一种新的码率兼容 LDPC 码打孔方案

苏和光[①] 夏树涛*[①②]
[①](清华大学深圳研究生院 深圳 518055)
[②](东南大学移动通信国家重点实验室 南京 210096)

摘 要：该文研究码率兼容 LDPC 码的打孔问题。Ha 等人研究 LDPC 码打孔时提出的 Grouping 和 Sorting 方案使低 k-SR 节点的数目最大化，它相对于随机打孔有了很大增益，但此方案只保证有一个存活校验节点。该文研究多个存活校验节点的作用，提出 MSCN 打孔方案最大化打孔节点的存活校验节点数。AWGN 信道上的仿真结果显示，低码率时 MSCN 方案具有比 Grouping 和 Sorting 方案更为优越的性能。理论推导及实验结果均表明，码率兼容 LDPC 码打孔时，多个存活校验节点有益于译码性能的提升。

关键词：LDPC 码；码率兼容；Tanner 图；打孔
中图分类号：TN911.22 **文献标识码**： A **文章编号**：1009-5896(2011)10-2334-06
DOI: 10.3724/SP.J.1146.2010.01202

A Novel Puncturing Scheme for Rate-compatible LDPC Codes

Su He-guang[①] Xia Shu-tao[①②]
[①](*Graduate School at Shenzhen, Tsinghua University, Shenzhen* 518055, *China*)
[②](*National Mobile Communications Research Laboratory of the Southeast University, Nanjing* 210096, *China*)

Abstract: This paper considers the issue of finding good puncturing patterns for rate-compatible LDPC codes. When studying the puncturing of LDPC codes, Ha, *et al.* obtain a so-called grouping and sorting puncturing scheme to maximize the number of lower k-SR nodes. Though the grouping and sorting scheme outperforms random puncturing schemes, it guarantees only single survived check node. This paper investigates the effect of multiple survived check nodes, and proposes an effective puncturing scheme called MSCN, which maximizes the number of survived check nodes. Simulation results over AWGN channels show that the proposed MSCN scheme is superior to the grouping and sorting scheme at low rates. By theoretical analysis and experimental results, it is indicated that multiple survived check nodes enhance the decoding performance of LDPC codes.

Key words: LDPC codes; Rate-compatible; Tanner graph; Puncturing

1 引言

在无线通信系统等应用场景中，信道是时变的，为使信道的吞吐率最大化，人们需要根据信道条件的不同自适应地使用不同的编码码率。只使用一对编译码器的码率兼容 LDPC 码大大降低了硬件实现的复杂度，成为时变信道中差错控制方法的优先选择。打孔是构造码率兼容 LDPC 码的有效方法，但它可能会引起性能的损失。为尽量避免较大的性能损失，需要解决的关键问题是如何寻找有效的打孔模式。

Ha 等人[1]研究表明好的打孔模式是存在的，且在一定码率范围内对一个 LDPC 码打孔后得到的码仍可能是好码[2]，然而其结果基于无限码长假设。文献[3,4]的工作是有限码长的，但其打孔方法对变量节点的选择是随机的，易损失重要节点，相应节点可能因此无法恢复，高码率时该情况尤为严重。为解决该问题，文献[5]研究译码时打孔节点恢复过程，提出 Grouping 和 Sorting 方案。Grouping 算法将变量节点分成不同集合 $G_1, G_2, \cdots, G_K$，依次最大化各集合中节点的数目。在每一个集合中，节点被打孔的次序由 Sorting 算法来确定。虽然 Grouping 和 Sorting 方案性能优于随机打孔的方案，但它只保证有一个存活校验节点。

本文研究多个存活校验节点的作用，提出最大化存活校验节点数(MSCN)方案寻找打孔模式。仿真结果表明，在 AWGN 信道上，低码率时本文提出的方案优于 Grouping 和 Sorting 方案。本文后续部分组织如下：第 2 节简要介绍一些基本概念和高斯近似密度进化；第 3 节分析多个存活校验节点的作用，介绍本文方案的基本思想，并对有限长 LDPC 码提出 MSCN 打孔方案；第 4 节对提出的打孔方案进行仿真实验；最后是结论。

