國 立 交 通 大 學

電信工程研究所

碩 士 論 文

穿刺位元的設計針對碼率相容的低密度檢測碼

Design of Puncture Patterns for Rate-Compatible LDPC Codes

研究生：劉殷呈

指導教授：蘇育德 教授

中 華 民 國 一百零一 年 九 月

穿刺位元的設計針對碼率相容的低密度檢測碼

Design of Puncture Patterns for Rate-Compatible LDPC Codes

研 究 生：劉殷呈　　Student：Yin-Cheng Liu

指導教授：蘇育德　教授　　Advisor：Prof. Y. T. Su

國 立 交 通 大 學

電信工程研究所

碩 士 論 文

A Thesis

Submitted to the Institute of Communications Engineering

in partial fulfillment of the requirements

for the Degree of

Master of Science

in

Communications Engineering

at the

National Chiao Tung University

June 2012

Hsinchu, Taiwan

公元 2012 年六月

穿刺位元的設計針對碼率相容的低密度檢測碼

學生：劉殷呈　　指導教授：蘇育德 教授

國立交通大學

電信工程學系碩士班

中文摘要

為了使通道的使用更加有效率，我們會針對同一個錯誤更正碼依據不同的通道狀況使用不同的碼率來傳送資料。為了使穿刺碼的錯誤更正能力的損失降低，我們針對有限長度的低密度檢測碼提供一個建立穿刺位元的方法。我們考慮了穿刺位元的恢復能力以及對其他位元造成的影響來設計我們的演算法並以數學推導得知有哪些參數將對這些產生作用。最後我們以模擬結果以及數據觀測與分析得知我們設計的方向與方法能夠有效降低改錯能力的損失。

Design of Puncture Patterns for Rate-Compatible LDPC Codes

Student : Yin-Cheng Liu Advisor : Y. T. Su

Institute of Communication Engineering

National Chiao Tung University

Abstract

In this thesis, we study puncturing schemes for finite-length rate-compatible low-density parity-check codes. A new bit-by-bit puncturing pattern searching scheme is proposed. The ultimate goal of the proposed method is to improve the recovery error probability of the punctured bits. We also take into account the detrimental effects on previous punctured and unpunctured bits brought about by the new selected punctured. Given the bit locations which have been punctured, a new one is chosen from the set of candidate bits by i) examining its recovery capability (which depends on the number and reliabilities of its connected check node message) and ii) assessing the impact a candidate bit may make. Numerical experimental results show that the proposed scheme outperforms existing puncturing methods. The superiority and robustness of our scheme are further verified by some observed statistics and are consistent with a Gaussian approximation based analytic prediction.

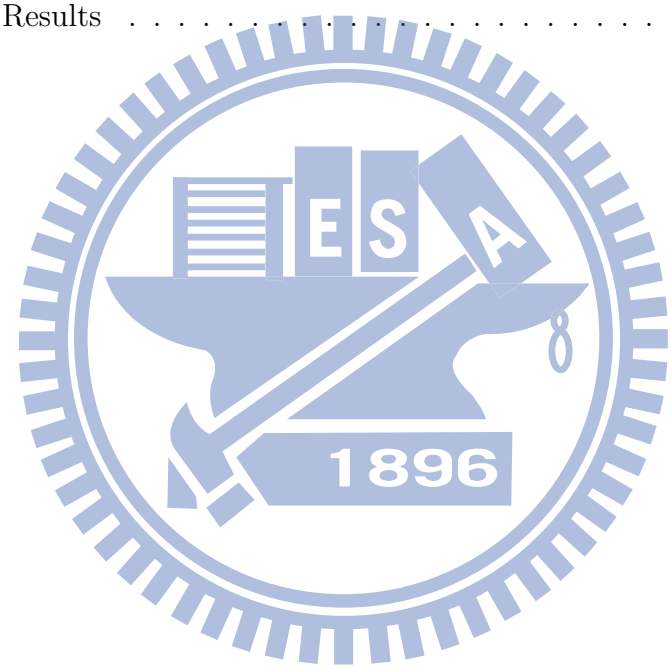
致謝

首先，感謝蘇育德老師的教導，由專題生到研究生，老師的教學與提供優質的研究環境讓學生能夠認真的從事研究並能夠有效率的完成各項研讀與研究。老師也常在 meeting 過後給予我們生活上的啟發，使學生受益良多。

再來要感謝家人的支持與鼓勵，讓我在求學期間能夠無後顧之憂並感受家的溫暖。感謝實驗室的學長姐、同學以及學弟妹，有了你們讓我的碩士生活更加多采多姿。最後感謝 tofar(張致遠)學長、大師兄(翁健家)以及 partner 許晏誠，使得我在研究上得到許多協助與啟發。

實驗室透過聚餐與活動，讓我在求學上增添不少活力，也讓學生和老師間有著更多的認識並讓大家的相處更加融洽。

Contents



List of Figures

Chapter 1

Introduction

Coding schemes that adapt to the channel condition by adjusting the code rate can improve the channel bandwidth efficiency while achieving the required bit error rate (BER) performance. An efficient solution which need only a single codec to implement various code rates requirements is the class of rate-compatible codes. This solution uses a low-rate "mother" code and offers several higher rate codes through puncturing so that only a subset of the original codewords is used to meet a higher rate need. Of course, puncturing modifies the mother code's structure and its distance spectrum and leads to degraded error-rate performance due to the incomplete transmission of coded bits. To minimize the performance loss, punctured bits need to be properly selected. Optimal and near-optimal puncturing patterns for some popular convolutional codes have been intensively studied through semi-analytic and computer-aided approaches.

The class of low-density parity-check (LDPC) codes, which was first introduced by Gallager [8] in early 1960s and rediscovered by Mackay [9] [10] in 1990s, provide near-capacity performance when the so-called belief propagation (BP) or sum-product algorithm (SPA) [11] is used for decoding. It is only natural that one looks for LDPC codes when considering adaptive coding applications. The asymptotic analysis and design of rate-compatible puncturing schemes for LDPC code ensemble has been studied by several experts [1]-[3]. However, the optimization results may not be applicable in non-asymptotic regime (finite length codes). Various puncturing algorithms for finite

length LDPC codes have been suggested in [4]-[6].

We investigate the effects of a punctured bit on the SPA decoding of an LDPC code by examining the massage-passing flows in the associated code graph and derive several guidelines for minimizing the detrimental puncturing effects. More specifically, our puncturing algorithm is designed according to the following principles: 1) Increasing the probability of correct recovery of the punctured nodes. 2) Protecting the nodes which are affected by the punctured nodes. 3) Guaranteeing no stopping sets are contained among punctured nodes.

Chapter 2

Low-Density Parity-Check Codes

Over a BSC channel, the BER performance of low-density codes improves exponentially with the block length by using the maximum likelihood (ML) decoding scheme. Over an AWGN channel, the error probability is upper-bounded by an exponentially decreasing function of the block length [8]. An iterative decoding algorithm that achieves near-capacity performance was developed in [8]. Gallager's iterative decoding algorithm is later recognized as a special instance of the class of so-called belief propagation (BP) algorithms. BP refers to the process of message passing in a graph in which the messages often correspond to some probabilistic or statistic values associated with the nodes of the underlying graph.

2.1 Definitions

An (N, K) LDPC code with K information bits and $(N-K)$ parity bits is a linear block code for which the $M \times N$ parity check matrix $\mathbf{H}$ has a low density of ones [8] where M is the number of check equations. Code rate $R = \frac{K}{N}$ is interpreted as the average number of information bits carried by each code bits.

2.2 Tanner Graph

A Tanner graph [12] is a bipartite graph which is composed of variable nodes and check nodes and edges which connect the two types of nodes. If the jth row and the ith

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

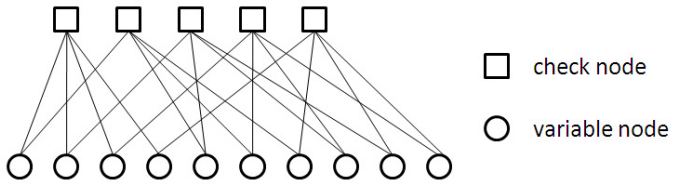


Figure 2.1: The parity check matrix and the corresponding Tanner graph.

column in $\mathbf{H}$ is 1, the check node c_j is connected to variable node v_i. A simple example is shown in Fig. 2.1. The N variable nodes in Tanner graph correspond to N columns of $\mathbf{H}$; the M check nodes in Tanner graph correspond to M rows of $\mathbf{H}$. The number of ones in the ith column of $\mathbf{H}$, $deg(v_i)$, is the degree of variable node v_i; The number of ones in the jth column of $\mathbf{H}$, $deg(c_j)$, is the degree of check node c_j. $d_{v,max}$ and $d_{c,max}$ denote the maximum variable node and check node degree. The variable node and check node degree-distribution polynomials from a "node perspective" are $\lambda(x) = \sum_{i=2}^{d_v} \lambda_i x^i$ and $\rho(x) = \sum_{j=2}^{d_c} \rho_j x^j$, where λ_i and ρ_j denote the fractions of the number of degree-i variable and degree-j check nodes, respectively. Some commonly used notation are defined as follows. $\mathcal{N}(j)$ is the set of the variable nodes (VNs) connected to the jth check node (CN). $\mathcal{M}(i)$ is the set of the check nodes (CNs) connected to the ith variable node (VN).

2.3 Message Passing

Message passing decoding is operated cooperatively and iteratively to decode a received sequence. Intrinsic information and extrinsic information are introduced in message passing principle. An example is shown in Figure 2.2. The message that an

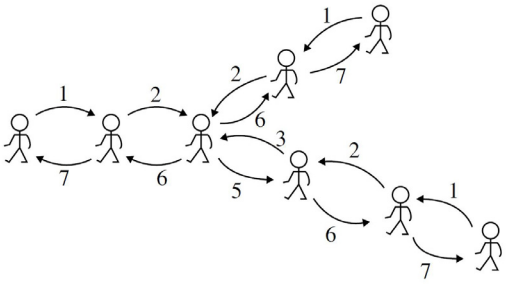


Figure 2.2: Soldier counting example

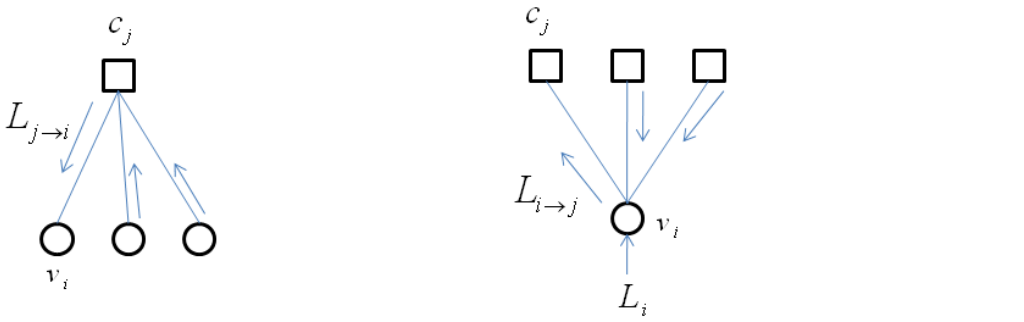
arbitrary soldier X passes to arbitrary neighboring soldier Y is equal to the sum of all incoming messages, plus one for soldier X, minus the message that soldier Y had just sent to soldier X. The sum of the number that a soldier receives from any one of his neighbors plus the number that the soldier passes to that neighbor is equal to the total number of soldiers [16]. The message which is sent from X to Y won't pass back from Y to X in the same iteration. $\mathcal{N}(X)$ is the set of neighbors of X and I_X is the intrinsic information of X and the extrinsic information sent from X to Y is $I_{X\to Y} = I_X + \sum_{Z\in\mathcal{N}(X)\setminus Y} I_{Z\to X}$.

2.4 Sum product Algorithm

Assume $\mathbf{c} = [c_0, c_1.....c_{N-1}]$ is the codeword defined by $C = \{c \in F_2^n : \mathbf{H}\mathbf{c}^T = 0\}$, where F_2 denotes the binary Galois field. After mapping, $x_i = 1-2c_i$ $(\forall\, i = 0 \sim (N-1))$ are transmitted. $\mathbf{y} = [y_0, y_1.....y_{N-1}]$ is the sequence of samples at the output of receiver over the AWGN channel with mean 0 and variance $\sigma^2 = N_0/2$. The log-likelihood ratio (LLR) of x_i:

$$L_i = L(x_i|\mathbf{y}) = log(\frac{\Pr(x_i = 1|\mathbf{y})}{\Pr(x_i = -1|\mathbf{y})}) = 2y_i/\sigma^2 \tag{2.1}$$

$L_{i\to j}$ denotes the message passsed from ith VN to jth CN. $L_{j\to i}$ denotes the message passsed from jth CN to ith VN. Let $\alpha_{i\to j} = \text{sign}(L_{i\to j})$, and $\beta_{i\to j} = |L_{i\to j}|$. Define $\phi(x) = -\log(\tanh(x/2)) = \log(\frac{e^x+1}{e^x-1})$.



message passed from check node c_j to variable node v_i

message passed from variable node v_i to check node c_j

Figure 2.3: An illustrate of LLR message passing between variable and check nodes.

The sum-product algorithm is listed.

- Initialization: Let $L_{i\to j} = L_i \ \forall \ i = 1 \sim N$.
- Step 1. Check-to-variable message updating:

$$L_{j\to i} = \prod_{i' \in \mathcal{M}(j)\backslash i} \alpha_{i'\to j} \times \phi\Big(\sum_{i' \in \mathcal{M}(j)\backslash i} \phi(\beta_{i'\to j})\Big), \tag{2.2}$$

- Step 2. Variable-to-check message updating:

$$L_{i\to j} = L_i + \sum_{j' \in \mathcal{M}(i)\backslash j} L_{j'\to i} \tag{2.3}$$

- Step 3. LLR computation:

$$L_i^{total} = L_i + \sum_{j \in \mathcal{M}(i)} L_{j\to i} \tag{2.4}$$

- Step 4. Make decision: If $L_i^{total} < 0$, let $\hat{c}_i = 1$. Else, let $\hat{c}_i = 0$.
- Step 5. If the number of iteration reach the maximum setting value or $\mathbf{H}\hat{\mathbf{c}}^T = \mathbf{0}$, stop decoding. Otherwise, go back to Step 1.

Chapter 3

Overview of Puncturing LDPC Codes

Transmitting a subset of parity bits of error-correcting codes is called puncturing, and it is assumed that the decoder knows the locations of punctured nodes. Over time-varying channels where channel state information (CSI) is available at transmitter, flexible code rate is desired to improve channel bandwidth efficiency. One can encode at a higher/lower code rate when the channel becomes more/less reliable, respectively.

In multi-carrier sub-channels, to promote error-correcting performance, if the fading gain of one sub-channel is higher/lower than a threshold, we use/discard it. Power reallocation/puncturing patterns are employed to used/discarded sub-channels, respectively. Moreover, there is a limit value existing for the total number of sub-channels which are discarded based on code characteristics.

