基于改进罚函数的 LDPC 码分层调度 ADMM 惩罚译码

王 彪[1,2],慕建君[1],焦晓鹏[1],王钟斐[2]
(1. 西安电子科技大学计算机科学与技术学院,陕西西安 710071;
2. 宝鸡文理学院数学与信息科学学院,陕西宝鸡 721013)

摘 要: 通过增加伪码字的代价,基于交替方向乘子法(Alternating Direction Method of Multipliers,ADMM)的惩罚译码方法可以改善低密度奇偶校验(Low-Density Parity-Check,LDPC)码低信噪比区域的译码性能,同时具有低的译码复杂度.而减少 ADMM 惩罚译码的欧几里德投影次数、选择合适的消息调度策略和设计有效的罚函数是提高 ADMM 惩罚译码速度的三种重要方法.为了进一步提高 ADMM 惩罚译码速度,通过利用 Wei 等人提出的方法来减少欧几里德投影的次数,本文设计了基于 I-l_1-PF 罚函数的水平分层调度与垂直分层调度策略的两种 LDPC 码 ADMM 惩罚译码方法.仿真实验表明,与现有 ADMM 惩罚译码方法相比较,所设计的译码方法不仅具有较好的译码性能,而且能够显著降低 LDPC 码译码的平均迭代次数和平均译码时间.

关键词: 低密度奇偶校验码;交替方向乘子法;罚函数;惩罚译码;分层调度
中图分类号: TN911.22 **文献标识码:** A **文章编号:** 0372-2112 (2020)04-0827-06
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2020.04.025

ADMM Penalized Decoding with Layered Scheduling for LDPC Codes Based on Improved Penalty Function

WANG Biao[1,2],MU Jian-jun[1],JIAO Xiao-peng[1],WANG Zhong-fei[2]
(1. *School of Computer Science and Technology,Xidian University,Xi'an,Shaanxi* 710071,*China*;
2. *School of Mathematics and Information Science,Baoji University of Arts and Sciences,Baoji,Shaanxi* 721013,*China*)

Abstract: By making the pseudocodewords more costly,the penalized decoding method based on alternating direction method of multipliers (ADMM) can improve the decoding performance for low-density parity-check (LDPC) codes at low signal-to-noise ratios and also has low decoding complexity. Reducing the number of Euclidean projection in ADMM penalized decoding,selecting the appropriate message scheduling strategy and designing effective penalty function are three important methods to increase the ADMM penalized decoding speed. In order to increase the ADMM penalized decoding speed further,by using the method proposed by Wei et al to reduce the number of Euclidean projections,this paper designs two kinds of ADMM penalized decoding methods with the horizontal layered scheduling and the vertical layered scheduling strategy for LDPC codes based on the I-l_1-PF penalty function. Simulation results show that the designed methods not only have better decoding performance but also significantly reduce the average number of iterations and the average decoding time compared with the existing ADMM penalized decoding methods.

Key words: low-density parity-check (LDPC) codes;alternating direction method of multipliers (ADMM);penalty function;penalized decoding;layered scheduling

1 引言

目前,低密度奇偶校验(Low-Density Parity-Check,LDPC)码迭代译码的研究受到国内外许多学者的普遍关注[1~7].基于交替方向乘子法(Alternating Direction Method of Multipliers,ADMM)的 LDPC 码线性规划(Linear Programming,LP)译码方法可以降低 LP 译码方法的译码复杂度,但其低信噪比区域的译码性能弱于置信

收稿日期:2017-02-15;修回日期:2019-08-28;责任编辑:孙瑶
基金项目:国家自然科学基金(No. 61971322,No. 61977051,No. 61602010);重庆市/信息产业部计算机网络与通信技术重点实验室开放基金(No. CY-CNCL-2017-03);陕西省自然科学基础研究计划(No. 2014JM1027);陕西省教育厅专项科研计划(No. 17JK0047);宝鸡市科技计划(No. 15RKX-1-5-8,2018JH-18);宝鸡文理学院校级重点项目(No. ZK12094,No. ZK2017001)

传播(Belief Propagation,BP)译码方法[5].近年来,针对基于 ADMM 的 LDPC 码 LP 译码方法的研究主要集中在降低译码复杂度来提高译码速度[6~13]和改善 LDPC 码低信噪比区域的译码性能[14~16]两个方面.

提高 LDPC 码 ADMM 译码速度的研究主要有两类方法.

