have all of their neighbours in $C \cup \tilde{M}$. Moreover, V is the set of variable nodes in $\mathcal{N}(C \cup \tilde{M})$ that have all their neighbours in $C \cup \tilde{M}$. In the next step, if the set V is an stopping set, then this set is clearly a GLMSS. Otherwise, there is no GLMSS on C.

Example 4.1.1. Consider the graph shown in Figure 4.2. There are only 3 LMSSs in

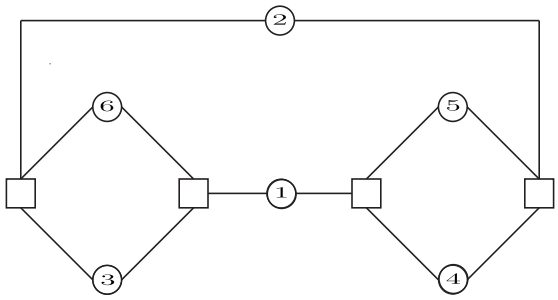


Figure 4.2: A bi-regular graph with 3 LMSSs

this graph. Table 4.1 shows these LMSSs along with the multiplicity of the children of different sizes for each LMSS with respect to Genie, LM, SSB and nSSB. For example, Table4.1 shows that the LMSS $\{1,2,3,4,5,6\}$ has 5 problematic children of size 2 with respect to LM (children that are SSSs). These children are $\{1,2\},\{5,6\},\{3,4\},\{4,6\}$ and $\{3,5\}$.
The exact probabilities of failures for different algorithms on this graph based on Theorem 4.1.10 are:

$$1-P_s^{Genie}\{V_l(G)\} = 2(\delta^2)(1-(\delta^2(1-\delta)^2+4\delta^3(1-\delta)+\delta^4))+(5\delta^4(1-\delta)^2+6\delta^5(1-\delta)+\delta^6). \quad (4.2)$$

$$1-P_s^{LM}\{V_l(G)\} = 2(2\delta(1-\delta)+\delta^2)((1-\delta)^4+2\delta(1-\delta)^3)+5\delta^2(1-\delta)^4+16\delta^3(1-\delta)^3+15\delta^4(1-\delta)^2 + 6\delta^5(1-\delta)+\delta^6. \quad (4.3)$$

$$1-P_s^{nSBB}\{V_l(G)\} = 2(\delta^2)((1-\delta)^4+4\delta(1-\delta)^3)+12\delta^3(1-\delta)^3+15\delta^4(1-\delta)^2+6\delta^5(1-\delta)+\delta^6. \quad (4.4)$$

Table 4.1: LMSSs of the graph in Figure 4.2 and the multiplicity of their failure children of different sizes with respect to different VB MPAs.

LMSS	child size	SS	SSS	RSSS	nRSSS
$\{3,6\}$	1	0	2	2	0
	2	1	1	1	1
$\{4,5\}$	1	0	2	2	0
	2	1	1	1	1
$\{1,2,3,4,5,6\}$	2	0	5	4	0
	3	0	16	16	12
	4	5	15	15	15
	5	6	6	6	6
	6	1	1	1	1

The numerical values of these probabilities, along with other examples, will be considered in Section 5.4. Also the discussion on the analysis for this graph will be postponed to that section.

The complexity of finding the exact probability of error using Theorem 4.1.10 is prohibitive for large graphs. However, for small values of δ, where we are working with super sparse signals, the complexity of finding this probability can be reduced dramatically. In the next section, we will focus on that region of the error curve, and we will introduce the concept of error floor in CS.

4.2 Error Floor Analysis

The main application of the expressions for exact probability of error of a given graph is in the estimation of the error floor. Error floor is a well-known phenomenon in coding theory and it has been several years that researchers are working on this topic in coding theory. However, in CS, to the best of our knowledge, there is no specific work which specifies this phenomenon despite its applications in solving the problems with super sparse signals. It is a common belief in CS community that most

of the algorithms will perform very well for signals with very few non-zero elements, in other words, it is assumed that the probability of failure for very sparse signals is zero. However, a careful inspection of the error probability at small values of δ reveals the fact that the performance of the recovery algorithm will change if the value of δ becomes less than a certain amount. To the best of our knowledge, this phenomenon has not yet been observed for any algorithm in CS, and this thesis is the first of its kind for tackling this problem in CS. The change in the slope of the graph at very small values of δ again indicates the close relation of MPAs in CS and in LDPC decoding. In the remainder of this section, we will investigate this issue in VB algorithms in CS.

In fact, error floor deals with the probability of failure of a graph in very small values of δ. Therefore, this restriction will enable us to simplify the expressions for the probability of failure, so that one can easily find the error floor of a given algorithm over a given graph.

In a given graph, with n variable nodes, it is clear that the expected value of the number of non-zero elements in the graph is δn. That is, we expect that approximately δn variable nodes become non-zero. Therefore, the probability of having less or more than δn non-zero variables is low. As the size of the LMSSs in a graph also varies from small values to larger values in the order of the size of the graph itself, then for a given value of δ, the probability of failure of some LMSSs is high, and at the same time, this probability for some others is relatively small. Therefore, for each value of δ, there are certain LMSSs that are contributing to the algorithm's failure. These LMSSs are called *active*, while all others are referred to as being *inactive*.

The activity status of an LMSS is highly dependent on the size of its children. If an LMSS only contains children with small size, this LMSS will be active only for small values of δ; on the other hand, if an LMSS contains children with variant sizes, this LMSS will be active for a wide range of values of δ, corresponding to the size of its children. There is a lower limit, $\delta_l(S)$, and an upper limit, $\delta_u(S)$, on δ values for which an LMSS S is active.

The lower limit can be determined by the LMSS itself; however, $\delta_u(S)$ is dependent on the other larger LMSSs in the graph. In fact, the lower limit is determined by the first term in (4.1), and the upper limit is also determined by the second term in that equation. This matter shows that the activity of an LMSS is closely related to the harmfulness of the LMSS. More precisely, the harmfulness of an LMSS introduced in

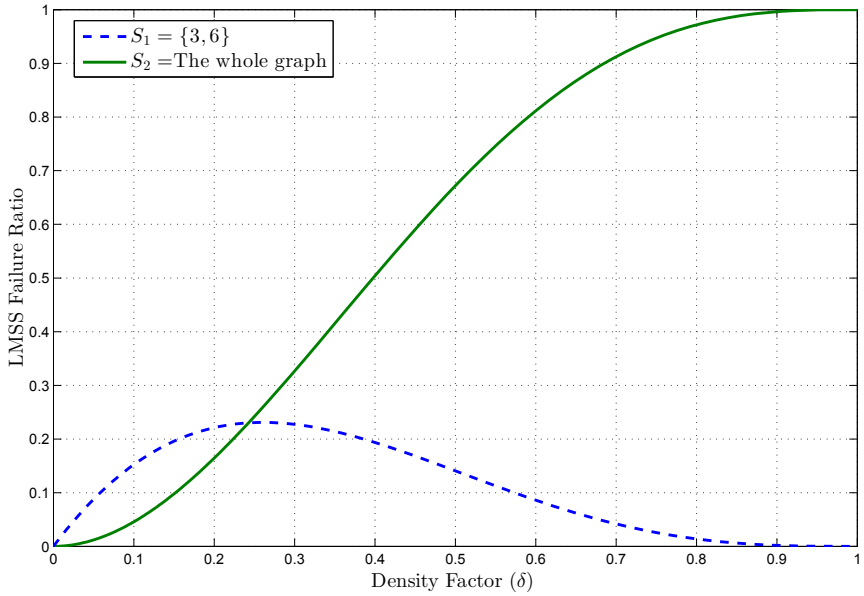


Figure 4.3: Activity range of two LMSSs in the graph shown in Figure 4.2

Theorem 4.1.6 can also determine the value of $\delta_l(S)$.

Example 4.2.1. Consider the graph shown in Figure 4.2. This graph has 3 LMSSs. These LMSSs are $S_1 = \{3, 6\}$, $S_2 = \{1, 2, 3, 4, 5, 6\}$ and $S_3 = \{4, 5\}$. The activity ranges of LMSSs S_1 and S_2 of this graph under LM algorithm are shown in Figure 4.3. This figure shows the contribution of the failure of each LMSS, individually. For example, the dashed line shows the probability that only S_1 fails. In fact, the LMSS failure ratio is defined as the probability that a certain algorithm fails to recover the variable nodes in the LMSS.

Since we are dicussing LM algorithm in this example, we have to consider SSS children of LMSSs. As can be seen from the figure, S_1 is a small LMSS with only children of size 1 and 2, which makes S_1 active in small values of δ; however, the range of the size of the children in S_2 is from 2 to 6 which makes its activity range being very wide. Also, since there is no other larger LMSS in the graph there is no upper limit defined for S_2. This matter clearly, can only happen for the whole set of variable nodes as the LMSS of the graph.

A deeper look at the contribution of each LMSS to the total failure probability of the graph under LM algorithm is also depicted in Figure 4.4. It can be seen that as we

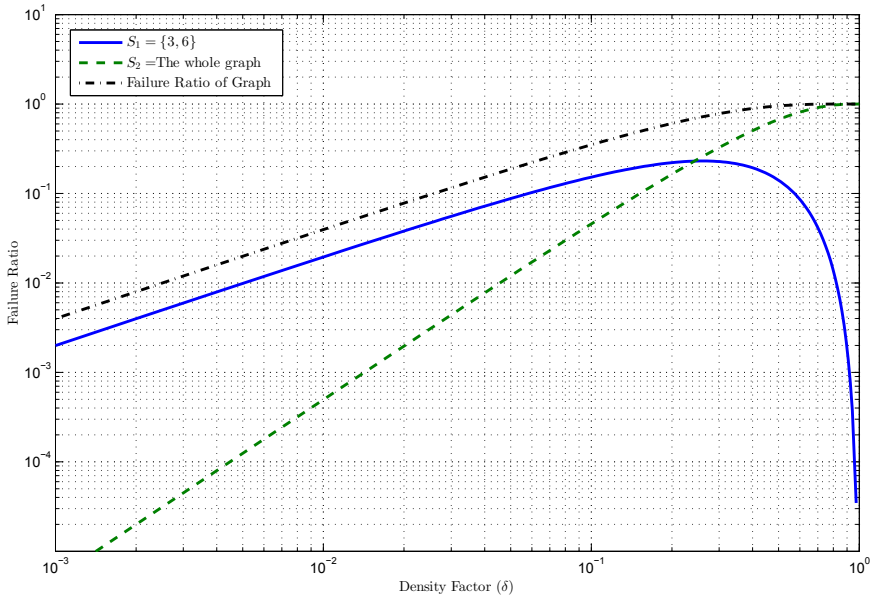


Figure 4.4: Contribution of each LMSS in the total failure probability of the graph shown in Figure 4.2.

move toward the smaller values of δ from the point where $\delta = 1$, at the beginning, we only have the contribution of S_2, but as we move on, S_1 will also join S_2 in the error. Interestingly, at some point in the curve, S_1 becomes dominant, and the final error follows the contribution of S_1 which is much more than S_2. In fact, the error contribution will be handed over, during this journey, between S_2 and S_1; since this hand over is done smoothly in this particular example, we cannot see a considerable degradation in the slope of the error curve, where the slope of the curve is defined as the least power of δ in the failure probability expression for each LMSS. On the other hand, for larger graphs, this hand over is usually not as smooth, and as a result, we will see a significant change in the slope of the curve. Also, when we are working in small values of δ, after the final hand over, there will be no more transition, and thus, the slope of the error curve will be dominated by the dominant LMSSs. This region is the so called error floor in coding theory. Note that the contribution of S_3 as the other LMSS in this graph is the same as S_1 due to the symmetry in the graph. One can also see that the total failure probability does not follow S_1 curve at the error floor, which implies the contribution of S_3.