Punctured codes have another advantage with hybrid automatic-repeat-request (HARQ) protocols. A transmitter sends partial parity bits by puncturing a mother code. If the receiver fails to recover the message, the receiver progressively requests additional parity bits which were previously punctured.

3.1 Introduction to Puncturing at TX and RX

In puncturing, incomplete coded bits are transmitted to receiver, and the error-rate performance may suffer from this. For reducing the performance loss, punctured

nodes need to be properly selected. There are $\binom{N}{x}$ choices for selecting x VNs from N VNs to puncture. Assume that these choices form a set denoted by $\boldsymbol{S}$, the optimal puncture pattern $\boldsymbol{P}$ could be obtained by $\boldsymbol{P} = \arg\min_{\boldsymbol{P}\in\boldsymbol{S}} \sum_{i=1}^{N} P_e(v_i)$, where $P_e(v_i)$ is the error probability of v_i. However, the exhaustive search is NP-hard (it needs to take exponential time) so that the method is hard to implement. Several preceding papers took greedy strategy to design algorithms based on bit-by-bit selection for finding puncturing patterns. The concepts of these algorithms can be understood through observing the iterative message-passing decoding. The decoder needs to know which VNs are punctured, and the error probability of these nodes for random decision is 1/2 (i,e, LLR = 0). Instead of random decision for these nodes, the punctured nodes need to acquire message from their connected CNs with decoding iterations. The process that a punctured node first receives a nonzero message from at least one connected check node is called to be recovered. The error probability of recovered message is called recovery error probability. The recovery procedure is described as follows. Step 1. The unpunctured nodes which received channel values will pass message to some punctured nodes. Step 2. The recovered punctured nodes will then pass message to some other unrecovered punctured nodes. Step 3. Proceeding 2 with decoding iterations and so forth. The different number of decoding iterations for a punctured node needs to be recovered causes different error rate on them. A punctured node which is recovered in the kth decoding iteration is called k-step-recoverable (k-SR) punctured node, and the recovery order of the punctured node is k. Notice that unpunctured nodes are defined as 0-SR. A k-SR punctured node v_i has at least one connected CN c_j, such that the set $\mathcal{N}(j) \setminus \{i\}$ contains at least one $(k-1)$-SR node while the others are m-SR, where $0 \leq m \leq k-1$ [5]. We show an example in Figure 3.1. The recovered message of a k-SR punctured node is from m-SR ($m < k$) punctured nodes', so the information on the average is more unreliable statistically. For a k-SR punctured node, the CNs which pass messages to it in the kth iteration is called Survived check node (SCNs). More-

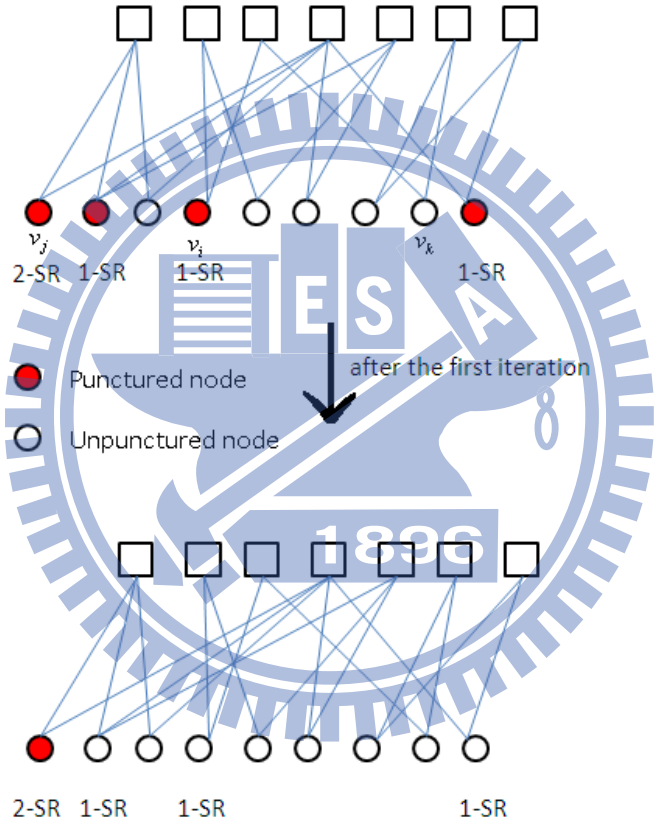


Figure 3.1: recovery of puncture nodes

over, for an unpunctured node, we further define the CNs which pass messages to this unpunctured node in the first iteration as SCNs In Figure 3.1, the number of SCNs of v_i and v_j is two and the number of SCNs of v_k is 1. Some expressions need to be defined as followes. $\mathcal{G}(i) = \{x | x \neq i, x \in \mathcal{N}(j), j \in \mathcal{M}(i)\}$ is the neighboring VNs of the VN v_i. The neighboring VN which is punctured is called neighboring punctured node; the neighboring VN which is unpunctured is called neighboring unpunctured node.

3.2 Puncturing Infinite Length LDPC Codes

The asymptotic analysis and design of rate-compatible puncturing schemes for LDPC code ensemble has been studied in [1]-[3]. In [1] [2], variable nodes with different degrees are divided into different groups and a puncturing distribution is defined to describe the puncturing proportions of each group. A Gaussian approximation method [14] is used to predict and compute the threshold–the lower bound of the theoretical required signal-to-noise ratio (SNR) for error-free decoding [13] for a code ensemble. When variable and check node degree distributions and the fraction of punctured nodes are given, the puncturing distribution is optimized by a linear programming technique to achieve the asymptotic threshold for the code ensemble. An extension work which additionally consider the code structure was proposed in [3]. A generalized check node degree distribution is introduced to describe different code structure: the check nodes are classified based on the degree distributions of their connected variable nodes. The authors [3] found out that the optimal puncturing distribution not only depends on code ensemble degree distribution but also on the code structure.

3.3 Puncturing Schemes for Finite Length LDPC Codes

The asymptotic analysis is suitable for infinite length LDPC codes, but it may have distortion when the optimized results are applied to finite length codes. The analysis of

finite length LDPC codes is more challenging. Several puncturing algorithms for finite length LDPC codes have been suggested in [4]-[6].

The idea in [5] is based on a fact that a punctured node will be recovered with reliable messages. First, the algorithm attempts to maximize the number of punctured nodes that are 1-SR. When no more 1-SR nodes can be found, it proceeds with 2-SR nodes, etc.. Second, in the algorithm, a certain SCN is selected for a punctured node and called guaranteed SCN. The recovery tree is introduced to observe the reliability of the guaranteed SCN information. A tree is built rooted in a punctured node v in the following way. First, v is linked to its guaranteed SCN and next link the SCN to all of its connected VNs excluding v. Then, repeat the preceding process for all new punctured nodes in the tree until every branch ends with an unpunctured node. The number of unpunctured nodes in the recovery tree of v is denoted as $S(v)$. When a punctured LDPC code is transmitted over a BEC with an erasure probability of ϵ, the probability of correct recovery of v in G_k which is composed of all k-SR nodes is expressed in a recursive form:

$$\Psi(v,\epsilon) = \begin{cases} 1-\epsilon & \text{, if } v \in \mathbf{G}_0 \\ \prod_{j=1}^{d_c-1} \Psi(r_j,\epsilon) & \text{, if } v \in \mathbf{G}_k (k>0) \end{cases}. \tag{3.1}$$

The recovery error probability of a punctured node v over a BEC with an erasure probability of ϵ is $(1-\Psi(v,\epsilon))$, where $\Psi(v,\epsilon) = (1-\epsilon)^{S(v)}$. The authors observe that the number of unpunctured nodes in the recovery tree of a punctured node is positive correlated to the probability of successful recovery. The candidate v with lower recovery order and with smaller $S(v)$ will be a new punctured node. A *grouping* algorithm based on the recovery order and the number of unpunctured nodes in the recovery tree is proposed. Within each group, *sorting* algorithm is proposed to determine the puncturing priority of these selected nodes.

Based on Gaussian approximation (GA), the authors [6] stated that more number of SCNs of a punctured node, the lower recovery error rate is. $C_i^{(k)}$ denotes a SCN i of the

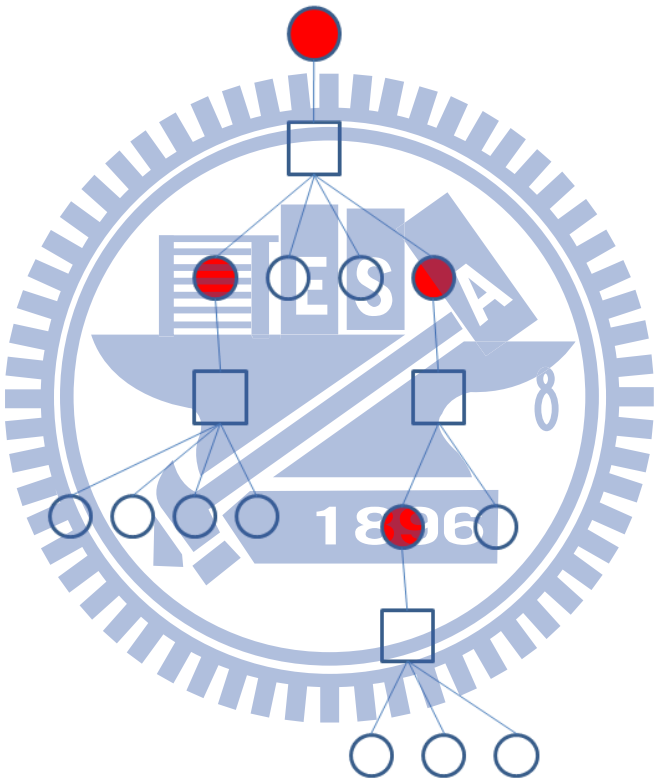


Figure 3.2: recovery tree of v

k-SR punctured node $v^{(k)}$. The value $S(C_i^{(k)})$ is the total number of unpunctured nodes under the SCN $C_i^{(k)}$. $m_{u,i}^{(k)}$ denotes the mean LLR value from the survived check node i of a k-SR punctured node to the k-SR punctured node. $m_v^{(k)}$ denotes the mean LLR value from a k-SR punctured node to the survived check node of a $(k+1)$-SR punctured node. $m_v^{(0)}$ is the mean LLR of channel value. $N_{SC}(v^{(k)})$ denotes the number of survived check nodes of the k-SR punctured node $v^{(k)}$. When observing the k-SR punctured node, it is assumed that the m-SR ($m < k$) punctured nodes only has a SCN. Based on the assumption, the recovery error probability of a punctured node $v \in G_k$ over an Additive White Gaussian Channel with Gaussian Approximation is $P_e^{(R)}(v^{(k)}) = Q(\sqrt{m_v^{(k)}/2})$, where $m_v^{(k)} = \sum_{i=1}^{N_{sc}(v^{(k)})} \Phi^{-1}(1-[1-\Phi(m_v^{(0)})]^{S(C_i^{(k)})})$, and

$$\Phi(x) = \begin{cases} 1 - \frac{1}{\sqrt{4\pi x}} \int \tanh \frac{u}{2} e^{-(u-x)^2/4x} du & , \text{if } x > 0 \\ 1 & , \text{if } x = 0 \end{cases}. \tag{3.2}$$

In the simulation, all punctured nodes are 1-SR and every punctured node has the same number of SCNs. The result is shown that the performance of the punctured code with more SCNs is better.

In [7], another puncturing scheme with better performance compared with those in [5] was proposed. The punctured nodes are selected as far apart from each other in the Tanner graph of the code as possible. In each round, the candidates are composed of the VNs which are at least a distance of four away from punctured nodes which had been selected in the round, and have at least one connected check node which connects to all unpunctured nodes. A new punctured node is randomly selected from candidates, and the neighboring variable nodes are excluded from the candidate set in the round. The punctured-patterns-selecting procedure are terminated when the candidate set is empty in a round and the set will be regenerated for the next round. Each round of the scheme returns a set of punctured nodes that the average error probability of an unpunctured node is minimum after 1st iteration of the message passing. When the kth round ($k \geq 1$) is processed, the punctured nodes may be m-SR, where $1 \leq m \leq k$. Moreover, *Additional puncturing scheme* (A-puncturing scheme) is proposed to achieve

higher puncturing code rate. It randomly selects an unpunctured node that connects to the least number of check nodes that involve only one punctured node selected in the previous puncturing scheme as a new punctured node. This ensures that when the punctured node is selected in A-puncturing scheme, the reduction in the probability of correctly recovery of 1-SR punctured nodes selected in the previous puncturing scheme is minimized.

3.3.1 Some thoughts about existing puncturing schemes

The recovery tree [5] is based on some assumptions. Assumption 1: If the maximum recovery order of punctured nodes is k, the mother code is either cycle-free or has a girth larger than $2(k+1)$. Assumption 2: Every punctured node is connected to its guaranteed SCN only, that is, it does not consider the actual number of SCNs. Assumption 3: The unpunctured nodes only pass their channel values even after they have received updated messages. Based on Assumption 1, the nodes are not repeated in the tree or no cycles exist in the tree. Based on Assumptions 2 and 3, a punctured node v with a smaller $S(v)$ has a smaller recovered error probability.

Although multiple SCNs are considered in [6], the DE with GA of a k-SR punctured node is based on the assumption that each m-SR ($m < k$) punctured node only has a SCN. Besides, despite of [5] or [6], the message of unpunctured nodes are not updated increment of iteration number; that is, unpunctured nodes only pass the received channel values out. Moreover, the method only cares about the number of SCNs, but which CNs will be the guaranteed SCNs are not considered, and which candidate with the number of SCNs as a punctured node is also not considered. Moreover, We know that the fixed number of guaranteed SCNs is restricted by the degree of VNs, and the achievable puncturing code rate is restricted by the fixed number of guaranteed SCNs. The authors don't provide a method to find the optimal number of guaranteed SCNs to enhance the overall performance. It is also not known if using a fixed number of guaranteed SCNs is

a better strategy than the one using different number of guaranteed SCNs for different punctured nodes.

In [7], although the punctured nodes in each round are at least a distance of four away from each other, the edge-distance among punctured nodes in all rounds is not expected to be so. In A-puncturing scheme, less 1-SR punctured nodes selected in previous scheme whose SCNs decrease due to the new punctured node, but it doesn't concern about the $k(>1)$-SR punctured nodes selected in previous puncturing scheme and these new punctured nodes selected in A-puncturing scheme.