(1)降低 ADMM 译码中欧几里德投影(Euclidean Projection)运算的计算复杂度,因为欧几里德投影是 LDPC 码 ADMM 译码中最复杂最耗时的运算.通过引入"割查找(cut search)"算法来确定投影点所在的面,Zhang 等人提出了欧几里德投影的一种有效计算方法[6];而将迭代解向量投影到单纯形也是降低欧几里德投影计算复杂度的一种重要方法[7].当然,直接减少欧几里德投影次数(而不是降低欧几里德投影的计算复杂度)也能够降低 ADMM 译码复杂度[8].最近,Jiao 等人提出了一种基于查表法的方法来实现欧几里德投影,降低了 LDPC 码的 ADMM 译码的复杂度[9].

(2)设计有效的消息调度策略.目前,大部分 ADMM 译码算法采用的是 Flooding 消息调度策略[5~8].为了提高基于 ADMM 的 LP 译码收敛速度,Debbabi 等人设计了水平分层(Horizontal Layered,HL)[10]和垂直分层(Vertical Layered,VL)[11]两种消息分层调度 ADMM 译码方法;同时,Jiao 等人设计了逐点(Node-Wise)消息调度的 ADMM 译码算法也可以提高译码速度[12].此外,超松弛(Over-Relaxation)ADMM 和早停止(Early Termination)技术也是提高 LDPC 码 ADMM 译码速度的重要实现方法[13].

为了改善 ADMM 译码时 LDPC 码低信噪比区域译码性能,Liu 等人通过引入罚函数提出了一种改进的 ADMM 译码方法(ADMM 惩罚译码方法)[14].该方法主要通过增加伪码字的代价(即惩罚伪码字)来实现迭代译码,改善了 LDPC 码低信噪比区域的译码性能.基于加权的惩罚因子,Jiao 等人设计了一种改进的 ADMM 惩罚译码方法,该方法显著改善了 LDPC 码的译码性能[15].通过提高罚函数在 0 和 1 两点附近的斜率,Wang 等人设计了两种改进罚函数(I-l_1-PF 和 I-l_1-PF),基于这两种改进罚函数的 ADMM 惩罚译码不但改善了原有 ADMM 惩罚译码方法的译码性能而且提高了译码速度[16].

目前,减少欧几里德投影次数、选择合适的消息分层调度策略和设计改进的罚函数是三种分别从不同角度提高 ADMM 惩罚译码速度的方法.为了进一步提高 ADMM 惩罚译码速度,通过利用 Wei 等人提出的方法来减少欧几里德投影的次数,本文设计了两种基于 I-l_1-PF 罚函数的水平分层调度与垂直分层调度策略的 ADMM 惩罚译码方法.仿真结果表明本文的两种译码方法具有较好的译码性能,并且显著提高了译码速度.

2 基于 ADMM 的 LDPC 码惩罚译码及其消息调度策略

设 LDPC 码 C 的码长为 n,其校验矩阵为 m 行 n 列矩阵 $\boldsymbol{H}$,集合 $I=\{1,2,\cdots,n\}$ 表示 $\boldsymbol{H}$ 中所有变量节点下标的集合,$J=\{1,2,\cdots,m\}$ 表示 $\boldsymbol{H}$ 中所有校验节点下标的集合.$\boldsymbol{H}$ 中元素 $H_{ji}=1$ 表示校验节点 c_j 与变量节点 v_i 相关联.集合 $N_v(i)$ 表示与变量节点 v_i 相关联的所有校验节点集合,$d_{v_i}=|N_v(i)|$ 表示 v_i 的度;$N_c(j)$ 表示与校验节点 c_j 相连的变量节点集合,$d_{c_j}=|N_c(j)|$ 表示校验节点 c_j 的度.

设经加性高斯白噪声(Additive White Gaussian Noise,AWGN)信道发送码字为 $\boldsymbol{x}=\{x_1,x_2,\cdots,x_n\}^{\mathrm{T}}\in C$,$x_i\in[0,1]$,$i\in I$,接收序列记为 $\boldsymbol{r}=\{r_1,r_2,\cdots,r_n\}^{\mathrm{T}}$,$\boldsymbol{r}$ 对应的对数似然比向量记为 $\boldsymbol{\gamma}=\{\gamma_1,\gamma_2,\cdots,\gamma_n\}^{\mathrm{T}}$,其中

$$\gamma_i=\log\left(\frac{\Pr(r_i|0)}{\Pr(r_i|1)}\right) \tag{1}$$

$\Pr(\cdot)$ 表示括号内代表的事件发生的概率.引入辅助变量 $\boldsymbol{z}_j\in\boldsymbol{R}^{d_{c_j}}$ $(j\in J)$ 可得到 LDPC 码 LP 译码数学模型为[5]

$$\begin{cases}\min & \boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{x}\\ \text{s.t.} & \boldsymbol{T}_j\boldsymbol{x}=\boldsymbol{z}_j,\boldsymbol{z}_j\in\boldsymbol{P}_{d_{c_j}},\ \forall j\in J\end{cases} \tag{2}$$

其中校验多胞体 $\boldsymbol{P}_{d_{c_j}}$ 表示所有长度为 d_{c_j} 且含有偶数个 1 的二进制向量构成的凸包[6],$d_{c_j}\times n$ 二进制转换矩阵 $\boldsymbol{T}_j$ 从 x 中选出第 j 个校验节点所关联的 d_{c_j} 个分量.