2010-11-05 收到，2011-07-13 改回
国家自然科学基金(60972011)，高等学校博士学科点专项基金和东南大学移动通信国家重点实验室开放研究基金(2011D11)资助课题
*通信作者：夏树涛 xiast@sz.tsinghua.edu.cn

2　预备知识

2.1 基本概念及定义

LDPC 码可以用 Tanner 图来表示。其校验矩阵的行对应 Tanner 图的校验节点，列对应变量节点。对一个节点 V，定义 $N(V)$为 V 的邻居。所谓打孔，就是将一个码中的每个码字都删除一个或多个分量，从二分图上看即是删除一个或多个变量节点，被删除的变量节点称为打孔变量节点。打孔模式定义为打孔变量节点的集合。若打孔变量节点 V 有至少一个邻居校验节点 C，这个校验节点 C 的其它邻居(除了 V)都是没有打孔的，那么称 V 是一步可恢复的(One Step Recoverable, 1-SR)[5−7]。所谓变量节点恢复，是指它第 1 次从邻居中得到非零的对数似然比(LLR)信息，而这个提供非零 LLR 信息的校验节点 C 被称为存活校验节点。一般地，一个 k 步可恢复(k-SR)的打孔变量节点 V 拥有至少一个邻居 C，满足 $N(C)\backslash V$ 包含至少一个(k−1)步可恢复节点，而其它节点是 m 步可恢复的，$0 \le m \le k-1$。k-SR 节点可以在第 k 次迭代时被恢复，亦称 k-SR 节点的恢复水平为 k。k-SR 节点的集合记为 G_k。恢复树显示了一个打孔节点在恢复过程中的信息传递情况，图 1 是一个 3-SR 节点恢复树的例子。为了记录校验节点在打孔节点恢复过程中发挥的作用，我们引入一个新的概念“校验节点状态”。每一个校验节点都有一个状态来表示它恢复打孔变量节点的信息。校验节点状态用二元对(l, p)来表示，l 表示这个校验节点 j 所恢复的变量节点 i 的恢复水平，p 表示 i 的存活校验节点的数目。i 的所有存活校验节点组成一个集合 Partner(j)。

2.2 LDPC 译码及密度进化

LDPC 码一般用 BP 算法来译码，在译码过程中，信息更新规则[8]如下：

$$v = u_0 + \sum_{j=1}^{d_v-1} u_j \tag{1}$$

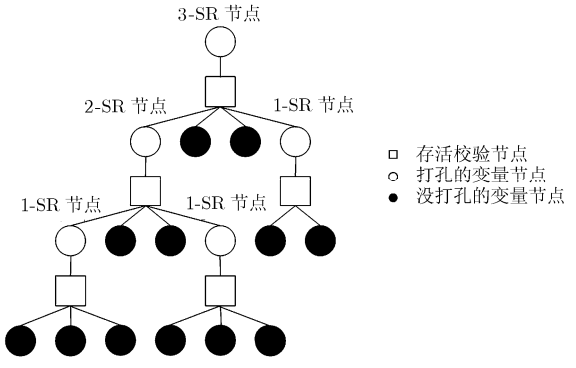

图 1 3-SR 节点恢复树

$$\tanh\frac{u}{2} = \prod_{i=1}^{d_c-1}\tanh\frac{v_i}{2} \tag{2}$$

v 及 v_i表示变量节点传出的信息，u 及 u_j表示校验节点传出的信息，u_0是从信道得到的信息。

采用高斯近似密度进化的方法[8]，分别在式(1)，式(2)等号两边取期望，得

$$m_v^{(l)} = m_{u_0} + (d_v - 1)m_u^{(l-1)} \tag{3}$$

$$E\left[\tanh\frac{u^{(l)}}{2}\right] = E\left[\prod_{i=1}^{d_c-1}\tanh\frac{v_i^{(l)}}{2}\right] = E\left[\tanh\frac{v^{(l)}}{2}\right]^{d_c-1} \tag{4}$$

l 是指迭代次数，m_{u_0} 是 u_0 的期望，m_u 和 m_v 分别是 u 和 v 的期望。

定义 $\phi(x)$ 为

$$\phi(x) = \begin{cases} 1-\dfrac{1}{\sqrt{4\pi x}}\displaystyle\int_{-\infty}^{\infty}\tanh\frac{u}{2}\exp\left(-\frac{(u-x)^2}{4x}\right)\mathrm{d}u, & x>0 \\ 1, & x=0 \end{cases}$$