Chapter 4

A New Puncturing Algorithm

4.1 Design Guidelines

To find an optimal puncturing set $\boldsymbol{P}$, exhaustive search can be used, but it is time-consuming. In contrast, finding the puncturing set in a bit-by-bit selection manner is a simpler approach. In this approach, a candidate that can minimize the average bit error rate (BER) is selected as a new punctured node; that is, a punctured node v_i is chosen by

$$i = \arg_i \min_{\substack{v_i \in \boldsymbol{V} \\ \boldsymbol{P} = \boldsymbol{P} \bigcup v_i}} \frac{1}{N} \sum_{j=1}^{N} P_e(v_j), \tag{4.1}$$

where $\boldsymbol{V}$ is the set of candidate punctured nodes, i.e., all unpunctured nodes, and $P_e(v_j)$ is the error probability of v_j. Based on (4.1), it is straightforward to see that the average BER of the N VNs is highly related to the recovery error probability of the candidate punctured node v_i and the error probability of other nodes if v_i is punctured. Consider the AWGN channel with the mean LLR of channel value μ_0 and use the Gaussian approximation (GA) to analyze the BER. Let $\mu_{v_i^{(0)}}^{(1)}$ denote the mean LLR of unpunctured node v_i in the first iteration and $\mu_{v_i}^{(k)}$ denotes the mean LLR of k-SR punctured node v_i in the kth iteration, where $k \in Z^+$. Let $\boldsymbol{S}_{\mathrm{CN}}(v_i)$ be the set of SCNs of v_i. Let $\alpha_j^{(m)}(v_i)$ be the number of m-SR nodes in the set $\{\mathcal{N}(j) \setminus i\}$, where c_j is the SCN of v_i. The

mean LLR of unpunctured nodes and 1-SR punctured nodes in the first iteration are as follows:

$$\begin{cases} \mu_{v_i^{(0)}}^{(1)} = \mu_0 + \sum\limits_{j \in \boldsymbol{S}_{\mathrm{CN}}(v_i^{(0)})} \phi^{-1}(1-[1-\phi(\mu_0)]^{deg(c_j)-1}) \\ \mu_{v_i}^{(1)} = \sum\limits_{j \in \boldsymbol{S}_{\mathrm{CN}}(v_i)} \phi^{-1}(1-[1-\phi(\mu_0)]^{deg(c_j)-1}) \end{cases}, \tag{4.2}$$

where $\phi(x)$ and $\phi^{-1}(x)$ are decreasing function and $0 < \phi(x), \phi^{-1}(x) < 1$. We assume that $\mu_{v,\ m-SR}^{(k-1)}$ is the average mean LLR of m-SR nodes in the $(k-1)$th iteration and we further obtain the mean LLR of k-SR punctured nodes in the kth iteration.

$$\mu_{v_i}^{(k)} = \sum_{j \in \boldsymbol{S}_{\mathrm{CN}}(v_i)} \phi^{-1}(1-\prod_{m=0}^{(k-1)}[1-\phi(\mu_{v,\ m-SR}^{(k-1)})]^{\alpha_j^{(m)}(v_i)}),\ k=2,3,... \tag{4.3}$$

The error probability of unpunctured node v_i in the first iteration is $Q(\sqrt{\mu_{v_i^{(0)}}^{(1)}/2})$ and the recovery error probability of k-SR punctured node v_i is $P_e^{(R)}(v^{(k)}) = Q(\sqrt{\mu_{v_i}^{(k)}/2})$. From (4.2) and (4.3), we know that the indicator parameters such as the number of SCNs, the degree of the SCNs and the connected nodes of the SCNs determine the error probability. It can be roughly known that a node v_i a) has more SCNs and the degree of these SCNs are smaller and b) the recovery order of the nodes which are connected to these SCNs are smaller, the error probability of v_i is lower. We further discuss about (4.1) by the indicator parameters to obtain several puncturing guidelines. The details are given as follows.

First of all, we show two examples to illustrate how the recovery error probability of a punctured node is affected. Consider the BEC with erasure probability ϵ in Figure 4.1, the probability of correct recovery of v_i is $(1-\epsilon)^2$; the probability of correct recovery of $v_{i'}$ is $2(1-\epsilon)^2-(1-\epsilon)^4$. Thus, the probability of correct recovery of $v_{i'}$ is larger than v_i in BEC. Similarly, we consider the AWGN channel in Figure 4.1. $\mu_{v_i}^{(1)} = \Phi^{-1}(1-[1-\phi(\mu_0)]^2)$; $\mu_{v_{i'}}^{(1)} = 2\Phi^{-1}(1-[1-\phi(\mu_0)]^2)$. The recovery error probability of $v_{i'}$ is smaller than v_i.

Guideline 1: The recovery error probability of the punctured node with more SCNs

will be lower, i,e, $|\boldsymbol{S}_{\mathrm{CN}}(v_i)| \uparrow \Rightarrow P_e^{(R)}(v_i) \downarrow$.

Another example is shown in Figure 4.2. Consider the BEC and assume that the average error probability of unpunctured nodes in the first iteration is ϵ_0 and the average recovery error probability of 1-SR punctured nodes is ϵ_1. Statistically, it is reasonable to assume that $\epsilon_1 > \epsilon_0$. The probability of correct recovery of v_i is $(1-\epsilon_1)+(1-\epsilon_0)(1-\epsilon_1)^2-(1-\epsilon_0)(1-\epsilon_1)^3$; the probability of correct recovery of $v_{i'}$ is $(1-\epsilon_1)+(1-\epsilon_0)^2(1-\epsilon_1)-(1-\epsilon_0)^2(1-\epsilon_1)^2$; the probability of correct recovery of $v_{i''}$ is $(1-\epsilon_1)+(1-\epsilon_0)(1-\epsilon_1)-(1-\epsilon_0)(1-\epsilon_1)^2$. Thus, if $\epsilon_1 > \epsilon_0$, the probability of correct recovery of $v_{i''}$ is larger than that of $v_{i'}$ and the probability of correct recovery of $v_{i'}$ is larger than that of v_i in BEC. Analogously, the AWGN channel is considered in Figure 4.2 and on the average $\mu^{(1)}_{v,\ 1-SR}$ is assumed to be smaller than $\mu^{(1)}_{v,\ 0-SR}$. $\mu^{(2)}_{v_i} = \Phi^{-1}(1-[1-\phi(\mu^{(1)}_{v,\ 1-SR})])+\Phi^{-1}(1-[1-\phi(\mu^{(1)}_{v,\ 0-SR})][1-\phi(\mu^{(1)}_{v,\ 1-SR})]^2)$; $\mu^{(2)}_{v_{i'}} = \Phi^{-1}(1-[1-\phi(\mu^{(1)}_{v,\ 1-SR})])+\Phi^{-1}(1-[1-\phi(\mu^{(1)}_{v,\ 0-SR})]^2[1-\phi(\mu^{(1)}_{v,\ 1-SR})])$; $\mu^{(2)}_{v_{i''}} = \Phi^{-1}(1-[1-\phi(\mu^{(1)}_{v,\ 1-SR})])+\Phi^{-1}(1-[1-\phi(\mu^{(1)}_{v,\ 0-SR})][1-\phi(\mu^{(1)}_{v,\ 1-SR})])$. Thus, $P_e^{(R)}(v_{i''}) = Q(\sqrt{\mu^{(2)}_{v_{i''}}/2}) < P_e^{(R)}(v_{i'}) = Q(\sqrt{\mu^{(2)}_{v_{i'}}/2}) < P_e^{(R)}(v_i) = Q(\sqrt{\mu^{(2)}_{v_i}/2})$. The differences of the recovery error probability among these three VNs result from the reliability of their SCNs, i.e., $R(c_j) < R(c_{j'}) < R(c_{j''})$, where $R(c_j)$ denotes the reliability of c_j.

Guideline 2: If $deg(c_{j1}) = deg(c_{j2})$ and c_{j1} connects to more unpunctured nodes and less punctured nodes, c_{j1} is more reliable than c_{j2}. Moreover, the smaller recovery order of these connected punctured nodes are, the more reliable of the CN is. For example, $R(c_j) < R(c_{j'})$ in Figure 4.2.

Guideline 3: If two CNs connect to the same number of punctured nodes, the CN with lower degree is more reliable. For example, $R(c_{j'}) < R(c_{j''})$ in Figure 4.2.

From the two examples, for reducing the recovery error probability of punctured nodes, the punctured nodes should have more SCNs and the reliability of these SCNs should be higher.

Second, we show an example in Figure 4.3 to discuss about the error probability

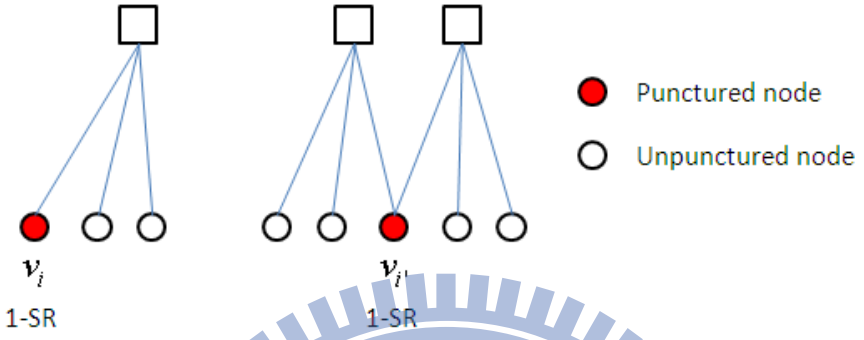


Figure 4.1: An example of the number of SCNs.

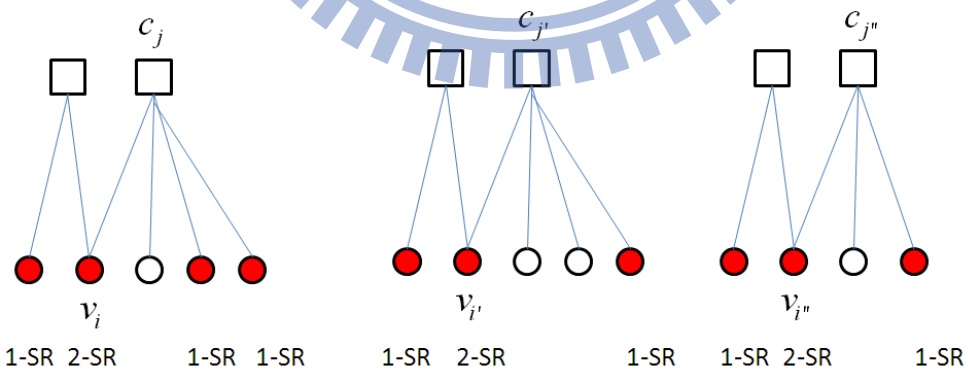


Figure 4.2: An example of check reliability.

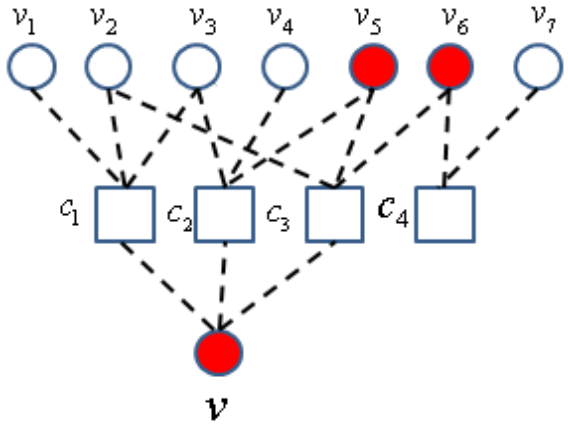


Figure 4.3: An example of the effect on the neighboring VNs of a punctured node.

of other nodes if a candidate is punctured. Consider the BEC with erasure probability ϵ, if v is unpunctured, the correct probability of v_1 in the first iteration is $(1-\epsilon)+(1-\epsilon)^3-(1-\epsilon)^4$; if v is punctured, c_1 cannot pass message to v_1 in the first iteration and the correct probability of v_1 in the first iteration is $(1-\epsilon)$. Consider the AWGN channel in Figure 4.3. If v is unpunctured, $\mu_{v_1^{(0)}}^{(1)} = \mu_0 + \Phi^{-1}(1-[1-\phi(\mu_0)]^3)$; if v is punctured, $\mu_{v_1^{(0)}}^{(1)} = \mu_0$. If a node is punctured, the error probability of the neighboring VNs of the punctured node will arise.

Guideline 4: If a node is punctured, the number of SCNs of some neighboring VNs will decrease such as v_1, v_2, v_3 and v_6 in Figure 4.3; the recovery order of some neighboring punctured nodes will increase such as v_5 in Figure 4.3. Thus, the error probability of these nodes are statistically higher.

Next, we classify the neighboring VNs whose SCNs will decrease if a candidate is punctured. One is the neighboring unpunctured nodes which connect to SCN of the punctured node if the recovery order of the punctured node is 1, for example, v_1, v_2, v_3 in Figure 4.2; Another is the neighboring punctured nodes of the punctured node despite of its recovery order, for example, v_5, v_6 in Figure 4.2. When a new punctured node is determined, we need to take care of the neighboring VNs.

Guideline 5: The recovery order of every punctured node needs to be under control such that the punctured nodes have lower recovery order and avoid stopping set contain only punctured nodes.

All in all, we select a new punctured node from candidates, the recovery error probability and the effect on other nodes need to be considered.

4.2 Discovering Good Puncture Patterns

4.2.1 Notations and Definitions

Let $\boldsymbol{P}^{(t)}$ denote the set of t-SR punctured nodes. If $\boldsymbol{A}$ is a set, let $|\boldsymbol{A}|$ be the cardinality of the set $\boldsymbol{A}$. Moreover, we introduce an *l-SR constraint* to control the recovery order of punctured nodes; that is when a candidate is punctured, the recovery order of neighboring punctured nodes of the candidate can be only less than or equal to l.

The next we define some quantities to measure which candidate is more suitable to be punctured. For some $v_i \in \boldsymbol{V}$, assume candidate v_i is punctured ($\boldsymbol{P} = \boldsymbol{P} \bigcup v_i$), we propose the following five subroutines to obtain appropriate information for puncturing:

- The number of SCNs of the neighboring punctured nodes of v_i are observed in Subroutine (A).
- The number of SCNs of v_i is obtained in Subroutine (B).
- The sum of the degree of these SCNs of v_i is obtained in Subroutine (C).
- If the recovery order of v_i is 1, the number of SCNs of the neighboring unpunctured nodes which connect to SCN of v_i are observed in Subroutine (D).
- If the recovery order of v_i is t and t is lager than 1, the number of m-SR ($0 \leq m \leq (t-1)$) nodes which connect to SCNs of v_i are observed in subroutine (E).

In Subroutine (B), (C) and (E), we examine the recovery error probability of the candidate punctured node. In Subroutine (A) and (D), we measure the influence on previous selected punctured nodes and related unpunctured nodes, which is caused when the candidate is selected to puncture.

Subroutine (A):

1. Let $\mathbf{\Gamma}^{(m)}(i) = \{k | k \in \mathcal{G}(v_i) \bigcap \boldsymbol{P}^{(m)}\}$ denote the set of the neighboring m-SR punctured nodes of v_i.

