LDPC码

引言

- Turbo码的发明引发了对基于图模型的编码设计和迭代译码算法的研究热潮
- Mackey等人发现由Gallager早在1960年提出的一种低密度奇偶校验码（Low Density Parity Check Codes，LDPC码）也是一种实用的好码
- 与Turbo码相比，LDPC码具有描述简单、解码复杂度低、实用灵活等特点

Gallager's 1960 Sc.D. thesis, entitled "Low Density Parity Check Codes," was published by the M.I.T. Press in 1963. An abbreviated version appeared earlier (January 1962) in the IRE Transactions on Information Theory：

[1] R. G. Gallager, "Low density parity check codes," IRE Trans. Information Theory, vol. IT-8, pp. 21–28, January 1962.
[2] R. G. Gallager, Low Density Parity Check Codes. Cambridge, MA: MIT Press, 1963.
[3] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," IEEE Trans. Inf. Theory, vol. 45, no. 2, pp. 399–431, Mar. 1999.

引言

Robert Gray Gallager

Born	May 29, 1931
Nationality	American
Fields	Information theory
Almamater	University of Pennsylvania MIT
Notable awards	Claude E. Shannon Award(1983) IEEE Medal of Honor (1990) Harvey Prize (1999) Marconi Prize (2003) Dijkstra Prize (2004)

引言

- LDPC码目前可达到的性能:
  - AWGN信道中接近信道容量的仿真结果
    - 冠军码：距离香侬极限0.0045dB
    - 码长为10^7：误码率为10^{-6}时距离香侬极限0.04dB
- 众多新一代通信标准中的信道编码方案:
  - DVB-S2
  - 中国地面广播电视标准-CMMB
  - WiMAX 802.16e
  - 802.11n提案
  - 深空探测推荐方案
  - 5G移动通信标准

CMMB: ½码率LDPC码为例
4608*9216的矩阵，每行有6个“1”

目录

LDPC码描述

- H矩阵
- 二分图
- 维度分布函数
- 参数

LDPC码描述

H矩阵

- LDPC码是一种线性分组码
- LDPC码的校验矩阵H是稀疏矩阵:
  - 每一行表示一个校验方程约束，包含j个码元
  - 每一列表示一个码元参加 k个校验方程
  - 任意两个校验方程包含至多一个相同码元

LDPC码描述

H矩阵举例(15*20)

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

H矩阵 二分图 维度分布函数 参数

LDPC码描述

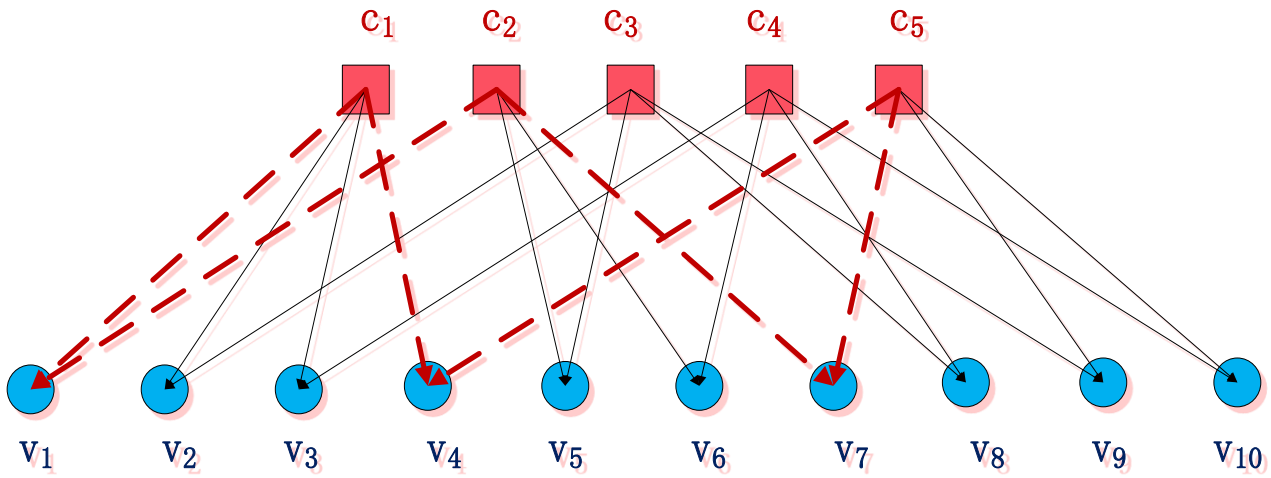
H矩阵

- LDPC码的校验矩阵
  - j和k为常数的H矩阵，其零空间构成正则LDPC码，记作（n, j, k ）正则LDPC码
  - j 和k不是常数的H矩阵，其零空间构成非正则LDPC码

LDPC码描述

二分图

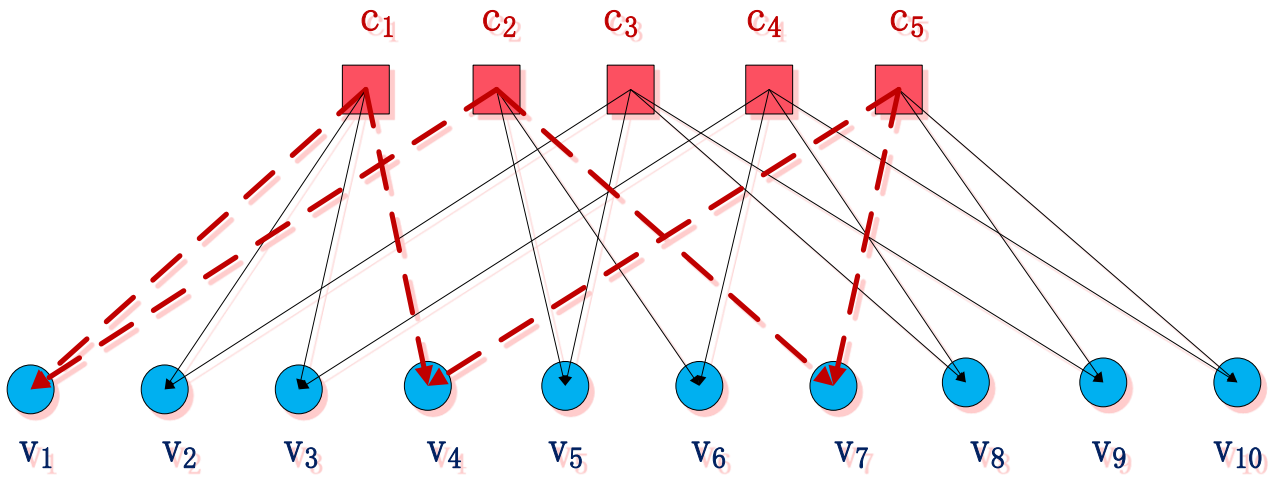
- LDPC码可以用二分图来表示.
- LDPC码对应的二分图包括两种节点:
  - 变量节点（variable nodes）
  - 校验节点（check nodes）



LDPC码描述

二分图

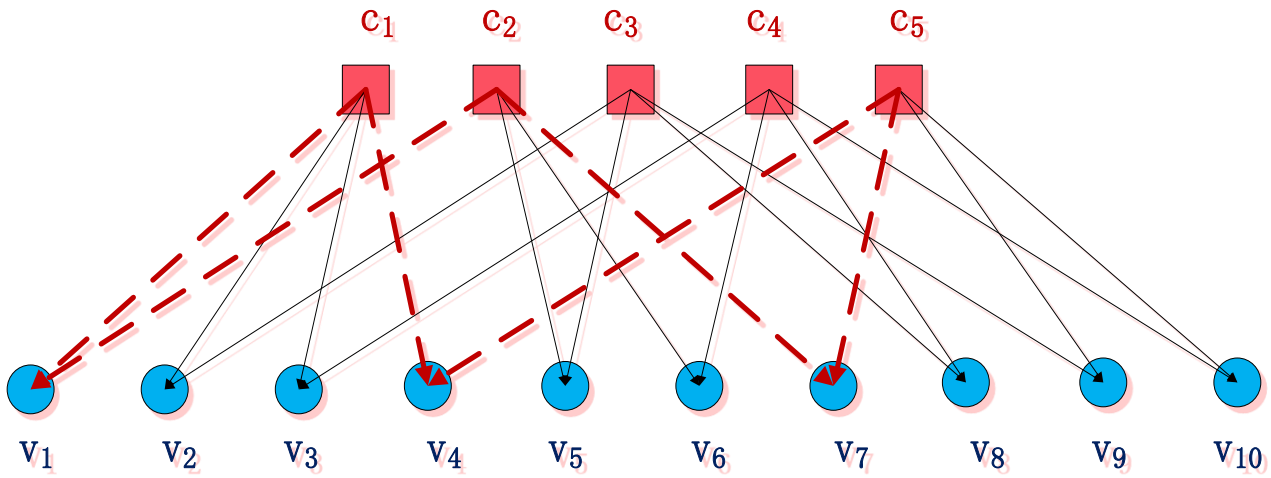
- 变量节点（Variable Node）与LDPC码的一个码元相对应，其个数等于码长*n*
- 校验节点（Check Node）与校验方程对应，其个数与校验矩阵*H*的行数相同
- 变量节点j和校验节点i之间的联线表示与之对应的第i校验方程中含有第j个码元



LDPC码描述

二分图

- 相互之间有连接线的节点互为邻点（Neighbor）
- 一个节点的邻点数量称为这个节点的维度（Degree）
- 回路（cycle，环）
  - 从一个端点出发，经过不重复的边回到出发端点
  - 经过的边数定义为回路的长度
  - 图中最短回路长度称为图的周长（girth）



LDPC码描述

维度函数

- 变量节点维度分布函数：

$$\lambda(x) := \sum_{i=2}^{d_v} \lambda_i x^{i-1}$$

λ_i 表示在二分图中，与维度为i的变量节点相联的边（Edge）占所有边的比例

d_v 表示变量节点的最大维度

LDPC码描述

维度函数

- 校验节点维度分布函数：

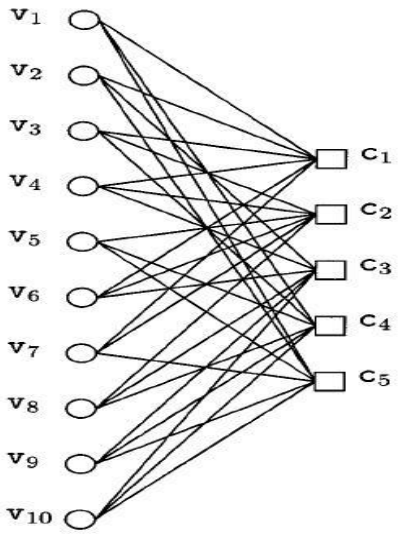
$$\rho(x) := \sum_{i=2}^{d_c} \rho_i x^{i-1}$$

ρ_i 表示在二分图中，与维度为i的校验节点相联的边（Edge）占所有边的比例

d_c 表示校验节点的最大维度

LDPC码描述

维度函数



- k=3, j=6的（3, 6）正则码中

$$\lambda(x) = x^2 \qquad \rho(x) = x^5$$

维度分布函数表示的LDPC码记为

$$\left(\lambda(x), \rho(x)\right)$$

LDPC码描述

参数

- 设总边数为B，则变量节点数n为

$$n=\sum_i \frac{B\cdot\lambda_i}{i}=B\sum_i \frac{\lambda_i}{i}=B\int_0^1 \lambda(x)dx$$

设变量节点数为n,则总边数B为

$$B=\frac{n}{\sum_i \frac{\lambda_i}{i}}=\frac{n}{\int_0^1 \lambda(x)dx}$$

LDPC码描述

参数

- 设总边数为B，则校验节点数m为

$$m=\sum_i \frac{B\cdot\rho_i}{i}=B\sum_i \frac{\rho_i}{i}=B\int_0^1 \rho(x)dx$$

设校验节点数为m,则总边数B为

$$B=\frac{m}{\sum_i \frac{\rho_i}{i}}=\frac{m}{\int_0^1 \rho(x)dx}$$

LDPC码描述

参数

- 维度为i的变量节点数:

$$B \cdot \lambda_i / i = n \frac{\lambda_i / i}{\sum_{j=2}^{d_v} \lambda_j / j} = n \frac{\lambda_i / i}{\int_0^1 \lambda(x) dx}$$

维度为i的校验节点数:

$$B \cdot \rho_i / i = m \frac{\rho_i / i}{\sum_{j=2}^{d_c} \rho_j / j} = m \frac{\rho_i / i}{\int_0^1 \rho(x) dx}$$

LDPC码描述

参数

- 码率:

$$r = 1 - \frac{m}{n}$$

$$= 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx}$$

目录

- LDPC码描述
- **LDPC码构造**
- 和积算法
- LDPC码解码
- 性能分析

LDPC码构造

- 构造目标
- 构造准则
- 构造方法

LDPC码构造----目标

- 构造一个二进制稀疏矩阵
  - 任意两行之间至多只有一个共同的位置有‘1’
  - 任意两列之间至多只有一个共同的位置有‘1’

- 衡量好坏的标准
  - 误码率
    - 通过仿真得到误码率曲线

LDPC码构造----准则

- 获得更好的误码性能
  - 更大的环
    - 译码采用迭代译码←前提：节点间传递的信息统计独立
    - 校验矩阵对应的二分图中有环存在→某一个节点发出的信息经过一个环长的传递会被传回本身，从而造成自身信息的叠加，破坏了独立的假设，影响译码的准确性
    - 希望：大环多，小环少，尤其避免最短环4（4-cycle）
    - 定理：LDPC码的任意一个长为L的环，满足L≥4，且L是2的倍数
  - 更大的最小码距
    - 高信噪比下降低error floor
  - 更均匀的环分布特性
    - 更快更均匀地收敛
  - 更规则的H矩阵结构
    - 便于工程实现
  - ……

LDPC码构造----方法

- 随机搜索
  - 最初的方法
- 图的方法
  - PEG（Progressive Edge Growth）算法
  - Bit-Filling算法
- 代数的方法（结构化的方法）
  - 有限几何方法
  - 基于RS码的方法
  - 组合数学的方法（BIBD平衡不完全区块设计）
  - 置换矩阵的方法