Clearly, in error floor region the only active LMSSs are the ones with the smallest children. Let us consider (4.1) again. Inside the summation, there are two terms, which together guarantee the failure of only one LMSS. The first term, in fact, considers the failure of the LMSS itself and the other term puts restriction on the rest of the graph to be verified. Clearly, if an LMSS S fails in the graph, other variable nodes will become verified if the set of non-zero elements of the rest of the graph does not contain a child of another GLMSS. If another LMSS also fails, then S is not the only LMSS which has failed, and this situation is not of interest in the analysis, as we are interested in the error floor region, which implies the region corresponding to the minimum number of eventually unverified variable nodes. However, we know that the error floor region is the region in which just one LMSS is failed which is the smallest one, and as a result, the probability of having another LMSS failed is almost zero. Therefore, if an LMSS fails in the error floor region the probability of having the rest of variable nodes verified is 1. Based on this observation, we can remove the second term in (4.1) as in the error floor region the terms with the smallest power of δ become dominant. Thus, we can simplify the equation for the error floor region as follows:

$$P_{ef}^{\mathcal{P}}(G) \approx \sum_{S \in \mathcal{LMSS}} P^{\mathcal{P}}\{S \ fails\}, \tag{4.5}$$

where $P_{ef}(G)$ is the probability of failure of graph G in the error floor region. [7] Moreover, we know that since δ is small then only those LMSSs that contains small size children, are active. Let $\mathcal{MLMSS}$ denote the set of LMSSs with the children of size x where x is the size of the smallest possible problematic set in a given graph and algorithm, and let us call these sets the LMSSs with minimal size children. Thus, we can simplify the equation further:

$$P_{ef}^{\mathcal{P}}(G) \approx \sum_{S \in \mathcal{MLMSS}} P^{\mathcal{P}}\{S \ fails\}. \tag{4.6}$$

Note that since the second term inside the summation in (4.1) is not useful any more in the error floor region, then the whole formulation can be applied to SBB algorithm,

[7] Note that the reason why we use $\mathcal{LMSS}$ instead of $G\mathcal{LMSS}$ in this expression is that according to Remark 6, when a graph G does not contain any disabled measurement nodes, which is the case here, then GLMSSs are exactly the same as LMSSs.

as well.[8] The only problem that SBB had for the exact analysis was the fact that we could not simply write a recursive formulation for its analysis. Therefore, (4.6) is applicable to all of the VB algorithms.
We are still able to simplify the error floor performance equation further.

Lemma 4.2.1. *The harmfulness of an LMSS S in error floor region is given by:*

$$P^{\mathcal{P}}\{S \ fails|\delta \to 0\} = J(S)\delta^{I(S)},$$

where $I(S)$ is the size of the smallest children of S, and $J(S)$ is the multiplicity of the smallest children of size $I(S)$.

Proof. The proof is trivial. □

Based on Lemma 4.2.1, one tight lower bound on the error floor of a given graph is given by:

$$P_{ef}^{\mathcal{P}}(G) \approx \sum_{S \in \mathcal{MLMSS}} J(S)\delta^{I(S)}. \tag{4.7}$$

The recent equation is an approximation of the (4.6), when $\delta \to 0$.

Example 4.2.2. Table 4.2 shows some known SSs that can be, possibly, the smallest LMSSs in a given graph. Some of these SSs are taken from [69], where some of the SS structures up to weight five are listed. In this example, we will see how harmful these structures can be for a VB algorithm in CS, and also we will address their relative harmfulness in the sense that in the same family of SSs, different non-isomorphic structures can have different harmfulness for an algorithm. [9] This issue is one of the most important contribution of the characterizations presented in Chapter 2. In other words, based on the characterization provided, we can exactly discuss how harmful an SS is for a given algorithm. Moreover, shown in Table 4.2 are the error floor estimations that can be caused by these structures if they appear as the smallest LMSSs in a graph.
There are some important and interesting points in this table. Firstly, as it can be seen, S_1 and S_2 are in the same family of SS as the degree distribution of the variable

[8] We do not need Lemma 4.1.9 any more.

[9] By family of SS, we mean the SS structures that have the same degree distribution on their set of variable nodes. Clearly, within a family of SS there are some non-isomorphic structures that creates the whole family.

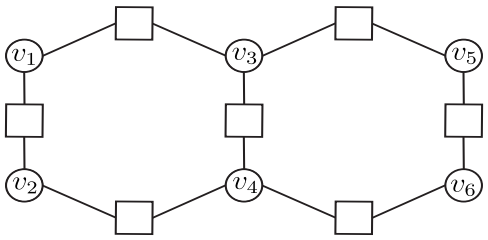


Figure 4.5: Example of a Stopping Set in a given graph

nodes in these two LMSSs are the same. However, there are differences between the nRSSS children of these two. More importantly, although the girth of S_2 (which is 4) is smaller than S_1 (which is 6), the performance of S_2 in non-binary SBB algorithm in the error floor region will be better than S_1. This shows that not always the smaller girths cause poorer performance, in the error floor region. The other issue worth mentioning in the table is the fact that sometimes by just a tiny change in the graph we can get reasonable improvement in the error floor. For instance S_4 is obviously a changed version of S_3, where the change is the swapping of an edge coming out of a single measurement node between two variable nodes. With only this slight change, the RSSS and nRSSS children of size 4 of S_3 have been completely removed in comparison to S_4. Furthermore, S_5 also shows that it is sometimes possible to reach to the performance of Genie algorithm in the error floor region using other practical VB MPAs.

Now as a simple numerical demonstration, consider the labeled LMSS shown in Figure 4.5, which is exactly S_3.

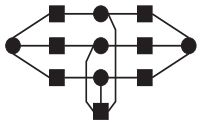
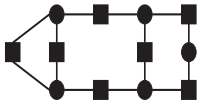
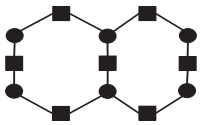
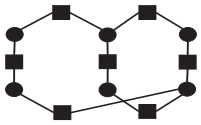
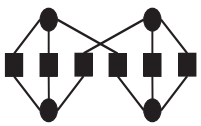
The list of children SSSs of this LMSS contains two children of size 3, $\{v_2, v_3, v_6\}$ and $\{v_1, v_4, v_5\}$, and also 8 children of size 4 and 6 children of size 5 and S itself. As a result, the probability of this LMSS being failed under LM algorithm is:

$$P^{LM}\{S\ fails\} = \ 2\delta^3(1-\delta)^3 + 8\delta^4(1-\delta)^2 + 6\delta^5(1-\delta) + \delta^6.$$

However, in Genie algorithm, this LMSS will be trapped only if the whole set S become non-zero. That is $P^{Genie}\{S\ fails\} = \delta^6$.

As a numerical example, to have a better sense of this difference, at $\delta = 0.1$, we have: $P^{LM}\{S\ fails\} = 0.0022$ and $P^{Genie}\{S\ fails\} = 1 \times 10^{-6}$. Furthermore, we see that in $P^{LM}\{S\ fails\}$ when δ is small, the most dominant term will be $2\delta^3$, corresponding

Table 4.2: Harmfulness of different stopping set structures

SS (LMSS in larger graphs)	**Children problematic sets**				
	child size	SS	SSS	RSSS	nRSSS
 S_1	1	0	0	0	0
	2	0	0	0	0
	3	0	4	0	0
	4	0	5	5	5
	5	1	1	1	1
	Error floor contribution	δ^5	$4\delta^3$	$5\delta^4$	$5\delta^4$
 S_2	1	0	0	0	0
	2	0	0	0	0
	3	0	4	0	0
	4	0	5	5	3
	5	1	1	1	1
	Error floor contribution	δ^5	$4\delta^3$	$5\delta^4$	$3\delta^4$
 S_3	2	0	0	0	0
	3	0	2	0	0
	4	0	8	2	2
	5	0	6	6	6
	6	1	1	1	1
	Error floor contribution	δ^6	$2\delta^3$	$2\delta^4$	$2\delta^4$
 S_4	2	0	0	0	0
	3	0	2	0	0
	4	0	8	0	0
	5	0	6	6	6
	6	1	1	1	1
	Error floor contribution	δ^6	$2\delta^3$	$6\delta^5$	$6\delta^5$
 S_5	1	0	0	0	0
	2	0	2	0	0
	3	0	4	4	0
	4	1	1	1	1
	Error floor contribution	δ^4	$2\delta^2$	$4\delta^3$	δ^4

to sets $\{v_2, v_3, v_6\}$ and $\{v_1, v_4, v_5\}$. In fact, in this particular example, this dominant part is responsible for almost 91% of the failures when $\delta = 0.1$.

As it is shown in the example 4.2.2, the harmfulness of an LMSS S in the error floor region is mainly given by $J(S)$ and $I(S)$. The smaller the children, the more harmful the LMSS will be. Similar to LDPC codes, the error floor phenomenon in VB MPAs in CS is due to the smallest problematic sets in the graph. Mathematically, as we decrease the value of δ the most dominant term in the harmfulness of the LMSS will be the one with the smallest power, or equivalently, the smallest $I(S)$. Moreover, among all of the LMSSs of a graph the one which has the smallest child will be the most harmful LMSS and is responsible for the error floor of the graph.
Based on (4.7) the problem of finding the error floor of the graph is equivalent to find the smallest children of the minimal active LMSSs under LM algorithm.

Theorem 4.2.2. *Finding the size of the minimum size SSS child S' in a given LMSS S is equivalent to the well-known Minimum Set Cover problem.*

Proof. Consider the universe of the Min Set Cover problem[10] to be $U = \mathcal{N}(S)$, and also assume that each variable node corresponds to the subset of measurement nodes to which it is connected. Based on Definition 4.1.2, a subset of variable nodes S' is called a child SSS if $\mathcal{N}(S') = \mathcal{N}(S)$, which equivalently means that S' covers the universe. Therefore, each child SSS, in fact, corresponds to a cover set for the universe. That is, finding the minimum size child SSS is equivalent to finding the minimum size cover set for the universe. □

Although one corollary from Theorem 4.2.2 is the fact that finding the smallest children in a given LMSS is in NP-Complete, as the Min Set Cover problem itself is a well-known NP-Complete problem [70], we should point out that we are working with small size LMSSs, in error floor region, and this means that the size of the problem will not be too large to make it intractable. In other words, even the exhaustive search algorithm (see Algorithm 3) for this problem can perform very well as the size of the minimal active LMSSs is usually small, in error floor region. Therefore, the main issue in this topic will be finding the minimal active LMSS, which has been considered vastly in the literature [67, 68], and is beyond the scope of this thesis.

[10] Interested reader is referred to [70] and reference therein for the introduction to Min Set Cover Problem and its related issues.

Chapter 5

Numerical Results

In this chapter, we will focus on simulation and numerical results of the analysis introduced in previous chapters. Firstly, we will focus on the difference between the finite length and asymptotic analysis, and we will show how finite length analysis differs from asymptotic one for individual graphs. In the next experiment, we will focus on the concentration result, and we will respond to the question that how and where it is possible to use the concentration results in VB algorithms. In the third part of this chapter, we will examine some ensemble of graphs to demonstrate that the average probability of failure that has been calculated using the analysis provided in Chapter 3 is matched with the numerical results. This experiment will also be repeated for all the VB algorithms, and also for the simplified method introduced for the LM algorithm. Furthermore, we will address the numerical results for the individual graph analysis, in the next experiments, and show that these results are also perfectly matched with the analytic results we acquired throughout the thesis. And finally, we will investigate the error floor analysis from the numerical perspective and discuss its different aspects.

The general approach for the numerical results is as follows unless otherwise stated: we will always simulate the results using the Monte Carlo simulation with 100,000 trials for experiments regarding average ensemble performance and 1,000,000 trials for experiments on individual graph performance. For the average ensemble, in each trial we will choose a graph randomly from an ensembles $\mathcal{G}(n, d_v, d_m)$ for given n, d_v and d_m, and for the individual graph performance, we will run the experiments on a specified graph. The input signal will be considered to be sparse, according to $\mathcal{K}$, the support set of the signal, which is randomly chosen in each trial from within the set of all elements of the input vector. Note that in the simulation results provided, we

will use both the deterministic number of elements in $\mathcal{K}$ and the probability that a variable node becomes non-zero, δ. That is, we sometimes assume that the number of non-zero elements of the graph is given and sometimes assume that δ is given. [1] In the latter case, we will consider δ as the probability that a variable node becomes non-zero independently from other variable nodes. Besides, every $v \in \mathcal{K}$ will be assigned a random value from uniform distribution function defined between $[0, 100]$. Note that the negative values can also be used, and they will not affect the performance of VB MPAs. Moreover, every edge in the sensing graph will have weights 1, except for non-binary SBB algorithm where the weights of the graph are chosen randomly from a continuous distribution. Besides, multiple parallel edges are allowed for a graph in the experiments regarding the average performance analysis. Nevertheless, the messages received at each node along its connected edges will be considered to be generated by different nodes. That is, the receiver node will not take parallel edges into account, since otherwise, the graph under consideration does not belong to $\mathcal{G}(n, d_v, d_m)$, and in fact, it belongs to another ensemble of irregular bipartite graphs. Also, For the experiments regarding individual graph analysis, we choose graphs in from $\mathcal{G}(n, d_v, d_m)$ that do not contain any parallel edges.