$\phi(x)$ 是在 $[0,\infty)$ 上的连续单调减函数，且 $\phi(0)=1$, $\phi(\infty)=0$。于是式(4)变为

$$1-\ \phi\left(m_u^{(l)}\right) = \left[1-\phi\left(m_v^{(l)}\right)\right]^{d_c-1} \tag{5}$$

$$m_u^{(l)} = \phi^{-1}\left\{1-\left[1-\phi\left(m_{u_0}+(d_v-1)m_u^{(l-1)}\right)\right]^{d_c-1}\right\} \tag{6}$$

3　打孔方案

3.1 多存活校验节点分析

当打孔变量节点 V 只有一个存活校验节点时，文献[5]给出了 V 接收到的 LLR 信息期望值的数学表达式，如引理 1 所示。

引理 1　对打孔变量节点，若都只有一个存活校验节点使其恢复，则打孔变量节点 V 接收到的 LLR 信息期望为

$$m_u(V) = \phi^{-1}\left(1-\left[1-\phi(m_{u_0})\right]^{S(V)}\right) \tag{7}$$

$S(V)$是 V 恢复树下非打孔变量节点的数目，u_0是从信道得到的 LLR 信息，m_{u_0} 是 u_0 的期望。

而关于多个存活校验节点的情况，我们有定理 1。

定理 1　设 l=1,⋯,k，打孔的 l-SR 节点都只有一个存活校验节点，且当打孔节点 V'，V'' 恢复水平相同时，其恢复树中各存活校验节点下边的非打孔变量节点数目相同 $S(C')=S(C'')$，C'，C'' 分别为 V'，V'' 的存活校验节点，则打孔的 (k+1)-SR 节点 V 接收到 LLR 信息期望为

$$\begin{aligned} m_u(V) &= \sum_{j=1}^{p}\phi^{-1}\left(1-\left[1-\phi\left(m_{u_0}\right)\right]^{S(C_j)}\right) \\ &= p\phi^{-1}\left(1-\left[1-\phi\left(m_{u_0}\right)\right]^{S(C_j)}\right) \end{aligned} \tag{8}$$

其中打孔变量节点 V 存在 p 个存活校验节点 C_1, $C_2,\cdots,C_p$, $S(C_j)$是在 V的恢复树中 C_j下边非打孔变量节点的数目。

证明 在(k+1)-SR 节点 V 的恢复树中，V 下边是存活校验节点 C_1, $C_2,\cdots$, C_p，而 $C_j(1\le j\le p)$下边含有 k-SR 变量节点和其他变量节点，设为 V_1, $V_2,\cdots,V_{d_c-1}$。变量节点 $V_i(1\le i\le d_c-1)$发送出去的 LLR 信息期望 $m_v(V_i)$等于接收到的 LLR 信息期望 $m_u(V_i)$，由引理 1，有

$$m_v(V_i)=m_u(V_i)=\phi^{-1}\left(1-\left[1-\phi(m_{u_0})\right]^{S(V_i)}\right) \tag{9}$$

即有

$$\left.\begin{aligned}1-\phi\left(m_v(V_i)\right)&=\left[1-\phi(m_{u_0})\right]^{S(V_i)}\\ E\left[\tanh\left(v_i/2\right)\right]&=\left[1-\phi(m_{u_0})\right]^{S(V_i)}\end{aligned}\right\} \tag{10}$$