2. The average number of SCN of the nodes in the set $\mathbf{\Gamma}^{(m)}(i)$ is

$$\bar{S}(\mathbf{\Gamma}^{(m)}(i)) = \begin{cases} \frac{1}{|\mathbf{\Gamma}^{(m)}(i)|} \sum\limits_{k \in \mathbf{\Gamma}^{(m)}(i)} |\boldsymbol{S}_{\mathrm{CN}}(v_k)| & , |\mathbf{\Gamma}^{(m)}(i)| \neq 0 \\ 0 & , |\mathbf{\Gamma}^{(m)}(i)| = 0 \end{cases} . \tag{4.4}$$

Subroutine (B):

1. Assume the recovery order of v_i is t, let the set $\boldsymbol{S}_{\mathrm{CN}}^{(t)}(v_i)$ be the SCNs of v_i,

2. The number of SCNs of v_i is $|\boldsymbol{S}_{\mathrm{CN}}^{(t)}(v_i)|$.

Subroutine (C):

Assume the recovery order of v_i is t, The sum of the degree of these SCNs of v_i is

$$d_{\mathrm{SCN}}(v_i) = \sum_{j \in \boldsymbol{S}_{\mathrm{CN}}^{(t)}(v_i)} deg(c_j) \tag{4.5}$$

Subroutine (D):

1. Assume the recovery order of v_i is 1, $\mathbf{\Lambda}(i) = \{k | k \in N(j) \setminus i \ , j \in \boldsymbol{S}_{\mathrm{CN}}^{(1)}(v_i)\}$ denotes the neighboring unpunctured nodes which connect to SCN of 1-SR v_i.

2. The average number of SCN of the nodes in the set $\mathbf{\Lambda}(i)$ is

$$\bar{S}(\mathbf{\Lambda}(i)) = \frac{1}{|\mathbf{\Lambda}(i)|} \sum_{k \in \mathbf{\Lambda}(i)} |\boldsymbol{S}_{\mathrm{CN}}(v_k)| \tag{4.6}$$

Subroutine (E):

1. Assume the recovery order of v_i is $t(>1)$, $\alpha_j^{(m)}(v_i)$ denotes the number of m-SR nodes in the set $\{\mathcal{N}(j) \setminus i\}$, where c_j is the SCN of v_i.

2. The sum of the number of m-SR nodes which each SCN of v_i connects is

$$\pi^{(m)}(v_i) = \sum_{j \in \boldsymbol{S}_{\mathrm{CN}}^{(t)}(v_i)} \alpha_j^{(m)}(v_i), \tag{4.7}$$

where $0 \leq m \leq (t-1)$.

4.2.2 Proposed Algorithm

In our algorithm, the candidate set is composed of all the unpunctured nodes and the candidate set is regenerated and reduced step by step such that a new punctured node is acquired. Each step is like a "sieve" to discard some candidates according to the quantities obtained by each subroutine. Several examples are illustrated to explain how to reduce the candidate set by each quantity. (Note that the candidate v_i is assumed to be punctured in the following examples (from Figure 4.4 to Figure 4.15).)

First, l-SR constraint is applied to candidates. If one of the nodes remained in the candidate set is chosen to be punctured, it would not induce the recovery order unexpected increment of the punctured nodes and the maximum recovery order of the new punctured node is $l+1$. For example, if 0-SR constraint is applied, the candidate v_i in in Figure 4.4 (a) is discarded; if 1-SR constraint is applied, the candidate v_i in Figure 4.5 (a) is discarded. Notice that when 0-SR constraint is applied, it means that the punctured nodes are at least a distance of four away in the Tanner graph.

Second, if a node remained in the candidate set after Subroutine (A) is punctured, the average number of SCNs of its neighboring punctured nodes are larger than that of selecting any node not belonging to the candidate set. Moreover, the neighboring punctured nodes with lower recovery order should have stronger protection (let it have more SCNs). If the recovery order of neighboring punctured nodes of a candidate are all higher than those of other candidates, the priority of the candidate as a new punctured node is lower. For example, $\bar{S}(\boldsymbol{\Gamma}^{(1)}(i))$=1 in Figure 4.6 (a); $\bar{S}(\boldsymbol{\Gamma}^{(1)}(i))$=1.5 in Figure 4.6

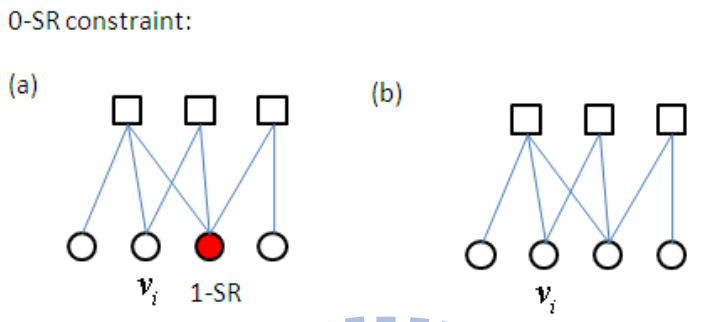


Figure 4.4: An example of l-SR constraint (I).

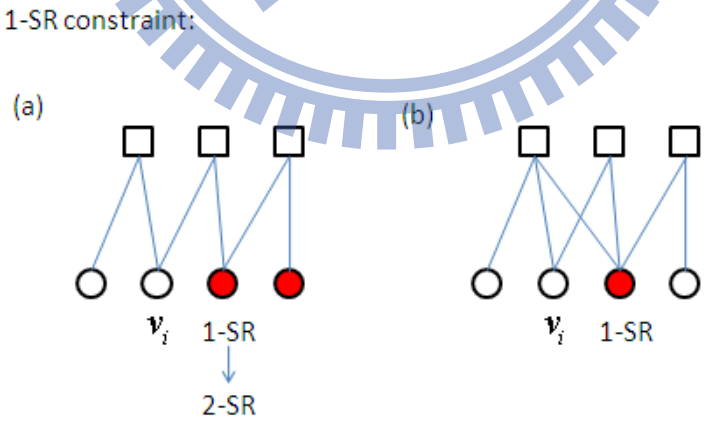


Figure 4.5: An example of l-SR constraint (II).

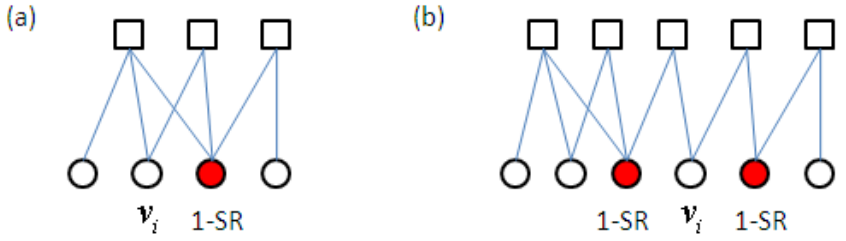


Figure 4.6: The neighboring punctured nodes of v_i (I).

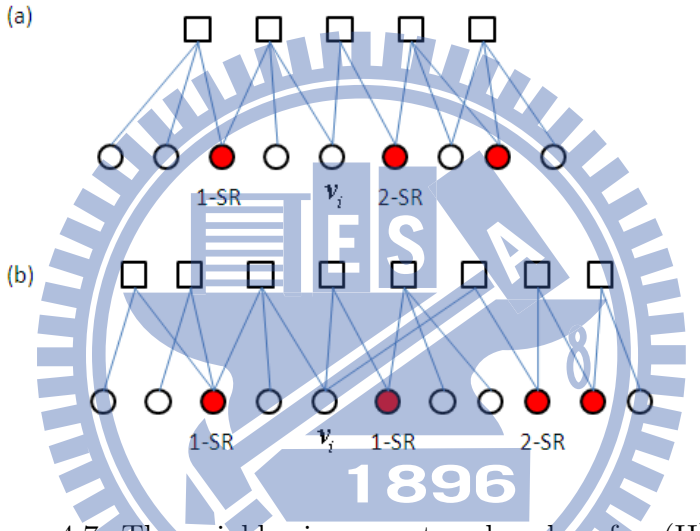


Figure 4.7: The neighboring punctured nodes of v_i (II).

(b), so v_i in Figure 4.6 (a) is discarded. $\bar{S}(\mathbf{\Gamma}^{(1)}(i))$=1 in Figure 4.7 (a); $\bar{S}(\mathbf{\Gamma}^{(1)}(i))$=1.5 in Figure 4.7 (b), no matter what the value $\bar{S}(\mathbf{\Gamma}^{(2)}(i))$ is, v_i in Figure 4.7 (a) is discarded. $\bar{S}(\mathbf{\Gamma}^{(1)}(i))$=1 in Figure 4.8 (a); $\bar{S}(\mathbf{\Gamma}^{(1)}(i))$=1 in Figure 4.8 (b); $\bar{S}(\mathbf{\Gamma}^{(1)}(i))$=0 in Figure 4.8 (c), so v_i in Figure 4.8 (c) is discarded. $\bar{S}(\mathbf{\Gamma}^{(2)}(i))$=1 in Figure 4.8 (a); $\bar{S}(\mathbf{\Gamma}^{(2)}(i))$=2 in Figure 4.8 (b), so v_i in Figure 4.8 (a) is discarded.

Third, The candidate punctured node with lower recovery order and more SCNs has a higher priority as a new punctured node. For example, $|\boldsymbol{S}_{\mathrm{CN}}^{(1)}(v_i)|$= 2 in Figure 4.9 (a); $|\boldsymbol{S}_{\mathrm{CN}}^{(1)}(v_i)|$= 3 in Figure 4.9 (b). In the comparison, v_i in Figure 4.9 (a) is discarded. $|\boldsymbol{S}_{\mathrm{CN}}^{(2)}(v_i)|$= 1 in Figure 4.10 (a); $|\boldsymbol{S}_{\mathrm{CN}}^{(2)}(v_i)|$= 2 in Figure 4.10 (b). In the comparison, v_i in Figure 4.10 (a) is discarded. The recovery order of v_i in Figure 4.11 (a) is 2; the

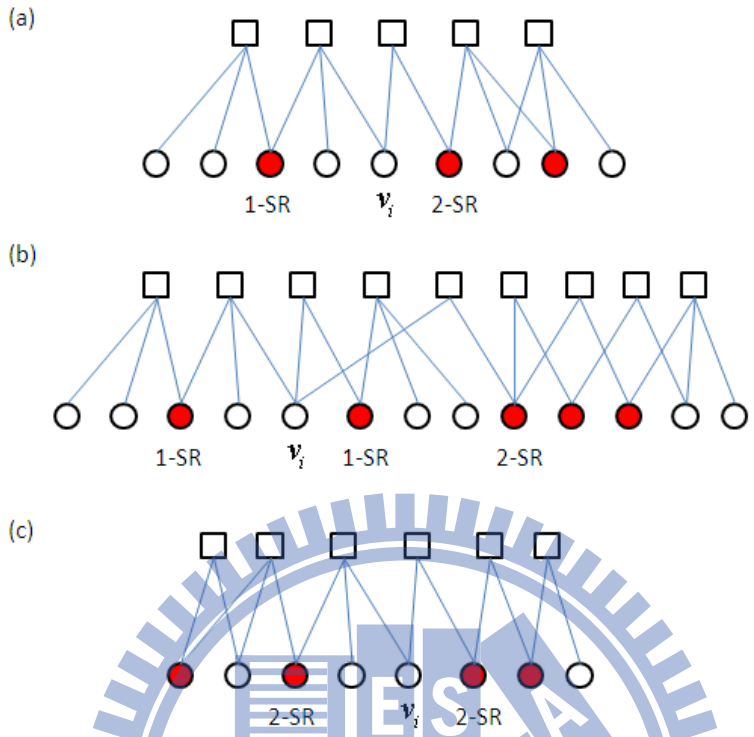


Figure 4.8: The neighboring punctured nodes of v_i (III).

recovery order of v_i in Figure 4.11 (b) and (c) is 1. The v_i in Figure 4.11 (a) is discarded. $|\boldsymbol{S}_{\mathrm{CN}}^{(1)}(v_i)|$= 2 in Figure 4.11 (b); $|\boldsymbol{S}_{\mathrm{CN}}^{(1)}(v_i)|$= 1 in Figure 4.11 (c). Thus, v_i in Figure 4.11 (c) is discarded. The next we see the sum of degree of SCNs conditionally on the candidates with the same number of SCNs. $d_{\mathrm{SCN}}(v_i)$=6 in Figure 4.12 (a); $d_{\mathrm{SCN}}(v_i)$=5 in Figure 4.12 (b). Thus, v_i in Figure 4.12 (a) is discarded. $d_{\mathrm{SCN}}(v_i)$=3 in Figure 4.13 (a); $d_{\mathrm{SCN}}(v_i)$=2 in Figure 4.13 (b). Thus, v_i in Figure 4.13 (a) is discarded.

Fifth, if the recovery order of the candidate punctured node is 1, we take care of the neighboring unpunctured nodes which connect to SCNs of the candidate. i,e., the amount of SCNs of these neighboring unpunctured nodes should be more. For example, $\bar{S}(\boldsymbol{\Lambda}(i))$=0.5 in Figure 4.14 (a); $\bar{S}(\boldsymbol{\Lambda}(i))$=1 in Figure 4.14 (b); in the comparison, v_i in Figure 4.14 (a) is discarded.

Finally, if the recovery order of the candidate punctured node is t and t is larger than 1, we need to concern more about the SCN reliability. Notice that if the recovery

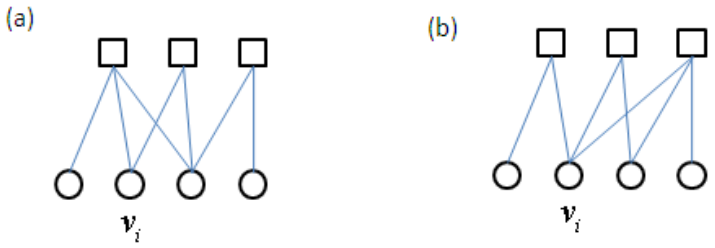


Figure 4.9: SCN of v_i (I).

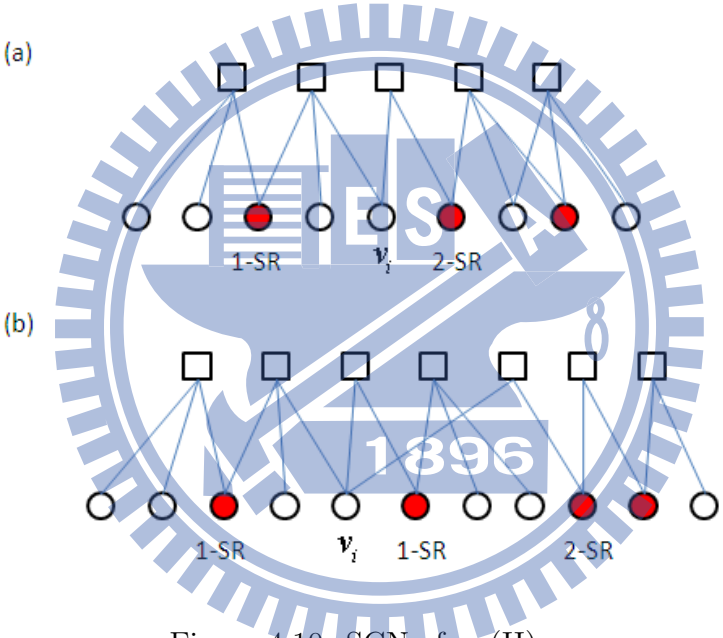


Figure 4.10: SCN of v_i (II).