5.1 Finite Length vs Asymptotic Regime

In the first numerical experiment, we will see how sizes of the graphs affect the success ratio curve of a given algorithm. For this purpose, we choose some random graphs with different compression ratios solved by different algorithms. We will change the sizes of the graphs to check the performance of it around its asymptotic curve. As discussed in [26], in the asymptotic regime, we will have a threshold, called the success threshold, before and after which the success ratio is 1 and 0, respectively. Shown in Figure 5.1 are different graphs and algorithms with different sizes, $n \in \{120, 240, 480\}$, and different degree distributions, $(d_v, d_m) \in \{(3, 4), (5, 6), (2, 5)\}$, to compare their finite length performance. For each (d_v, d_m) pair associated with each graph the success threshold is indicated using tables provided in [26]. The probability of success of each algorithm on each individual graph is, as expected, near its corresponding

[1] Translating these results from fixed point to δ can be easily done using the total probability theorem as follows:

$$P_e(G, \delta) = \sum_{\mathcal{K} \subseteq V_l} \delta^{|\mathcal{K}|}(1-\delta)^{|V_l|-|\mathcal{K}|} P_e(G \mid \mathcal{K})$$

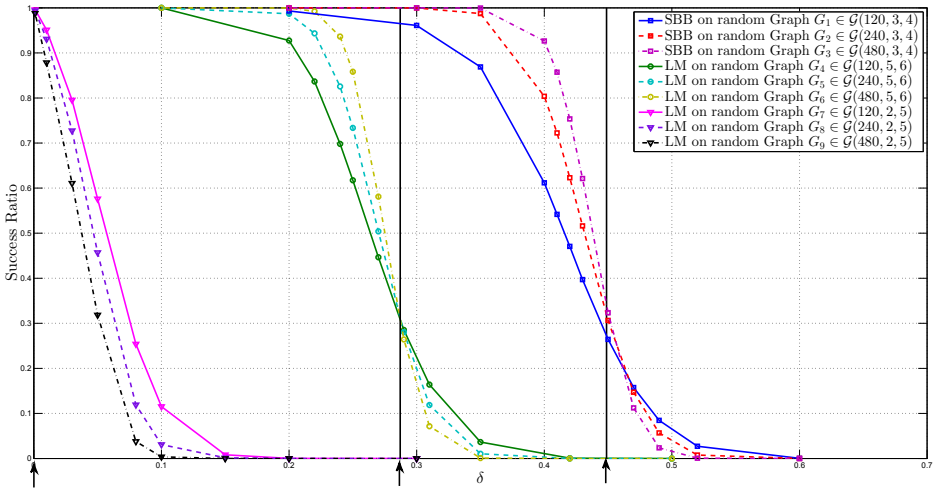


Figure 5.1: success ratio curves for finite length regime around their success ratios for some graphs with $n \in \{120, 240, 480\}$ and $(d_v, d_m) = \{(3,4),(5,6),(2,5)\}$ along with the success threshold in the corresponding ensemble.

success threshold. In fact, the success threshold is almost matched to the point of the intersection of curves of different-sized graphs in the finite length regime.
It can be seen that before the success threshold the performance of the algorithm improves as the size of the graph increases; however, after the success threshold the situation will be reversed. This will result in an interesting phenomenon in graphs with $d_v = 2$. Since in those graphs the success threshold is zero, then the performance of a graph will improve as the size of the graph decreases.

5.2 Concentration Results

In this section, we will run experiments by which we can verify the concentration result discussed in Theorem 1.5.1. There are two main questions that we need to answer regarding the finite length analysis. The first issue is about the validity of the concentration result for finite length regime. To answer this question, we will run LM algorithm with $1,000,000$ trials, for each point, on several graphs in a certain ensemble. Figure 5.2 shows the behavior of different graphs, with $n \in \{126, 252, 400\}$ and $(d_v, d_m) = (3,6)$, in the same ensemble. From each ensemble we choose two graphs at random, and we also add another graph using PEG algorithm [71]. The

PEG algorithm generally is a method of constructing graphs with large girths. In this algorithm, the edges of a graph will be added iteratively to a graph such that in each iteration the maximum possible girth is achieved in a greedy manner. Moreover, based on the results in [72] the larger the girth of the graph, the larger the smallest SS in the graph. In other words, removal of short cycles in the graph will result in the removal of short SSs in the graph. Down the road, according to the established connection between problematic sets in this work and SSs, we can simply guarantee that by removal of small SSs, other small problematic sets are also removed. At least, based on Diagram 2.2, we know that SSs are part of other problematic sets. Then removal of them will guarantee the removal of some of the problematic sets from the underlying graph.[2] Having the curves related to the graphs with large girth will give us a good insight on the performance of VB MPAs over those graphs, as well.

As can be seen in the figure, the curves corresponding to $n = 126$ are not close to each other in comparison to those curves related to $n = 252$ and $n = 400$. Moreover, when the value of δ is less than a certain amount, roughly 0.08 in this figure, then all of the curves will go apart from each other, which means that the concentration results do not hold true. This will motivate the second question: In which region of the performance curve can one guarantee the concentration result? It is shown in this figure that small values of δ in each curve are not governed by the concentration result. Basically, the concentration results do not work in the error floor region of the graphs.

Moreover, in all of the scenarios the graph that has been constructed by the Progressive Edge Growth (PEG) method, expectedly, has the best performance, especially, when the value of δ is small. Since PEG method removes small cycles of the graph then, as discussed, it is guaranteed that small size SSSs will also be removed. Since the performance of graphs constructed by PEG method outperforms other graphs in the same ensemble, it can be interpreted that these are actually small size SSSs that result in poor performance of the graph in small values of δ, as their removal will improve the performance in that region.

[2] More precise relations between problematic sets introduced in this work and girth of a graph can be considered as a future topic.

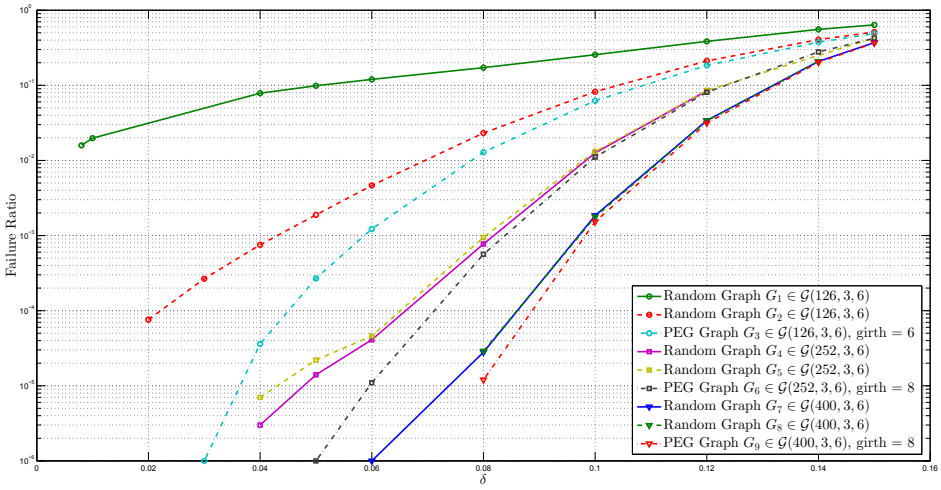


Figure 5.2: Failure probability of different graphs with different sizes in LM algorithm. The applicability of concentration results depend on the size of the graphs and different regions of the curves.

5.3 Average Performance Analysis and Applications

The finite length analysis of VB algorithms on the average performance of a given ensemble will be addressed in this section. Moreover, based on the analysis, we will be able to examine some features of VB algorithms, analytically.

5.3.1 Genie

We will start by the analysis of Genie algorithm from the simplified equations given in [55]. Firstly, we will consider some numerical results for some small size graphs. Table 5.1a shows how close the analytic results are to the numerical results for three small size graphs with $n \in \{6, 8, 10\}$, and $(d_v, d_m) \in \{(2,3),(3,4),(2,5)\}$. Note that the small discrepancy between the analytic and numerical results in tables are due to the small number of trials we used for the simulation. Increasing the number of trials will solve this issue. For instance, in Table 5.1a in the results for $(n, d_v, d_m) = (8,3,4)$ and $|\mathcal{K}| = 2$, if we increase the number of trials to 10,000,000 the gap between 0.0580 as the analytic value and 0.0571 as the simulation result will become small. In fact the value of the simulation result, i.e. 0.0571 will move towards the analytic result,

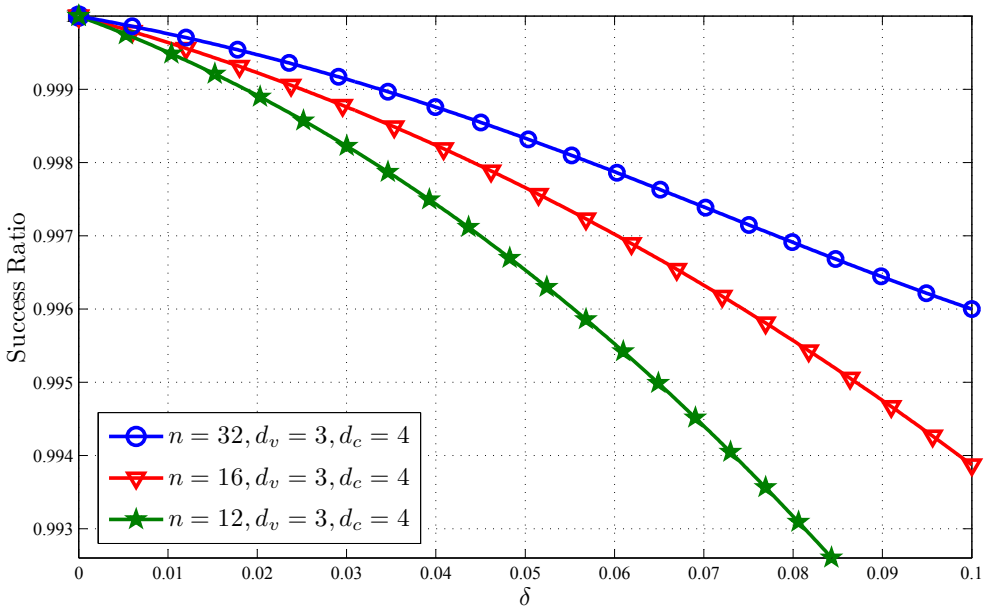


Figure 5.3: behavior of Genie algorithm around success ratio 1, with respect to the size of the graph

i.e. 0.0580.

As the next experiment on Genie algorithm, we will examine the fact that how this algorithm converges to success ratio 1 when $\delta \to 0$, with respect to the size of the graph. Shown in Figure 5.3 is the convergence behavior of the Genie algorithm around success ratio 1, when $(d_v, d_m) = (3, 4)$. As it can be seen by increasing the size of the graph the success ratio will improve. These results are provided using the analytic results, hence the size of the graphs are limited. However, a thorough discussion for different algorithms in larger sizes has been considered in Figure 5.1 using numerical results.

5.3.2 LM

For LM algorithm in the first experiment, we will provide some numerical examples for some small size graphs, and compare the results with the analytic results that we had in Chapter 3, and then we will consider the simplified versions of the algorithm to analyze larger graphs. Furthermore, using the analytic result, we will examine a famous phenomenon in iterative decoders, that will be introduced later,

and we will show that LM is not an exception. Finally for LM, we will examine its behavior around success ratio 1 with respect to different ensembles. Although the computational complexity of the simplified version of the algorithm for LM is much lower than the general methodology that we had in Section 3.1, this complexity is still exponential in the number of non-zero elements; as a result, we will be able to provide simulation results for larger graphs, but with limited number of non-zero elements. However, these results are enough for us to see the behavior of the system in small fraction of non-zero elements. For the first experiment, Table 5.1b shows how matched the analysis and characterization provided are to the numerical results in the LM algorithm. Generally speaking, as it can be seen, size of a graph is not the only important parameter in the performance of the graph. In fact, the compression ratio is another dominant parameter which affects the failure ratio. As it can be seen, in $\mathcal{G}(10, 2, 5)$, where the compression ratio is 0.4 even one non-zero element is enough to make a huge failure ratio while in $\mathcal{G}(8, 3, 4)$, even though the size of the graphs are smaller, the failure ratios are significantly lower.

The next experiment is dedicated to the simplified versions of the expressions for LM algorithm provided in Section 3.4. These expressions will allow us to run the algorithm for larger graphs, however, with limited number of non-zero elements. Table 5.2 shows how matched the simplified version of the analysis is to the numerical results. It has been shown that the result of the simplified version of expressions are accurate both for small size and moderate size graphs. Moreover, note that the simplified version of the expressions in LM algorithm is only provided for ensembles of graphs with $d_v = 2$. As it is discussed in Section 3.4 generalizing these expressions to larger left degrees is a lot more complicated and is avoided in this thesis. As it can be seen from the table, again, the compression ratio plays an important role. The failure ratios in $\mathcal{G}(12, 2, 3)$ are almost the same as the ones in $\mathcal{G}(40, 2, 5)$ despite the larger graphs that we have in the latter ensemble. This issue can restate the fact, that in some cases, we may need a certain amount of information from the sparse signal to be able to recover it in CS, and if we go beyond that amount by reducing the compression ratio, then the possibility to recover the signal we be dramatically reduced.