目录

- LDPC码描述
- LDPC码构造方法
- 和积算法
- LDPC码解码
- 性能分析

和积算法

置信度

- 采用概率对数比作为置信度

$$LLR(u)=\log\frac{P\{u=+1\}}{P\{u=-1\}}$$

符号的概率对数比与符号概率等价

$$P\{u=+1\}=\frac{e^{LLR(u)}}{1+e^{LLR(u)}}$$

$$P\{u=-1\}=\frac{1}{1+e^{LLR(u)}}$$

和积算法

基本思想

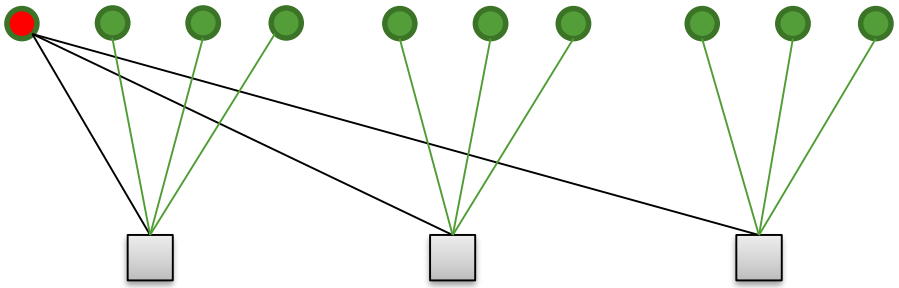
- 编码系统中，各个符号之间存在校验和关系
  - 已知各个校验和的置信度，求出参与校验的各个符号的置信度
  - 已知各个符号的置信度，求出各个校验和的置信度

和积算法

约束条件

开始推导前的假设与定义

- 数字符号 u_i 参与 N 个校验和 S_n，其中
  - u_i 之外的数字符号只出现一次
  - u_i 之外的数字符号相互独立

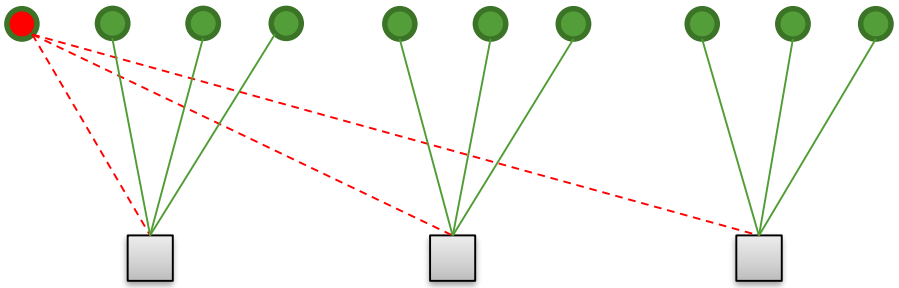


1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0
0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1

和积算法

从校验和的置信度得到符号的置信度

- 将数字符号 u_i从校验和 s_n 中除去，得到“修正校验和” s'_n
  - “修正校验和”的置信度➔数字符号 u_i 的置信度
  - 校验和包含的其余符号的置信度➔“修正校验和”的置信度



和积算法

从校验和的置信度得到符号的置信度

修正校验和举例:

- 有数字符号u_1，u_2，u_3满足校验关系
  - $S_1 = u_1+u_2+u_3=0$ ➔ $+1$；
- 去除符号u_1的作用，得到修正校验和
  - $S_1' = u_2+u_3=1$ ➔ -1, if $u_1=1$ （-1）；
  - $S_1' = u_2+u_3=0$ ➔ $+1$, if $u_1=0$（$+1$）；

和积算法

从校验和的置信度得到符号的置信度

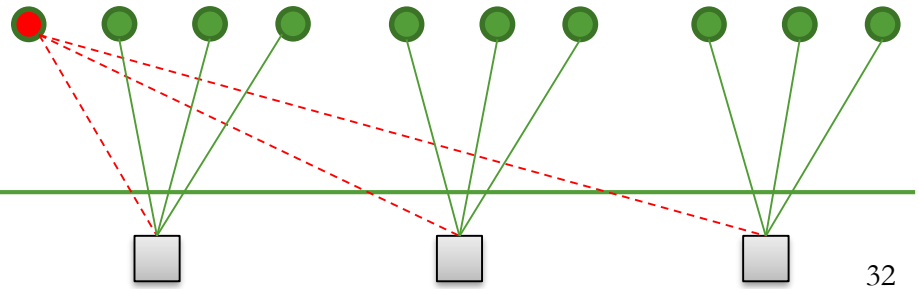
- 引入“修正校验和” S'_n后的概率

$$P(u_i=+1, s_n=+1,\ n=1,2,\cdots N)$$

$$=P(s_n=+1,\ n=1,2,\cdots N/u_i=+1)\cdot P(u_i=+1)$$

$$=P(s'_n=+1,\ n=1,2,\cdots N)\cdot P(u_i=+1)$$

$$=\prod_{n=1}^{N}P(s'_n=+1)\cdot P(u_i=+1)$$



和积算法

从校验和的置信度得到符号的置信度

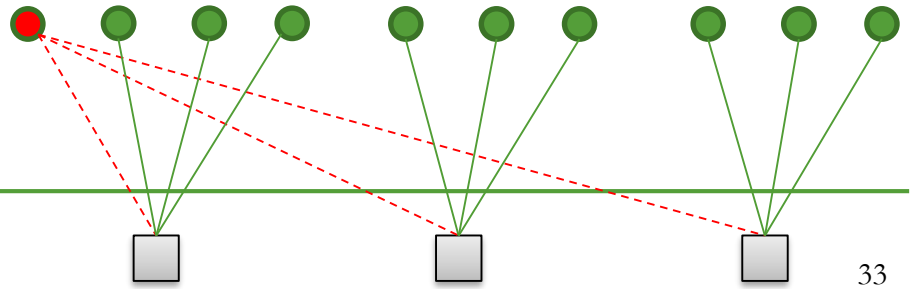
- 引入“修正校验和” S'_n后的概率

$$P(u_i=-1, s_n=+1,\ n=1,2,\cdots N)$$

$$=P(s_n=+1,\ n=1,2,\cdots N/u_i=-1)\cdot P(u_i=-1)$$

$$=P(s'_n=-1,\ n=1,2,\cdots N)\cdot P(u_i=-1)$$

$$=\prod_{n=1}^{N}P(s'_n=-1)\cdot P(u_i=-1)$$



和积算法

从校验和的置信度得到符号的置信度

$$LLR(u_i) = \log \frac{P(u_i = +1, s_n = +1,\ n = 1, 2, \cdots N)}{P(u_i = -1, s_n = +1,\ n = 1, 2, \cdots N)}$$

$$= \log \prod_{n=1}^{N} \frac{P(s'_n = +1)}{P(s'_n = -1)} + \log \frac{P(u_i = +1)}{P(u_i = -1)}$$

$$= \sum_{n=1}^{N} \log \frac{P(s'_n = +1)}{P(s'_n = -1)} + \log \frac{P(u_i = +1)}{P(u_i = -1)}$$

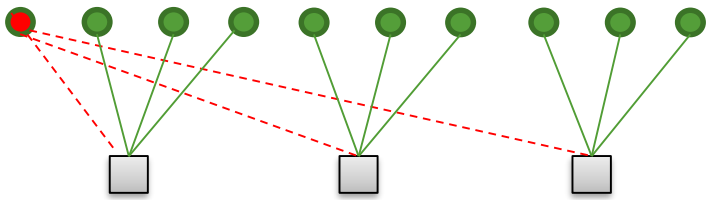
$$= \sum_{n=1}^{N} LLR(s'_n) + \log \frac{P(u_i = +1)}{P(u_i = -1)}$$

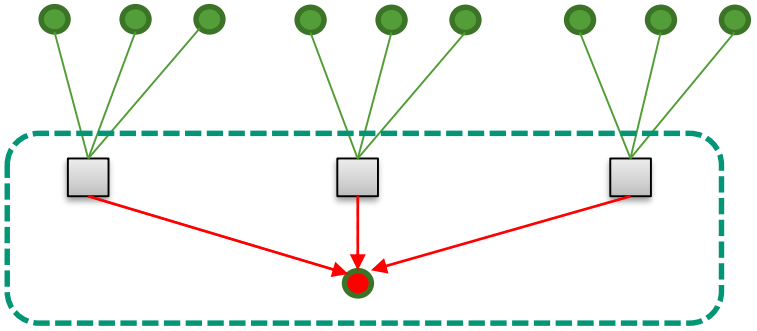
和积算法

从校验和的置信度得到符号的置信度

- 校验和约束下
  - 数字符号置信度 = 修正校验和置信度+ 先验概率对数比

$$LLR(u_i) = \sum_{n=1}^{N} LLR(s'_n) + \log \frac{P(u_i = +1)}{P(u_i = -1)}$$

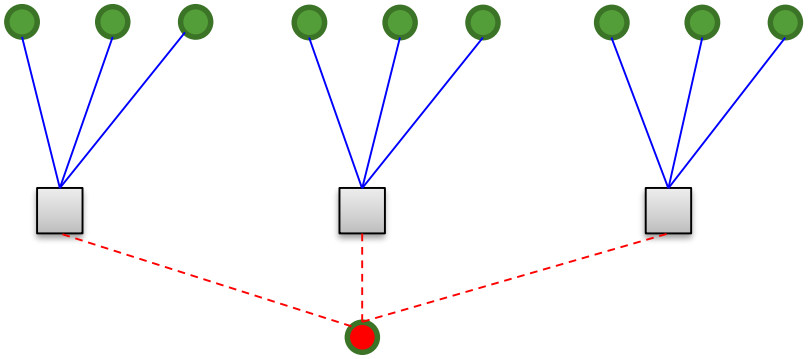




和积算法

从符号的置信度得到校验和的置信度

- 校验和约束下的置信度传递
  - “修正校验和”的置信度➔数字符号 u_i 的置信度
  - **符号的置信度➔校验和的置信度**



和积算法

从符号的置信度得到校验和的置信度

开始推导前的讨论

- 各个符号独立时
  - 0 ➔ +1 ， 1 ➔ -1
  - $$s_n = +1 \Leftrightarrow \prod_{i=1}^{k} u_i = +1; \quad s_n = -1 \Leftrightarrow \prod_{i=1}^{k} u_i = -1$$
  - $$LLR(s_n) = \log \frac{P(\prod_{i=1}^{k} u_i = +1)}{P(\prod_{i=1}^{k} u_i = -1)}$$

和积算法

从符号的置信度得到校验和的置信度

- 利用数学归纳法可证明：

$$P\left\{\prod_{i=1}^{k} u_i = +1\right\} = \left\{1 + \prod_{i=1}^{k}\left[2P\{u_i = +1\} - 1\right]\right\} \Big/ 2$$

$$P\left\{\prod_{i=1}^{k} u_i = -1\right\} = \left\{1 - \prod_{i=1}^{k}\left[2P\{u_i = +1\} - 1\right]\right\} \Big/ 2$$

和积算法

从符号的置信度得到校验和的置信度

- 续上页

$$LLR\{s_n\} = \log \frac{P\left\{\prod_{i=1}^{k} u_i = +1\right\}}{P\left\{\prod_{i=1}^{k} u_i = -1\right\}} = \log \frac{\left\{1 + \prod_{i=1}^{k}\left[2P\{u_i = +1\} - 1\right]\right\}/2}{\left\{1 - \prod_{i=1}^{k}\left[2P\{u_i = +1\} - 1\right]\right\}/2}$$

$$= \log \frac{\prod_{i=1}^{k}\left[P\{u_i = +1\} + \left(1 - P\{u_i = +1\}\right)\right] + \prod_{i=1}^{k}\left[P\{u_i = +1\} - \left(1 - P\{u_i = +1\}\right)\right]}{\prod_{i=1}^{k}\left[P\{u_i = +1\} + \left(1 - P\{u_i = +1\}\right)\right] - \prod_{i=1}^{k}\left[P\{u_i = +1\} - \left(1 - P\{u_i = +1\}\right)\right]}$$

和积算法

从符号的置信度得到校验和的置信度

$$P\{u=+1\}=\frac{e^{LLR(u)}}{1+e^{LLR(u)}}$$

$$P\{u=-1\}=\frac{1}{1+e^{LLR(u)}}$$

- 续上页

$$LLR\{s_n\}=\log\frac{\prod_{i=1}^{k}\left[P\{u_i=+1\}+P\{u_i=-1\}\right]+\prod_{i=1}^{k}\left[P\{u_i=+1\}-P\{u_i=-1\}\right]}{\prod_{i=1}^{k}\left[P\{u_i=+1\}+P\{u_i=-1\}\right]-\prod_{i=1}^{k}\left[P\{u_i=+1\}-P\{u_i=-1\}\right]}$$

$$=\log\frac{\prod_{i=1}^{k}\left[e^{LLR(u_i)}+1\right]+\prod_{i=1}^{k}\left[e^{LLR(u_i)}-1\right]}{\prod_{i=1}^{k}\left[e^{LLR(u_i)}+1\right]-\prod_{i=1}^{k}\left[e^{LLR(u_i)}-1\right]}$$

和积算法

从符号的置信度得到校验和的置信度

- 对比双曲正切函数 $\tanh(u/2)=\left(e^{u}-1\right)/\left(e^{u}+1\right)$
- 得到 $LLR(s_n)=\log\dfrac{1+\prod_{i=1}^{k}\tanh\left(LLR(u_i)/2\right)}{1-\prod_{i=1}^{k}\tanh\left(LLR(u_i)/2\right)}$
- 进一步化简得到

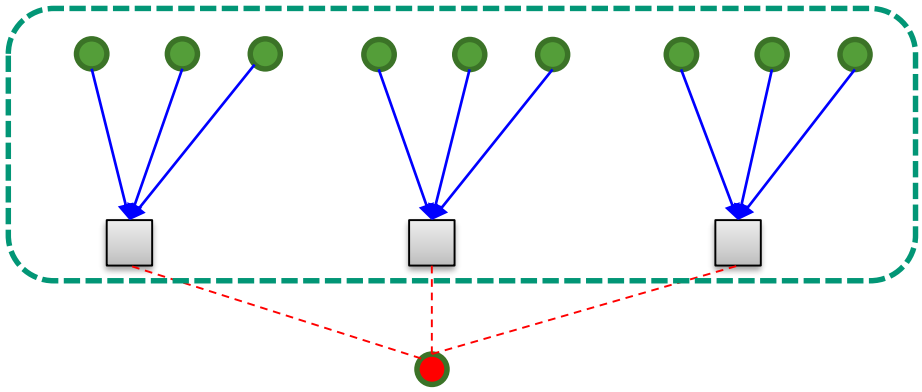
$$\tanh(LLR(s_n)/2)=\prod_{i=1}^{k}\tanh(LLR(u_i)/2)$$