通过将罚函数加入到(2)的目标函数中,可以得到基于 ADMM 的惩罚译码方法的目标函数为

$$f(\boldsymbol{x})=\boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{x}+\alpha\sum_{i\in I}g(x_i) \tag{3}$$

其中 α 为罚参数,$g(x)$ 为罚函数.目前,常用的罚函数主要有 l_1 罚函数 $g_1(x)=-|x-0.5|$ 和 l_2 罚函数 $g_2(x)=-(x-0.5)^2$.

目前,LDPC 码 ADMM 惩罚译码的消息调度策略主要有三类,即 Flooding 消息调度策略[5~8]、水平分层调度策略[10]和垂直分层调度策略[11].由于水平(或垂直)分层调度策略每次迭代时,校验(或变量)节点的消息更新能够充分利用本轮迭代中已经更新过的相关校验和变量节点信息,这就可以加快收敛速度,从而节省译码时间.因此,水平(或垂直)分层调度策略比 Flooding 消息调度策略具有更低的译码平均迭代次数和平均译码时间.

3 基于 I-l_1-PF 罚函数的分层调度 ADMM 惩罚译码

3.1 设计动机

当 LDPC 码的 ADMM 惩罚译码方法迭代一定的次

数以后,Wei 等人发现当前次迭代和上一次迭代的欧几里德投影向量几乎相同,而不需要再继续进行耗时的欧几里德投影运算[8]. 因此,通过减少欧几里德投影的次数(而不是降低欧几里德投影的计算复杂度)也可以降低 LDPC 码的 ADMM 惩罚译码复杂度,从而提高译码速度. 而译码中消息调度策略的设计也是影响 ADMM 惩罚译码速度的重要因素,当前大多数 ADMM 译码器均采用 Flooding 消息调度策略[5~8],而 Debbabi 等人从消息调度方式入手,设计了水平分层调度和垂直分层调度的 ADMM 惩罚译码方法,从而提高了 ADMM 惩罚译码的速度[10,11]. 同时,Wang 等人发现 LDPC 码的 ADMM 惩罚译码方法中罚函数的斜率也是影响译码速度的一个原因. 于是,通过提高罚函数在 0 和 1 两点附近的斜率设计了两种改进型罚函数[16],从而提高了 ADMM 惩罚译码速度.

总之,降低欧几里德投影运算的次数、选择合适的消息调度策略以及线性规划模型的目标函数中设计有效的惩罚函数可以提高 LDPC 码的 ADMM 惩罚译码速度. 受到这些方法的启发,通过将这三种提高译码速度的方法结合起来,本文设计了 LDPC 码 ADMM 惩罚译码的两种新方法来进一步提高译码速度.

3.2　设计的 ADMM 惩罚译码方法

通过利用 Wei 等人提出的方法来减少欧几里德投影的次数[8],本文设计了基于改进型罚函数(即文献[16]中提出的 I-l_1-PF)的水平分层调度和垂直分层调度策略的两种 ADMM 惩罚译码方法. 而且在这两种惩罚译码方法的实现中都借助超松弛 ADMM 和基于 $\boldsymbol{H}\boldsymbol{x}^{\mathrm{T}}=0$ 的早停止技术来提高译码速度. 下面分别给出了基于 I-l_1-PF 罚函数的水平分层和垂直分层两种分层调度的 LDPC 码 ADMM 惩罚译码算法,分别如算法 1 和算法 2 所示.

设 $\boldsymbol{z}_j^k$ 表示第 k 次迭代的辅助变量,$\boldsymbol{y}_j^k$ 表示第 k 次迭代的拉格朗日乘子向量,$\boldsymbol{L}_{j\to i}^k$ 表示第 k 次迭代中与变量节点 v_i 相关联的所有校验节点和信道传递给 v_i 的信息,$\boldsymbol{t}_i$ 表示变量节点 v_i 所收到信息的中间变量,$\boldsymbol{w}_j^k$ 表示第 k 次迭代中投影前的向量,$\boldsymbol{L}_{i\to j}^k$ 表示第 k 次迭代中变量节点 v_i 发送给其相关联的所有校验节点的信息,$\hat{c}_i$ 为译出的第 i 位码字比特.