In this part, we will consider a famous phenomenon in ensembles with $d_v = 2$. According to [61], when the fraction of edges connected to degree 2 variable nodes in an irregular LDPC code exceeds a certain threshold, then the probability of bit error cannot converge to zero no matter how large the block length and iteration numbers

are. Considering the close connection between MPAs in CS and LDPC decoding, the same argument is expected to be seen in the context of CS. As a motivating example, we will show that this phenomenon does exist in LM algorithm, when we have a regular bipartite graph with $d_v = 2$. Note that in the regular case, when all of the variable nodes have degree 2, the fraction of edges connected to variable nodes with degree 2 is 1 which is clearly greater than any threshold. Since the simplified expressions, that we have in place for LM algorithm, work quite well in small number of non-zero elements for ensembles of graphs with $d_v = 2$, we can use these expressions to see this phenomenon. That is, it is enough to see that for small fraction of non-zero elements, increasing n will not reduce the failure ratio. Figure 5.4a shows the reduction of failure ratio by increasing n; however, this does not contradict the recent discussion as this figure depicts the failure ratio versus number of non-zero elements. If we consider the failure ratio versus density factor δ, as depicted in Figure 5.4b, then we will see that all of the curves for different values of n overlap, which means that increasing n will not contribute to reduction of failure ratio when $d_v = 2$. This issue, in fact, restates that the success threshold for LM algorithm when $d_v = 2$ is zero. Also, what we showed in Figure 5.4b supports the curves that were shown in Figure 5.1 for LM with $d_v = 2$. In fact, curves in Figure 5.4b are depicted in small values of δ, while those in Figure 5.1 are showing the whole range of δ.

The other analytical result from the LM algorithm that also can prove the degree 2 phenomenon is depicted in Figure 5.5. As it can be seen in this figure, the slope of the curve corresponding to $d_v = 2$ is significantly greater than the two other curves, which empirically states that there is no trend in those curves to converge to one before $\delta = 0$. On the other hand, the slopes of the other curves corresponding to $d_v > 2$ are approximately zero. Another interesting point that can be interpreted from this figure is that the larger compression ratio will not always result in a better performance, as an example consider $\mathcal{G}(8, 3, 4)$ with compression ratio 0.75 which has better performance in large values of δ than $\mathcal{G}(10, 4, 5)$ with compression ratio 0.8 even though its size is also smaller. Also note that these analytic results are provided using the general method for average performance analysis of LM not the simplified version.

In this part of the numerical results, we aim at evaluating the convergence of the success ratio to 1, when the fraction of non-zero elements is small and also when

$d_v > 2$, in LM algorithm. Since the simplified version of the analysis cannot be employed for $d_v > 2$, then the following results have been extracted from the general methodology for average performance analysis. Shown in Figure 5.6 is the convergence of the success ratio in small fraction of non-zero elements, when n increases. The 99% success ratio for the scenarios depicted in Figure 5.6 occurs at $\delta = 0.0091$, 0.0193 and 0.410 for $n = 8$, $n = 16$ and $n = 32$, respectively, which shows the convergence speed of the algorithm curve to success ratio 1, when $d_v > 2$.

5.3.3 Non-binary SBB

In the next experiment, we will provide some numerical results for the analysis of the non-binary SBB algorithm. Unfortunately, since there is no explicit expression for the analysis of binary SBB, thus, we can only provide the numerical results for the non-binary case. Besides, since the analysis provided for the non-binary SBB is also complex, then we can only show the results for some small graphs. Table 5.1c shows how accurate the analytic results are in this algorithm comparing to the numerical results, for some small size graphs. It can be seen that in comparison to LM, non-binary SBB has much better performance, and sometimes outperforms LM in success probability by about one order of magnitude. Also, again the impact of the compression ratio is remarkable in this example, generally, as we increase the compression ratio the probability of failure will be decreased.

5.4 Individual Graph Analysis

In this section, we will focus on simulation and numerical results of the individual graph analysis introduced Chapter 4. We will mainly focus on the performance of the individual graphs, and we will examine the harmfulness of different problematic sets in given graphs. We will categorize problematic sets based on some of their features and their harmfulness, and we will also see how accurate the error floor estimation technique provided for problematic sets in this thesis is. Besides, we will also examine some small individual graphs to demonstrate that the analytical and numerical results are matched. From this point all simulations are based on 1,000,000 trials unless otherwise stated.

Table 5.1: Comparison between the analytical and numerical results of failure ratios of different VB MPAs over different ensembles.

(a) Genie

(n, d_v, d_c)	$\lvert\mathcal{K}\rvert$	(Analytic, Simulation)
$(6, 2, 3)$	1	(0.1818, 0.1820)
	2	(0.4000, 0.4014)
	3	(0.6727, 0.6750)
	4	(1.0000, 1.0000)
	5	(1.0000, 1.0000)
$(8, 3, 4)$	1	(0.0119, 0.0122)
	2	(0.0580, 0.0571)
	3	(0.1624, 0.1635)
	4	(0.3987, 0.3966)
	5	(0.8799, 0.8791)
	6	(1.0000, 1.0000)
$(10, 2, 5)$	1	(0.2105, 0.2106)
	2	(0.4582, 0.4581)
	3	(0.7420, 0.7392)
	4	(1.0000, 1.0000)

(b) LM

(n, d_v, d_c)	$\lvert\mathcal{K}\rvert$	(Analytic, Simulation)
$(6, 2, 3)$	1	(0.5325, 0.5321)
	2	(0.9501, 0.9502)
	3	(1.0000, 1.0000)
	4	(1.0000, 1.0000)
	5	(1.0000, 1.0000)
$(8, 3, 4)$	1	(0.1270, 0.1272)
	2	(0.5886, 0.5881)
	3	(0.9224, 0.9227)
	4	(0.9902, 0.9902)
	5	(1.0000, 1.0000)
	6	(1.0000, 1.0000)
$(10, 2, 5)$	1	(0.8233, 0.8231)
	2	(1.0000, 1.0000)
	3	(1.0000, 1.0000)
	4	(1.0000, 1.0000)

(c) non-binary SBB

(n, d_v, d_c)	$\lvert\mathcal{K}\rvert$	(Analytic, Simulation)
$(6, 2, 3)$	1	(0.1818, 0.1820)
	2	(0.6868, 0.6882)
	3	(1.0000, 1.0000)
	4	(1.0000, 1.0000)
	5	(1.0000, 1.0000)
$(8, 3, 4)$	1	(0.0306, 0.0301)
	2	(0.1878, 0.1885)
	3	(0.6358, 0.6362)
	4	(0.9597, 0.9601)
	5	(1.0000, 1.0000)
	6	(1.0000, 1.0000)
$(10, 2, 5)$	1	(0.2105, 0.2118)
	2	(0.8659, 0.8664)
	3	(1.0000, 1.0000)
	4	(1.0000, 1.0000)

5.4.1 Exact Probability of Failure

Given an individual graph, from (4.1), we are able to find the exact probability of failure for LM, non binary SBB and Genie algorithm. However, note that, since the complexity of the analysis is high, we are not able to see the results for large graphs. Nevertheless, although the graphs in this subsection are not large, they can be used to see how matched the analytical result and numerical results are. Consider the graph shown in Figure 4.2. Equations (4.2), (4.3) and (4.4) are the results of the exact analysis of this graph. To compare these analytic results with the simulation results, Table 5.3a is provided. As it can be seen from the table, the analytic results are closely matched with the numerical results. It can also be seen that the performance of non-binary SBB is close to the performance of Genie in the error floor when the value of δ is small. However, for LM algorithm the situation is completely different, and it performs inferior to Genie and non-binary SBB. This example demonstrates that the children nRSSS and children SS of the minimal active LMSSs in this graph, i.e. $\{3, 6\}$ and $\{4, 5\}$ are, to some extent, the same in terms of harmfulness; however, for the same LMSS, the SSS children are a lot more harmful. Furthermore, the analytic results for this graph based on (4.2), (4.3) and (4.4) are also depicted in Figure 5.7.

Table 5.2: Failure Ratio in LM Algorithm (Simplified Expressions)

(n, d_v, d_c)	$\lvert\mathcal{K}\rvert$	(Analytic, Simulation)
$(12, 2, 3)$	1	(0.2586, 0.2602)
	2	(0.5370, 0.5389)
	3	(0.8044, 0.8033)
	4	(0.9629, 0.9638)
	5	(0.9982, 0.9981)
	6	(1.0000, 1.0000)
	7	(1.0000, 1.0000)
$(40, 2, 5)$	1	(0.2424, 0.2424)
	2	(0.5022, 0.4989)
	3	(0.7461, 0.7455)
	4	(0.9147, 0.9154)
$(120, 2, 3)$ (10000) trials	1	(0.0251, 0.0256)
	2	(0.0505, 0.0489)
	3	(0.0763, 0.0803)
	4	(0.1025, 0.1036)
	5	(0.1291, 0.1249)
	6	(0.1561, 0.1463)
	7	(0.1836, 0.1890)

In the next experiment, we will consider the graph shown in Figure 5.8. The first step in the analysis of this graph is to find different children of LMSSs of this graph which are shown in Table 4.1. The probability of failure of this graph under VB algorithms is also shown in Table 5.3b. The results show how accurate the analytic results are. Unlike the previous graph, the performance of non-binary SBB in small values of δ is much worse than the performance of Genie. It follows that for the minimal active LMSS in this graph, the harmfulness of nRSSS children are more than SS children.

5.4.2 Error Floor Analysis

In this subsection, we will provide some analysis for different LMSS structures and their contribution to the performance of a given graph. We are mostly interested in small size problematic sets as they play the most important role in the error floor[3] region. The characterizations provided in Chapter 2 will be highly useful in this analysis, as it will pave the road for further research in this area specifically on the error floor region analysis. In the first experiment, we will inspect the dominant SSSs, and we will find how significantly they contribute in the failure of a given graph.
We start with a random graph $G_1 \in \mathcal{G}(126, 3, 6)$ whose performance is shown in Figure 5.2. As can be seen, the performance of this graph is very poor in comparison to other graphs. Full inspection of the structure of the graph leads us to the LMSS shown in Figure 5.9 in the graph.
Using this structure and the analysis provided in Chapter 4, we are able to estimate the failure probability of graph G_1. Table 5.4 shows the estimated values of failure probability and compare them with the numerical results. The numerical results for contribution of the LMSS in the error is evaluated by considering the number of trials in which the LMSS shown in Table 5.4 is eventually failed in the algorithm. In other words, we look for the cases in which all the measurement nodes in the LMSS is non-zero or equivalently at least one of the two trivial children of the LMSS remains eventually unverified in the algorithm.
Furthermore, the error floor of G_1 can also be evaluated by (4.5) and (4.6).

[3]Note that the correct term that should be used in the context of VB MPAs is the *failure floor*; however, in order to follow the terminology in from the coding theory we still use the term *error floor*.

Table 5.3: Comparison of the analytic and numerical results provided for the graph of Figure 4.2 under different VB algorithms. The format of the entries of the table is (Analytic Result, Numerical Result)

(a) Failure probability of the graph in Figure 4.2

δ	Genie	non-binary SBB	LM
0.1	(0.0202, 0.0206)	(0.0290, 0.0291)	(0.3505, 0.3504)
0.2	(0.0825, 0.0820)	(0.1316, 0.1312)	(0.6068, 0.6070)
0.3	(0.1878, 0.1874)	(0.2989, 0.2992)	(0.7815, 0.7812)
0.4	(0.3313, 0.3309)	(0.4972, 0.4961)	(0.8911, 0.8915)
0.5	(0.5000, 0.4993)	(0.6875, 0.6875)	(0.9531, 0.9530)
0.6	(0.6733, 0.6742)	(0.8392, 0.8390)	(0.9836, 0.9834)
0.7	(0.8263, 0.8261)	(0.9375, 0.9373)	(0.9959, 0.9959)
0.8	(0.9359, 0.9262)	(0.9851, 0.9849)	(0.9994, 0.9994)
0.9	(0.9901, 0.9900)	(0.9989, 0.9988)	(1.0000, 1.0000)

(b) Failure Ratio of the graph in Figure 5.8

δ	Genie	non-binary SBB	LM
0.1	(0.0654, 0.0651)	(0.2165, 0.2167)	(0.4964, 0.4966)
0.2	(0.2551, 0.2551)	(0.5504, 0.5494)	(0.7853, 0.7853)
0.3	(0.5060, 0.5064)	(0.7936, 0.7931)	(0.9233, 0.9230)
0.4	(0.7342, 0.7345)	(0.9241, 0.9241)	(0.9778, 0.9779)
0.5	(0.8887, 0.8883)	(0.9785, 0.9787)	(0.9951, 0.9951)
0.6	(0.9665, 0.9666)	(0.9957, 0.9957)	(0.9993, 0.9993)
0.7	(0.9938, 0.9938)	(0.9995, 0.9995)	(0.9999, 1.0000)
0.8	(0.9995, 0.9995)	(1.0000, 1.0000)	(1.0000, 1.0000)
0.9	(1.0000, 1.0000)	(1.0000, 1.0000)	(1.0000, 1.0000)

Table 5.4: Failure Contribution of a small LMSS in LM for G_1

δ		Total failure probability of graph G_1 (Simulation)
0.008	0.0159	0.0159
0.01	0.0199	0.0198
0.04	0.0789	0.0785
0.05	0.0975	0.0986
0.06	0.1164	0.1197
0.08	0.1536	0.1716
0.1	0.19	0.2553
0.12	0.22	0.3847
0.14	0.26	0.5553
0.15	0.27	0.6376

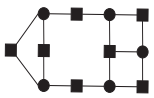
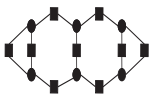
$$P_{ef}^{LM}(G_1) \approx 2\delta(1-\delta) + \delta^2 \approx 2\delta.$$

As it can be seen from Table 5.4, the estimation of failures are closely matched to the simulation results, in the error floor region. Moreover, as the fraction of non-zero elements in the graph increases, other SSSs also contribute to the failure; however, it may be hard to distinguish a few number of those SSSs and consider them as the dominant ones. In fact, there are more than a few number of SSS structures that contribute to the failure in the region with larger fraction of non-zero variables.