于是由式(4)，关于 C_j处的信息有

$$\begin{aligned}E\left[\tanh\frac{u_j}{2}\right]&=E\left[\prod_{i=1}^{d_c-1}\tanh\frac{v_i}{2}\right]=\prod_{i=1}^{d_c-1}E\left[\tanh\frac{v_i}{2}\right]\\&=\prod_{i=1}^{d_c-1}\left[1-\phi(m_{u_0})\right]^{S(V_i)}=\left[1-\phi(m_{u_0})\right]^{\sum_{i=1}^{d_c-1}S(V_i)}\\&=\left[1-\phi(m_{u_0})\right]^{S(C_j)}\end{aligned} \tag{11}$$

即有

$$\begin{aligned}1-\phi\left(m_u(C_j)\right)&=\left[1-\phi(m_{u_0})\right]^{S(C_j)},\\ m_u(C_j)&=\phi^{-1}\left(1-\left[1-\phi(m_{u_0})\right]^{S(C_j)}\right)\end{aligned} \tag{12}$$

由式(3)，对(k+1)-SR 节点 V，接收到的 LLR 信息期望等于其存活校验节点传来的 LLR 信息期望之和，故

$$m_u(V)=\sum_{j=1}^{p}m_u(C_j)=\sum_{j=1}^{p}\phi^{-1}\left(1-\left[1-\phi\left(m_{u_0}\right)\right]^{S(C_j)}\right) \tag{13}$$

由定理条件，有 $S(C_1)=S(C_2)=\cdots=S(C_p)$，所以

$$\begin{aligned}m_u(V)&=\sum_{j=1}^{p}\phi^{-1}\left(1-\left[1-\phi\left(m_{u_0}\right)\right]^{S(C_j)}\right)\\&=p\phi^{-1}\left(1-\left[1-\phi\left(m_{u_0}\right)\right]^{S(C_j)}\right)\end{aligned} \tag{14}$$

证毕

在 AWGN 信道中，假设满足独立同分布和对称条件，节点 V'的错误概率为[8]

$$P_e(V')=Q\left(\frac{m_u(V')}{\sqrt{\sigma^2(V')}}\right)=Q\left(\sqrt{\frac{m_u(V')}{2}}\right) \tag{15}$$

$\sigma^2(V')$是节点 V'发送的 LLR 信息的方差。不难看出，$P_e(V')$是关于 $m_u(V')$的单调递减函数，$m_u(V')$的值越大，则 $P_e(V')$越小。

根据定理 1，在其他条件相同的情况下，若节点 V'拥有多个存活校验节点，相比于只有一个存活校验节点的情况，其 LLR 信息期望值会更大。由式(15)及 $P_e(V')$的单调性可知，多个存活校验节点会降低节点 V'的差错概率。

3.2 打孔算法

基于前边的分析，本文提出了一个新的打孔算法，它能够最大化一个打孔节点的存活校验节点数目。初始，若一个变量节点被打孔，就把 $N(V)$定为 V 的存活校验节点。打孔数较多时，新打孔的节点不可避免地会破坏前打孔节点的存活校验节点，带来性能损失。为了减少这种损失，新选打孔节点必须对前打孔节点的影响最小，控制存活校验节点减少的速度。故一个新打孔节点，其邻居中包含前打孔节点的存活校验节点的数目必须是最小的。若打孔节点 V 只有一个存活校验节点 C，则 C 的邻居中除了 V 外所有其它变量节点都不能再被打孔，否则有些前打孔节点会因此而无法恢复。我们把不能被打孔的变量节点的集合称为 G_0，C 的邻居除 V 外的所有变量节点都应添到 G_0 中。打孔节点和 G_0 中的节点都称为已确定节点。