算法 1　基于 I-l_1-PF 罚函数的水平分层调度 ADMM 惩罚译码算法

1: $\forall j\in J, i\in N_c(j): z_j^0=0.5, \boldsymbol{y}_j^0=0, \boldsymbol{L}_{j\to i}^0=0.5$

2: $\forall i\in I, j\in N_v(i): \boldsymbol{t}_i=\sum_{j\in N_v(i)}(\boldsymbol{L}_{j\to i}^0)-\frac{\gamma_i}{\mu}$

3: for all $k=1\to$(iter_max) when 终止条件为假 do

4:　对于所有校验节点 $j\in J$

5:　　根据文献[16]中式(10)计算 $\boldsymbol{L}_{i\to j}^{k+1}$

6:　　计算 $\boldsymbol{w}_j^k=\rho\boldsymbol{L}_{i\to j}^k+(1-\rho)\boldsymbol{z}_j^{k-1}+\boldsymbol{y}_j^{k-1}$,
　　if $\exists\ |\boldsymbol{w}_j^k-\boldsymbol{w}_j^{k-1}|>\varepsilon$
　　then $\boldsymbol{z}_j^k=\prod_{P_{d_{c_j}}}(\boldsymbol{w}_j^k)$
　　else $\boldsymbol{w}_j^{k-1}=\boldsymbol{w}_j^k$

7:　　更新 $\boldsymbol{y}_j^k=\boldsymbol{y}_j^{k-1}+\rho\boldsymbol{L}_{i\to j}^k+(1-\rho)\boldsymbol{z}_j^{k-1}-\boldsymbol{z}_j^k$
　　和 $\boldsymbol{L}_{j\to i}^k=\boldsymbol{z}_j^k-\boldsymbol{y}_j^k$

8:　　计算 $\boldsymbol{t}_i=\boldsymbol{t}_i-\boldsymbol{L}_{j\to i}^{k-1}+\boldsymbol{L}_{j\to i}^k$

9:　　按照 $\hat{c}_i=[\sum_{j\in N_v(i)}(\boldsymbol{L}_{i\to j})]>0.5$ 进行硬判决得到解 x

10:　　if $\boldsymbol{H}\boldsymbol{x}^{\mathrm{T}}=0$ exit

11: end for

算法 2　基于 I-l_1-PF 罚函数的垂直分层调度 ADMM 惩罚译码算法

1: $\forall j\in J, i\in N_c(j): z_j^0=0.5, \boldsymbol{y}_j^0=0$

2: $\forall i\in I, j\in N_v(i): \boldsymbol{L}_{i\to j}^0=0.5, \boldsymbol{t}_i=\sum_{j\in N_v(i)}(\boldsymbol{L}_{j\to i}^0)-\frac{\gamma_i}{\mu}$

3: $\forall i\in I, j\in N_v(i): \boldsymbol{L}_{i\to j}^1=x_i$(按照文献[16]中式(10)计算)

4:　for all $k=1\to$(iter_max) when 终止条件为假 do

5:　　对于所有变量节点 $i\in I$

6:　　计算 $\boldsymbol{w}_j^k=\rho\boldsymbol{L}_{i\to j}^k+(1-\rho)\boldsymbol{z}_j^{k-1}+\boldsymbol{y}_j^{k-1}$,
　　if $\exists\ |\boldsymbol{w}_j^k-\boldsymbol{w}_j^{k-1}|>\varepsilon$
　　then $\boldsymbol{z}_j^k=\prod_{P_{d_{c_j}}}(\boldsymbol{w}_j^k)$
　　else $\boldsymbol{w}_j^{k-1}=\boldsymbol{w}_j^k$

7:　　更新 $\boldsymbol{y}_j^k=\boldsymbol{y}_j^{k-1}+\rho\boldsymbol{L}_{i\to j}^k+(1-\rho)\boldsymbol{z}_j^{k-1}-\boldsymbol{z}_j^k$
　　和 $\boldsymbol{t}_i=\boldsymbol{t}_i-\boldsymbol{w}_j^{k-1}+\boldsymbol{w}_j^k$

8:　　根据文献[16]中式(10)计算 $\boldsymbol{L}_{i\to j}^{k+1}$

9:　　按照 $\hat{c}_i=[\sum_{j\in N_v(i)}(\boldsymbol{L}_{i\to j})]>0.5$ 进行硬判决得到解 $\boldsymbol{x}$