和积算法

从符号的置信度得到校验和的置信度

- 校验和约束下
  - 校验和置信度一半的双曲正切 = 各个符号置信度一半的双曲正切连乘的积

$$\tanh(LLR(s_n)/2) = \prod_{i=1}^{k} \tanh(LLR(u_i)/2)$$



和积算法

小结

- 从校验和置信度得到符号置信度（求和）

$$LLR(u_i) = \sum_{n=1}^{N} LLR(s'_n) + \log \frac{P(u_i = +1)}{P(u_i = -1)}$$

- 从符号置信度得到校验和置信度（乘积）

$$\tanh(LLR(s_n)/2) = \prod_{i=1}^{k} \tanh(LLR(u_i)/2)$$

- 利用上述校验关系得到符号置信度进行迭代解码的算法被称为和积算法（sum-product）

和积算法

小结

- 通常可以将积算法简化为

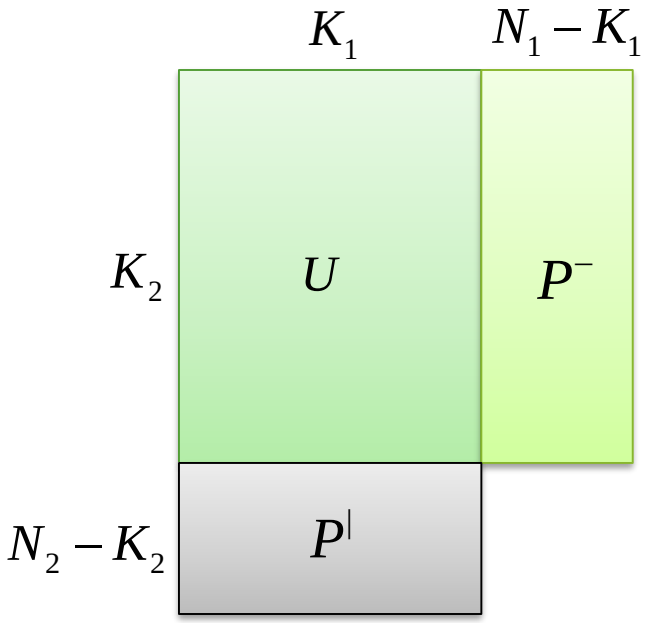
$$\tanh(LLR(s_n)/2) = \prod_{i=1}^{k} \tanh(LLR(u_i)/2)$$

$$LLR(s_n) \approx \prod_{i=1}^{k} sign(LLR(u_i)) \times \min(|LLR(u_i)|)$$

“最小和算法”

和积算法
应用举例

- 一个简单奇偶校验（SPC）组合码的译码
  - $K_1 \times K_2$个信息比特排列成的长方形阵列
  - 附加阵列和是和信息比特对应的校验比特
  - 符号“|”和“—”分别表示列和行



