基于查找表的 ADMM 译码算法中量化算法优化研究

刘华军[1] 唐诗迪[1] 张迪科[2] 夏巧桥[1]
(华中师范大学物理科学与技术学院 武汉 430079)[1] (武汉大学电子信息学院 武汉 430072)[2]

摘 要 在基于 ADMM 的线性规划译码中,待投影向量向校验多胞体进行欧几里得投影计算是最复杂和耗时的部分。基于查找表的 ADMM-LDPC 译码算法通过简单的查表操作来替代复杂的投影运算,简化了投影过程,提升了算法的效率,但消耗了大量的内存资源。之后研究者提出了非均匀量化方法,该方法虽然极大地减少了内存消耗,但是所采用的量化方案的计算复杂度较高,从而使得该方法在量化段数较多的条件下难以实现。针对该问题,文中提出了一种新的非均匀量化方法。首先,针对不同的码字,在不同信噪比条件下,通过实验统计待投影向量中元素的分布特性,探究其分布规律,并设计相应的函数作为量化映射关系;然后,采用差分进化算法对函数的参数进行优化,从而得出在该函数下的最优量化方案,最终确定量化函数。仿真实验表明:与已有的量化方法相比,文中设计的非均匀量化方法具有不受量化段数、精度等因素影响的优点;且针对不同的码字,所提方法在高信噪比下均能达到 0.05 dB 左右的性能增益。

关键词 LDPC 码,查找表,非均匀量化,ADMM 惩罚译码
中图法分类号 TN911.22 **文献标识码** A **DOI** 10.11896/jsjkx.181001871

Study on Optimization of Quantization Algorithm in ADMM Decoding Algorithm Based on Lookup Table

LIU Hua-jun[1] TANG Shi-di[1] ZHANG Di-ke[2] XIA Qiao-qiao[1]
(School of Physical Science and Technology, Central China Normal University, Wuhan 430079, China)[1]
(School of Electronic Information, Wuhan University, Wuhan 430072, China)[2]

Abstract In the linear programming decoding based on ADMM, the Euclidean projection calculation of the projection vector to the check multi-cellular body is the most complex and time-consuming part. The ADMM-LDPC decoding algorithm based on look-up tables replace the time complexity of algorithm with simple table look-up operations to simplify the projection process and improve the efficiency of the algorithm, however it expends much memory consumption. Later, the researchers proposed the nonuniform quantization method, which can decrease the memory consumption effectively, but the computation complexity of the quantitative method is too high, so that the way is difficult to achieve when the number of segments is too much. For this problem, this paper proposed a new nonuniform quantization method. Firstly, for different code-words, the distribution characteristics of elements in the vector are calculated to be projected under different SNRs conditions by experiment, its distribution rules are explored and the corresponding function is designed as the quantitative mapping relation. Then, differential evolution algorithm is used to optimize the parameters of the function, the optimal quantization schema under the function is obtained, and the quantization function is finally determined. The simulation results show that, compared with the existing quantitative methods, the nonuniform method proposed in this paper has the advantage of not being affected by the number of quantization segment, precision and other factors. What's more, it achieves about 0.05 dB performance improvement for different code words in high SNRs.

Keywords LDPC codes, Look-up tables, Nonuniform quantization, Penalized ADMM decoder

1 引言

对于低密度奇偶校验(Low-Density Parity-Check, LDPC)码的译码问题,Feldman[1]首次提出将凸优化理论与纠错码译码问题相结合,并将最大似然译码模型转化为线性规划译码(Linear Programming, LP)模型来进行译码,奠定了线性规划译码的理论基础。线性规划译码在很多方面都比置信传播(Belief Propagation, BP)算法更具优势,例如:线性规划译码算法具有最大似然特性[2],在高信噪比下仍具有较好的误码率[3-4];并且还能消除 LDPC 码中短环的影响,提升译码性

到稿日期:2018-10-09 返修日期:2019-03-24 本文受国家自然科学基金项目(61501334),华中师范大学中央高校基本科研业务费(CCNU16A05028)资助。

刘华军(1994-),男,硕士,主要研究方向为高速实时数据处理平台、LDPC 等;**唐诗迪**(1997-),男,主要研究方向为 LDPC 码的译码;**张迪科**(1994-),男,硕士,主要研究方向为信道编解码;**夏巧桥**(1987-),男,博士,讲师,主要研究方向为高速实时数据处理平台、LDPC 等,E-mail:xqq2947759@163.com(通信作者)。

能。然而，该算法的复杂度较高，内存消耗大。在此基础上，Barman[5]提出一种基于交替方向乘子法（Alternating Direction Method of Multipliers，ADMM）的线性规划译码算法（ADMM-based Linear Programming decoding，ADMM-LP），该算法有效地降低了译码的复杂度。之后，为使算法具有更好的译码性能，Jiao 等[6-7]提出带有惩罚项的 ADMM-LP 译码算法（ADMM-based Linear Programming decoding with Penalty Function，ADMM-PD），在惩罚项的作用下，大大提升了算法的译码性能。Wang 等[8]提出了通过重新构造惩罚函数的方法来进一步提升译码性能，但相较于 BP 译码算法，该算法仍有较高的计算复杂度。