10:　　if $\boldsymbol{H}\boldsymbol{x}^{\mathrm{T}}=0$ exit

11: end for

说明 1　在算法 1 中,第 1 ~ 2 行进行参数的初始化. 第 5 行利用文献[16]中 I-l_1-PF 罚函数进行变量节点更新. 第 6 行的作用是利用 Wei 等人提出的方法来减少欧几里德投影的次数[8]. 第 7 行利用超松弛技术来提高译码速度,其中 ρ 为超松弛参数. 第 8 行更新中间变量 $\boldsymbol{t}_i$. 第 9 ~ 10 行对译出的码字比特位进行硬判决,并按照 $\boldsymbol{H}\boldsymbol{x}^{\mathrm{T}}=0$ 判断是否终止译码.

说明 2　在算法 2 中,第 1 ~ 3 行进行参数的初始化. 第 6 行的作用是利用 Wei 等人提出的方法来减少欧几里德投影的次数[8]. 第 7 行利用超松弛技术来提高译码速度,其中 ρ 为超松弛参数. 第 8 行利用文献[16]中 I-l_1-PF 罚函数进行变量节点更新. 第 9 ~ 10 行对译出的码字比特位进行硬判决,并按照 $\boldsymbol{H}\boldsymbol{x}^{\mathrm{T}}=0$ 判断是否终止译码.

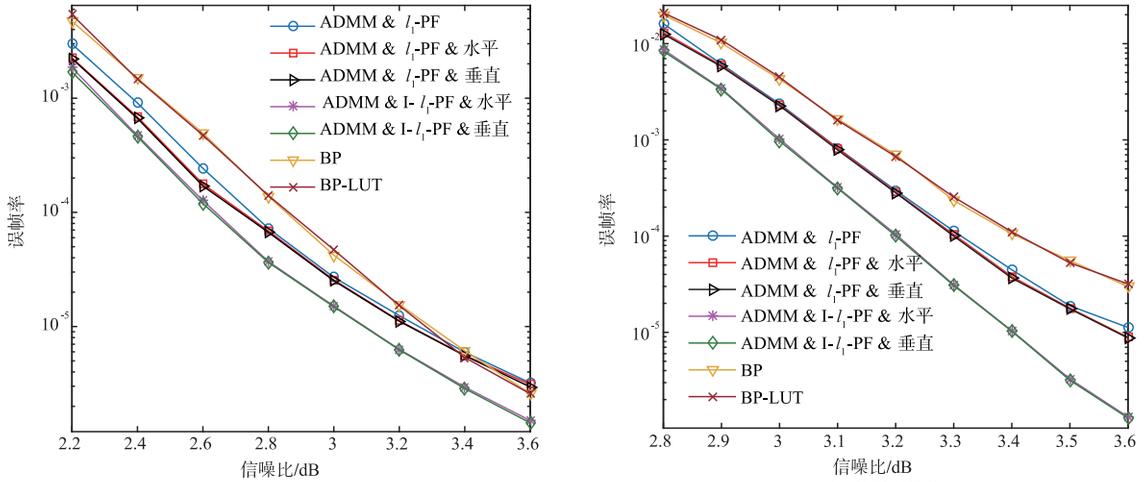
4　仿真分析

本文的仿真选择 IEEE 802.16e 标准中两个典型的 LDPC 码[17]:码率为 0.5 的(576,288)码 C_1 和码率为

0.75 的(1152,288)码 C_2. 从译码性能、平均迭代次数和平均译码时间三个方面,对本文设计的两种基于 I-l_1-PF 罚函数的分层调度 ADMM 惩罚译码方法、原来基于 l_1 罚函数的 ADMM 惩罚译码方法、两种基于 l_1 罚函数并采用减少欧几里德投影次数技术的原有分层调度 ADMM 惩罚译码方法、LDPC 码经典的对数域 BP 译码方法及基于查表法的对数域 BP(BP-LUT)译码方法[18]进行了比较. 在仿真实验中,所有 ADMM 惩罚译码方法的实现都借助超松弛 ADMM 和基于 $\boldsymbol{H}\boldsymbol{x}^{\mathrm{T}}=0$ 的早停止技术,并设定超松弛参数为 1.9,容差值为10^{-5},最大迭代次数为 1000,两种 BP 译码方法的最大迭代次数为 100. 仿真时,按照文献[5]中的方法来优化原有 ADMM 惩罚译码方法以及两种原有分层调度 ADMM 惩罚译码方法中的拉格朗日因子 μ 与罚参数 α,按照差分进化(Differential Evolution,DE)[15]方法来优化本文方法中拉格朗日因子μ、罚参数α与罚参数β. 本文所有仿真实验的运行环境为:64 位 Win7 操作系统、Intel(R) Core(TM) i5-3470 处理器、3.2GHz 主频、8GB 内存以及 VC6.0 的编程环境.