和积算法

应用举例

发送码字

- 对2×2个信息比特进行图中形式的奇偶校验编码
- BPSK调制

U_{11}	U_{12}	P_1^{-}
U_{21}	U_{22}	P_2^{-}
$P_1^{\mid}$	$P_2^{\mid}$	

+	+	+
+	-	-
+	-	

和积算法

应用举例

接收符号

- 经过AWGN信道，接收码字的信道似然比信息作为先验信息置信度

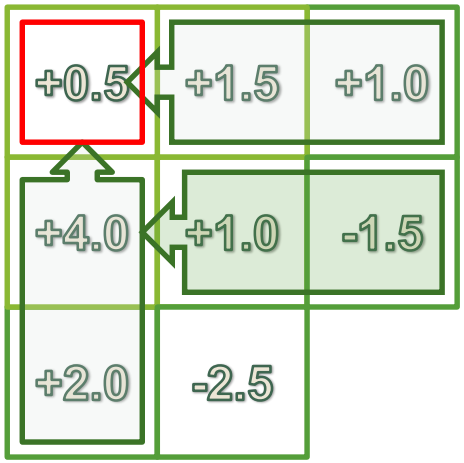
+0.5	+1.5	+1.0
+4.0	+1.0	-1.5
+2.0	-2.5	

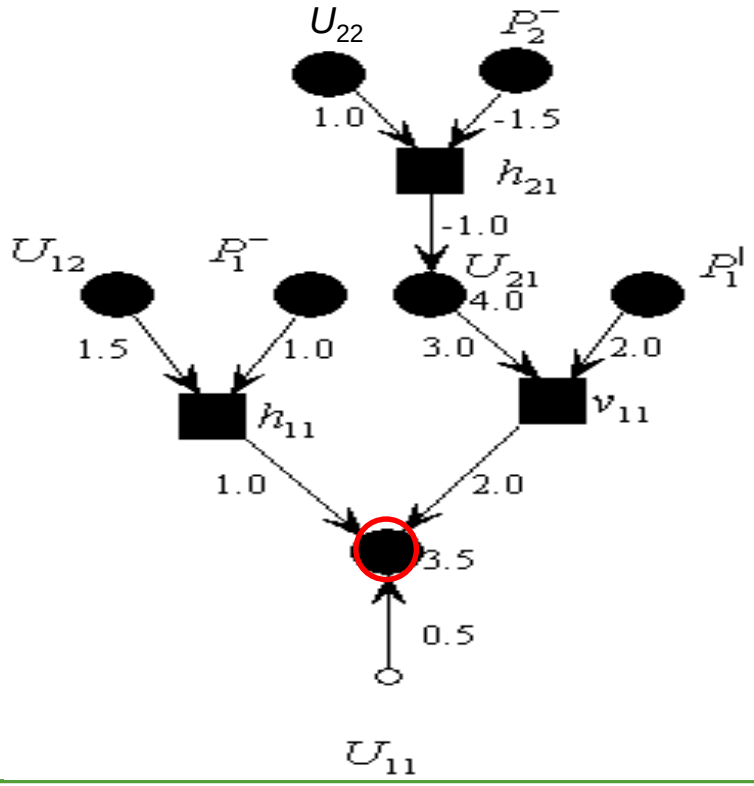
和积算法

应用举例

解码——置信度传播

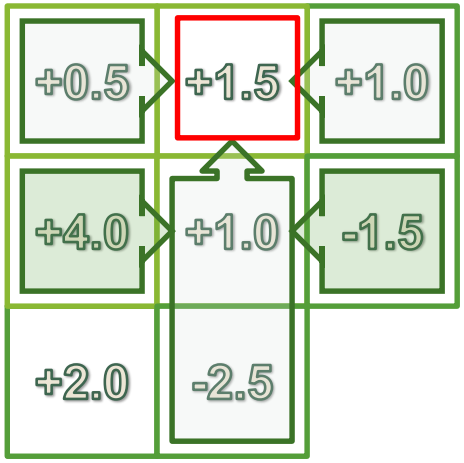
- 左上角比特的置信度传播





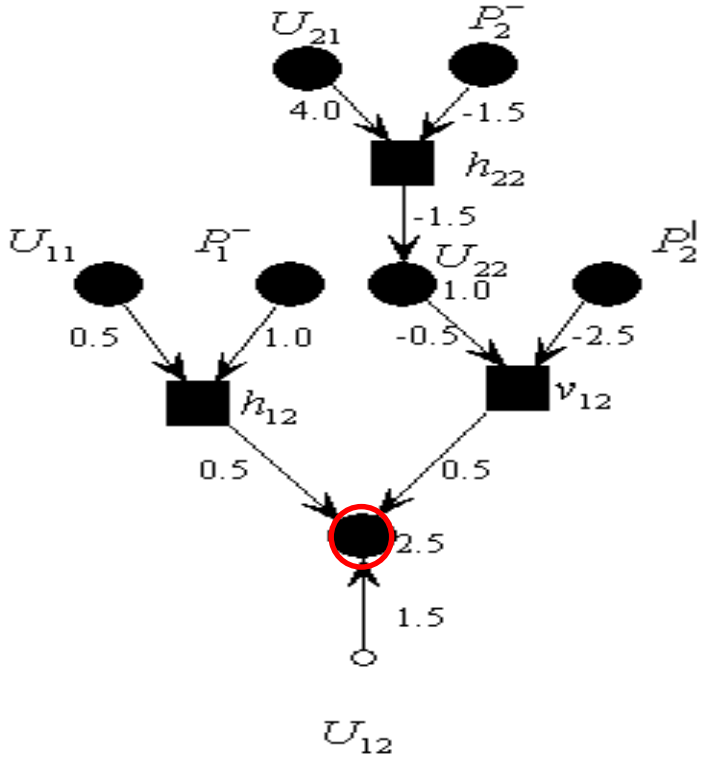
和积算法

应用举例



解码——置信度传播

- 右上角比特的置信度传播



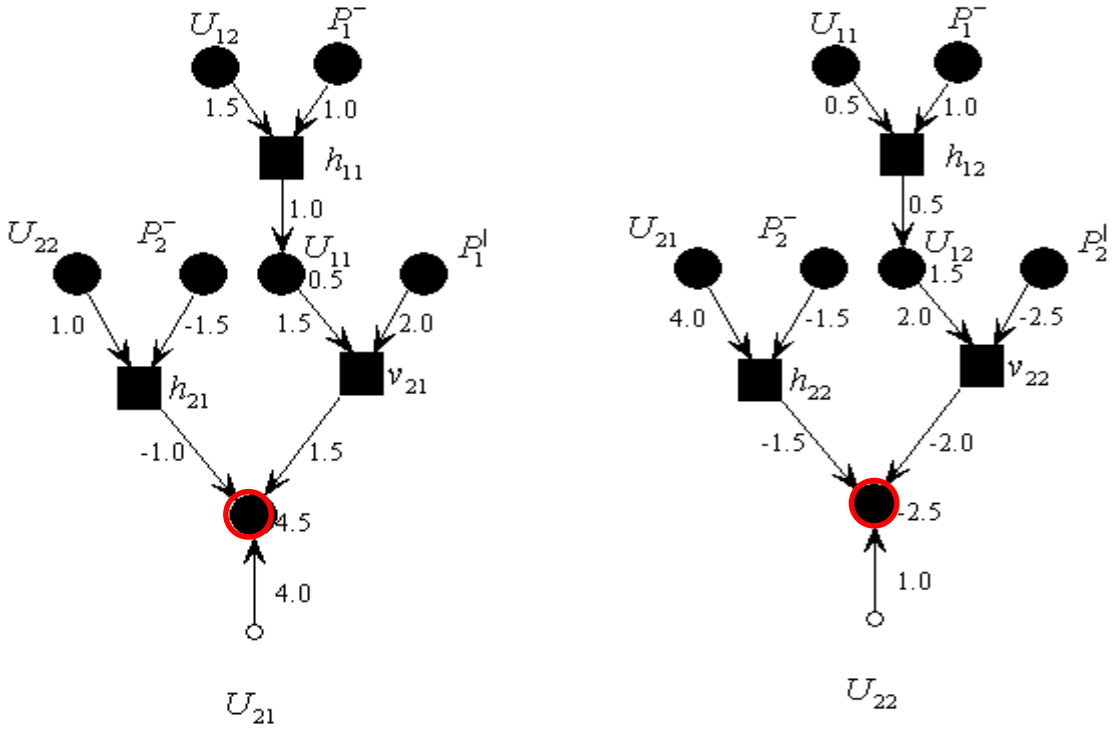
和积算法

应用举例

+0.5	+1.5	+1.0
+4.0	+1.0	-1.5
+2.0	-2.5	

解码——置信度传播

- 第二行两个信息比特的置信度传播

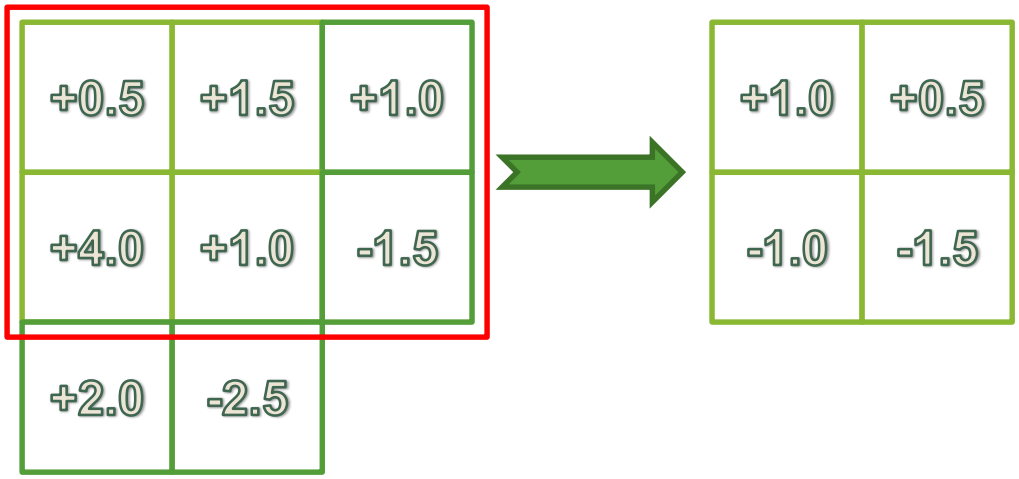


U_{11}	U_{12}	P_1^-
U_{21}	U_{22}	P_2^-
$P_1^\|$	$P_2^\|$	

和积算法

应用举例

- 利用行校验关系对2行信息比特进行解码
  - 从信息比特置信度得到相应“修正校验和”的置信度（积）



和积算法

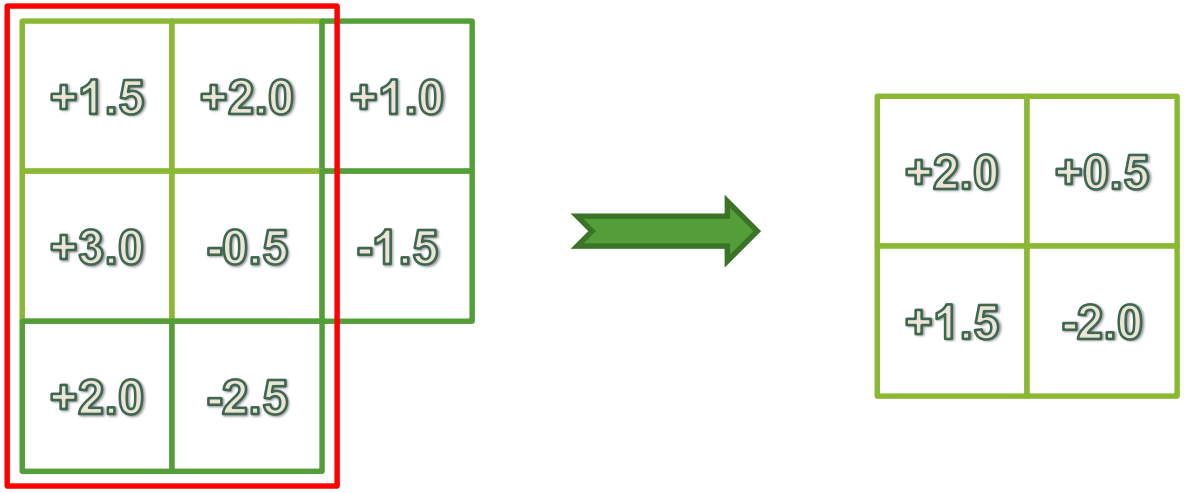
应用举例

- 利用行校验关系对2行信息比特进行解码
  - 将所有“修正校验和”置信度与先验信息置信度求和（和）

和积算法

应用举例

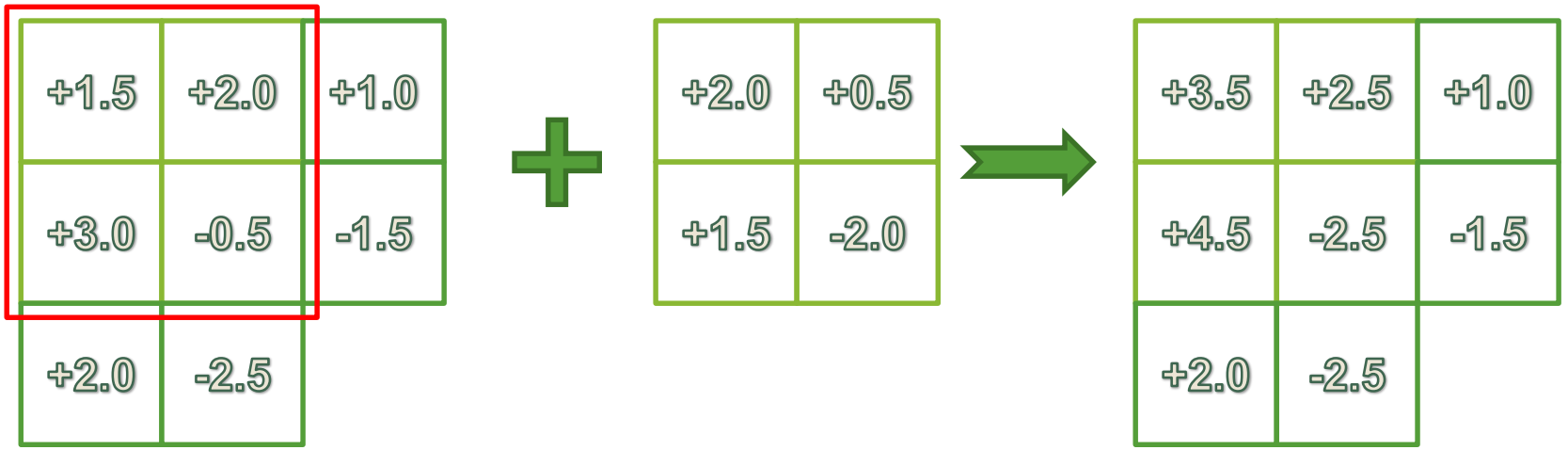
- 利用列校验关系对2列信息比特进行解码
  - 从信息比特置信度得到相应“修正校验和”的置信度（积）

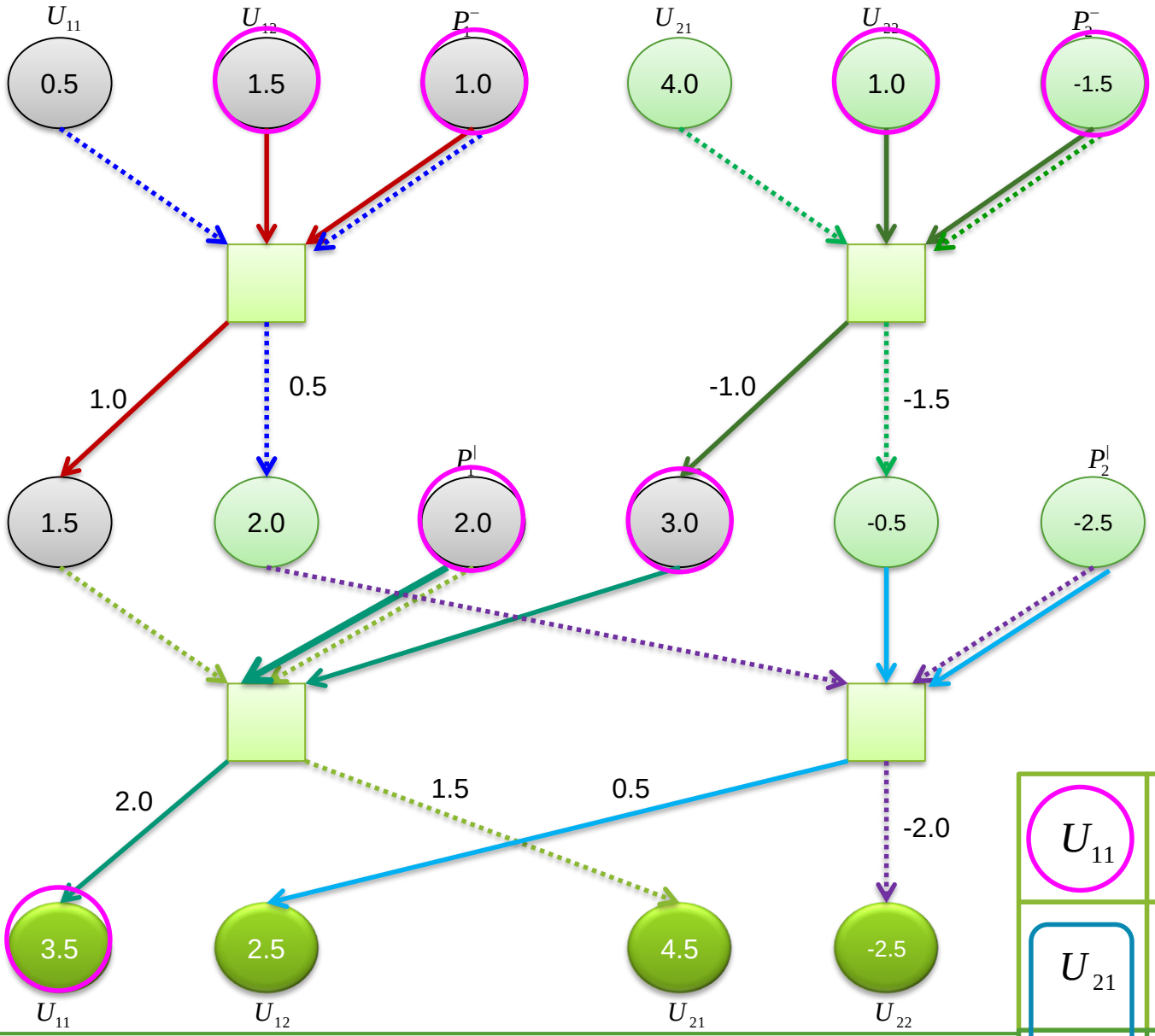


和积算法

应用举例

- 根据校验和置信度得到信息比特置信度
  - 每个信息比特都分别参与了行与列两个校验方程
  - 将所有“修正校验和”置信度与先验信息置信度求和（和）





U_{11}	U_{12}	P_1^-
U_{21}	U_{22}	P_2^-
$P_1^\|$	$P_2^\|$	

和积算法

应用举例

+0.5	+1.5	+1.0
+4.0	+1.0	-1.5
+2.0	-2.5	

- 根据最终置信度进行硬判决
  - 如果符合所有校验关系，结束译码
  - 如果不符合校验关系，进行迭代译码
    - 行解码时，加上列修正校验和置信度作为先验信息
    - 列解码时，加上行修正校验和置信度作为先验信息

+3.5	+2.5	+1.0
+4.5	-2.5	-1.5
+2.0	-2.5	

+	+	+
+	-	-
+	-	

目录

- LDPC码描述
- LDPC码构造方法
- **和积算法**
- **LDPC码解码**
- 性能分析

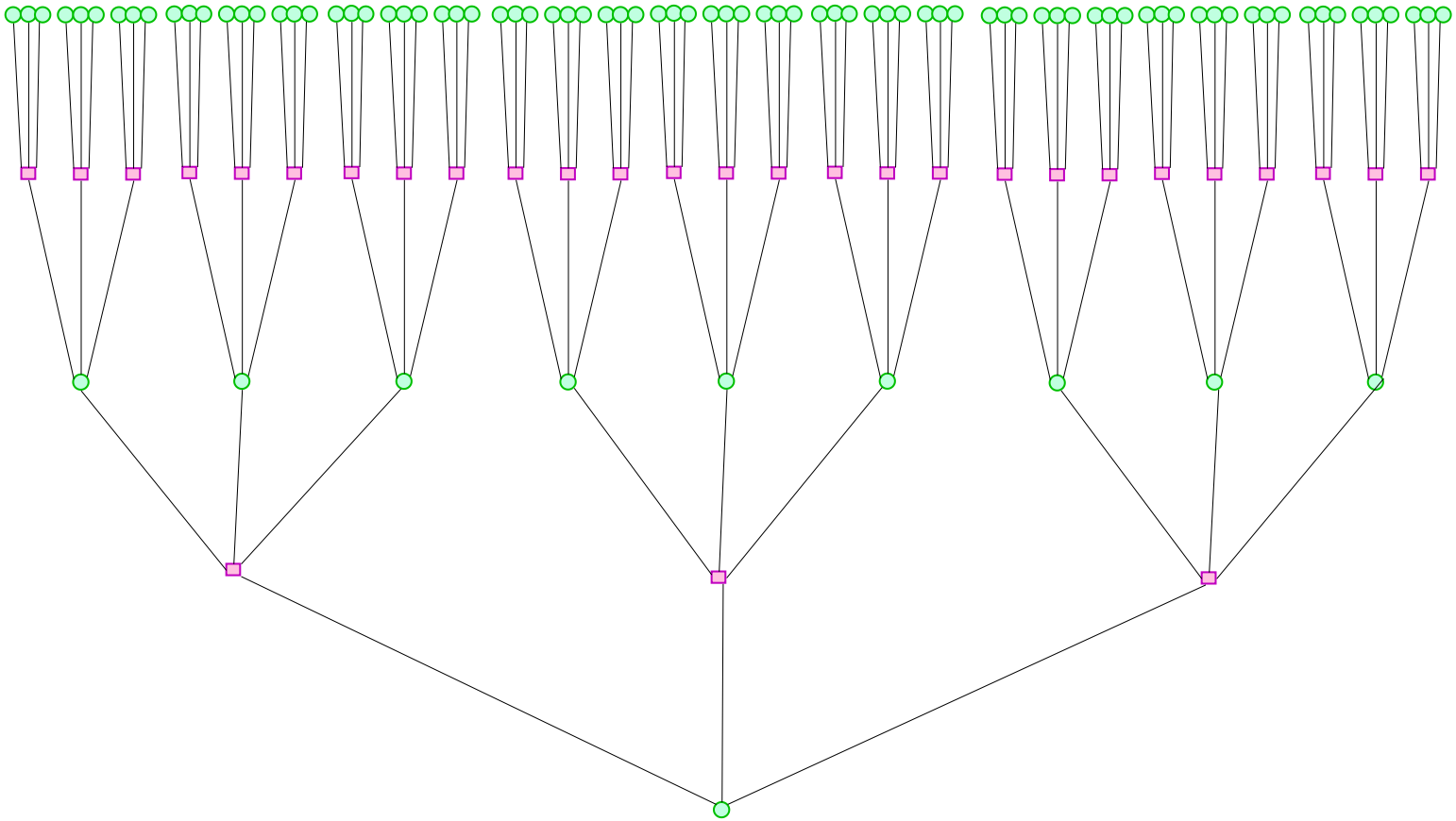
LDPC码解码

每个符号的检测解码：

- 每个符号参加了j个校验方程
- 每个校验方程有k个符号参与校验
- 采用软判决迭代解码
  - 每个符号的置信度 = j个修正校验和的置信度 + 接收信号先验信息置信度（和算法）
  - 每个修正校验和的置信度来自于其余k-1个符号的置信度（积算法）

LDPC码解码

- LDPC码的分层结构图



LDPC码解码

- 符号置信度等于各个修正校验和置信度之和

$$LLR(u_i) = \sum_{n=1}^{j} LLR(s'_n) + \log \frac{P(u_i = +1)}{P(u_i = -1)}$$

各个修正校验和的置信度来自于其余参与校验的符号

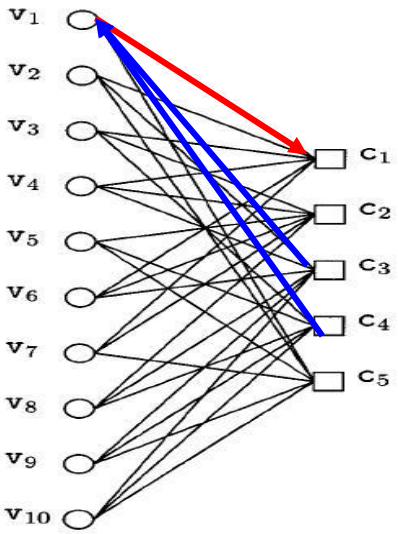
$$\tanh(LLR(s_n)/2) = \prod_{i=1}^{k} \tanh(LLR(u_i)/2)$$

LDPC码解码

- 修正校验和涉及到的其余符号的置信度，来自于它们各自参加的其余修正校验和置信度
- 这些其余的修正校验和置信度，又来自于参与这些校验的符号置信度
- ……

- 单个符号逐一进行检测解码效率较低，可以**并行处理**！

LDPC码解码

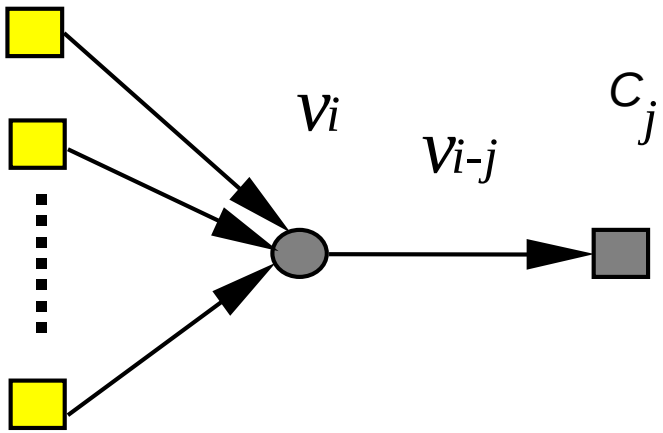


和积算法的置信度传播

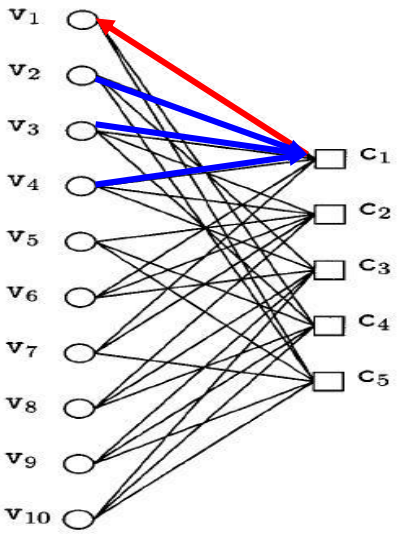
- 设校验节点j与变量节点i连接
- 变量节点i到校验节点j（Sum）
  - 求与变量节点i相连接的所有校验节点传递给i的置信度信息之和
  - 减去校验节点j传来的置信度信息
  - 传递给校验节点j