存活校验节点数目最大化算法(MSCN)如表 1 所示。

表 1 MSCN 算法

//**Step 0 处理状态(0, 0)**
Candidate←{1,⋯,n-1,n}; ∀ j($\le j\le m$), State(j)←(0,0);
　∀ i($1\le i\le n$), Level(i)←0; nowPS←(0,0);
While Candidate≠ ϕ , do
　从 Candidate 中随机选择节点 V 作为打孔节点；Candidate ← Candidate\$N(N(V))$; State($j$)=(1, degree($V$)), ∀ j∈$N(V)$;
　Level (V) =1;
End While
k ←1; //**Step 1 处理状态(*k*, *d*)**
while 1, do 若所有变量节点都已确定，结束循环终止查找。d←变量节点的最大度数。
　while d>1, do
　prePS ←prePS∪nowPS; nowPS←(k,d);
　X={j:State(j)= nowPS };
　Y = { j : State(j) ≠ nowPS, State(j) ∉ prePS};
　　Candidate ← $N(X)\backslash\{N(Y)$, fixed nodes};
　If Candidate= ϕ , then　d ←d-1; continue; End if
　Candidate*←Candidate∩N({j:State(j)∈prePS});　//**Step 1.0**
　While Candidate*≠ ϕ , do 从 Candidate*选|$N(V)\cap$ { j: State(j) ∈prePS }|最大的点 V; **Update**;
　End While

//检查是否产生$(k,0)$状态，是的话执行 **Step 1.1**，否则跳到 **Step 1.2**

Candidate*←Candidate ∩ $N(\{j$: State$(j)=(k,0)\})$; prePS←prePS∪$(k,0)$; **//Step 1.1**

While Candidate$^*\neq\phi$, do 从 Candidate*选$|N(V)\cap\{j$: State$(j)\in$prePS$\}|$最大的点 V; **Update**;

End While

Candidate* ←Candidate; **//Step 1.2**

While Candidate$^*\neq$, do 从 Candidate*中随机选择节点 V 作为打孔节点；**Update**; End while

d←d −1;

End While

k ←k+1;

End While

Update:

(1)Candidate and Candidate*:

Level(V)←min{State(j).k: $j\in N(V)$}+1; A←{j:$j\in N(V)$, State(j) =nowPS}; B←{j:$j\in N(V)$, State(j) ∈ prePS}; Candidate←Candidate \ $N(\{j$: $j\in B\})$, Candidate* ←Candidate*\$N(\{j$:$j\in B\})$; ∀ $j\in A$, if (State(j).l =Level(V))Candidate←Candidate\$N$(Partner($j$)\{$j$},$j\in A$), Candidate* ←Candidate*\$N$(Partner($j$)\ {$j$},$j\in A$); else Candidate←Candidate\$N$(Partner($j$), $j\in A$), Candidate*←Candidate*\ N(Partner(j), $j\in A$); End if

(2)State updating for $N(V)$:

Count←$|\{j$:State(j).l= Level(V)−1, $j\in N(V)\}|$;

∀ $j\in N(V)$, if(State(j).l =Level(V))State (j).p←0;

else State (j) ← (Level(V), Count); End if

(3)State updating for Partner(j)\{j},$j\in N(V)$:

∀ $j'\in$Partner(j)\{j}, $j\in N(V)$, State (j').p←State (j').p −1;

If (State (j').p =1), then N (j') is marked to be fixed nodes; G_0←G_0 ∪ $N(j')\backslash\{V\}$; Candidate←Candidate\$N(j')$; Candidate*←Candidate* \$N(j')$; End if

在 Step 0 中，选取状态为 nowPS 的校验节点的邻居来打孔，此步中得到的打孔节点都是 1-SR。Step 1.0 中，prePS 是之前处理的状态。寻找点 V 满足$|N(V)\cap\{j$: State$(j)\in$prePS $\}|$最大，打孔时对 nowPS 校验节点的邻居影响就会尽可能小。Update(1)中，Level(V)指打孔节点 V 的恢复水平；为保证新打孔节点具有最多的存活校验节点，把 $N(\{j$:$j\in B\})$从候选变量节点中删除；而为使前打孔节点受到的影响最小化，使其存活校验节点数只减少 1，故从候选变量节点中删去 N(Partner(j)\{j}, $j\in A$)。

为了更好地理解本文提出的算法，以图 2 表示的码作为例子。最初，打孔候选节点是$\{V_1, V_2,\cdots, V_{10}\}$，校验节点状态为(0,0)。选 V_1 打孔，则 $N(N(V_1))$ 必须从候选节点中去掉，以最大化 V_1 存活校验节点数目。C_1 和 C_2 状态变为(1,2)。以同样方式选 V_8 作为下一个打孔节点，更新后打孔候选节点集合为空。