4.1 译码性能

图 1 给出了码 C_1 和 C_2 在不同信噪比下七种译码方法的译码性能. 从图 1 可以看出,本文所设计两种方法的译码性能大致相同,并且均优于原 ADMM 惩罚译码方法、原分层调度 ADMM 惩罚译码方法及两种 BP 译码方法. 对码 C_1,当误帧率为 3×10^{-5}时,与三种原有 ADMM 惩罚译码方法相比较,本文方法获得了大约 0.15dB 的编码增益;与两种 BP 译码方法相比较,本文方法获得了大约 0.2dB 的编码增益. 类似地,对码 C_2,当误帧率为10^{-5}时,与三种原有 ADMM 惩罚译码方法相比较,本文方法获得了大约 0.2dB 的编码增益;尽管图 1(b)未绘出两种 BP 译码方法误帧率为10^{-5}时的性能曲线,但与两种 BP 译码方法相比较,本文方法明显获得了超过 0.2dB 的编码增益,这说明在较高信噪比区域本文方法的译码性能明显优于两种 BP 译码方法的译码性能. 表中的标注代表相应的译码方法,例如标注"ADMM & I-l_1-PF & 水平"表示本文所设计的基于 I-l_1-PF 罚函数的水平分层调度 ADMM 惩罚译码方法.



(a) 码C_1的译码性能 (b) 码C_2的译码性能

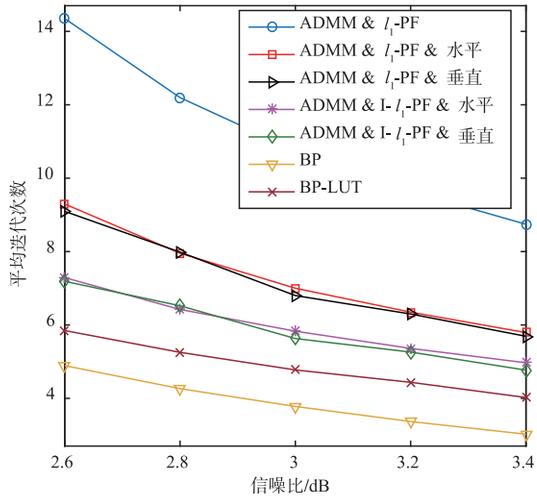
图1 码C_1和C_2的译码性能

4.2 平均迭代次数

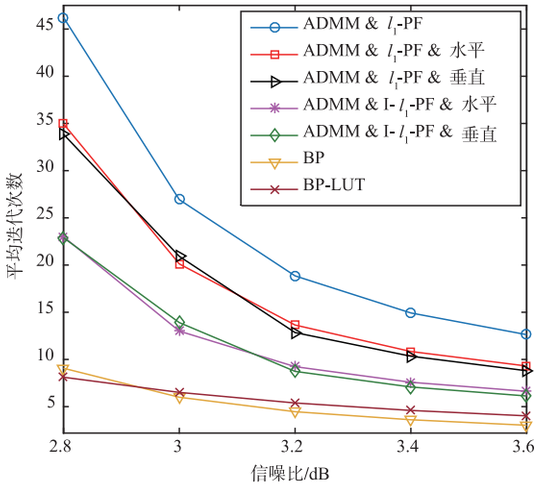
图 2 给出了码 C_1 和 C_2 在不同信噪比下七种译码方法的平均迭代次数. 从图 2 可以看出,本文所设计两种方法的平均迭代次数大致相同,且明显低于原 ADMM 惩罚译码方法和原分层调度 ADMM 惩罚译码方法的平均迭代次数,但要高于两种 BP 译码方法的平均迭代次数. 对码 C_1,当信噪比为 2.8dB 时,本文所设计方法的平均迭代次数比原 ADMM 惩罚译码方法和原分层调度 ADMM 惩罚译码方法的平均迭代次数分别降低了 49.2% 和 21.6%,比对数域 BP 译码方法和基于查表法的对数域 BP 译码方法的平均迭代次数分别增加了 51.9% 和 22.9%. 而对码 C_2,当信噪比为 2.8dB 时,与原 ADMM 惩罚译码方法和原分层调度 ADMM 惩罚译码方法相比较,本文所设计方法的平均迭代次数能够分别降低了 50.2% 和 34.2%;但是,本文所设计方法的平均迭代次数比对数域 BP 译码方法与基于查表法的对数域 BP 译码方法的平均迭代次数分别增加了 1.53 倍和 1.83 倍.