Another point that can be perceived from Table 5.4 is that although graph G_1 only has one structure of the kind shown in Figure 5.9, i.e. the multiplicity of that dominant SSS is 1, the results show that existence of one such harmful SSS is enough to dramatically deteriorate the performance of the whole graph.

In the next experiment, we will find the probability of failure of another random graph in the same ensemble, in the error floor region, $G_2 \in \mathcal{G}(126, 3, 6)$, whose performance is also depicted in Figure 5.2. In this graph, the smallest LMSSs and their contribution

Table 5.5: Failure Contribution of small LMSSs in LM for G_2

δ			Total Failure Probability (Simulation)
0.02	3.92e-5	1.63e-5	7.60e-5
0.03	1.31e-4	5.55e-5	2.66e-4
0.04	3.07e-4	1.32e-4	7.52e-4
0.05	5.94e-4	2.61e-4	1.882e-4
0.06	0.0010	4.55e-4	4.636e-3
0.08	0.0024	0.0011	0.0231
0.1	0.0054	0.0022	0.0823
0.12	0.0076	0.0038	0.2112
0.14	0.0119	0.0060	0.4054
0.15	0.0144	0.0075	0.5115

to the total failure of the graph are shown in Table 5.5. Note we have found these LMSSs using simple simulation. That is, by setting a very small value for δ, i.e. δ in error floor region, and capturing the failure instances in the graph. Using this approach, one would be able to find some of the dominant LMSSs of the graph, though it is not guaranteed to find all of them.[4] These LMSSs are the ones that trapped the algorithm for those small values of δ. Then we used all of the SSS children of those LMSSs to find the contribution of the LMSSs in the total failure probability. Again, similar to G_1, in G_2, we are also able to estimate the error floor, almost perfectly. Addition of the failure contribution of the two LMSSs shown in Table 5.5 will give us a close approximation of the total failure probability, in the error floor region.

In the next experiment, we have shown the error floor estimation of some graphs with different compression ratios under LM algorithm. Figure 5.10 shows the accuracy of the error floor estimation for these graphs.

[4]One might use different approaches in the literature for finding small and dominant SSs in a given graph [68].

Based on (4.6), the error floor approximations on each of the graphs shown in Figure 5.10 are as follows:

$$P_{ef}^{LM}(G'_1) \approx 2\delta^2(1-\delta)^2 + 4\delta^3(1-\delta) + \delta^4.$$
$$P_{ef}^{LM}(G'_2) \approx 2\delta^3(1-\delta)^3 + 8\delta^4(1-\delta)^2 + 6\delta^5(1-\delta) + \delta^6.$$
$$P_{ef}^{LM}(G'_3) \approx 2\delta^4(1-\delta)^4 + 12\delta^5(1-\delta)^3 + 18\delta^6(1-\delta)^2 + 8\delta^7(1-\delta) + \delta^8.$$

In fact, these approximations are extracted from the dominant LMSSs in the graphs. Although there are a number of terms involved in these expressions, the dominant terms are the first terms (see (4.7)). That is, since we are working on small values of δ, then the following approximations are valid as discussed in Section 4.2.

$$P_{ef}^{LM}(G'_1) \approx 2\delta^2.$$
$$P_{ef}^{LM}(G'_2) \approx 2\delta^3.$$
$$P_{ef}^{LM}(G'_3) \approx 2\delta^4.$$

In the next experiment, we will examine the difference between the error floor level of different algorithms over a given graph. Figure 5.11 shows the error floor region of the graph G'_1 in Genie, LM, binary SBB and non-binary SBB. The error floor regions corresponding to each of the algorithms for this graph are as follows:

$$P_{ef}^{Genie}(G'_1) \approx 2\delta^4.$$
$$P_{ef}^{nSBB}(G'_1) \approx 9\delta^4.$$
$$P_{ef}^{SBB}(G'_1) \approx 4\delta^3.$$
$$P_{ef}^{LM}(G'_1) \approx 2\delta^2.$$

Note that, based on these expressions, it follows that the slope of the error floors is approximately the power of δ, when we use the logarithmic scale for the curves. As a result, the slope of the error floor can closely be approximated by the size of its smallest problematic set, as it is expected. In this particular example the slope of the curve for Genie, nSBB, SBB, and LM are respectively, 4, 4, 3 and 2, in the error

floor region considering the logarithmic scale.
The dominant LMSSs in the non-binary SBB algorithm are 3 different sets. These sets consist of an LMSS of size 4 which only has 1 nRSSS child of size 4, and there are also two more LMSSs each of size 5. One of them has 5 nRSSS children of size 4 and the other has 3 nRSSS children of size 4. Clearly, the LMSS of size 5 with 5 children of size 4 is more harmful than the other two. Interestingly, this shows that the minimum size LMSS in a graph is not necessarily the most harmful one, as in this case the LMSS of size 4 is less harmful than those of size 5.
Another interesting point in the analysis of this graph is that the error floor in Genie and nSBB are relatively close to each other and they only differ in a constant term. This issue can be seen in Figure 5.11, as well. The performance curve of nSBB algorithm will become parallel to the performance curve of Genie algorithm, and for nSBB it is $\log 9$ dB above the Genie's curve. This issue shows that as we move on towards smaller values of δ the difference between performance of Genie and nSBB will not be changed, while this difference will be increased for LM and SBB as the slopes of their performance curves are not parallel to that of Genie.

5.4.3 Problematic Set Distribution

Clearly the size of a problematic set is an important factor in its harmfulness. In this subsection, we will see how different size problematic sets will contribute to the final failure of an algorithm. Before that, we first cluster the problematic sets as (a, b) problematic sets where a is the size of the problematic set and b is the size its associated set. Furthermore, a and b also can be interpreted as the number of non-zero elements of the failure pattern and number of zero elements of the failure pattern, respectively. Using this classification, we will be able to find a relation between the harmfulness of a problematic sets and the size of the problematic set and its associated set, namely $H-$set or $R-$set. Moreover, we will be able to see relative harmfulness of different problematic sets.
We run 10,000 trials for Monte Carlo simulations on different graphs under different algorithms, and we will examine the number of failures in the graph that are caused by problematic sets in different (a, b) classes. Obviously, since Genie does not contain any zero valued variable node being unverified, all of its classes are of type $(a, 0)$. As a result, we only perform the analysis for LM, SBB, and nSBB algorithms, where b can be non-zero.

We start by comparing different VB MPAs over a randomly selected graph in $\mathcal{G}(60, 2, 3)$. The distribution of failures for different values of δ are shown in Tables 5.6a, 5.7a and 5.8a. It can be seen that for SBB and non-binary SBB algorithms, in this particular graph, all the failures are for (a, b) classes where a is larger than b. whereas in LM algorithm there are a few number of entries in which $b > a$. This is in fact, due to the nature of $H-$set in LM which is always larger than $R-$set in SBB and $H-$set in non-binary SBB. In fact, the fraction of zero unverified variable nodes in LM is larger than the other two algorithms and also between SBB and non-binary SBB, that fraction is larger in SBB. Note that although the zero unverified elements in LM and non-binary SBB are both $H-$sets, in non-binary SBB zero failures are absolutely less than LM's. This is because of the nature of nRSSSs in comparison to SSSs. In fact, nRSSSs potentially accept less number of variable nodes in their corresponding $H-$set than SSSs. Moreover, among (a, b) classes with a given b, those with smaller a are, in most cases, the dominant ones. This could be because of the multiplicity of those sets being larger, or more importantly, because of the fact that the probability of having less number of elements being non-zero is higher than having a large number of elements being non-zero as the values of δ are small here. On the other hand, in (a, b) classes with a given a, there is a certain b for which the harmfulness is maximum. The measurement nodes in the neighbourhood of a problematic set is determined by its non-zero values. Therefore, if number of non-zero elements, or equivalently, a is given, then there will be a certain capacity for zero variable nodes to be connected to those measurement nodes. Among different algorithms, in non-binary SBB, nRSSSs happen to have the least capacity. (See Definition 2.0.8 and 2.0.9 for the relation of the neighbouring measurement nodes with $H-$sets and $R-$sets, which are responsible for failure of zero variable nodes.) Moreover, as the value of δ increases the failures occur over (a, b) sets with larger a, which demonstrate the fact that for larger δ problematic sets with larger size will be activated.

The graph considered in Tables 5.6a, 5.7a and 5.8a is a graph with $d_v = 2$. In terms of the graphs with $d_v = 2$, we know that the smallest SSs are always cycles [72]. That is, a minimum SS of size x in these graphs is, in fact, a cycle of length $2x$ in the graph. For this particular realization, as discussed previously, if we remove the short cycles of the graph using some famous techniques such as Progressive Edge Growth (PEG) [71], then we will see a considerable improvement in the performance of the graph as it cab be seen from Tables 5.6b, 5.7b and 5.8b. Along with the improvement

in the performance of all of VB MPAs using the PEG graph, it can also be seen that in the range of δ that are shown in those tables, the distribution of (a, b) problematic sets are almost the same for all VB MPAs over the PEG graph, by the only difference being the number of failures in each non-zero (a, b) problematic set. This shows that in this range of δ, the main source of differences between the performance of different VB MPAs is the short cycles or more precisely, small problematic sets.

To have a better picture on the amount of improvement by using a PEG graph, depicted in Figure 5.12 are the performance curve of the random graph and the graph constructed by PEG method under non-binary SBB algorithm. Also, note that the degree distribution of the PEG graph is the same as the random one with a slight difference of having two measurement nodes one with degree 4 and the other with degree 2. That is, the PEG graph can almost be considered as a graph in the same ensemble as the random graph. In Figure 5.12, as an example, we can see that when the fraction of non-zero elements equals 0.2, the success probability of non-binary SBB algorithm has about 40% improvement, compared to the random graph. This is because of the fact that for small number of non-zero elements most of the failure patterns in the random graph are caused by SSs of size 2 or 3 ($(2, 0)$ and $(2, 1)$ nRSSSs), or equivalently, cycles of length 4 or 6, as it can be seen in Table 5.8a. Note that, such an improvement can only be achieved in the cases where the concentration result is not applicable, that is for scenarios where either the size of the graph is small or in the error floor region. In this particular example, we used a graph of size 60 for which the concentration result does not hold true.