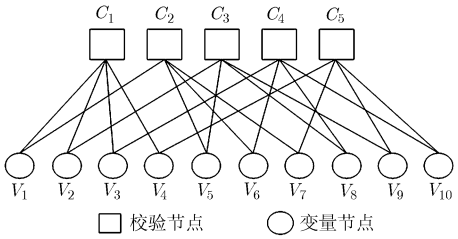

图 2 (2, 4)-规则码的例子

C_3 和 C_4 状态变为(1,2)。至此得到两个 1-SR 节点，并保证它们有尽可能多的存活校验节点。接下来的选择中，新打孔节点会使前打孔节点的存活校验节点数目减少，这种减少必须最小化。建立集合为打孔候选节点：N({校验节点 j:State(j)=(1,2)})\已确定节点。集合为$\{V_2, V_3, V_4, V_5, V_6, V_7, V_9, V_{10}\}$。$V_4$ 只与一个(1,2)校验节点相连，故选 V_4 打孔。V_4 的邻居 C_5 , C_1 状态分别变为(1,1)和(1,0)。C_1 与两个 1-SR 节点 V_1, V_4 相邻，故现在不能恢复它们中任一个。V_1 只剩下一个存活校验节点 C_2, C_2 状态变为(1,1)。选 V_4 后，选 V_2 打孔，其邻居中只有一个(1,2)校验节点。V_2 是一个 2-SR 节点。至此所有变量节点均确定，或者被打孔或者在 G_0 中，终止寻找。最后得到打孔变量节点是$\{V_1, V_8, V_4, V_2\}$。

3.3 算法分析与比较

MSCN 算法 Step 0 步骤产生的打孔节点均属 1-SR 节点，且打孔节点的邻居都是存活校验节点。以变量节点度数 d_v，校验节点度数 d_c 的规则 LDPC 码为例进行说明。此时 1-SR 节点 V 的存活校验节点数都有 d_v 个，根据定理 1，节点 V 接收到 LLR 信息期望值为

$$m_u(V)=d_v\phi^{-1}\left\{1-\left[1-\phi\left(m_{u_0}\right)\right]^{S(C_j)}\right\}$$
$$=d_v\phi^{-1}\left\{1-\left[1-\phi\left(m_{u_0}\right)\right]^{d_{c-1}}\right\} \quad (16)$$

相对于只有一个存活校验节点的情况，期望值达到 d_v 倍，故能更快趋向收敛值。若打孔目标码率较高，则需打孔节点数也相应较多，Step 0 中产生的 1-SR 节点并不足以支持高码率，故需继续打孔。Step 1 中处理状态(k, d)，此时产生的打孔节点 V 主要是 $(k+1)$-SR 节点，算法使 V 的存活校验节点数尽可能多。但是对变量节点 V 打孔，会难以避免地影响到 1-SR 节点，新打孔节点可能会破坏原先的存活校验节点，使其不能恢复 V, V 的存活校验节点数变少。MSCN 算法严格控制了 V 存活校验节点数减少的速度。尽管如此，当要求码率足够高，新打孔节点足够多时，V 的存活校验节点数可能退化到变为 1，此时与 Grouping/Sorting 方案所保证的 1 个存

活校验节点相一致。由上边分析可知，低码率时 MSCN 算法能使打孔节点的存活校验节点数最大化，LLR 信息期望值较高，加快译码收敛速度并降低出错概率；但若打孔到高码率，由于新打孔节点会破坏原先存活校验节点，最坏情况下时原打孔节点只有一个存活校验节点，此时 MSCN 算法便不再具有明显优势。

Grouping 算法使低 k-SR 变量节点的数目最大化[5,9]，而 MSCN 算法则是使打孔变量节点的存活校验节点数目最大化。MSCN 算法在恢复打孔变量节点时，一次译码迭代中打孔变量节点就可以得到多个存活校验节点传输给它的信息。根据定理 1，当打孔节点的恢复水平相同时，多个存活校验节点能够比单一存活校验节点提供更多的 LLR 信息，使变量节点得到的 LLR 信息期望值更大，加快译码收敛速度；并由式(15)，更大的 LLR 信息期望值能降低出错概率。这就是低码率时 MSCN 算法在性能上优于 Grouping/Sorting 算法的原因。