如何降低算法的计算复杂度，提高算法的效率，是线性规划译码问题中的一个主要研究方向。后续研究发现，在基于 ADMM 的线性规划译码中，向量向校验多胞体上的欧几里得投影是最复杂、计算量最大的一部分。为此，Zhang 等提出一种基于割查找的投影算法来降低 ADMM-LP 译码算法的计算复杂度[9]；Wei 等[10]提出了一种基于迭代方法的投影算法，避免了投影过程中复杂的排序操作。Zhang 等[11]将原先的投影算法简化为一种具有线性时间复杂度的投影算法；之后，Wasson 等提出了一种适合硬件实现的投影算法[12]；Wei 等[13]提出通过减少投影运算中无用的投影运算来降低系统运算量。另外，Jiao 等[14]提出建立查找表的方法，用简单的查表操作来替代复杂的投影运算，从而提升算法的运算效率。之后，他们[15]使用非均匀量化的方法对待投影向量进行量化，以进一步减少内存消耗。除了通过对投影算法进行优化之外，Debbabi 等[16-17]也使用改变算法调度的方式来提升算法的运行效率。

与文献[14]采用的均匀量化相比，文献[15]所使用的方法在量化段数相同的情况下具有更好的译码性能，即可以使用更少的量化段数来达到更好的译码性能，从而大大减少了内存耗损。但文献[15]中的方法在选取最优量化方案的过程中，计算复杂度太高，在选取段数较多的量化方案中很难得到实现，并且决定最优方案的判决条件是原始向量进行欧几里得投影运算后得到的结果与量化后的向量投影后得到的结果之间的均方误差最小，而实际上二者的均方误差最小并不意味着译码性能最好，因此，该方法在选取过程中存在着局限性。针对该问题，本文进一步探究了待投影向量中元素的分布规律，并使用函数形式来表示该规律，最终采用差分进化方法对函数的参数进行优化，从而得出在该函数下的最优量化方案。实验表明，相较于文献[15]中的方法，本文方法对任意段数的量化方案均适用，且在高信噪比时，译码性能可以得到 0.05 dB 左右的提升。

2　基于 ADMM-LP 的 LDPC 译码算法

2.1　基于 ADMM-LP 的惩罚译码算法

对于一个校验矩阵为 $\boldsymbol{H}(m\times n)$ 的 LDPC 码，其包含 m 个校验节点 c_j 和 n 个变量节点 v_i。其中，$j\in J\equiv\{1,2,\cdots,m\}$ 和 $i\in I\equiv\{1,2,\cdots,n\}$ 分别表示 $\boldsymbol{H}$ 矩阵行、列元素的下标索引。$N_c(j)=\{i|\boldsymbol{H}_{j,i}=1\}$ 和 $N_v(i)=\{j|\boldsymbol{H}_{j,i}=1\}$ 分别表示 $\boldsymbol{H}$ 矩阵中每一行、每一列中非零元素的索引。同时，$d_{c_j}=|N_c(j)|$ 和 $d_{v_i}=|N_v(i)|$ 分别表示 LDPC 码中校验节点与变量节点的维度，若所有的校验节点具有相同的维度 d_c，所有的变量节点具有相同的维度 d_v，则该 LDPC 码为规则的 LDPC 码；反之，LDPC 码为不规则的 LDPC 码。对最大似然译码算法而言，我们可将其转化成如式(1)所示的优化问题：

$$\begin{aligned}&\min\ f(\boldsymbol{x})=\boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{x}\\&\text{s. t.}\ \ \boldsymbol{P}_j\boldsymbol{x}=\boldsymbol{z}_j,\boldsymbol{z}_j\in P_{d_j},\forall j\in J\end{aligned}\tag{1}$$

其中，$\boldsymbol{z}_j$ 是一个辅助向量，使其满足 $\boldsymbol{z}_j=\boldsymbol{P}_j\boldsymbol{x}$ 对所有的 $j\in J$ 均成立。$\boldsymbol{\gamma}=[\gamma_1,\gamma_2,\cdots,\gamma_n]^{\mathrm{T}}$ 表示码字通过信道之后的对数似然比向量，且 $\gamma_i=\log[\frac{P_r(r_i/x_i=0)}{P_r(r_i/x_i=1)}]$，$i\in I$。$\boldsymbol{P}_j$ 表示与第 j 个校验节点相对应的大小为 $d_{c_j}\times n$ 的转换矩阵。例如，若校验矩阵中存在一行 $h_j=\{101010\}$，则转换矩阵为：

$$\boldsymbol{P}_j=\begin{bmatrix}1&0&0&0&0&0\\0&0&1&0&0&0\\0&0&0&0&1&0\end{bmatrix}$$