In the next experiment, we consider another random graph in ensemble $\mathcal{G}(100, 4, 5)$ in larger values of δ, i.e. $\delta \in \{0.3, 0.35, 0.4\}$. Tables 5.9, 5.10 and 5.11 are provided to show how different (a, b) SSSs, RSSSs and nRSSSs contribute to the final failure probability of the selected graph under the LM, binary SBB and non-binary SBB algorithms. As δ increases, it can be seen that the failures also will move to problematic sets with larger a and b values, which means that for each specific value of δ, there are a group of problematic sets that are active. In small values of δ, only small problematic sets are active; however, in larger values of δ, as shown in these tables, there are larger problematic sets that are active. Furthermore, as it can be seen, in LM algorithm, there are number of failures occur in (a, b) problematic sets

Table 5.6: Distribution of (a, b) SSS problematic sets in LM for graphs with $n = 60$, $m = 40$ and $d_v = 2$, at different values of δ. (a) Random graph, (b) PEG graph

(a) Graph chosen randomly

δ	a \ b	0	1	2	3	4	5	6	7
0.1	2	187	246	781	0	0	0	0	0
	3	8	95	212	213	33	0	0	0
	4	2	20	46	70	73	25	0	0
	5	0	3	17	26	40	22	10	1
0.15	2	252	293	1048	0	0	0	0	0
	3	24	206	333	368	48	0	0	0
	4	9	36	137	211	116	50	0	0
	5	0	12	60	86	135	73	42	1
	6	2	3	13	48	50	81	38	30
	7	0	2	3	10	21	28	53	31
	8	0	2	4	5	12	27	31	25
0.25	2	170	171	531	0	0	0	0	0
	3	13	217	273	272	28	0	0	0
	4	8	84	186	256	138	46	0	0
	5	5	34	105	116	202	94	42	8
	6	2	10	42	95	110	161	73	51
	7	2	5	21	57	125	104	148	59
	8	0	0	14	36	59	149	128	116
	9	0	0	6	12	44	81	113	109

(b) Graph constructed by PEG

δ	a \ b	0	1	2	3	4	5	6	7
0.1	2	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0
	4	0	0	9	0	0	0	0	0
	5	0	1	11	0	0	0	0	0
0.15	2	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0
	4	0	0	20	0	0	0	0	0
	5	0	9	35	0	0	0	0	0
	6	0	9	20	8	0	0	0	0
	7	0	3	11	4	1	0	0	0
	8	0	0	1	4	2	0	0	0
0.25	2	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0
	4	0	0	64	0	0	0	0	0
	5	0	53	271	10	0	0	0	0
	6	2	84	224	57	0	0	0	0
	7	2	37	123	38	14	2	0	0
	8	2	10	37	51	42	9	0	0
	9	0	7	20	41	55	19	3	0

Table 5.7: Distribution of (a, b) RSSS problematic sets in binary SBB for graphs with $n = 60$, $m = 40$ and $d_v = 2$, at different values of δ. (a) Random graph, (b) PEG graph

(a) Graph chosen randomly

δ	a \ b	0	1	2	3	4	5	6	7
0.1	2	184	287	656	0	0	0	0	0
	3	9	118	180	97	0	0	0	0
	4	3	38	47	51	13	0	0	0
	5	0	4	15	15	17	0	0	0
0.15	2	277	378	1066	0	0	0	0	0
	3	20	256	392	226	0	0	0	0
	4	12	58	164	131	49	0	0	0
	5	5	28	53	76	54	2	0	0
	6	0	5	19	34	27	18	0	0
	7	0	0	7	15	26	8	3	1
	8	0	1	3	0	10	10	4	5
0.25	2	320	316	964	0	0	0	0	0
	3	39	411	492	320	0	0	0	0
	4	32	163	349	307	117	0	0	0
	5	9	85	207	234	133	2	0	0
	6	7	31	92	193	103	85	2	0
	7	3	9	72	98	140	78	28	0
	8	0	6	21	81	101	119	23	12
	9	0	1	13	33	71	75	79	25

(b) Graph constructed by PEG

δ	a \ b	0	1	2	3	4	5	6	7
0.1	2	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0
	4	0	0	5	0	0	0	0	0
	5	0	1	3	0	0	0	0	0
0.15	2	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0
	4	0	0	18	0	0	0	0	0
	5	0	6	40	2	0	0	0	0
	6	2	7	25	7	0	0	0	0
	7	0	1	11	6	3	0	0	0
	8	0	0	1	3	2	0	0	0
0.25	2	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0
	4	0	0	83	0	0	0	0	0
	5	0	54	266	10	0	0	0	0
	6	5	88	205	49	0	0	0	0
	7	7	39	104	48	14	0	0	0
	8	2	17	24	42	48	8	0	0
	9	0	4	12	34	44	22	3	0

Table 5.8: Distribution of (a, b) nRSSS failures in non-binary SBB for graphs with $n = 60$, $m = 40$ and $d_v = 2$, at different values of δ. (a) Random graph, (b) PEG graph

(a) Graph chosen randomly

δ	a \ b	0	1	2	3	4	5	6	7
0.1	2	308	491	0	0	0	0	0	0
	3	21	54	0	0	0	0	0	0
	4	6	20	27	18	0	0	0	0
	5	4	4	2	0	0	0	0	0
0.15	2	552	905	0	0	0	0	0	0
	3	59	184	0	0	0	0	0	0
	4	17	97	87	38	0	0	0	0
	5	3	20	40	3	0	0	0	0
	6	2	8	13	20	12	2	0	0
	7	0	0	10	10	8	2	0	0
	8	0	0	0	7	3	4	0	0
0.25	2	823	1171	0	0	0	0	0	0
	3	143	370	0	0	0	0	0	0
	4	81	328	324	120	0	0	0	0
	5	29	133	206	35	0	0	0	0
	6	5	72	69	126	54	16	0	0
	7	5	24	81	105	73	21	0	0
	8	0	12	38	87	69	38	5	1
	9	1	4	23	48	73	35	15	1

(b) Graph constructed by PEG

δ	a \ b	0	1	2	3	4	5	6	7
0.1	2	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0
	4	0	0	3	0	0	0	0	0
	5	0	0	4	0	0	0	0	0
0.15	2	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0
	4	0	0	22	0	0	0	0	0
	5	0	6	48	0	0	0	0	0
	6	0	10	25	6	0	0	0	0
	7	0	3	7	3	2	0	0	0
	8	0	0	3	7	0	0	0	0
0.25	2	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0
	4	0	0	74	0	0	0	0	0
	5	0	42	262	0	0	0	0	0
	6	2	90	195	62	0	0	0	0
	7	4	44	111	38	15	0	0	0
	8	2	17	34	46	33	9	0	0
	9	1	4	8	39	64	15	1	0

with $b > a$; however, in other algorithms, we always have $a > b$. Moreover, in this graph, most of the failures in LM algorithm have occurred in (a, b) problematic sets with $b = a$, whereas in other algorithms in most harmful (a, b) problematic sets b is much smaller than a. Again, this issue, reveals the fact, that $H-$set corresponding to an SSS is more likely to have larger size than other associated sets in VB MPAs.

Table 5.9: Distribution of dominant (a, b) SSSs in a random graph chosen from $\mathcal{G}(100, 4, 5)$ in LM algorithm.

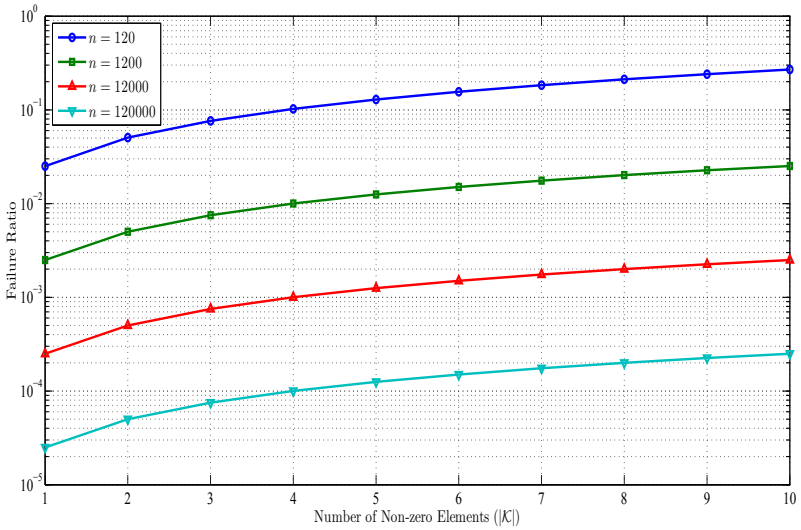
δ	a \ b	0	5	10	15	20	25	30	35	40	45	50	55	60	65
0.3	20	0	0	3	5	1	0	0	0	0	0	0	0	0	0
	25	0	0	1	7	19	11	1	0	0	0	0	0	0	0
	30	0	0	0	1	22	27	24	10	5	0	0	0	0	0
	35	0	0	1	1	6	13	24	10	7	5	1	0	0	0
	40	0	0	0	0	0	0	3	3	1	1	0	2	0	0
0.35	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	20	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	25	0	0	1	3	11	6	1	0	0	0	0	0	0	0
	30	0	0	0	0	19	29	17	6	2	0	0	0	0	0
	35	0	0	0	2	6	17	43	41	18	6	5	1	0	0
	40	0	0	0	0	1	6	20	23	12	9	9	3	0	0
	45	0	0	0	0	0	0	2	7	7	13	5	0	0	0
	50	0	0	0	0	0	0	0	1	2	2	3	0	0	0
0.4	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	20	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	25	0	0	0	2	3	1	0	0	0	0	0	0	0	0
	30	0	0	1	7	6	9	2	2	0	0	0	0	0	0
	35	0	0	0	0	5	16	31	22	12	3	3	0	0	0
	40	0	0	0	0	1	8	27	28	29	30	22	1	2	0
	45	0	0	0	0	0	2	10	13	39	42	31	12	0	0
	50	0	0	0	0	0	0	3	7	29	25	10	0	0	0
	55	0	0	0	0	0	0	0	4	2	4	0	0	0	0

Table 5.10: Distribution of dominant (a, b) RSSSs in a random graph chosen from $\mathcal{G}(100, 4, 5)$ in binary SBB algorithm.

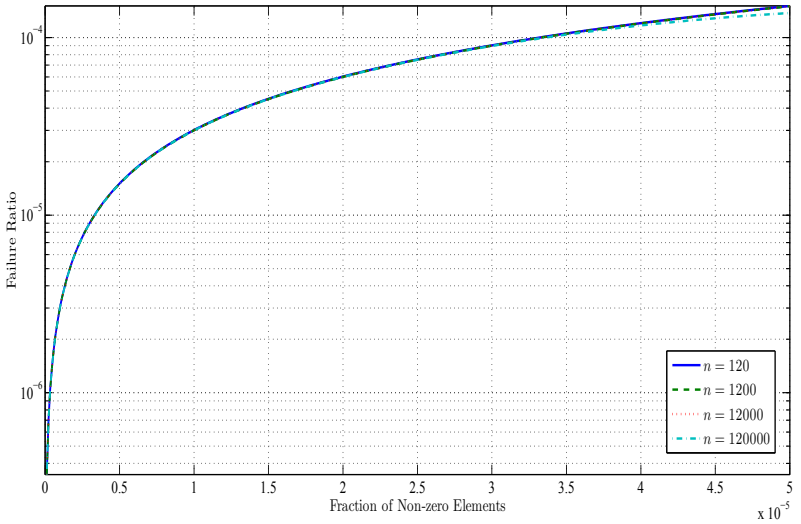
δ	a \ b	0	5	10	15	20	25	30	35	40	45	50	55	60	65
0.3	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	25	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	30	0	0	1	3	1	0	0	0	0	0	0	0	0	0
	35	0	0	0	0	7	4	2	0	0	0	0	0	0	0
	40	0	0	0	0	0	2	0	0	0	0	0	0	0	0
0.35	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	20	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	25	0	1	4	2	0	0	0	0	0	0	0	0	0	0
	30	0	0	3	9	3	0	0	0	0	0	0	0	0	0
	35	0	0	1	5	17	8	3	1	0	0	0	0	0	0
	40	0	0	1	0	5	10	5	5	0	0	0	0	0	0
	45	0	0	0	0	1	2	2	1	3	1	1	0	0	0
	50	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0.4	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	20	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	25	0	1	3	1	0	0	0	0	0	0	0	0	0	0
	30	0	0	7	9	1	0	0	0	0	0	0	0	0	0
	35	0	0	1	15	24	5	2	1	0	0	0	0	0	0
	40	0	0	0	6	14	21	20	3	0	0	0	0	0	0
	45	0	0	0	0	3	12	10	7	3	8	1	0	0	0
	50	0	0	0	0	0	1	2	2	14	17	3	0	0	0
	55	0	0	0	0	0	0	0	2	3	2	0	0	0	0

Table 5.11: Distribution of dominant (a, b) nRSSSs in a random graph chosen from $\mathcal{G}(100, 4, 5)$ in non-binary SBB algorithm.