4 仿真分析

本节对 MSCN 算法进行实验仿真，并与 Grouping/Sorting 算法比较。实验分别针对规则码和非规则码两种情况，母码码长为 1152，码率为 0.5。规则码由 PEG 算法[10]构造生成，变量节点度数为 3，校验节点度数为 6。非规则码是采用 802.16e 标准码率为 0.5 的 LDPC 码。

实验中设置最大译码迭代次数为 50 次。为使实验数据具有平稳性，实验运行中要求实验样本至少有 20000 个码字分组或至少观察到 200 个分组错误。为了得到其它码率的新码，需要打孔的节点数目如表 2 所示。

表 2 码率及打孔数

码率	打孔数
0.60	192
0.65	265
0.70	329

表 3，表 4 显示了不同算法下 i-SR 节点分布的情况，G_i 是 i-SR 节点的集合，而 G_0 是指没有打孔的节点。由表可知，两种算法产生的打孔节点集合中都是 G_1 最大，低 i-SR 变量节点的数目要比高 i-SR 节点的多，即大部分打孔变量节点的恢复水平很低，在译码时基本只需要一两次迭代就可以恢复。两种算法相比较，Grouping/Sorting 算法最大化低 i-SR 节点的数目，而 MSCN 算法最大化 i-SR 节点的存活校验节点数目，故无论规则码还是非规则码，Grouping/Sorting 算法所产生的 G_1 都比 MSCN 算法的 G_1 大，但是 MSCN 算法所产生的总打孔节点数目大于 Grouping/Sorting 算法的，能支持更大的码率范围。

图 3 和图 4 分别是规则 LDPC 码和非规则 LDPC 码采用两种不同打孔方案时的比特错误概率曲线。在图中 0.6-Grouping/Sorting 是指打孔采用的是 Grouping/Sorting 算法，打孔后得到的新码码率为 0.6。类似地，0.6-Proposed Algorithm 是指打孔采用的是我们提出的 MSCN 算法，打孔后得到的新码码率为 0.6，其它依此类推。

表 3 i-SR 变量节点集合分布情况(规则码情形)

	G_0	G_1	G_2	G_3	G_4
Grouping/Sorting	746	338	64	4	0
MSCN	728	192	145	80	7

表 4 i-SR 变量节点集合分布情况(非规则码情形)

	G_0	G_1	G_2	G_3	G_4
Grouping/Sorting	723	364	62	3	0
MSCN	720	240	144	48	0

从图中可以看出，本文所提的 MSCN 算法具有比 Grouping/Sorting 算法更优越的性能。虽然 Grouping/Sorting 算法得到的 G_1 集合更大，但 MSCN 算法保证每一个打孔节点的存活校验节点尽可能多，降低了错误概率，故能拥有更佳的性能。图 3 和图 4 中，码率 0.6, 0.65 时，MSCN 算法比 Grouping/Sorting 算法在 BER 为 10^{-5} 时都具有 0.2 dB 的增益；但到了高码率，新打孔节点会破坏前打孔节点的存活校验节点，前打孔节点的存活校验节点数逐渐减少，最坏情况下甚至可能只有一个存活校验节点，变得与 Grouping/Sorting 算法一样，而且运用了比较多的高 i-SR 节点，故在高码率时 MSCN 算法的优势逐渐下降。MSCN 算法在低、高码率具有不同的性能表现，这些不同的性能表现与第 3.1 节中的理论分析是不矛盾的。理论分析表明，多个存活校验节点能提高译码收敛速度，并且降低错误概率。低码率时 MSCN 算法打孔的变量节点具有多个存活校验节点，其性能很符合理论分析的结果；而到了高码率，为了满足高码率的要求，需要的打孔数目较多，MSCN 算法中原先的打孔节点难以避免地受到新打孔节点的影响，原打孔节点不再具有多个存活校验节点，最坏情况下甚至只拥有一