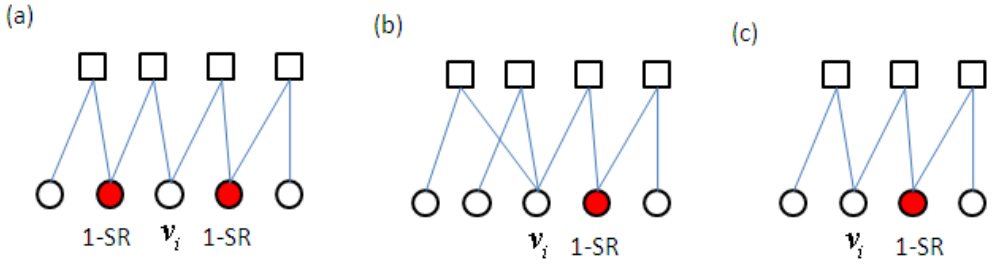


Figure 4.11: Recovery order and SCN of candidate punctured node v_i.

Forth, conditionally on the candidates with the same SCN number, the candidate punctured node with lower degree of their SCNs has a higher priority as a new punctured node. For example, $d_{\mathrm{SCN}}(v_i)$=6 in Figure 4.12 (a); $d_{\mathrm{SCN}}(v_i)$=5 in Figure 4.12 (b). Thus, v_i in Figure 4.12 (a) is discarded. $d_{\mathrm{SCN}}(v_i)$=3 in Figure 4.13 (a); $d_{\mathrm{SCN}}(v_i)$=2 in Figure 4.13 (b). Thus, v_i in Figure 4.13 (a) is discarded.

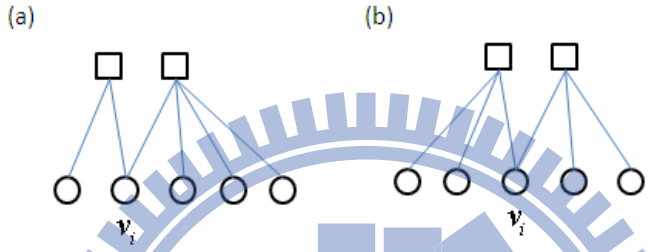


Figure 4.12: Degree of SCN of v_i (I).

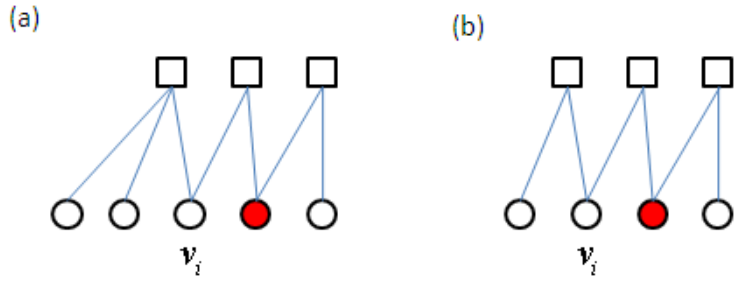


Figure 4.13: Degree of SCN of v_i (II).

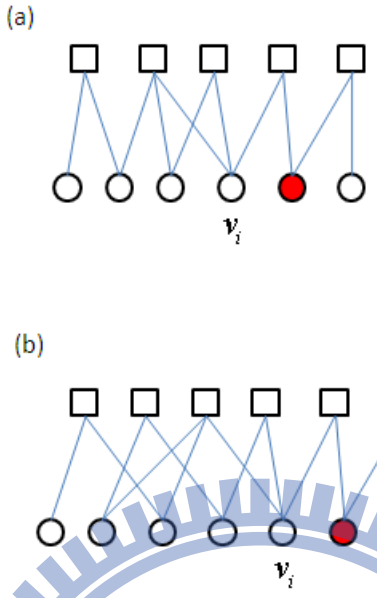


Figure 4.14: Unpunctured nodes which connect to SCN of v_i (the recovery order is 1).

order of the candidate punctured node is 1, the reliability of SCNs has been considered by CN degree. Conditionally on the candidates with the same number of SCNs and the same sum of the degree of SCNs, we compare $\pi^{(m)}(v_i)$, where $0 \leq m \leq (t-1)$. If a SCN connects to more unpunctured nodes and less punctured nodes and the recovery order of these punctured nodes are lower, the reliability of the SCN is higher. For example, $\pi^{(0)}(v_i)$=2 in Figure 4.15 (a); $\pi^{(0)}(v_i)$=3 in Figure 4.15 (b); thus, v_i in Figure 4.15 (a) is discarded.

The proposed algorithm is listed in Table 4.1. After step 1, the maximum recovery order of every punctured node is under control. In step 2, the SCN number of previous punctured nodes is considered when a new punctured node is determined. Based on taking care of previous selected punctured nodes, the recovery order and SCN number of a new punctured node are considered in step 3. We further discuss the SCN reliability of a new punctured node in step 4 and step 5. The SCN number of unpunctured nodes is thought in step 5. In step 6, the new punctured node is randomly selected from the candidate set and then the candidate set is regenerated by all unpunctured nodes and

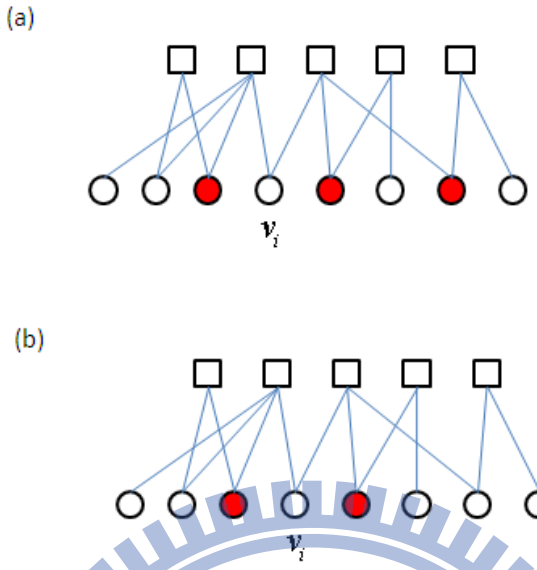


Figure 4.15: SCN reliability

go back to step 1.

Table 4.1: Algorithm of Puncture Patterns Design

Initialization:

Set $l = 0$, $\boldsymbol{P} = \phi$, $\boldsymbol{V} = \{1, 2, ..., N\}$, and $t = 1$.

Step 1:

The candidate set $\boldsymbol{V}$ is regenerated such that the candidates in $\boldsymbol{V}$ satisfy l-SR constraint.

If $\boldsymbol{V} = \boldsymbol{\phi}$

set $l = l + 1$ and $t = 1$.

reset $\boldsymbol{V} = \{1, 2, ..., N\} \setminus \boldsymbol{P}$.

If $l = N$, exit the algorithm.

go back to step 1.

end if

Step 2:

for $m = 1 : l$

Obtain $\bar{S}(\boldsymbol{\Gamma}^{(m)}(i))$ with $\boldsymbol{P}$ by Subroutine (A), $\forall\ v_i \in \boldsymbol{V}$

$\boldsymbol{V'} = \{i | \bar{S}(\boldsymbol{\Gamma}^{(m)}(i)) = \max_{j \in \boldsymbol{V}} \bar{S}(\boldsymbol{\Gamma}^{(m)}(j))\}$

$\boldsymbol{V} = \boldsymbol{V'}$. (If $\max_{j \in \mathbf{V}} \bar{S}(\boldsymbol{\Gamma}^{(m)}(j)) = 0$, $\boldsymbol{V}$ is unchanged.)

end for

Step 3:

If no candidate punctured nodes is t-SR and $t < (l + 1)$, set $t = t + 1$.

Obtain $|\boldsymbol{S}_{CN}^{(t)}(v_i)|$ with $\boldsymbol{P}$ by Subroutine (B), $\forall\ v_i \in \boldsymbol{V}$

$\boldsymbol{V'} = \{i |\ |\boldsymbol{S}_{CN}^{(t)}(v_i)| = \max_{j \in \boldsymbol{V}} |\boldsymbol{S}_{CN}^{(t)}(v_j)|\}$

$\boldsymbol{V} = \boldsymbol{V'}$.

Step 4:

Obtain $d_{SCN}(v_i)$ with $\boldsymbol{P}$ by Subroutine (C), $\forall\ v_i \in \boldsymbol{V}$

$\boldsymbol{V'} = \{i | d_{SCN}(v_i) = \min_{j \in \boldsymbol{V}} d_{SCN}(v_j)\}$

$\boldsymbol{V} = \boldsymbol{V'}$.

Step 5:

if $t = 1$

Obtain $\bar{S}(\boldsymbol{\Lambda}(i))$ with $\boldsymbol{P}$ by Subroutine (D), $\forall\ v_i \in \boldsymbol{V}$

$\boldsymbol{V'} = \{i | \bar{S}(\boldsymbol{\Lambda}(i)) = \max_{j \in \boldsymbol{V}} \bar{S}(\boldsymbol{\Lambda}(j))\}$

end if

else if $t > 1$

for $m = 0 : (t - 1)$

Obtain $\pi^{(m)}(v_i)$ with $\boldsymbol{P}$ by Subroutine (E), $\forall\ v_i \in \boldsymbol{V}$

$\boldsymbol{V'} = \{i | \pi^{(m)}(v_i) = \max_{j \in \boldsymbol{V}} \pi^{(m)}(v_j)\}$

$\boldsymbol{V} = \boldsymbol{V'}$. (If $\max_{j \in \boldsymbol{V}} \pi^{(m)}(v_j) = 0$,$\boldsymbol{V}$ is unchanged.)

end for

end if

Step 6:

Randomly select one node v_i from $\boldsymbol{V}$.

$\boldsymbol{P} = \boldsymbol{P} \bigcup v_i$.

$\boldsymbol{V} = \{1, 2, ..., N\} \setminus \boldsymbol{P}$.

If $|\boldsymbol{P}|$ is reached, exit the algorithm.

otherwise, go back to Step 1.

Chapter 5

Numerical Results and Related Discussions

5.1 Relationship Between Indicator Parameters and Error Probability

Several indicator parameters are observed in analyzing the simulation results. The indicator parameters include the number of SCNs and the degree of the SCNs and the number of un-/punctured nodes which a SCN connects for each recovery order of punctured nodes. Notice that the reliability of a CN is determined by its degree and the connected nodes. In most cases, the recovery error probability can be inferred from these indicator parameters. However, some cases are more complicated, it needs to observe more detailed information, for example, the recovery order of punctured nodes which connect to a SCN and the error rate of the nodes which a SCN connects.

The QC code with $N = 576$, $K = 384$, $M = 192$, and $R = 2/3$ in IEEE 2005 802.11n is simulated. The degree of VNs is 2, 3, 4; The degree of CNs is 9. When the puncturing code rate is 3/4, the recovery order of punctured nodes in [7] and proposed scheme are 1, 2. First, we see the 1-SR nodes. The average number of SCNs which a 1-SR punctured node has is approximately equal in the two schemes and the data is

shown in Figure 5.1. The average number of unpunctured nodes which connect to a SCN of a 1-SR node is also approximately equal in Figure 5.2. We can infer the average recovery error probability of 1-SR nodes in the two schemes are approximately equal and the simulation result is shown in Figure 5.4. Next, we see the 2-SR nodes. The average number of SCNs which a 2-SR punctured node has in proposed scheme is more than that in [7] and the SCN reliability of 2-SR nodes in two schemes is approximately equivalent, so we can deduce that the average recovery error probability of 2-SR nodes in proposed scheme is better than that in [7]. After 2-SR nodes are recovered, all the message passing in Tanner graph works and thus the 2-SR nodes with better recovery improves the performance of 1-SR nodes.

Both the average BER of unpunctured nodes and two recovery order of punctured nodes are decreasing functions of the iteration number, as is evidenced by the simulation results shown in Figure 5.5. In Figure 5.6, we show that in the 100th iteration, the average BER of unpunctured nodes, 1-SR and 2-SR nodes outperform than those in [7], respectively. Another simulation is the QC code with puncturing code rate 4/5, the recovery order of punctured nodes in [7] are 1, 2, 3, 4; in proposed scheme are 1, 2, 3. The average number of SCNs which a 1-SR punctured node has and the reliability of these SCNs are approximately equivalent in the two schemes. The average recovery error probability of 1-SR nodes in the two schemes is also approximately equal. The average number of SCNs of 2-SR or 3-SR nodes and the average number of unpunctured nodes which connect to a SCN of a 2-SR or 3-SR node in proposed scheme are all more than those in [7], respectively. Moreover, in Figure 5.9, the average number of 1-SR and 2-SR nodes which a SCN of a 3-SR node connects in proposed scheme are less than those in [7], respectively. The average recovery error probability of 2-SR, 3-SR nodes in proposed scheme outperform than those in [7], respectively. When the puncturing code rate of the QC code achieves 5/6, the average number of SCNs of 1-SR nodes in proposed scheme is less than that in [7] and the SCN reliability of 1-SR nodes is approximately

equal, the average recovery error probability of 1-SR nodes in proposed scheme are worse than that in [7]. The situations of other punctured nodes with different recovery order in proposed scheme are all better, therefore the average recovery error probability of them outperform than those in [7]. The neighboring VNs of 1-SR nodes in proposed scheme are mainly composed of unpunctured nodes and 2-SR nodes, after 2nd iteration, the error rate of 1-SR nodes are mainly affected by them. With increment of iteration number, these nodes which have better recovery in proposed scheme pass more reliable message out such that the average error rate of all kinds of nodes are better than [7]. The analysis of simulation data about length-504 PEG code is similar to above description.

5.2 Simulation Results

Computer-simulated bit error rate (BER) and frame error rate (FER) performance of two different puncturing schemes for LDPC codes are reported in this section. One scheme is proposed in paper [7] and another is proposed scheme. The (576, 192) 2/3-rate LDPC code defined in 2005-802.11n, the PEG (504,252) 0.5-rate LDPC code and the (1920, 640) 1/3-rate Gallager LDPC code are used in simulations. Sum product decoding algorithm is taken and the maximum number of iteration is set to be 100.

Figure 5.37 depicts the BER performance of the QC code. When the puncturing code rate is 3/4, 4/5 and 5/6, the proposed scheme outperforms [7] by about 0.24 dB at BER=$3 * 10^{-6}$, 0.25 dB at BER=$6 * 10^{-6}$ and 0.875 dB at BER= 10^{-5}, respectively.

Figure 5.38 depicts the FER performance of the QC code. When the puncturing code rate achieves 3/4, 4/5 and 5/6, the proposed scheme offers about 0.24 dB gain around FER=10^{-4}, 0.26 dB gain at FER=$2 * 10^{-4}$ and 1 dB gain at FER= $4 * 10^{-4}$ against [7], respectively.

Figure 5.39 depicts the BER performance of the PEG code. When the puncturing code rate is 2/3 and 3/4, the proposed scheme outperforms [7] with 0.1 dB distance at BER=$6 * 10^{-6}$ and 1 dB distance at BER=$2 * 10^{-5}$, respectively.