δ	a \ b	0	5	10	15	20	25	30	35	40	45	50	55	60	65
0.3	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	25	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	30	0	0	0	2	1	0	0	0	0	0	0	0	0	0
	35	0	0	0	0	1	3	1	0	0	0	0	0	0	0
	40	0	0	0	0	1	2	0	1	0	0	0	0	0	0
0.35	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	25	0	0	2	2	0	0	0	0	0	0	0	0	0	0
	30	0	0	4	9	3	0	0	0	0	0	0	0	0	0
	35	0	0	3	11	10	7	0	0	0	0	0	0	0	0
	40	0	0	0	1	7	9	2	2	0	0	0	0	0	0
	45	0	0	0	0	3	2	3	2	1	2	0	0	0	0
	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.4	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	20	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	25	0	2	2	0	0	0	0	0	0	0	0	0	0	0
	30	0	0	6	9	2	0	0	0	0	0	0	0	0	0
	35	0	0	2	19	16	4	0	0	0	0	0	0	0	0
	40	0	0	0	4	16	24	15	1	1	0	0	0	0	0
	45	0	0	0	0	2	16	18	13	2	6	3	0	0	0
	50	0	0	0	0	0	0	2	3	7	5	4	0	0	0
	55	0	0	0	0	0	0	0	1	1	0	0	0	0	0



(a) Failure Ratio vs number of non-zero elements. $d_v = 2,\ d_m = 3$



(b) Failure Ratio vs fraction of non-zero elements. $d_v = 2,\ d_m = 3$

Figure 5.4: Degree 2 variable nodes phenomenon in LM algorithm

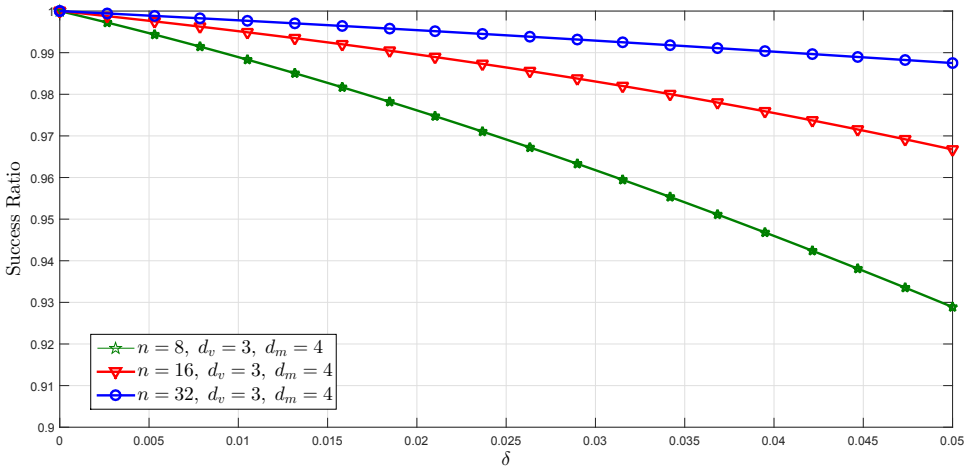


Figure 5.5: Comparison of analytic results of the analysis of different ensembles of graphs in LM.

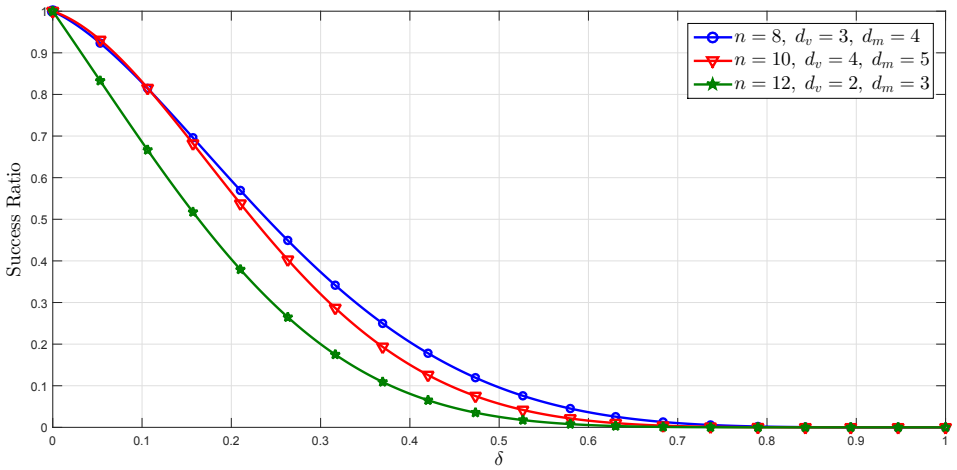


Figure 5.6: Convergence of the success ratio to 1 when $d_v > 2$ in LM algorithm

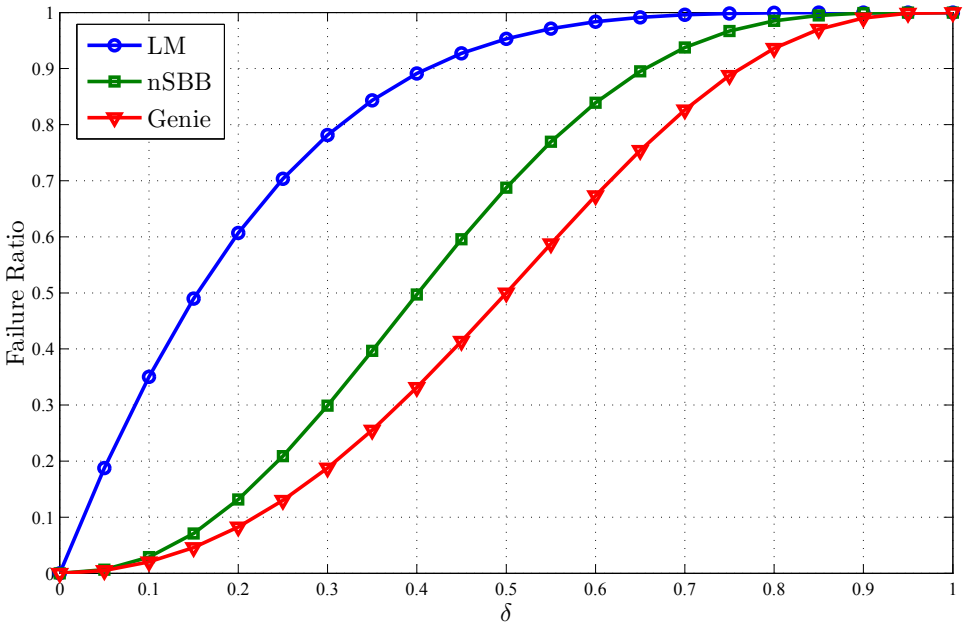


Figure 5.7: The exact probability of failure of the graph shown in Figure 4.2 for different algorithms.

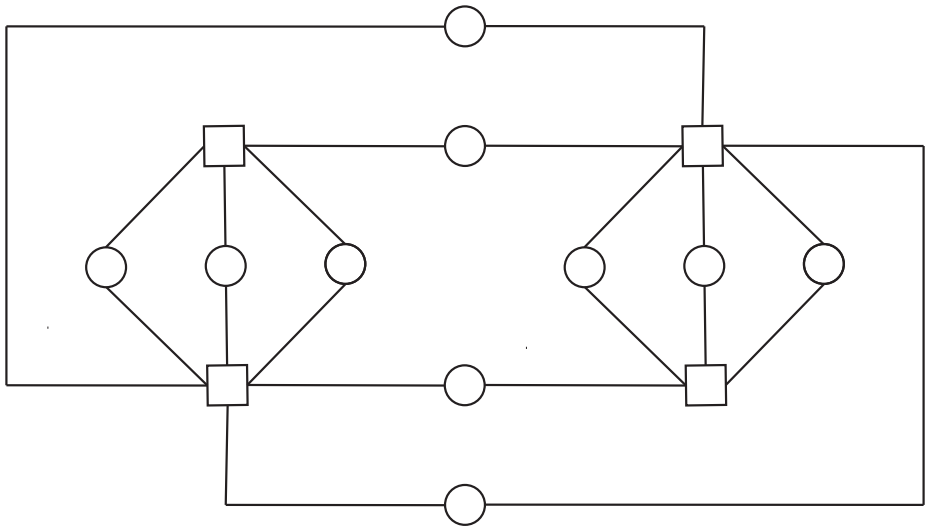


Figure 5.8: A random graph with $n = 10$, $m = 4$ and $d_v = 2$

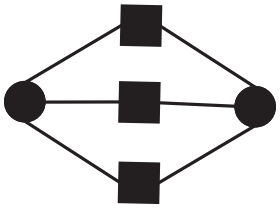


Figure 5.9: Dominant Stopping Set in G_1

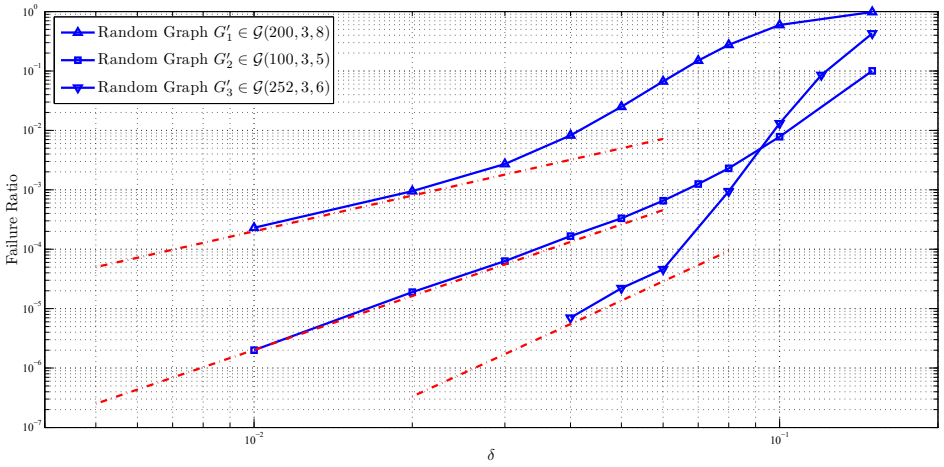


Figure 5.10: Comparison between the simulation and estimation of the error floor for graphs with different dimension and compression ratios under LM algorithm. Dashed lines are the error floor estimation based on dominant LMSSs.

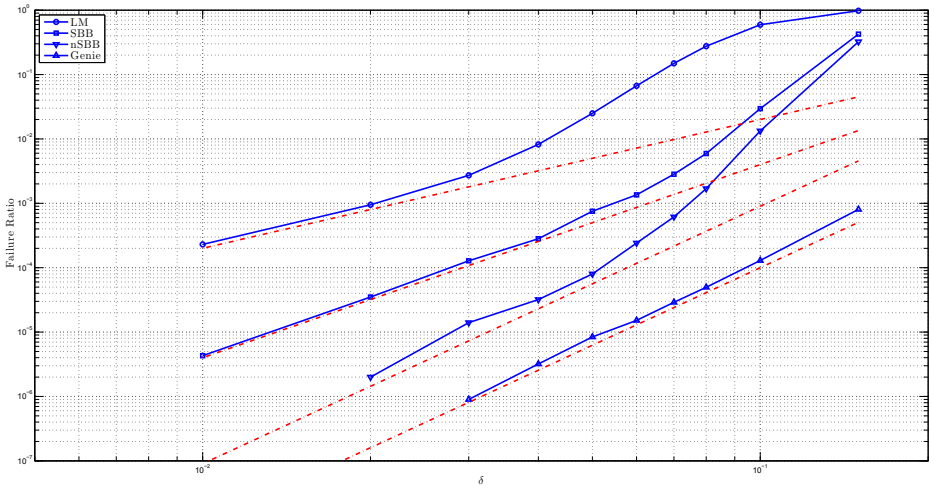


Figure 5.11: Comparison between the simulation and estimation of the error floor for a given random graph $G_1' \in \mathcal{G}(200, 3, 8)$ under different VB MPAs. Dashed lines are the error floor estimation based on dominant LMSSs.

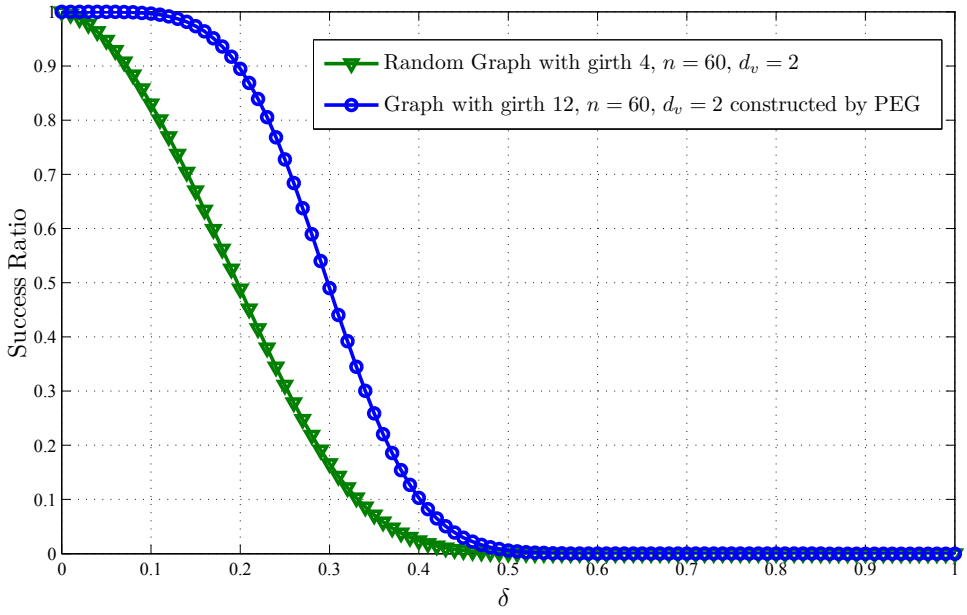


Figure 5.12: Comparison of the performance of non-binary SBB algorithm over a random graph with $n = 60$, $d_v = 2$ and girth 4 and another graph with $n = 60$, $d_v = 2$ and girth 12 constructed by the PEG method.