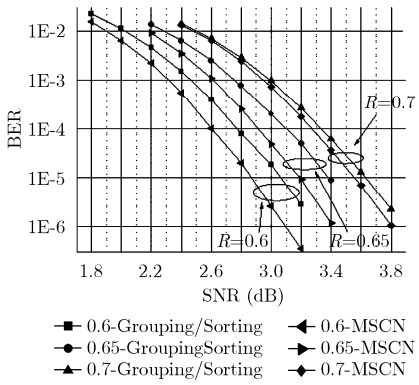

图 3 规则 LDPC 码的 BER

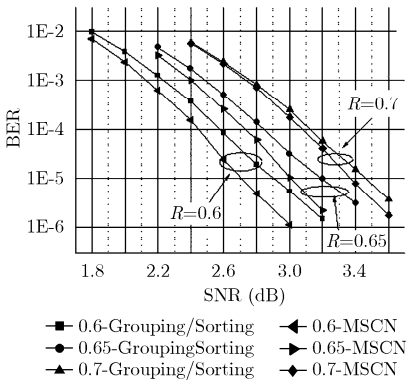

图 4 非规则 LDPC 码的 BER

个存活校验节点，由于存活校验节点变少了，故性能也下降，优势不明显，这从反面印证了理论分析的结果。实验的结果与第 3 节中对算法的分析与比较是一致的。

5 结论

本文研究码率兼容 LDPC 码多存活校验节点的作用，理论分析表明多个存活校验节点在提高译码收敛速度的同时还能降低错误概率。基于这种分析，我们提出了一种新颖有效的打孔方案，这种方案最大化了打孔变量节点的存活校验节点数目，它与 Ha 等人提出的 Grouping 和 Sorting 方案在寻找节点和更新的规则上差别很大。AWGN 信道上的实验仿真结果表明，低码率时本文所提出的方案在性能上优于 Ha 等人的方案，而高码率时优势不明显。

参 考 文 献

[1] Ha J, Kim J, and McLaughlin S W. Rate-compatible puncturing of low-density parity-check codes [J]. *IEEE Transactions on Information Theory*, 2004, 50(11): 2824–2836.

[2] Ha J and McLaughlin S W. Optimal puncturing distributions for rate compatible low-density parity-check codes[C]. IEEE International Symposium on Information Theory, Yokohama, Japan, Jun./Jul. 2003: 233.

[3] Li J and Narayanan K. Rate-compatible low density parity check codes for capacity-approaching ARQ scheme in packet data communications[C]. International Conference on Communications, Internet, and Information Technology, Virgin Islands, US, Nov. 2002: 201–206.

[4] Yazdani M R and Banihashemi A H. On construction of rate-compatible low-density parity-check codes [J]. *IEEE Communications Letters*, 2004, 8(3): 159–161.

[5] Ha J, Kim J, Klinc D, *et al.*. Rate-compatible punctured low-density parity-check codes with short block lengths[J]. *IEEE Transactions on Information Theory*, 2006, 52(2): 728–738.

[6] Kim J, Ramamoorthy A, and McLaughlin S W. The design of efficiently-encodable rate-compatible LDPC codes [J]. *IEEE Transactions on Communications*, 2009, 57(2): 365–375.

[7] Shi C and Ramamoorthy A. Design and analysis of E^2RC codes using EXIT chart[C]. IEEE ICC, Dresden, Germany, June 2009: 1–5.

[8] Chung S Y, Richardson T J, and Urbanke R L. Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation [J]. *IEEE Transactions on Information Theory*, 2001, 47(2): 657–670.

[9] Vellambi B N and Fekri F. Finite-length rate-compatible LDPC codes: a novel puncturing scheme [J]. *IEEE Transactions on Communications*, 2009, 57(2): 297–301.

[10] Hu X Y, Eleftheriou E, and Arnold D M. Regular and irregular progressive edge-growth tanner graphs [J]. *IEEE Transactions on Information Theory*, 2005, 51(1): 386–398.

苏和光：　男，1985 年生，硕士生，研究方向为信道编码.

夏树涛：　男，1972 年生，教授，博士生导师，研究方向为信道编码和网络编码等.