4.3 平均译码时间

图 3 给出了码 C_1 和 C_2 在不同信噪比下七种译码方法的平均译码时间. 可以看出,本文所设计两种方法的平均译码时间大致相同,且明显低于原 ADMM 惩罚译码方

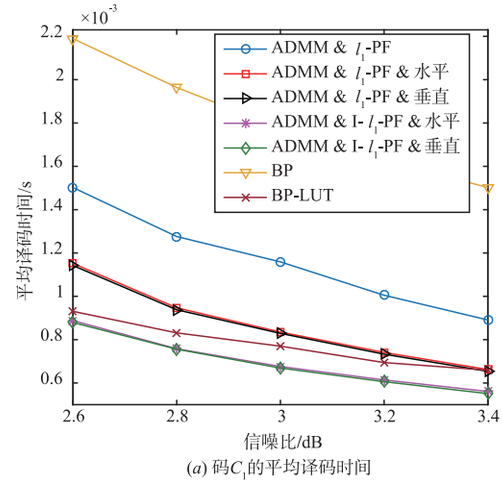


(*a*) 码C_1的平均迭代次数

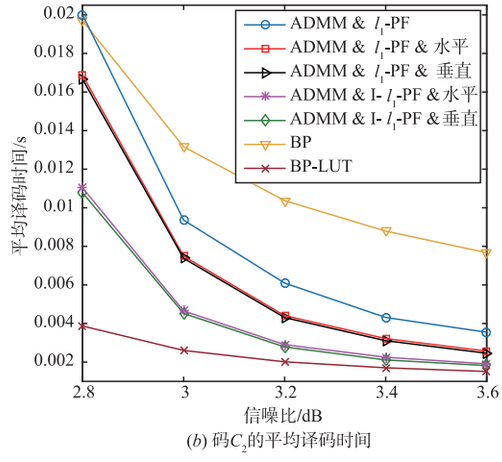


(*b*) 码C_2的平均迭代次数

图2　码C_1和C_2的平均迭代次数



(*a*) 码C_1的平均译码时间



(*b*) 码C_2的平均译码时间

图3　码C_1和C_2的平均译码时间

法和原分层调度 ADMM 惩罚译码方法的平均译码时间. 对码 C_1,当信噪比为 2.8dB 时,本文方法的平均译码时间比原 ADMM 惩罚译码方法和原分层调度 ADMM 惩罚译码方法的平均译码时间分别节省了 40.9% 和 22.9%. 而对码 C_2,当信噪比为 2.8dB 时,本文方法的平均译码时间比原 ADMM 惩罚译码方法和原分层调度 ADMM 惩罚译码方法的平均译码时间分别节省了 44.9% 和 34.6%. 而与两种 BP 译码方法相比较时,对码 C_1,本文所设计方法的平均译码时间明显少于 BP 译码方法的平均译码时间,略少于基于查表法的 BP 译码方法的平均译码时间;而对码 C_2,本文所设计方法的平均译码时间明显少于 BP 译码方法的平均译码时间,而多于基于查表法的 BP 译码方法的平均译码时间. 但是,随着信噪比的增加,本文所设计方法的平均译码时间逼近基于查表法的 BP 译码方法的平均译码时间.

5　结束语

为了能够更进一步提高 LDPC 码的 ADMM 惩罚译码速度,通过 Wei 等人提出的方法来减少欧几里德投影的次数,利用水平分层调度与垂直分层调度策略,本文分别设计了两种基于 I-l_1-PF 罚函数的分层调度 ADMM 惩罚译码方法. 仿真实验表明,所设计的两种译码方法的译码性能、平均迭代次数和平均译码时间大致相同,且与现有 ADMM 惩罚译码方法相比较,所设计的译码方法不仅具有较好的译码性能,而且能够显著降低 LDPC 码译码的平均迭代次数和提高译码速度. 同时,与两种 BP 译码方法相比较,本文所设计的译码方法具有更好的译码性能,但高码率 LDPC 码(例如码 C_2)的译码速度低于基于查表法的 BP 译码方法. 因此,如何加快高码率 LDPC 码的 ADMM 惩罚译码速度有待于进一步研究.