和积算法的置信度传播
变量节点i到校验节点j（Sum）

$$v_{i\to j} = u_{0\to i} + \sum_{k=1,k\neq j}^{d_i^v} c_{k\to i}$$



LDPC码解码

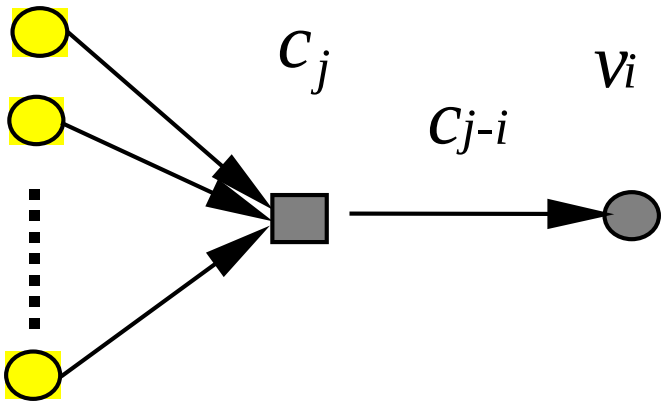


和积算法的置信度传播

- 校验节点j到变量节点i（Product）
  - 求与校验节点j相连接的所有变量节点传递给j的置信度信息双曲正切之积
  - 除以变量节点i传来的置信度信息的双曲正切
  - 对双曲正切之积求artanh
  - 传递给变量节点i

和积算法的置信度传播
校验节点j到变量节点i（Product）

$$c_{j\to i} = 2ar\tanh\left[\prod_{k=1,k\neq i}^{d_j^c}\tanh\left(\frac{v_{k\to j}}{2}\right)\right]$$

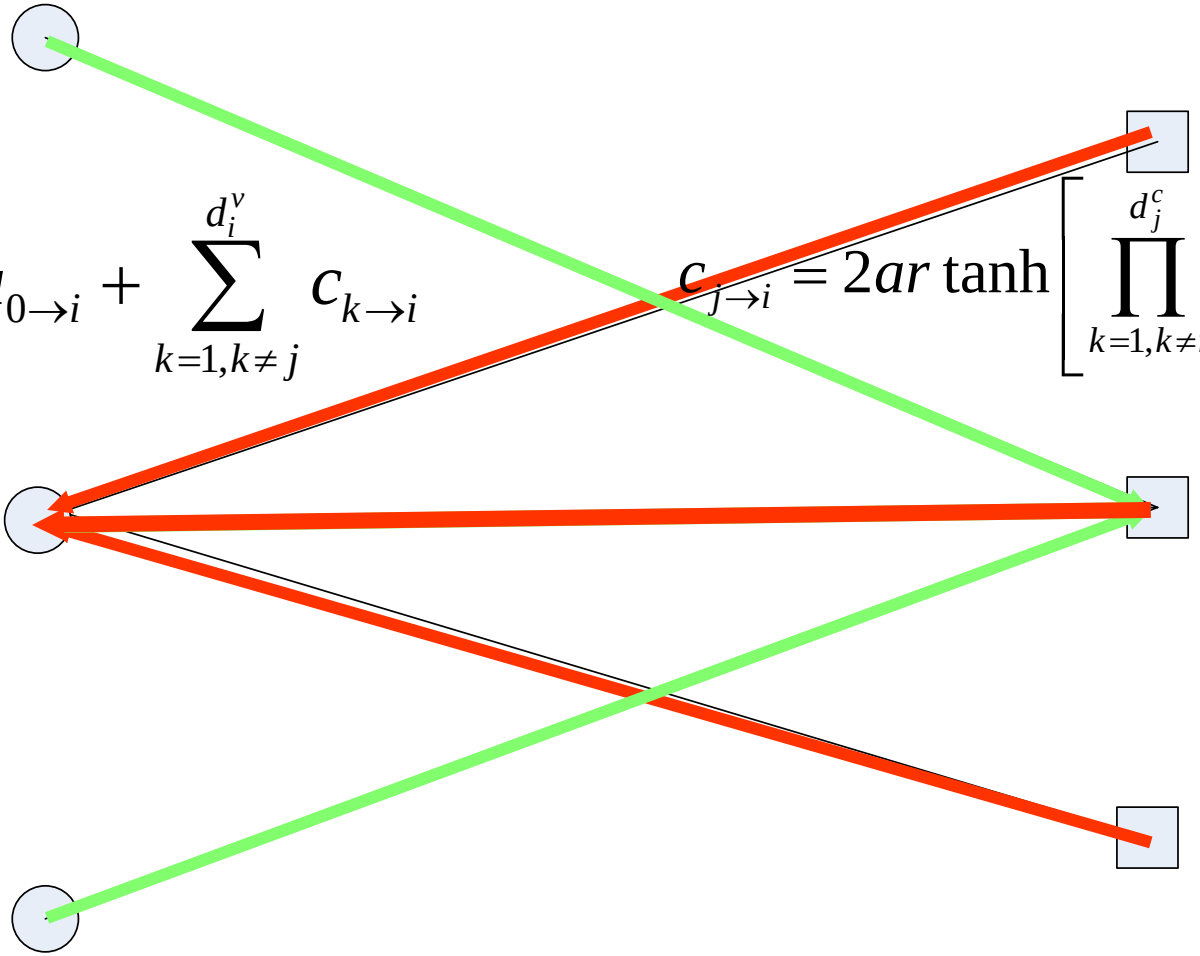


LDPC码解码

和积算法的置信度传播

$$v_{i\to j} = u_{0\to i} + \sum_{k=1,k\neq j}^{d_i^v} c_{k\to i}$$

$$c_{j\to i} = 2ar\tanh\left[\prod_{k=1,k\neq i}^{d_j^c} \tanh\left(\frac{v_{k\to j}}{2}\right)\right]$$



Vi

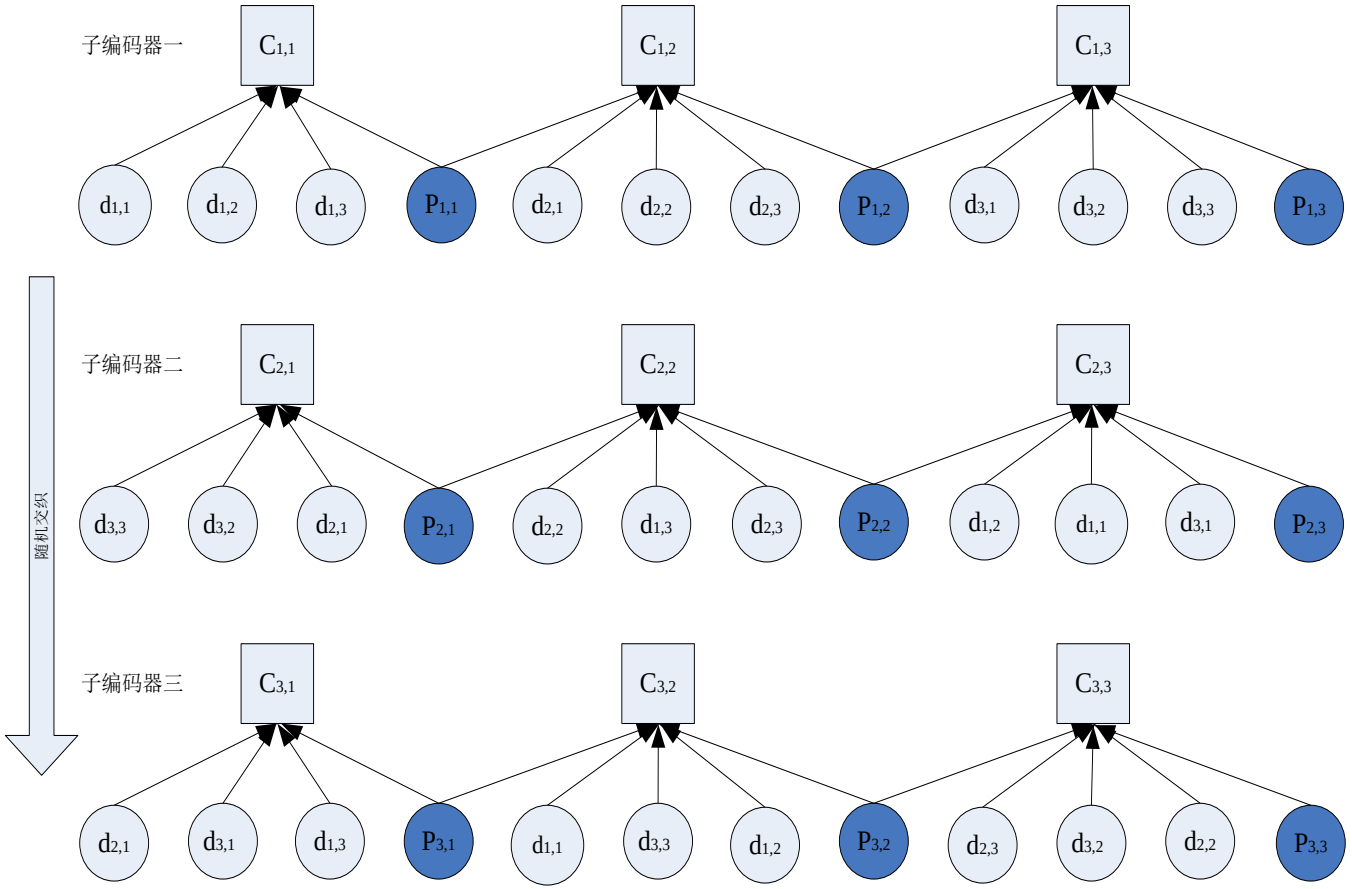
Cj

LDPC码解码

和积算法（Sum-Product ）

- 每次迭代结束之后，需要进行判决
  - 求与变量节点i相连接的所有校验节点传递给i的置信度信息之和，判决
  - 满足校验方程约束，则结束译码
  - 不满足校验方程约束，则继续迭代

LDPC码解码----和积算法举例



LDPC码解码----和积算法举例

子编码器	信息比特顺序								
子编码器一	1	2	3	4	5	6	7	8	9
子编码器二	9	8	4	5	3	6	2	1	7
子编码器三	4	7	3	1	9	2	6	8	5

LDPC码解码----和积算法举例

- 输入信息比特[101011001]
  - 系统码输出为 $[DP_1P_2P_3]$ 形式

    [101011001001101111]
  - 在信道中传输时采用BPSK调制，则编码输出

$$[-1,1,-1,1,-1,-1,1,1,-1,1,1,-1,-1,1,-1,-1,-1,-1]$$

LDPC码解码----和积算法举例

- 在AWGN信道中
  - $L_c \cdot y = \log \frac{p(y \mid x=+1)}{p(y \mid x=-1)} = 4 \cdot \frac{E_s}{N_0} \cdot y$
  - 当$SNR = 10\lg \frac{E_s}{N_0} = 0dB$时，$\frac{E_s}{N_0} = 1$
  - 此时 $L_c \cdot y = 4y$

LDPC码解码----和积算法举例

假设接收序列为：

$$\left.\begin{array}{l}-2.4448,1.5867,0.02484,\\ -0.11428,-2.5639,-2.3157,\\ 1.6011,-0.47525,-0.39747,\end{array}\right\} D$$

$$1.4183,2.3154,0.43241,\} P_1$$

$$-1.1902,2.1111,-0.81348,\} P_2$$

$$-2.0499,-0.83743,-1.1697\} P_3$$

LDPC码解码----和积算法举例

- 变量节点传递给校验节点置信度信息为外信息与信道信息之和

图1-1：由于外信息为零，子解码器一的第一次迭代中传递的信息

$$L_c \cdot y = 4y$$

图1-2：子解码器一中的校验节点传递给变量节点的置信度信息为

$$c_{j\to i} = 2ar\tanh\left(\prod_{k=1,k\neq i}^{d_j^c}\tanh\left(\frac{v_{k\to j}}{2}\right)\right)$$

LDPC码解码----和积算法举例

图1-1

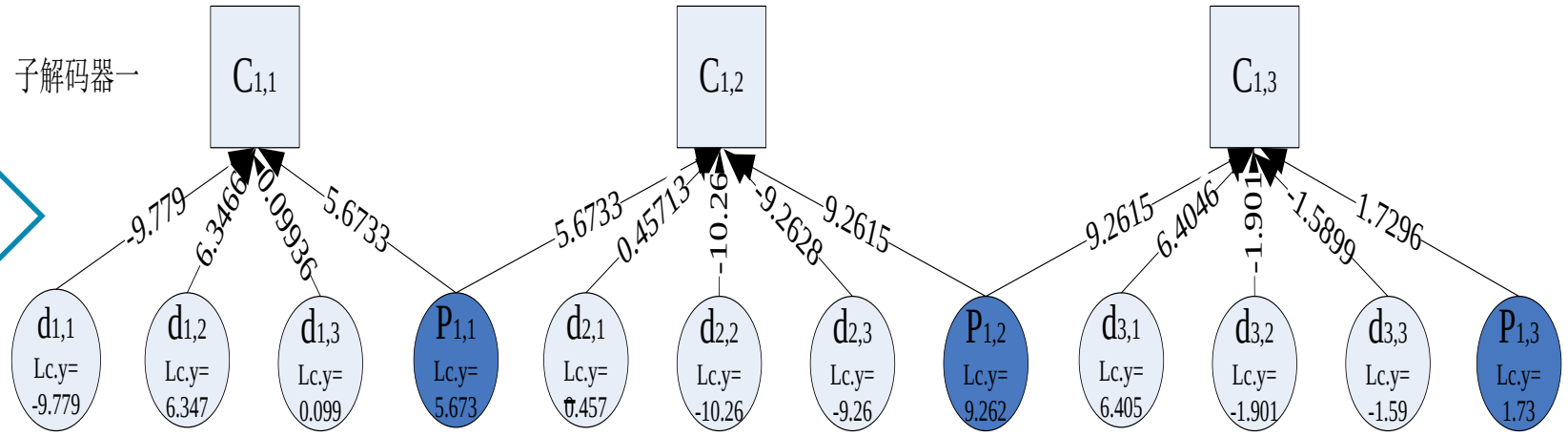
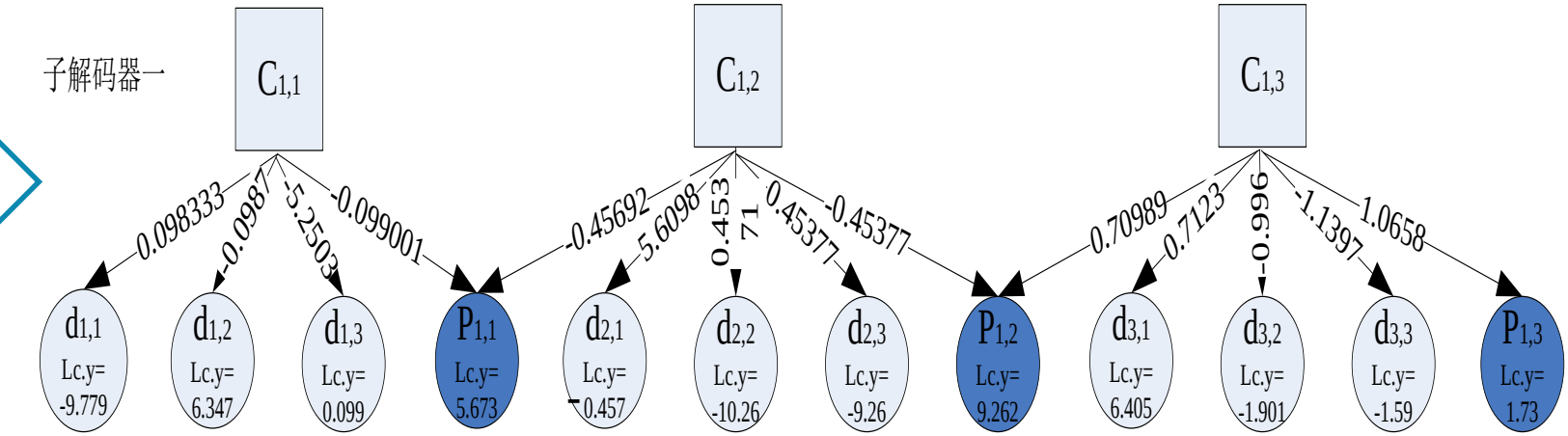