Ratio of recovery order

Recovery order	Paper [7]	Proposed Scheme
1	99.87%	99.2%
2	0.13%	0.8%

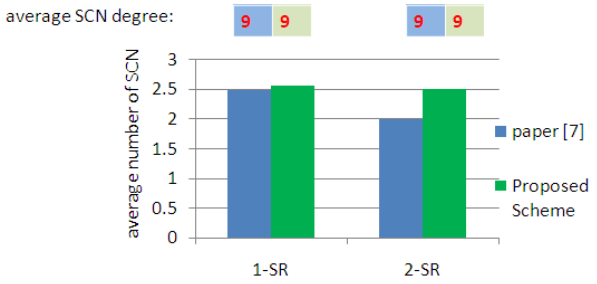


Figure 5.1: Ratio of SR order and the number and the degree of SCN. (QC576 R=3/4)

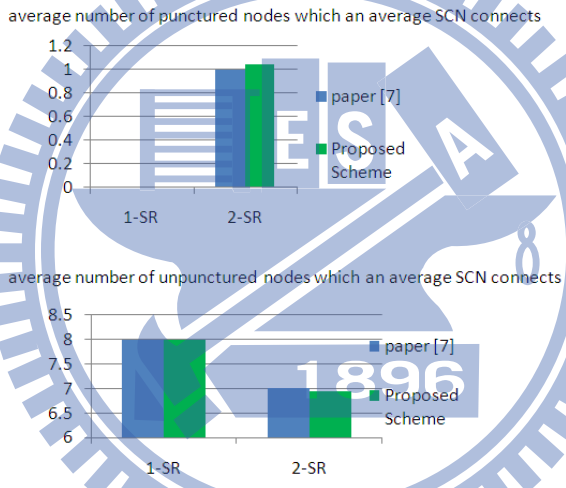


Figure 5.2: Average number of un-/punctured nodes which a SCN connects. (QC576 R=3/4)

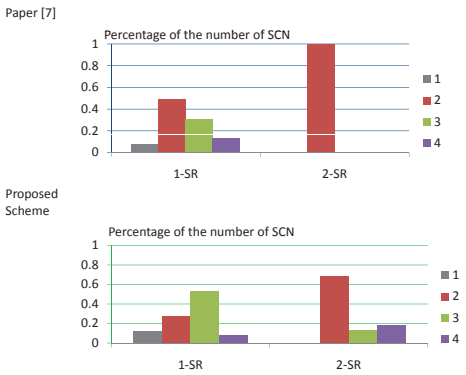


Figure 5.3: The percentage of the number of SCN. (QC576 R=3/4)

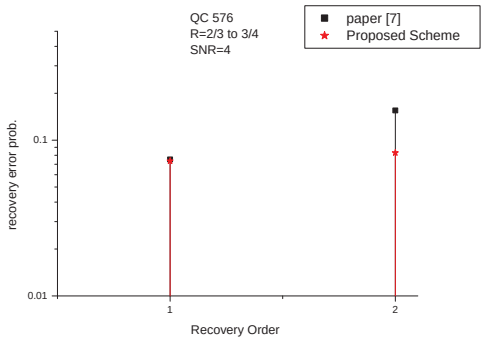


Figure 5.4: recovery error probability (QC576 R=3/4)

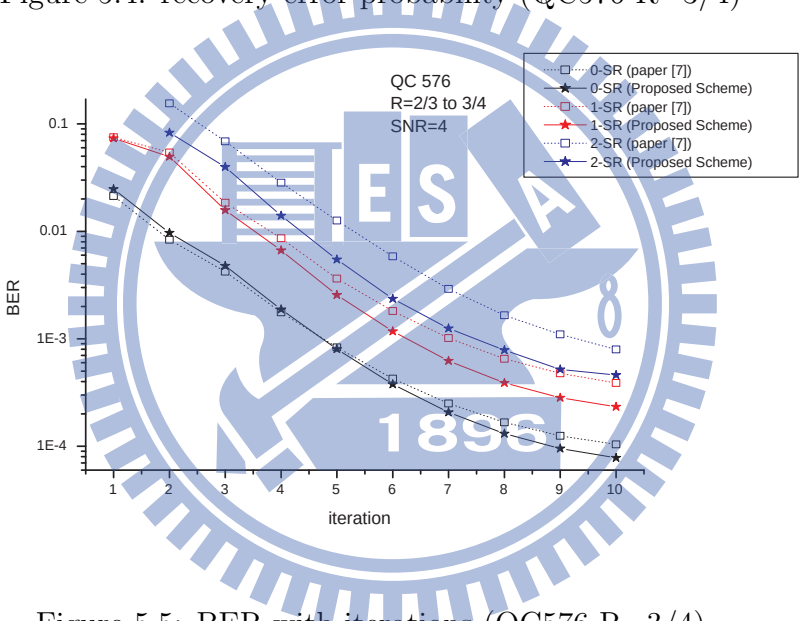


Figure 5.5: BER with iterations (QC576 R=3/4)

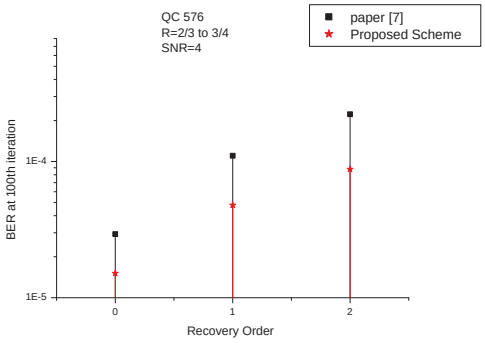


Figure 5.6: BER at 100th iteration (QC576 R=3/4)

Ratio of recovery order

Recovery order	Paper [7]	Proposed Scheme
1	88.54%	69.37%
2	10.65%	30.51%
3	0.77%	0.12%
4	0.03%	

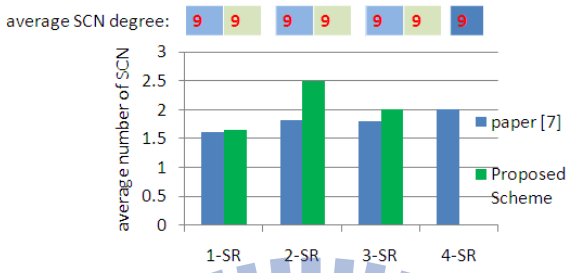


Figure 5.7: Ratio of SR order and the number and the degree of SCN. (QC576 R=4/5)

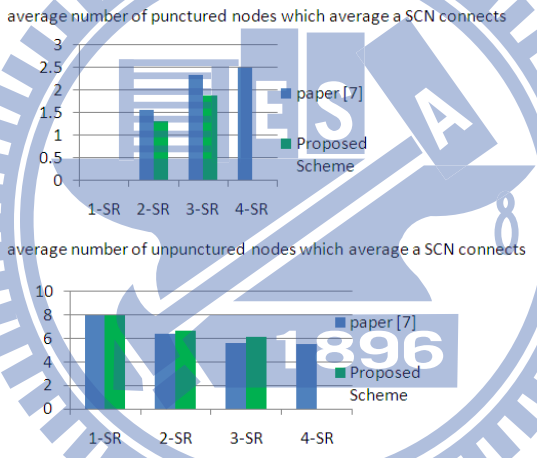


Figure 5.8: Average number of un-/punctured nodes which a SCN connects. (QC576 R=4/5)

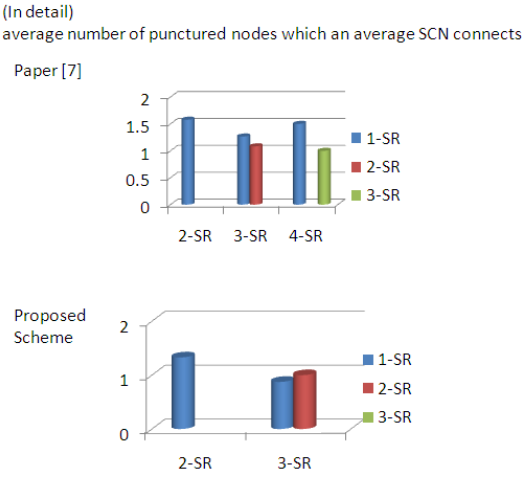


Figure 5.9: (In detail) Average number of punctured nodes which a SCN connects. (QC576 R=4/5)

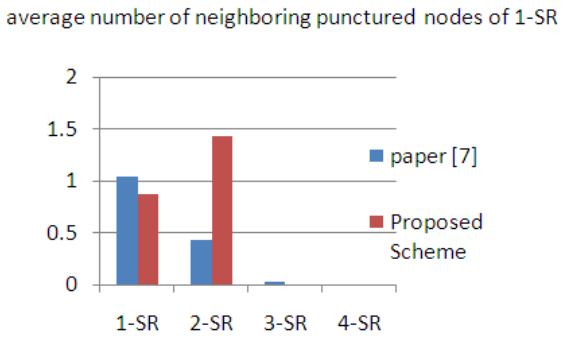


Figure 5.10: Average number of neighboring punctured nodes of 1-SR. (QC576 R=4/5)

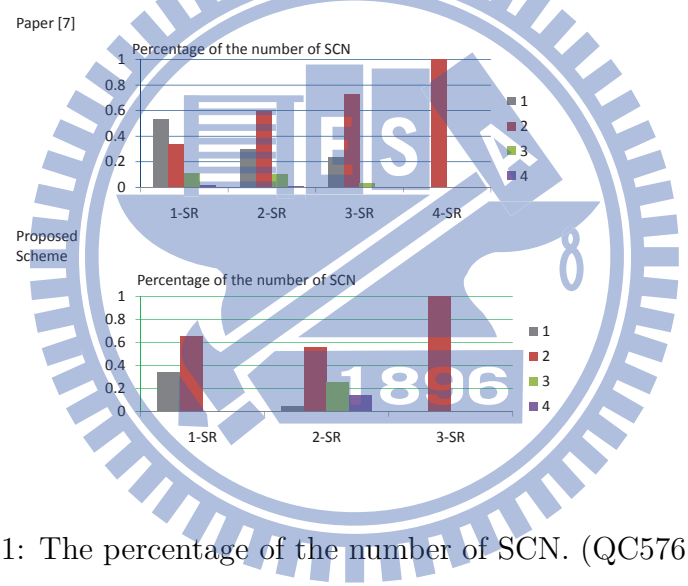


Figure 5.11: The percentage of the number of SCN. (QC576 R=4/5)

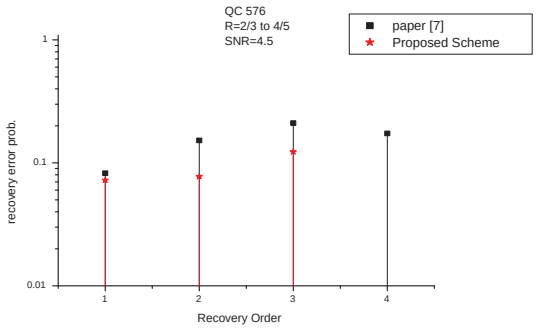


Figure 5.12: recovery error probability (QC576 R=4/5)

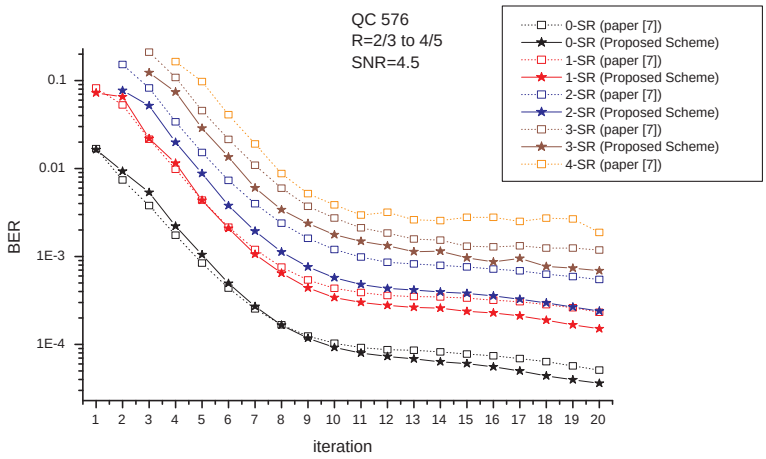


Figure 5.13: BER with iterations (QC576 R=4/5)

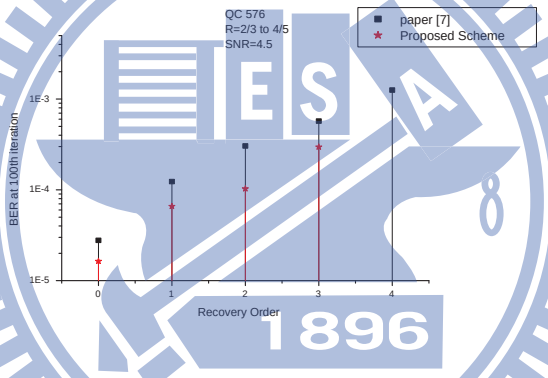


Figure 5.14: BER at 100th iteration (QC576 R=4/5)

Ratio of recovery order

Recovery order	Paper [7]	Proposed Scheme
1	67%	50%
2	21%	35%
3	8.4%	9.3%
4	3.2%	5.2%
5	0.85%	0.8%
6	0.15%	0.12%

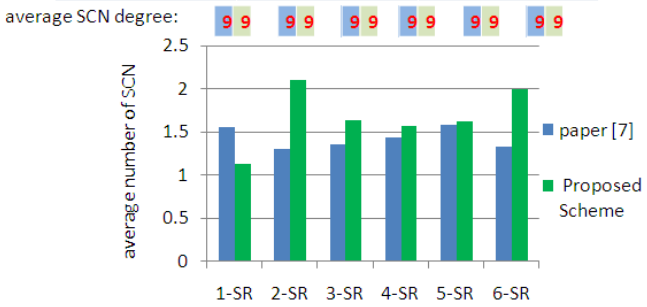


Figure 5.15: Ratio of SR order and the number and the degree of SCN. (QC576 R=5/6)

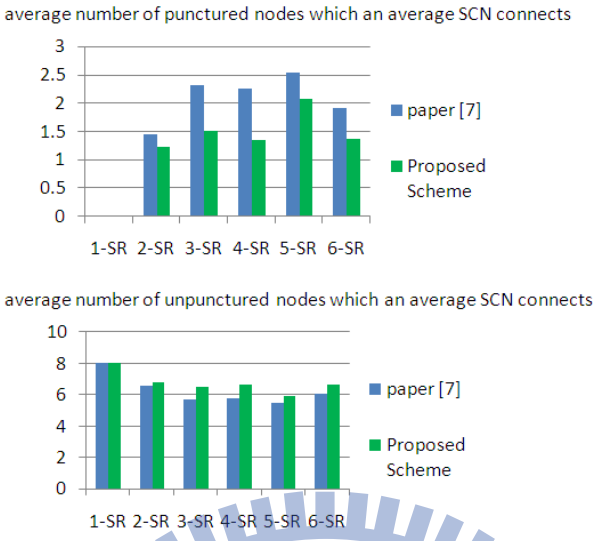


Figure 5.16: Average number of un-/punctured nodes which a SCN connects. (QC576 R=5/6)