参考文献

[1] 陈海强,梁奇,黎相成,等. 瑞利信道下基于广义阈值函

数的 LDPC 译码算法[J]. 电子学报,2017,45(1):16-21.
CHEN Hai-qiang, LIANG Qi, LI Xiang-cheng, et al. LDPC decoding algorithm with generalized threshold-function over Rayleigh fading channel[J]. Acta Electronica Sinica, 2017,45(1):16-21. (in Chinese)

[2] 孙友明,陈海强,黎相成,等. 基于节点子集和 k 阶信息截断的多元 LDPC 译码算法[J]. 电子学报,2017,45(8):1925-1930.
SUN You-ming, CHEN Hai-qiang, LI Xiang-cheng, et al. Decoding algorithm for non-binary LDPC codes based on node-subset and k-order message truncation[J]. Acta Electronica Sinica,2017,45(8):1925-1930. (in Chinese)

[3] 陈海强,罗灵山,孙友明,等. 基于大数逻辑可译 LDPC 码的译码算法研究[J]. 电子学报,2015,43(6):1169-1173.
CHEN Hai-qiang, LUO Ling-shan, SUN You-ming, et al. Decoding algorithms for majority-logic decodable LDPC codes[J]. Acta Electronica Sinica, 2015, 43(6):1169-1173. (in Chinese)

[4] Kschischang F R, Frey B J, Loeliger H A. Factor graphs and the sum-product algorithm[J]. IEEE Transactions on Information Theory,2001,47(2):498-519.

[5] Barman S, Liu Xi-shuo, Draper S C, et al. Decomposition methods for large scale LP decoding[J]. IEEE Transactions on Information Theory,2013,59(11):7870-7886.

[6] Zhang Xiao-jie, Siegel P H. Efficient iterative LP decoding of LDPC codes with alternating direction method of multipliers[A]. Proceedings of IEEE International Symposium on Information Theory[C]. Turkey:IEEE,2013. 1501-1505.

[7] Zhang Guo-qiang, Heusdens R, Kleijn W B. Large scale LP decoding with low complexity[J]. IEEE Communications Letters,2013,17(11):2152-2155.

[8] Wei Hao-yuan, Jiao Xiao-peng, Mu Jian-jun. Reduced-complexity linear programming decoding based on ADMM for LDPC codes[J]. IEEE Communications Letters, 2015, 19(6):909-912.

[9] Jiao Xiao-peng, Mu Jian-jun, He Yu-Cheng, et al. Efficient ADMM decoding of LDPC codes using look-up tables[J]. IEEE Transactions on Communications,2017,65(4):1425-1437.

[10] Debbabi I, Gal B L, Khouja N, et al. Fast converging ADMM penalized algorithm for LDPC decoding[J]. IEEE Communications Letters,2016,20(4):644-647.

[11] Debbabi I, Gal B L, Khouja N, et la. Comparison of different schedulings for the ADMM based LDPC decoding[A]. Proceedings of 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)[C]. France:IEEE,2016. 51-55.

[12] Jiao Xiao-peng, Mu Jian-jun, Wei Hao-yuan. Reduced complexity node-wise scheduling of ADMM decoding for LDPC codes[J]. IEEE Communications Letters,2017,21(3):472-475.

[13] Boyd S, Parikh N, Chu E, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers[J]. Foundations and Trends in Machine Learning,2011,3(1):1-122.

[14] Liu Xi-shuo, Draper S C. The ADMM penalized decoder for LDPC codes[J]. IEEE Transactions on Information Theory,2016,62(6):2966-2984.

[15] Jiao Xiao-peng, Wei Hao-yuan, Mu Jian-jun, et al. Improved ADMM penalized decoder for irregular low-density parity-check codes[J]. IEEE Communications Letters, 2015,19(6):913-916.

[16] Wang Biao, Mu Jian-jun, Jiao Xiao-peng, et al. Improved penalty functions of ADMM penalized decoder for LDPC codes[J]. IEEE Communications Letters, 2017, 21(2):234-237.

[17] LDPC Coding for OFDMA PHY. IEEE Standard C802.16e-05/0066r3[S]. 2005.

[18] Hu Xiao-yu, Eleftheriou E, Arnold D M, et al. Efficient implementations of the sum-product algorithm for decoding LDPC codes[A]. Proceedings of IEEE Global Telecommunications Conference[C]. USA:IEEE,2001. 1036-1036E.

作者简介

王 彪 男,1982 年 4 月出生于天津市宝坻区. 2018 年获西安电子科技大学工学博士学位. 现为宝鸡文理学院数学与信息科学学院讲师. 主要研究方向为差错编码控制技术.
E-mail:wangbiao401@sina.com

慕建君 男,1965 年 3 月出生于陕西省吴堡县. 2002 年毕业于西安电子科技大学,博士. 现为西安电子科技大学教授,博导. 主要研究方向为 LDPC 码、闪存系统纠错码技术.
E-mail:jjmu@xidian.edu.cn