图1-2



LDPC码解码----和积算法举例

- 子解码器一的译码信息作为子解码器二的外信息输入

图2-1：子解码器二中变量节点传递给校验节点的置信度

图2-2：子解码器二中的校验节点传递给变量节点的置信度信息为

$$c_{j\to i} = 2ar\tanh\left(\prod_{k=1,k\neq i}^{d_j^c}\tanh\left(\frac{v_{k\to j}}{2}\right)\right)$$

LDPC码解码----和积算法举例

图2-1

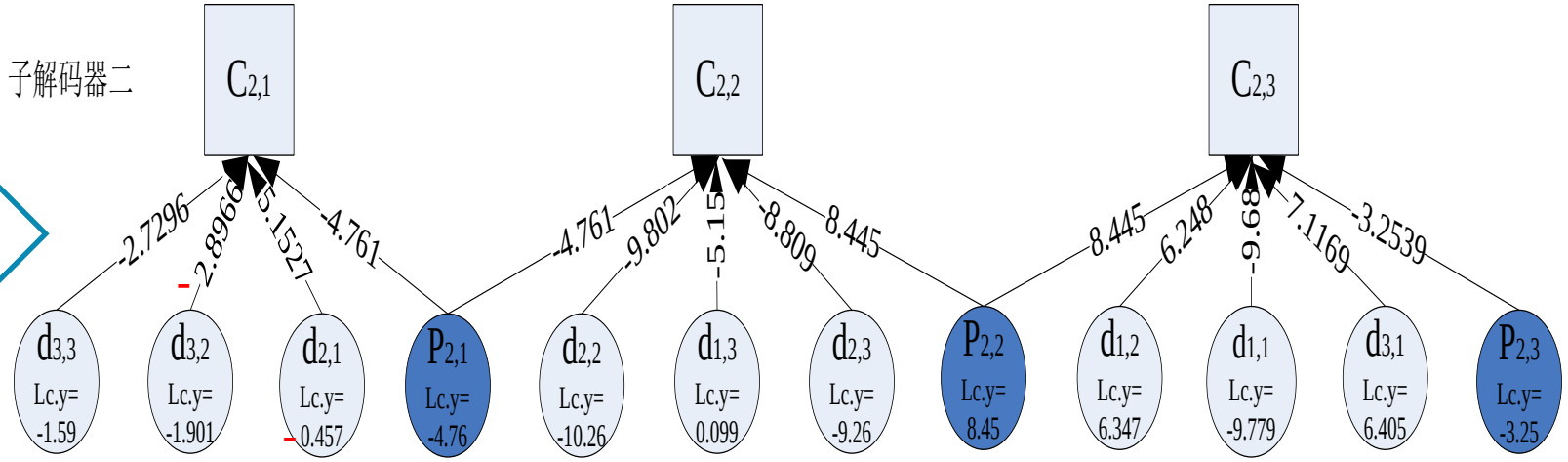
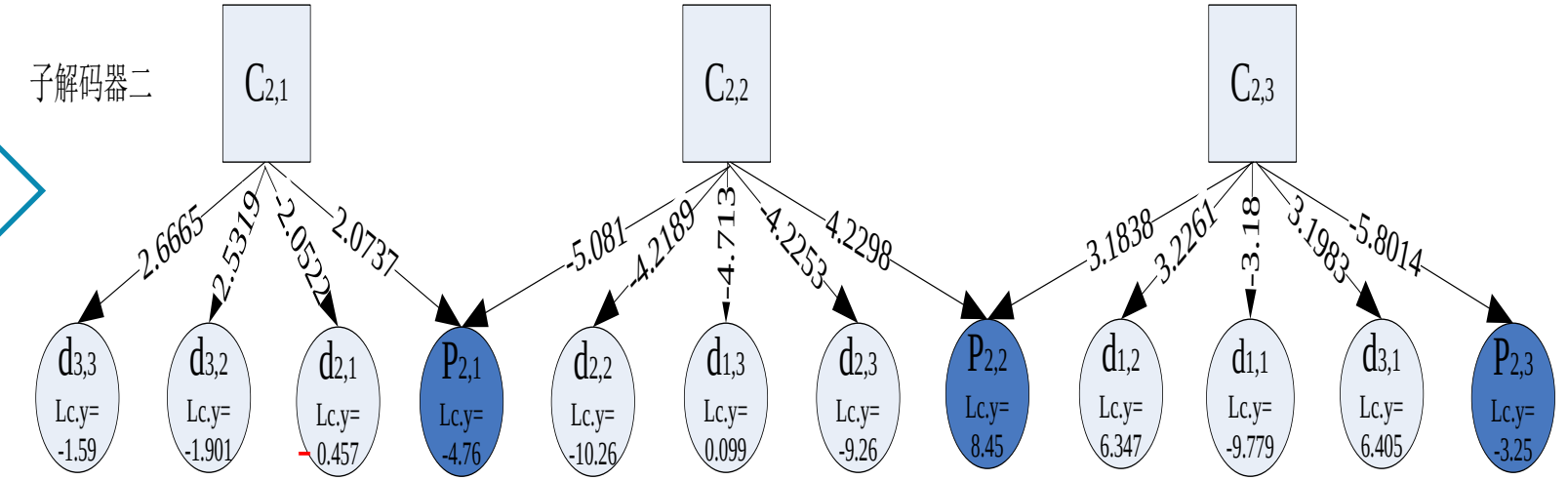


图2-2



LDPC码解码----和积算法举例

- 子解码器二的译码信息作为子解码器三的外信息输入

图3-1：子解码器三中变量节点传递给校验节点的置信度

图3-2：子解码器三中的校验节点传递给变量节点的置信度信息为

$$c_{j\to i} = 2ar\tanh\left(\prod_{k=1,k\neq i}^{d_j^c}\tanh\left(\frac{v_{k\to j}}{2}\right)\right)$$

LDPC码解码----和积算法举例

图3-1

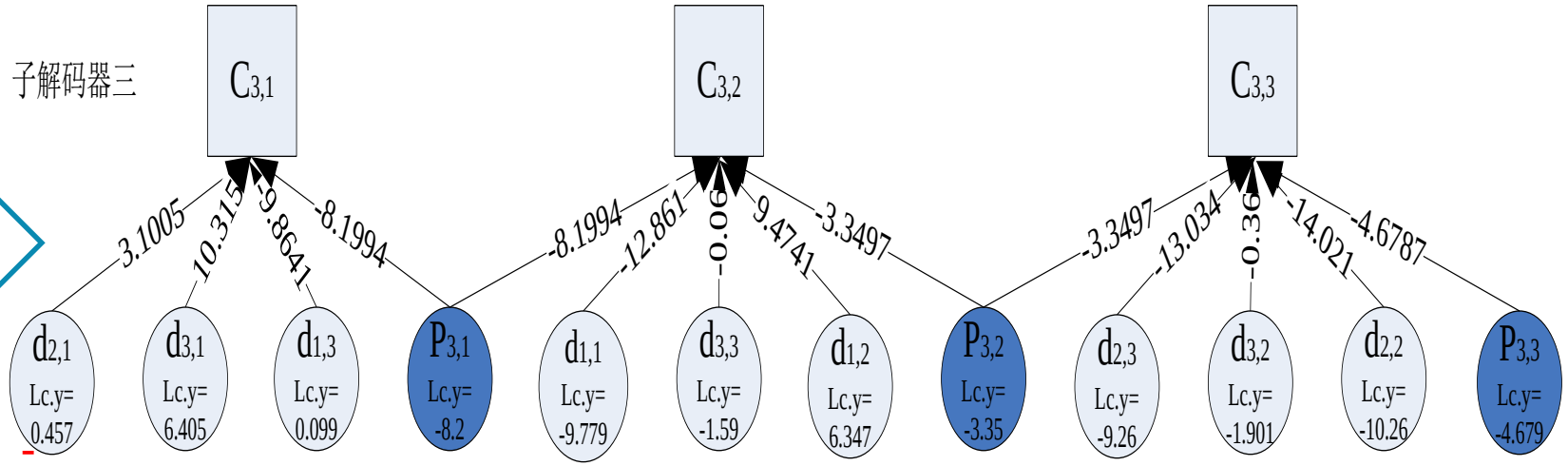
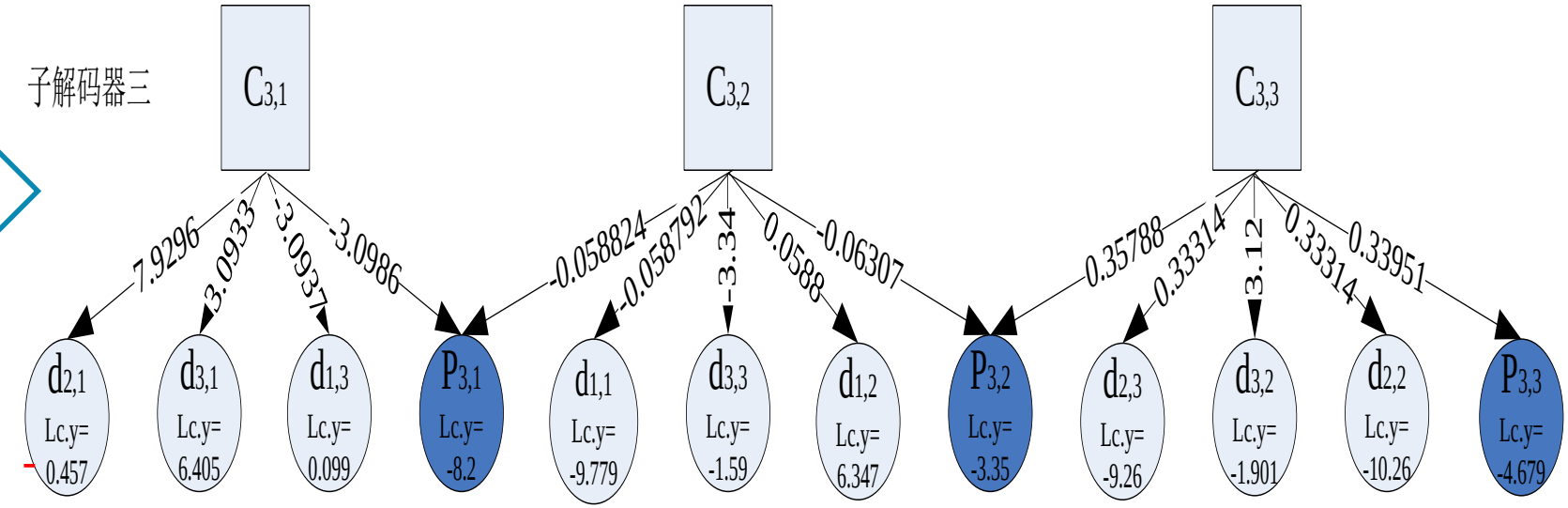


图3-2



LDPC码解码----和积算法举例

- 完成所有子解码器解码后进行判决
  - 判决信息为所有子解码器得到的解码信息与信道信息之和

LDPC码解码----和积算法举例

	$L_c \cdot y_i$	$\wedge_{1e}(d_i)$	$\wedge_{2e}(d_i)$	$\wedge_{3e}(d_i)$	判决信息	判决
$d_{1,1}$	-9.779	0.098333	-3.1802	-0.05879	-12.92	-1
$d_{1,2}$	6.3466	-0.09867	3.2261	0.0588	9.5329	1
$d_{1,3}$	0.099361	-5.2503	-4.7131	-3.0937	-12.958	-1
$d_{2,1}$	-0.45713	5.6098	-2.0522	7.9296	11.03	1
$d_{2,2}$	-10.256	0.45371	-4.2189	0.33314	-13.688	-1
$d_{2,3}$	-9.2628	0.45377	-4.2253	0.33314	-12.701	-1
$d_{3,1}$	6.4046	0.7123	3.1983	3.0933	13.408	1
$d_{3,2}$	-1.901	-0.9956	2.5319	3.1151	2.7504	1
$d_{3,3}$	-1.5899	-1.1397	2.6665	-3.3397	-3.4028	-1

续上表

$P_{1,1}$	5.6733	-0.55592	0	0	5.1773	1
$P_{1,2}$	9.2615	0.25612	0	0	9.5176	1
$P_{1,3}$	1.7296	1.0658	0	0	2.7954	1
$P_{2,1}$	-4.761	0	-3.0073	0	-7.7683	-1
$P_{2,2}$	8.4445	0	7.4137	0	15.858	1
$P_{2,3}$	-3.2539	0	-5.8014	0	-9.0553	-1
$P_{3,1}$	-8.1994	0	0	-3.1574	-11.357	-1
$P_{3,2}$	-3.3497	0	0	0.29481	-3.0549	-1
$P_{3,3}$	-4.6787	0	0	0.33951	-4.3392	-1