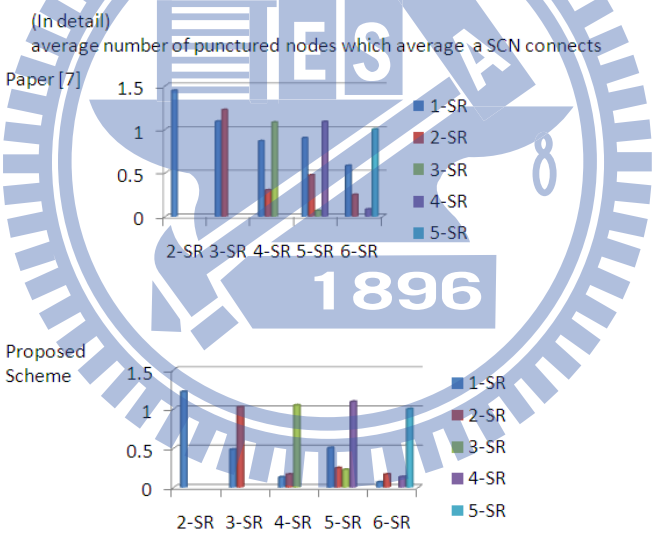


Figure 5.17: (In detail) Average number of punctured nodes which a SCN connects. (QC576 R=5/6)

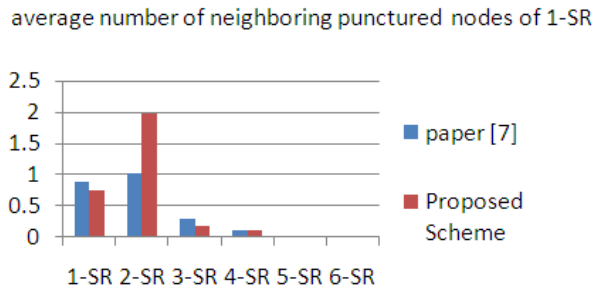


Figure 5.18: Average number of neighboring punctured nodes of 1-SR. (QC576 R=5/6)

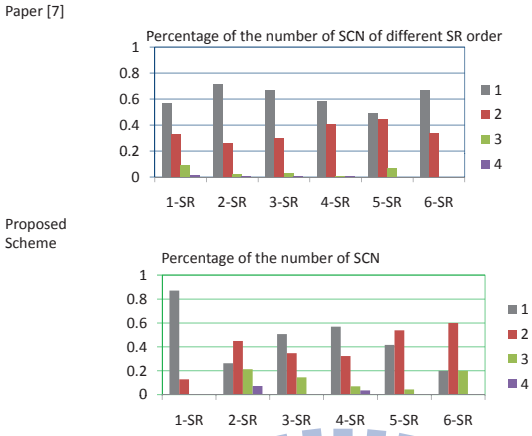


Figure 5.19: The percentage of the number of SCN. (QC576 R=5/6)

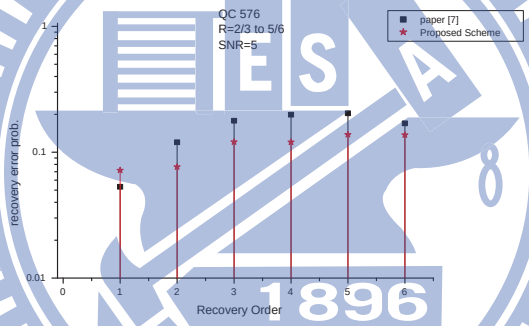


Figure 5.20: recovery error probability (QC576 R=5/6)

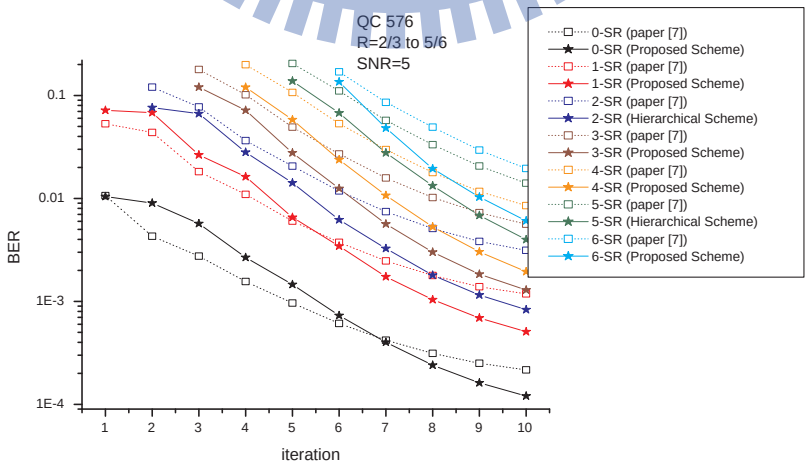


Figure 5.21: BER with iterations (QC576 R=5/6)

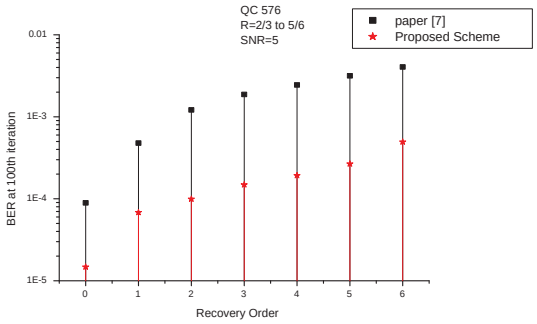
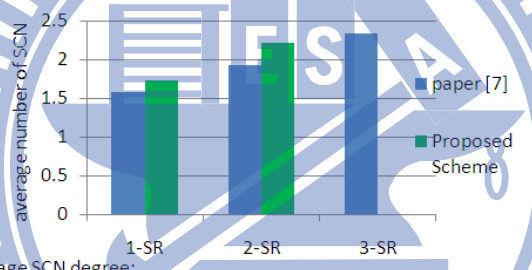


Figure 5.22: BER at 100th iteration (QC576 R=5/6)

Ratio of recovery order

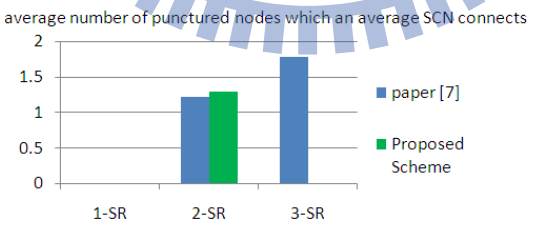
Recovery order	Paper [7]	Proposed Scheme
1	92.85%	72%
2	7.08%	28%
3	0.067%	0%



average SCN degree:

5.7091	5.76452	6
5.79255	5.57881	

Figure 5.23: Ratio of SR order and the number and the degree of SCN. (PEG504 R=2/3)



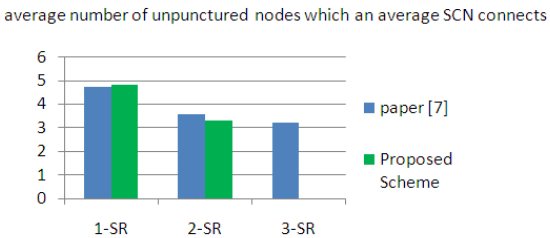


Figure 5.24: Average number of un-/punctured nodes which a SCN connects. (PEG504 R=2/3)

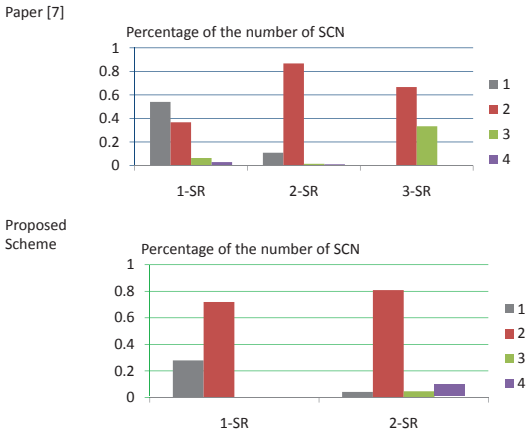


Figure 5.25: The percentage of the number of SCN. (PEG504 R=2/3)

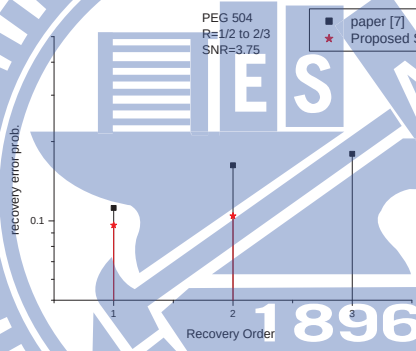


Figure 5.26: recovery error probability (PEG504 R=2/3)

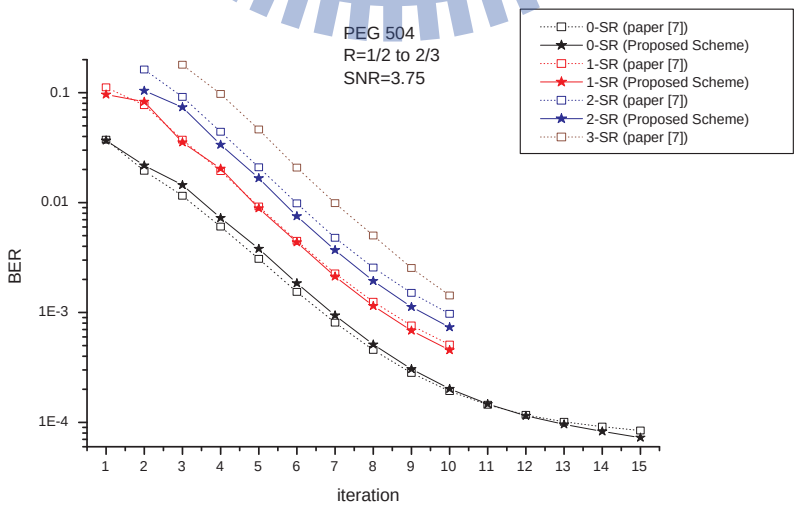


Figure 5.27: BER with iterations (PEG504 R=2/3)

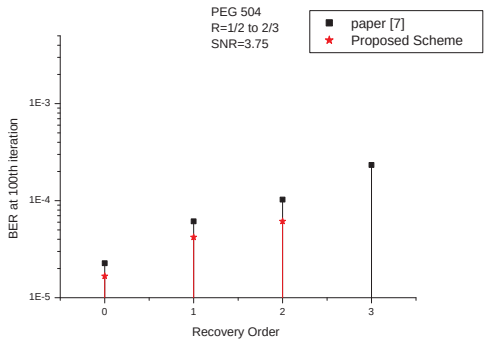
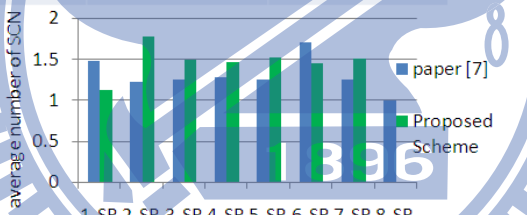


Figure 5.28: BER at 100th iteration (PEG504 R=2/3)

Ratio of recovery order

Recovery order	Paper [7]	Proposed Scheme
1	62%	46%
2	22%	36%
3	8.9%	11%
4	4.5%	5.7%
5	1.6%	1.3%
6	0.54%	0.26%
7	0.12%	0.051%
8	0.034%	0%



average SCN degree:							
5.69537	5.75171	5.79712	5.76002	5.83084	5.7625	5.75	5.5
5.7991	5.6502	5.72535	5.7701	5.79257	5.8888	5.75	

Figure 5.29: Ratio of SR order and the number and the degree of SCN. (PEG504 R=3/4)

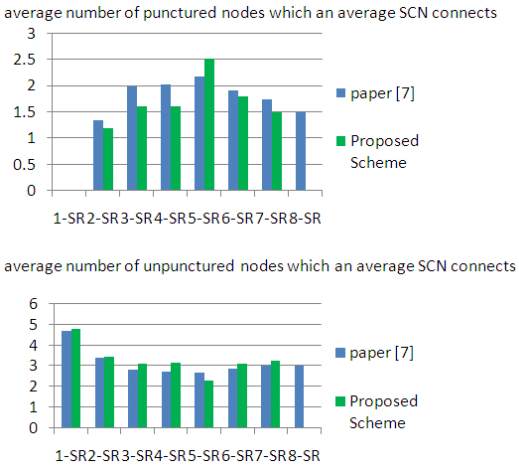


Figure 5.30: Average number of un-/punctured nodes which a SCN connects. (PEG504 R=3/4)

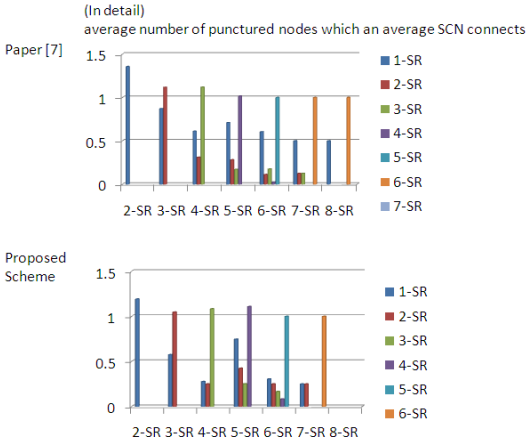


Figure 5.31: (In detail) Average number of punctured nodes which a SCN connects. (PEG504 R=3/4)

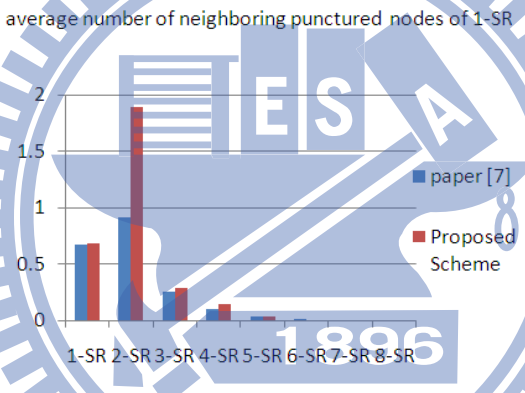


Figure 5.32: Average number of neighboring punctured nodes of 1-SR. (PEG504 R=3/4)

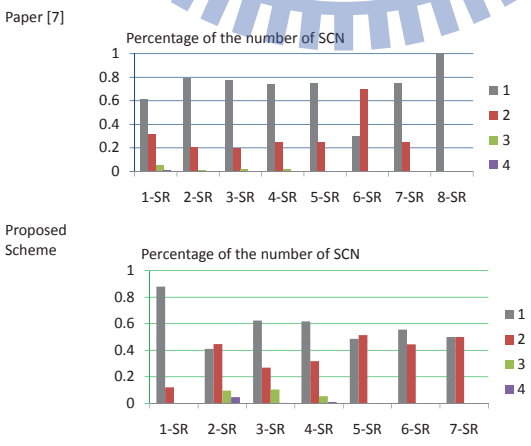


Figure 5.33: The percentage of the number of SCN. (PEG504 R=3/4)