Chapter 6

Conclusion and Future Works

In this thesis, we characterized the problematic sets for VB MPAs in CS, and we showed, that using this characterization, one can find average probability of failure for a given ensemble of finite length graphs under VB MPAs by providing the analytical expressions for the analysis. To reduce the complexity of the analysis, we also provided a simplified version of the expressions for sensing matrices with $d_v = 2$. Furthermore, we investigated the connections between the problematic sets introduced in this thesis with the well-known SSs. As a result, we were able to come up with stronger relations between CS and LDPC decoding, confirming that Genie algorithm is, in fact, the same as LDPC decoding over BEC using belief propagation decoder.

Moreover, using the characterizations provided, we tackled the problem of individual graph analysis for finite length graphs under VB MPAs in CS. We, then, derived general expressions for the exact probability of failure of a given VB MPA over a given graph. Full investigation of the expressions led us to the discovery of error floor phenomenon in VB MPAs, which is similar to its counter part in coding theory. Furthermore, using the characterizations developed in the thesis, we provided some interesting discussions on the performance of VB algorithms over graphs in the error floor region, including effects of different algorithms and problematic set on the error floor, introducing dominant problematic sets and their contribution in the error floor, along with a discussion on the harmfulness of each problematic set and the idea of error floor estimation by finding only the dominant stopping sets rather than the dominant problematic sets. In fact, the latter idea give us the chance to use a number of available approaches in the literature to find the dominant stopping sets, and then use them to estimate the error floor in VB MPAs. Numerous examples were, also, given for the exact analysis and error floor evaluation. Finally, we provided simulation

results to demonstrate how closely matched the analysis and the simulation results are both in individual graph and error floor estimation.

The whole work presented in the thesis can also shed some lights for future research on VB MPAs. For instance, using the in-depth relation provided between problematic sets in coding and CS, one can formulate even more simplified versions of the analysis, in terms of the computational complexity, specifically by employing tools in coding theory. Moreover, the failure characterizations introduced in this work can be used for new graph construction methods, with good performance in a given ensemble. Furthermore, the error floor analysis paves the road for future research in this topic. Specifically, one important question that remains to be answered is that if there exists error floor phenomenon in other algorithms in CS. The answer to this question will enrich both the theoretical and practical aspects of CS, especially for super-sparse signals. In addition, the exact probability of failure, given for a VB MPA over a given graph, can also be employed, even in coding theory, for further research on SSs. In particular, the method that has been used in this work for the estimation of error floor, which uses dominant SSs to reach to dominant problematic sets using the notion of mother and child problematic sets, can be employed in coding to tackle the error floor estimation in more complicated problems.

List of References

[1] D. L. Donoho, "Compressed sensing," *IEEE Transaction on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[2] A. J. Jerri, "The Shannon sampling theorem - Its various extensions and applications: A tutorial review," *Proceedings of the IEEE*, vol. 65, no. 11, pp. 1565–1596, 1977.

[3] E. J. Candès, "Compressive sampling," in *Proceedings of the International Congress of Mathematicians*, pp. 1433–1452, 2006.

[4] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. E. Kelly, and R. G. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, p. 83, 2008.

[5] J. Haupt and R. Nowak, "Compressive sampling vs. conventional imaging," in *IEEE International Conference on Image Processing*, pp. 1269–1272, 2006.

[6] J. Ma, G. Plonka, and M. Hussaini, "Compressive video sampling with approximate message passing decoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, pp. 1354–1364, Sept 2012.

[7] L.-W. Kang and C.-S. Lu, "Distributed compressive video sensing," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1169–1172, 2009.

[8] J. Prades-Nebot, Y. Ma, and T. Huang, "Distributed video coding using compressive sampling," in *IEEE Picture Coding Symposium*, pp. 1–4, 2009.

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

[10] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, "Compressed sensing MRI," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 72–82, 2008.

[11] U. Gamper, P. Boesiger, and S. Kozerke, "Compressed sensing in dynamic MRI," *Magnetic Resonance in Medicine*, vol. 59, no. 2, pp. 365–373, 2008.

[12] S. F. Cotter and B. D. Rao, "Sparse channel estimation via matching pursuit with application to equalization," *IEEE Transactions on Communications*, vol. 50, no. 3, pp. 374–377, 2002.

[13] G. Taubock and F. Hlawatsch, "A compressed sensing technique for OFDM channel estimation in mobile environments: Exploiting channel sparsity for reducing pilots," in *EEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2885–2888, 2008.

[14] W. U. Bajwa, A. Sayeed, and R. Nowak, "Sparse multipath channels: Modeling and estimation," in *IEEE 13th Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop, DSP/SPE*, pp. 320–325, 2009.

[15] Z. Tian and G. B. Giannakis, "Compressed sensing for wideband cognitive radios," in *IEEE International Conference on Acoustics, Speech and Signal Processing, (ICASSP)*, vol. 4, pp. IV–1357–IV–1360, 2007.

[16] Y. L. Polo, Y. Wang, A. Pandharipande, and G. Leus, "Compressive wide-band spectrum sensing," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pp. 2337–2340, 2009.

[17] I. F. Akyildiz, B. F. Lo, and R. Balakrishnan, "Cooperative spectrum sensing in cognitive radio networks: A survey," *Physical Communication*, vol. 4, no. 1, pp. 40–62, 2011.

[18] L.-H. Chang and J.-Y. Wu, "Compressive-domain interference cancellation via orthogonal projection: How small the restricted isometry constant of the effective sensing matrix can be?," in *IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 256–261, April 2012.

[19] G. Shi, J. Lin, X. Chen, F. Qi, D. Liu, and L. Zhang, "UWB echo signal detection with ultra-low rate sampling based on compressed sensing," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 55, no. 4, pp. 379–383, 2008.

[20] D. Ge, X. Jiang, and Y. Ye, "A note on the complexity of lp minimization," *Mathematical programming*, vol. 129, no. 2, pp. 285–299, 2011.

[21] R. Berinde, A. C. Gilbert, P. Indyk, H. Karloff, and M. J. 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, 2008.

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

[23] M. Duarte, M. Wakin, and R. Baraniuk, "Wavelet-domain compressive signal reconstruction using a Hidden Markov Tree model," in *IEEE International Conference on Acoustics, Speech and Signal Processing,*, pp. 5137–5140, March 2008.

[24] D. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck, "Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit," *IEEE Transaction on Information Theory*, vol. 58, pp. 1094–1121, Feb 2012.

[25] E. J. Candes, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on pure and applied mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.

[26] Y. Eftekhari, A. Heidarzadeh, A. H. Banihashemi, and I. Lambadaris, "Density evolution analysis of node-based verification-based algorithms in compressed sensing," *IEEE Transactions on Information Theory*, vol. 58, no. 10, pp. 6616–6645, 2012.

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

[28] W. Xu and B. Hassibi, "Efficient compressive sensing with deterministic guarantees using expander graphs," in *IEEE Information Theory Workshop,*, pp. 414–419, 2007.

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

[30] F. Zhang and H. Pfister, "Verification decoding of high-rate LDPC codes with applications in compressed sensing," *IEEE Transaction on Information Theory*, vol. 58, pp. 5042–5058, Aug 2012.

[31] M. Akçakaya, J. Park, and V. Tarokh, "Low density frames for compressive sensing," in *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pp. 3642–3645, 2010.

[32] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[33] E. J. Candes, "The restricted isometry property and its implications for compressed sensing," *Comptes Rendus Mathematique*, vol. 346, no. 9, pp. 589–592, 2008.

[34] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A simple proof of the restricted isometry property for random matrices," *Constructive Approximation*, vol. 28, no. 3, pp. 253–263, 2008.

[35] M. Rudelson and R. Vershynin, "Geometric approach to error-correcting codes and reconstruction of signals," *International mathematics research notices*, vol. 2005, no. 64, pp. 4019–4041, 2005.

[36] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[37] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM journal on scientific computing*, vol. 20, no. 1, pp. 33–61, 1998.

[38] D. Needell and R. Vershynin, "Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 310–316, 2010.

[39] V. Chandar, "A negative result concerning explicit matrices with the restricted isometry property," *Tek. Rep.*, 2008.

[40] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[41] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, pp. 18914–18919, 2009.

[42] S. Rangan, "Generalized approximate message passing for estimation with random linear mixing," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*, pp. 2168–2172, 2011.

[43] 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 *IEEE Information Theory Workshop, Punta del Este.*, pp. 160–164, 2006.

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

[45] F. Zhang and H. D. Pfister, "Verification decoding of high-rate LDPC codes with applications in compressed sensing," *IEEE Transactions on Information Theory*, vol. 58, no. 8, pp. 5042–5058, 2012.

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

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

[48] R. Pedarsani, K. Lee, and R. K, "Capacity-approaching phasecode for low-complexity compressive phase retrieval," in *IEEE International Symposium on Information Theory Proceedings (ISIT), Accepted for presentation*, 2015.

[49] X. Li, S. Pawar, and K. Ramchandran, "Sub-linear time compressed sensing using sparse-graph codes," in *IEEE International Symposium on Information Theory Proceedings (ISIT), Accepted for presentation*, 2015.

[50] Z. Zeinalkhani and A. H. Banihashemi, "Low-complexity detection of zero blocks in wideband spectrum sensing," in *IEEE International Symposium on Information Theory Proceedings (ISIT), Accepted for presentation*, 2015.

[51] M. G. Luby and M. Mitzenmacher, "Verification-based decoding for packet-based low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 120–127, 2005.

[52] F. Zhang and H. Pfister, "Analysis of verification-based decoding on the q-ary symmetric channel for large q," *IEEE Transactions on Information Theory*, vol. 57, pp. 6754–6770, Oct 2011.

[53] M. Bayati and A. Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 764–785, 2011.

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

[55] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Transaction on Information Theory*, vol. 48, no. 6, pp. 1570–1579, 2002.

[56] J. Zhang and A. Orlitsky, "Finite-length analysis of LDPC codes with large left degrees," in *IEEE International Symposium on Information Theory,*, p. 3, 2002.

[57] A. Amraoui, A. Montanari, T. Richardson, and R. Urbanke, "Finite-length scaling for iteratively decoded LDPC ensembles," *IEEE Transactions on Information Theory*, vol. 55, no. 2, pp. 473–498, 2009.

[58] A. Orlitsky, K. Viswanathan, and J. Zhang, "Stopping set distribution of LDPC code ensembles," *IEEE Transaction on Information Theory*, vol. 51, no. 3, pp. 929–953, 2005.

[59] A. R. Krishnan, S. Sankararaman, and B. Vasic, "Graph-based iterative reconstruction of sparse signals for compressed sensing," in *10th International Conference on Telecommunication in Modern Satellite Cable and Broadcasting Services (TELSIKS)*, vol. 1, pp. 133–137, 2011.

[60] V. Ravanmehr, L. Danjean, B. Vasic, and D. Declercq, "Interval-passing algorithm for non-negative measurement matrices: Performance and reconstruction analysis," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, pp. 424–432, Sept 2012.

[61] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 599–618, 2001.

[62] O. Ore, "Theory of equivalence relations," *Duke Mathematical Journal*, vol. 9, no. 3, pp. 573–627, 1942.

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

[64] T. Richardson, "Error floors of LDPC codes," in *Proceedings of the annual Allerton conference on communication control and computing*, vol. 41, pp. 1426–1435, 2003.

[65] K. Fu and A. Anastasopoulos, "Stopping-set enumerator approximations for finite-length protograph LDPC codes," in *IEEE International Symposium on Information Theory,*, pp. 2946–2950, June 2007.

[66] M. Hirotomo, Y. Konishi, and M. Morii, "A probabilistic algorithm for finding the minimum-size stopping sets of LDPC codes," in *IEEE Information Theory Workshop,*, pp. 66–70, May 2008.

[67] E. Rosnes and O. Ytrehus, "An efficient algorithm to find all small-size stopping sets of low-density parity-check matrices," *IEEE Transaction on Information Theory*, vol. 55, pp. 4167–4178, Sept 2009.

[68] G. Richter, "Finding small stopping sets in the Tanner graphs of LDPC codes," in *4th International Symposium on Turbo Codes and Related Topics; 6th International ITG-Conference on Source and Channel Coding*, pp. 1–5, 2006.

[69] S. Ranganathan, D. Divsalar, K. Vakilinia, and R. Wesel, "Design of high-rate irregular non-binary LDPC codes using algorithmic stopping-set cancellation," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 711–715, June 2014.

[70] S. Chakravarty and A. Shekhawat, "Parallel and serial heuristics for the minimum set cover problem," *The Journal of Supercomputing*, vol. 5, no. 4, pp. 331–345, 1992.

[71] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Progressive edge-growth Tanner graphs," in *IEEE Global Telecommunications Conference*, vol. 2, pp. 995–1001, 2001.

[72] A. Orlitsky, R. Urbanke, K. Viswanathan, and J. Zhang, "Stopping sets and the girth of Tanner graphs," in *IEEE International Symposium on Information Theory (ISIT)*, p. 2, 2002.