LDPC码解码----和积算法举例

- 将上述判决结果映射回‘0’，’1’序列
- 将译码判决输出序列代入校验方程组

$$
\begin{cases}
(\text{方程}C_{1,1})\ d_{1,1}+d_{1,2}+d_{1,3}+P_{1,1}=0\\
(\text{方程}C_{1,2})\ d_{2,1}+d_{2,2}+d_{2,3}+P_{1,2}+P_{1,1}=0\\
(\text{方程}C_{1,3})\ d_{3,1}+d_{3,2}+d_{3,3}+P_{1,3}+P_{1,2}=0\\
(\text{方程}C_{2,1})\ d_{3,3}+d_{3,2}+d_{2,1}+P_{2,1}=0\\
(\text{方程}C_{2,2})\ d_{2,2}+d_{1,3}+d_{2,3}+P_{2,2}+P_{2,1}=0\\
(\text{方程}C_{2,3})\ d_{1,2}+d_{1,1}+d_{3,1}+P_{2,2}+P_{2,3}=0\\
(\text{方程}C_{3,1})\ d_{2,1}+d_{3,1}+d_{1,3}+P_{3,1}=0\\
(\text{方程}C_{3,2})\ d_{1,1}+d_{3,3}+d_{1,2}+P_{3,2}+P_{3,1}=0\\
(\text{方程}C_{3,3})\ d_{2,3}+d_{3,2}+d_{2,2}+P_{3,2}+P_{3,3}=0
\end{cases}
$$

LDPC码解码----和积算法举例

- 符合所有方程，则没有误码，译码结束
- 不符合所有方程，有误码，迭代译码:
  - 子解码器二和三的译码信息作为子解码器一的外信息
  - 校验位P的外信息为零

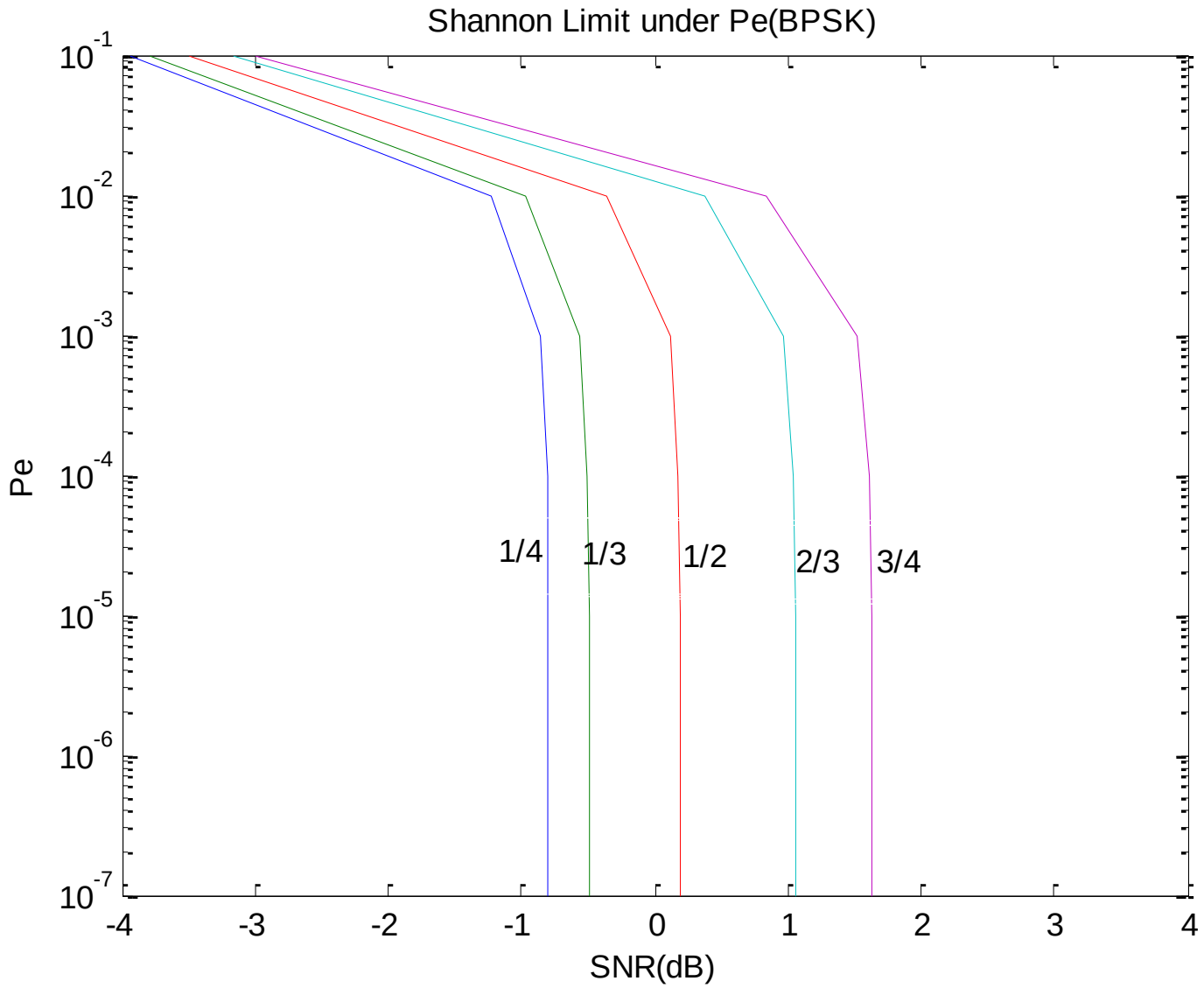
LDPC码解码

和积算法解码分析

- LDPC码存在信道阈值现象
  - 当信道条件好于阈值时，随着解码迭代次数的增加，误码率可以趋近于零
  - 当信道条件差于阈值时，随着解码迭代次数的增加，误码率趋于一个大于零的值

目录

- LDPC码描述
- LDPC码构造方法
- 和积算法
- LDPC码解码
- 性能分析


Shannon Limit under Pe(BPSK)
Pe
SNR(dB)
1/4
1/3
1/2
2/3
3/4

Source： Richardson, T.J.; Shokrollahi, M.A.; Urbanke, R.L.; Design of capacity-approaching irregular low-density parity-check codes. Information Theory, IEEE Transactions on, Volume: 47 , Issue: 2， 2001

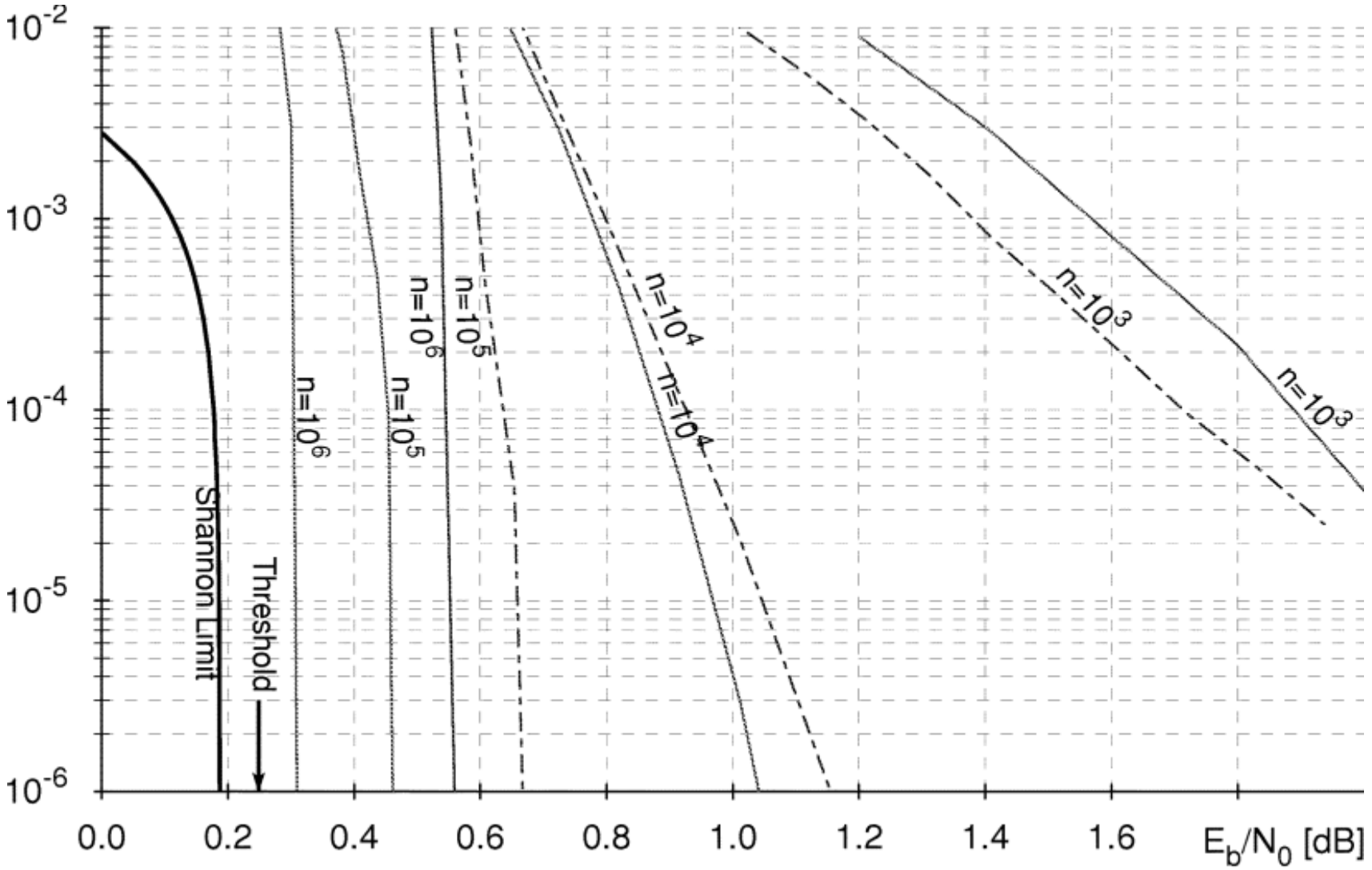


Fig. 3. Comparison between bit-error rates achieved by turbo codes (dashed curves) and LDPC codes (solid curves) . All codes are of rate one-half. Observe that longer LDPC codes outperform turbo codes and that the gap becomes the more significant the larger n is chosen. For short lengths, it appears that the structure of turbo codes gives them an edge over LDPC codes despite having a lower threshold.

DVBSII 标准

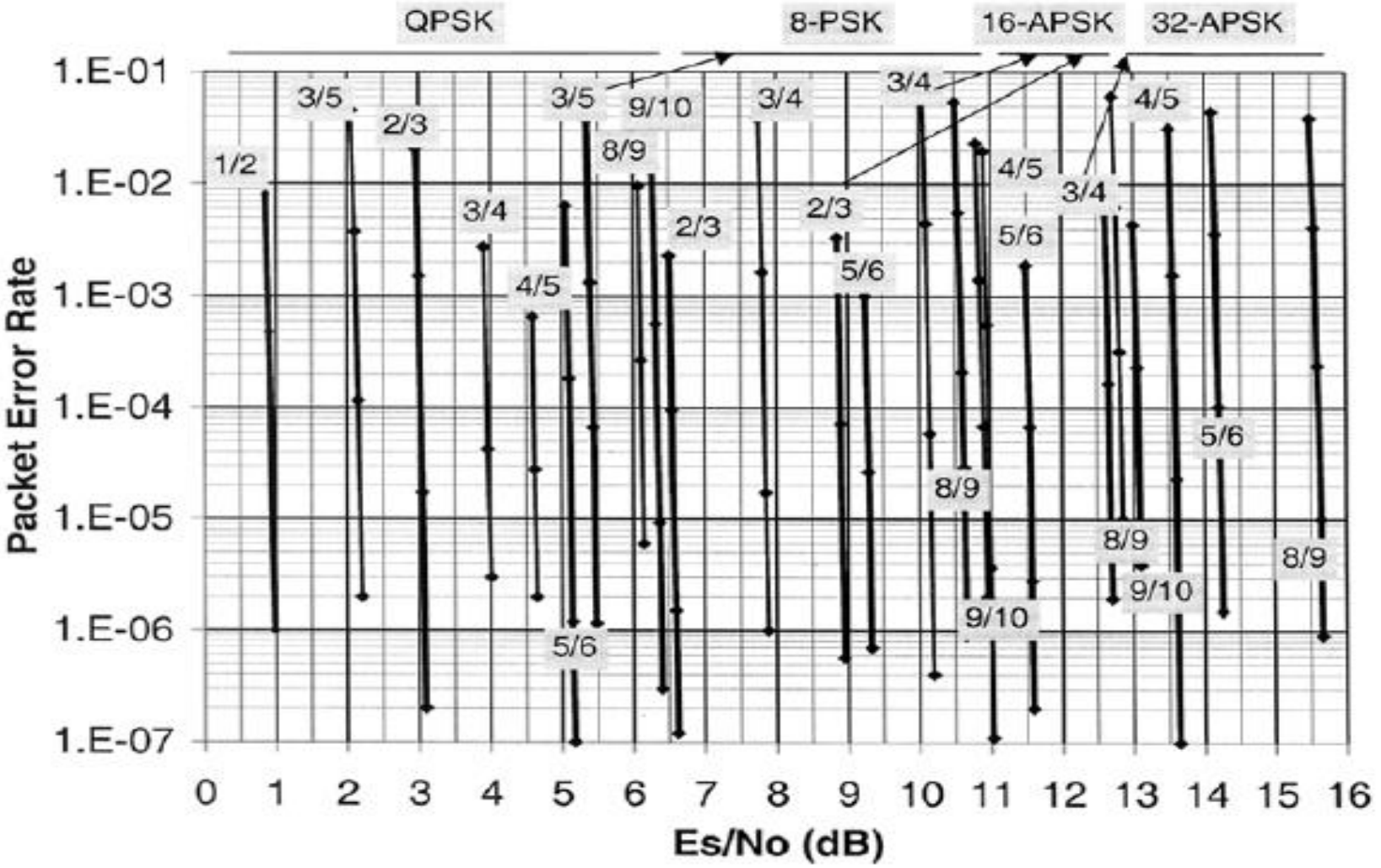
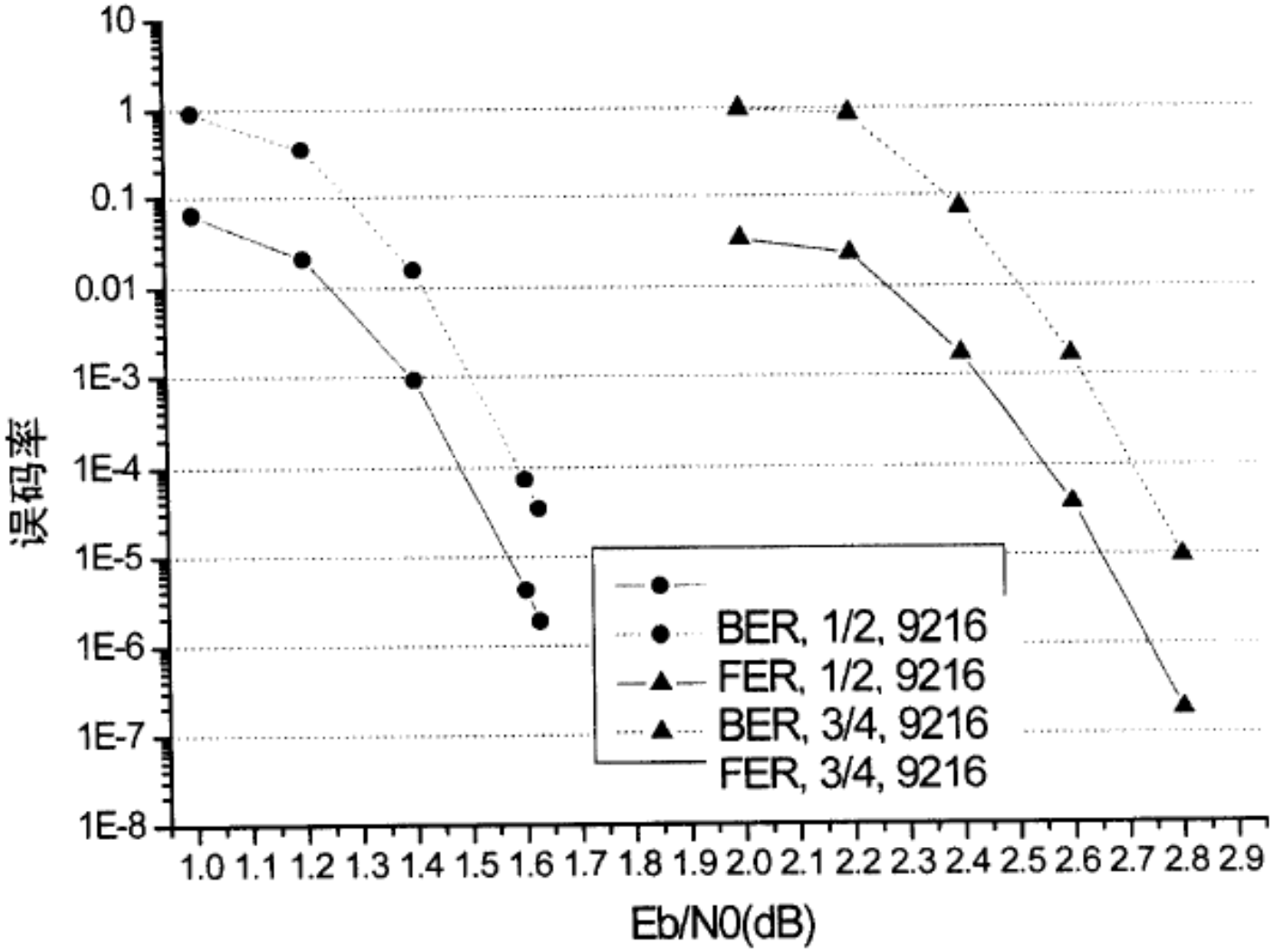