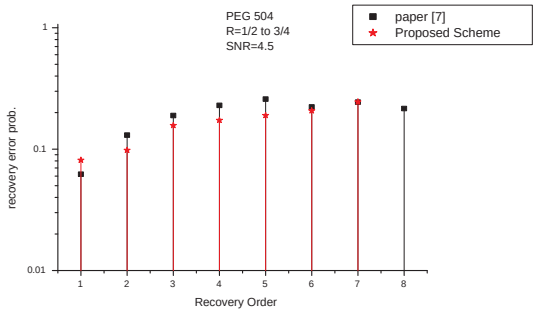


Figure 5.34: recovery error probability (PEG504 R=3/4)

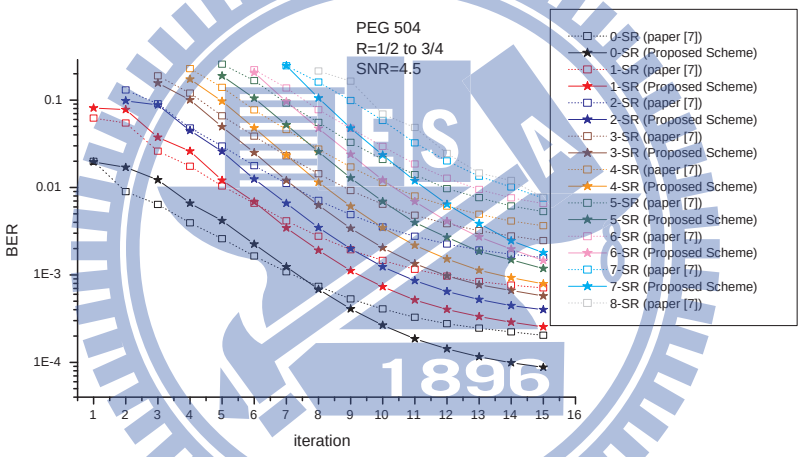


Figure 5.35: BER with iterations (PEG504 R=3/4)

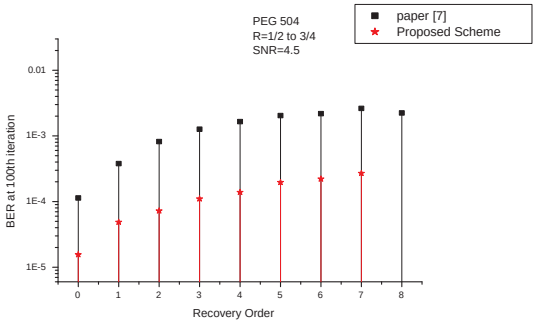


Figure 5.36: BER at 100th iteration (PEG504 R=3/4)

Figure 5.40 depicts the FER performance of the PEG code. When the puncturing code rate achieves 2/3 and 3/4, the proposed scheme yields 0.2 dB gain around FER=$2*10^{-4}$ and 1.2 dB gain at FER=$6.5*10^{-4}$ against [7], respectively.

Figure 5.41 depicts the BER performance of the Gallager code. When the puncturing code rate is 2/3, the proposed scheme outperforms [7] by about 0.4 dB at BER=10^{-4}. Figure 5.42 depicts the FER performance of the code. When the puncturing code rate is 2/3, the proposed scheme offers about 0.4 dB gain around FER=10^{-3} against [7].

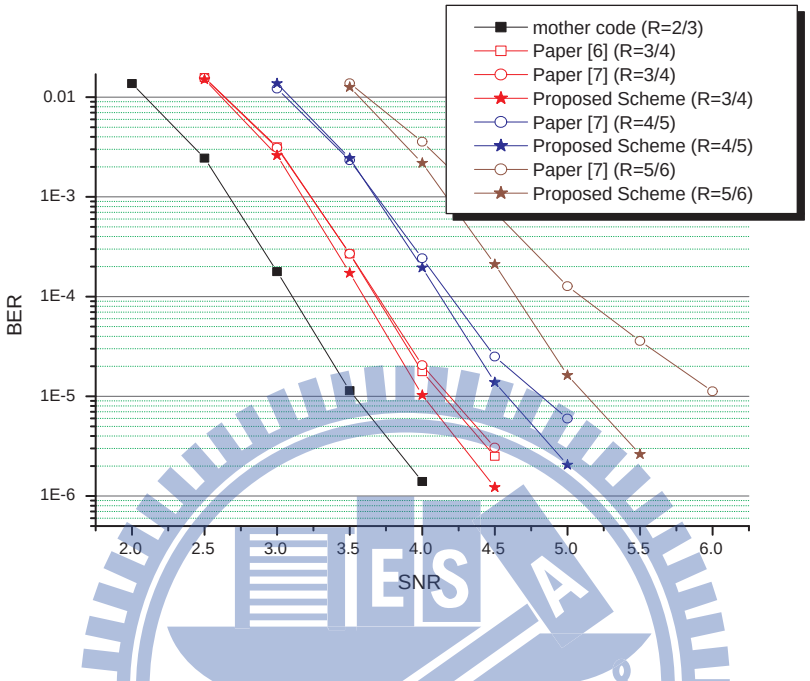


Figure 5.37: BER performance of puncturing QC code

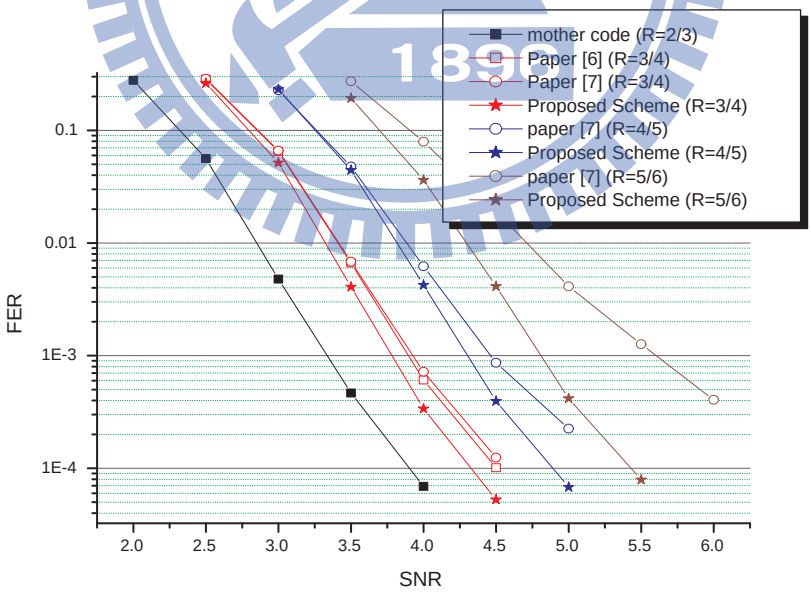


Figure 5.38: FER performance of puncturing QC code

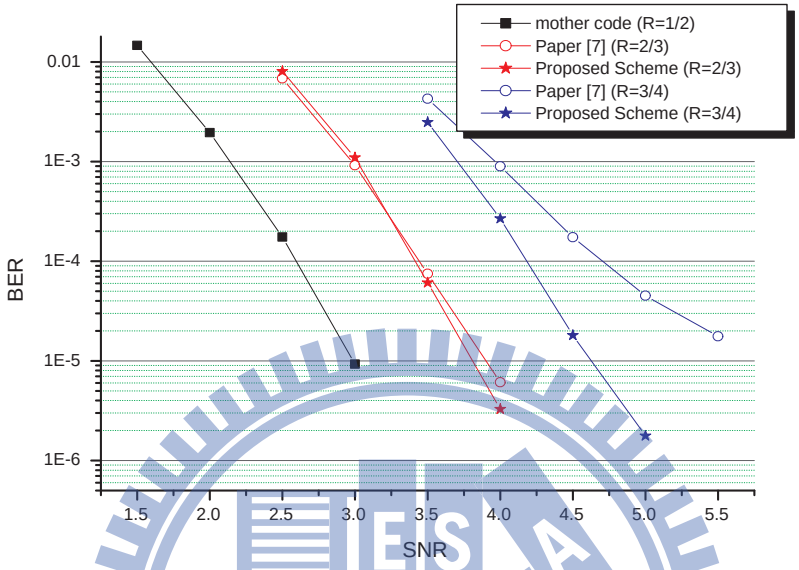


Figure 5.39: BER performance of puncturing PEG code

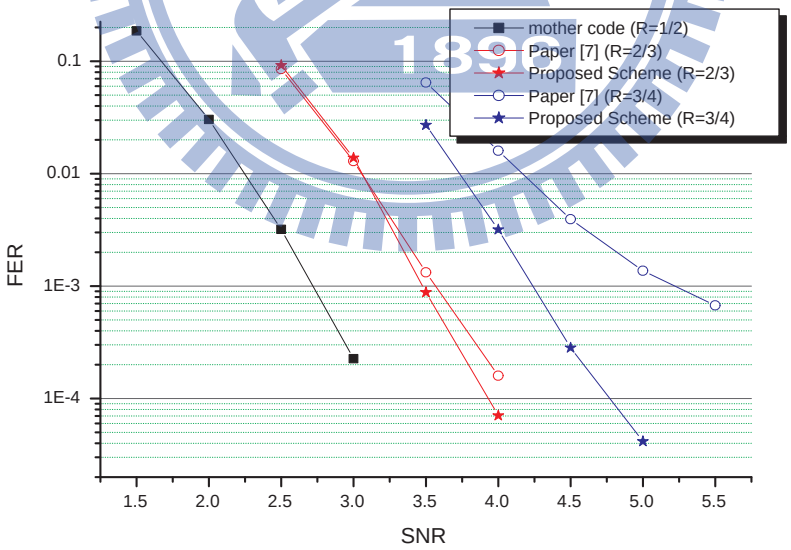


Figure 5.40: FER performance of puncturing PEG code

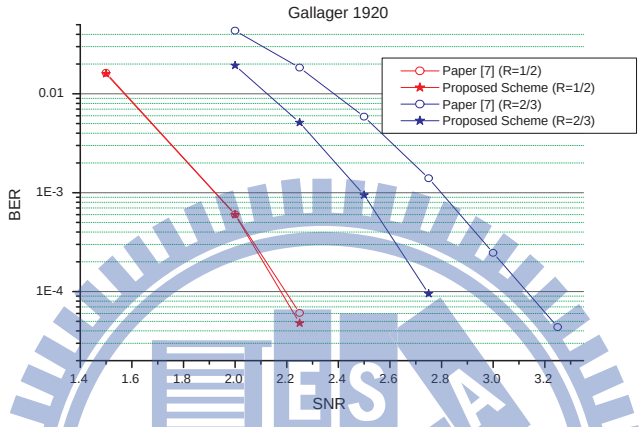


Figure 5.41: BER performance of puncturing Gallager code with mother code rate 1/3

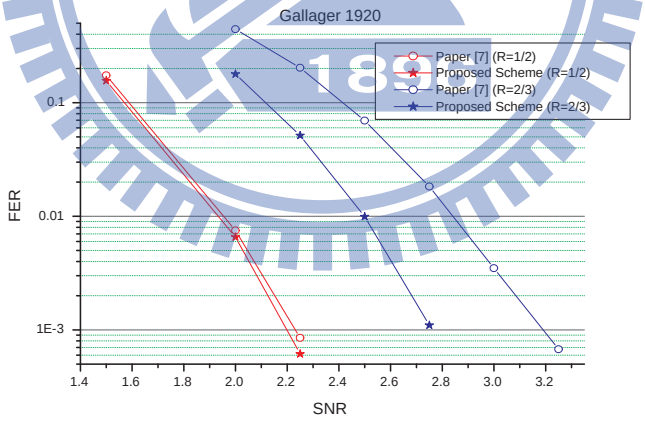


Figure 5.42: FER performance of puncturing Gallager code with mother code rate 1/3

Chapter 6

Conclusions

We present a puncture pattern design scheme for finite-length LDPC codes. For use in AWGN channels, we use GA to analyze the error probability of variable nodes. Some indicator parameters associated with the recovery capability of punctured nodes and the error-correcting performance of unpunctured nodes are obtained based on the analysis. The design guidelines are derived from the relationships between these parameters and error-rate performance. We also compare the decoding performance of our scheme with that of existing approaches. The experimental results prove that the performance of the proposed algorithm does offer better performance in comparison with the method in [7], which is consistent with the GA-based theoretical prediction.

Bibliography

[1] J. Ha and S. W. McLaughlin, "Optimal puncturing distributions for rate-Compatible low-density parity-check codes," *IEEE ISIT*, Yokohama, Japan, 2003, pp. 233.

[2] J. Ha, J. Kim, and S. W. McLaughlin, "Rate-compatible puncturing of low-density parity-check codes," *IEEE Trans. Inf. Theory*, VOL. 50, NO. 11, Nov. 2004.

[3] G. Richter, S. Stiglmayr, and M. Bossert "Optimized Asymptotic Puncturing Distributions for Different LDPC Code Constructions," *IEEE ISIT*, Seattle, USA Jul. 2006.

[4] J. Ha, J. Kim, and S. W. McLaughlin "Puncturing for Finite Length Low-Density Parity-Check Codes," *IEEE ISIT*, Chicago, USA, Jun/Jul. 2004.

[5] J. Ha, J. Kim, D. Klinc, and S. W. McLaughlin "Rate-Compatible Punctured Low-Density Parity-Check Codes With Short Block Lengths," *IEEE Trans. Inf. Theory*, VOL. 52, NO. 2, Feb. 2006.

[6] J Sunghoon Choiy, K. Nohz, J. H. Shinyy and J. Heo "Rate-Compatible Punctured LDPC Codes based on Recovery Tree," *ISITA* , Auckland, New Zealand, Dec. 2008.

[7] Badri N. Vellambi and F. Fekri, "Finite-Length Rate-Compatible LDPC Codes: A Novel Puncturing Scheme," *IEEE Trans. Commun.*, VOL. 57, NO. 2, Feb. 2009.

[8] R. G. Gallager *Low Density Parity Check Codes*, *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21-28, Jan. 1962.

[9] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Lett.*, vol. 32, no. 18, pp. 1645-1646, Aug. 1996.

[10] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, pp. 399-431, Mar. 1999.

[11] F. R. Kschischang, B. J. Frey, and Hans-Andrea Loeliger "Factor Graphs and the Sum-Product Algorithm," *IEEE Trans. Inf. Theory*, vol. 47, Feb. 2001.

[12] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, no. 9, pp. 533-547, Sep. 1981.

[13] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke "Analysis of sum product decoding of low-density parity-checkcodes using a Gaussian approximation," *IEEE Trans. Inf. Theory*, vol. IT-47, pp. 657-670, Feb. 2001.

[14] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, "Finite-length analysis of low-density parity-checkcodes on the binary erasure channel," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1570-1579, Jun. 2002.

[15] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linkoping Univ., Linkoping, Sweden, 1996.

[16] W.E. Ryan and S. Lin "Channel Codes: Classical and Modern," *Cambridge University Press,* 2009.