Fig. 3. Performance of LDPC codes over AWGN channel ($N = 64800$).

CMMB标准 (9216, R=0.5, 0.75)



IEEE 802.11N

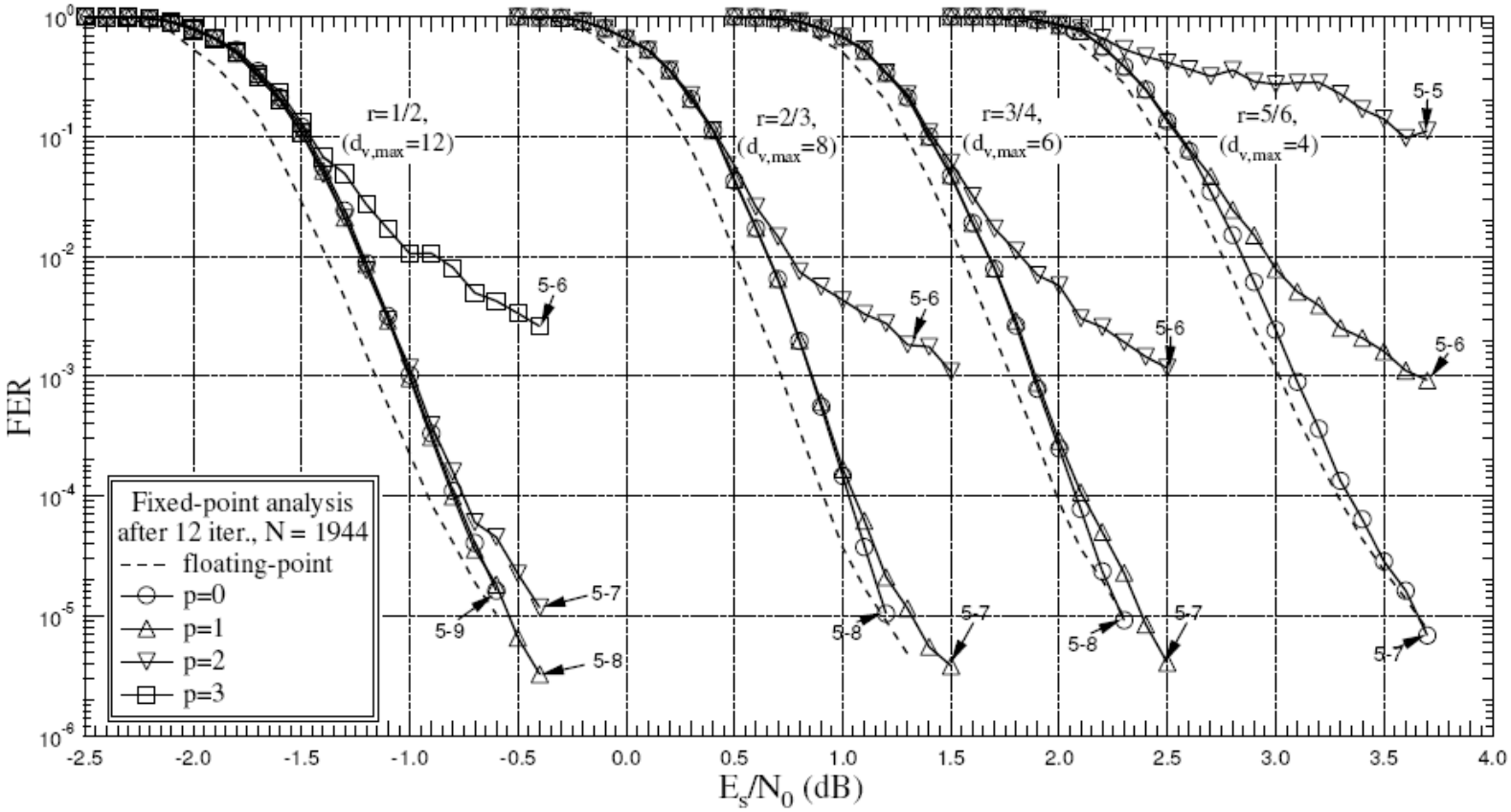
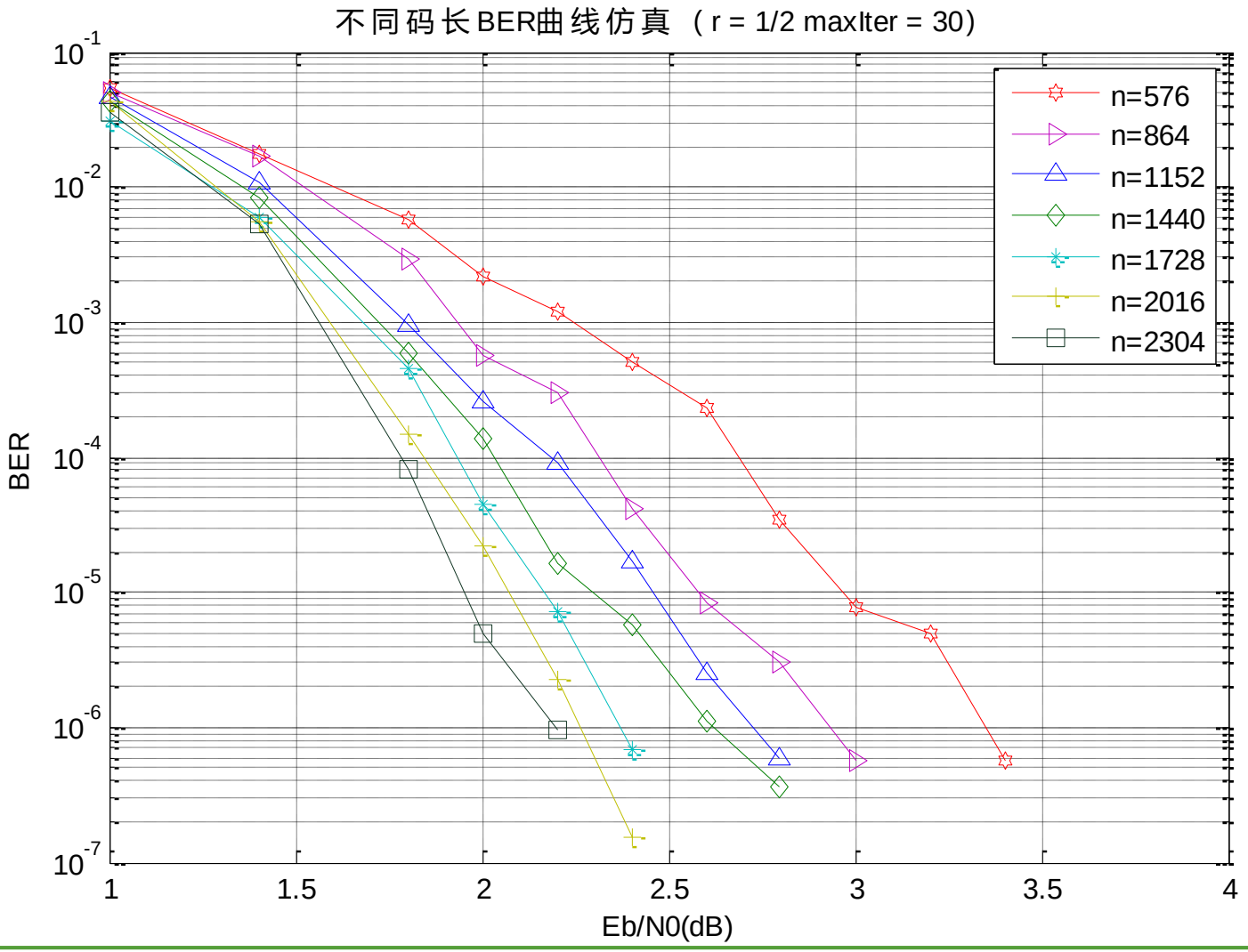
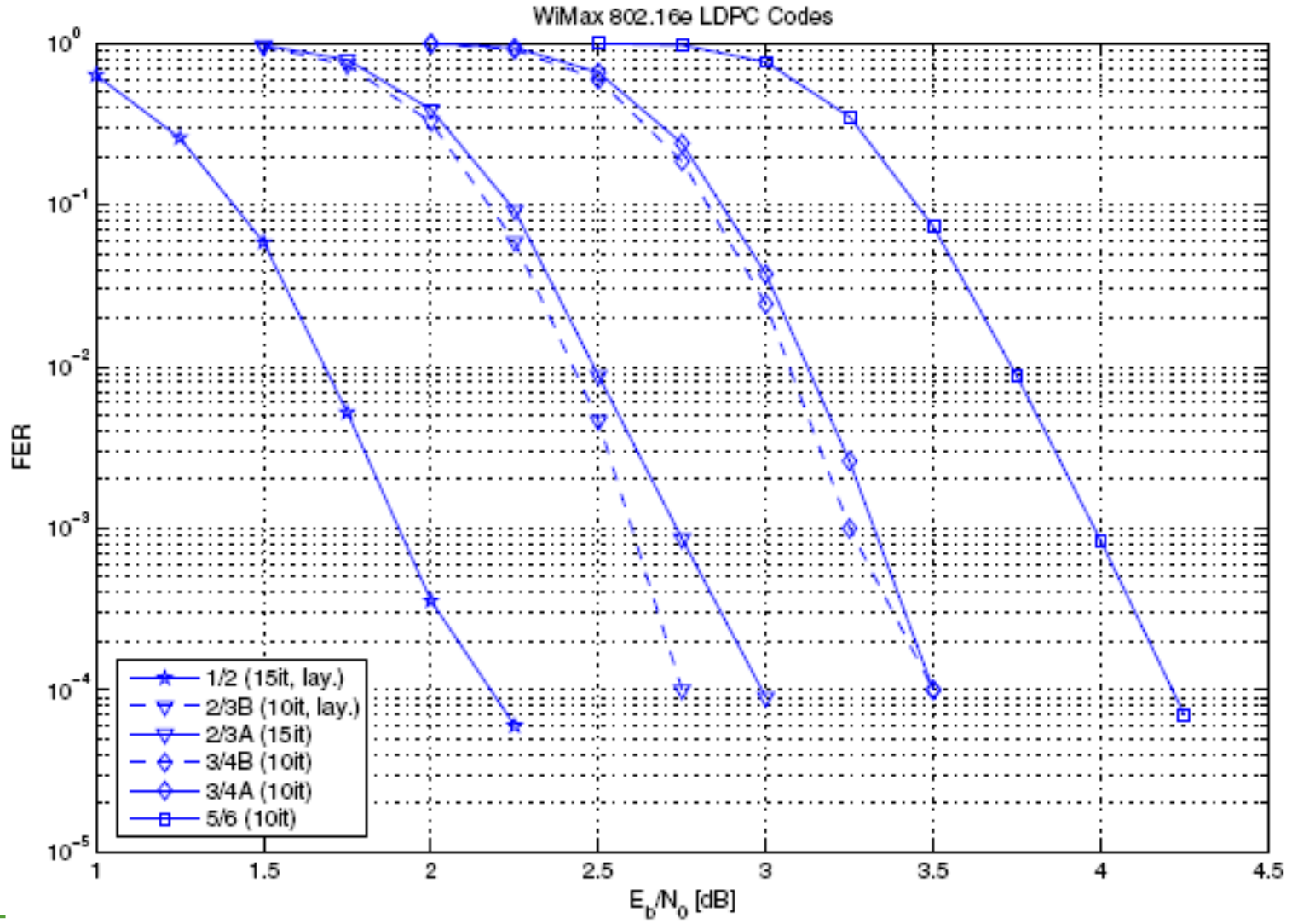


Fig. 7. FER performance of IEEE 802.11n LDPC codes with $N = 1944$ and $c2v$ messages on 5 bits.

IEEE 802.16E (码率1/2)



IEEE 802.16E (码长2304)



~ END ~