其中，P_{d_j} 表示第 j 个校验节点对应的校验多胞体，可表示为：

$$P_{d_j}=con(\{\in\{0,1\}^{d_j}\}\ \|e\|_1\ \text{is even}\})$$

对于式(1)所示的优化问题，我们亦可得到其增广拉格朗日式：

$$L_\mu(\boldsymbol{x},\boldsymbol{z},\boldsymbol{\lambda})=\boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{x}+\sum_{j\in J}(\boldsymbol{\lambda}_j^{\mathrm{T}}(\boldsymbol{P}_j\boldsymbol{x}-\boldsymbol{z}_j)+\frac{\mu}{2}\|\boldsymbol{P}_j\boldsymbol{x}-\boldsymbol{z}_j\|_2^2)\tag{2}$$

其中，$\boldsymbol{\lambda}$ 是与每个校验节点相对应的拉格朗日乘子向量，μ 是一个能提升算法收敛性并且大于零的惩罚参数。每次迭代的更新规律如式(3)－式(5)所示：

$$\begin{aligned}x_i^{k+1}&=\underset{x\in X}{\operatorname{argmin}}\,L_\mu(\boldsymbol{x},z^k,\lambda^k)\\&=\prod_{[0,1]}[\frac{1}{d_{v_i}}(\sum_{j\in N_v(i)}(z_j^k-\lambda_j^k)-\gamma_i/\mu)]\end{aligned}\tag{3}$$

$$z_j^{k+1}=\underset{z\in Z}{\operatorname{argmin}}\,L_\mu(x^{k+1},\boldsymbol{z}_j,\lambda_j^k)=\prod_{P_d}(v_j)\tag{4}$$

$$\lambda_j^{k+1}=\lambda_j^k+\mu(\boldsymbol{P}_jx^{k+1}-z_j^{k+1})\tag{5}$$

其中，$k\geqslant1$ 表示迭代次数。在式(3)中，$\prod_{[0,1]}(\boldsymbol{v})$ 表示将向量 $\boldsymbol{v}$ 投影到单位多面体中。在式(4)中，$\boldsymbol{v}=\boldsymbol{P}_j\boldsymbol{x}+\frac{\gamma_i}{\mu}$，$\prod_{P_{d_j}}(\boldsymbol{v})$ 表示向量 $\boldsymbol{v}$ 在校验多胞体 P_{d_j} 上的欧几里得投影运算。

为了进一步提升算法的译码性能，Liu 提出带有惩罚项的 ADMM-LP 译码算法，其核心是在惩罚项的作用下，变量节点的值尽可能远离判决门限 0.5。因此，式(1)中的优化问题也转变为如式(6)所示的优化问题：

$$\begin{aligned}&\min\ f(\boldsymbol{x})=\boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{x}+\prod_i g(x_i)\\&\text{s. t.}\ \ \boldsymbol{P}_j\boldsymbol{x}=\boldsymbol{z}_j,\boldsymbol{z}_j\in P_{d_j},\forall j\in J\end{aligned}\tag{6}$$

由式(6)可知，目标函数中引入了一个惩罚函数，从而导致 x 的更新过程发生变化，即由式(3)转变为式(7)：

$$x_i^{k+1}=\prod_{[0,1]}[\frac{1}{d_{v_i}}(\sum_{j\in N_v(i)}(z_j^k-\lambda_j^k)-(\gamma_i+g'(x_i))/\mu)]\tag{7}$$

Liu[7]也给出了几种典型的惩罚函数，其中，对于函数 $g(x)=-\alpha|x-0.5|$，我们记为 l_1 惩罚函数；对于函数 $g(x)=-\alpha(x-0.5)^2$，我们记为 l_2 惩罚函数。实验结果表明，引入惩罚函数后，算法具有更好的译码性能。把 l_1，l_2 惩罚函数分别代入式(8)中，则 x 的更新公式如式(8)、式(9)所示：

$$x_i=\begin{cases}\prod_{[0,1]}(\frac{1}{d_{v_i}}(t_i+\frac{\alpha}{\mu})),&t_i\geqslant\frac{d_{v_i}}{2}\\\prod_{[0,1]}(\frac{1}{d_{v_i}}(t_i-\frac{\alpha}{\mu})),&t_i\leqslant\frac{d_{v_i}}{2}\end{cases}\tag{8}$$

$$x_i=\prod_{[0,1]}\left(\frac{1}{d_{v_i}-\frac{2\alpha}{\mu}}\right)\left(t_i-\frac{\alpha}{\mu}\right)\tag{9}$$

其中，$t_i=\sum_{j\in N_v(i)}(z_j^{(i)}-\frac{\lambda_j^{(i)}}{\mu})-\frac{\gamma_i}{\mu}$。在式(8)、式(9)中，$\alpha$ 为一个大于 0 的常数，被称为惩罚系数，当 α 为 0 时，ADMM-PD 译码算法将退化为 ADMM-LP 译码算法。

2.2 基于查找表的 ADMM-LDPC 译码算法

为提升算法的效率，Jiao 等[14]提出用查找表的方法来替代式(4)中的欧几里得投影运算 $\prod_{P_{d_j}}(v)$，即 $\prod_{P_{d_j}}(u)=LUT_{d_j}[index]$。为此，在给定的量化区间$[a,b]$中，对待投影向量 $\boldsymbol{v}$ 进行 L 段的量化，使得 $u=Q(v)$，其映射函数如下所示：

$$Q(v)=\begin{cases}a, & v\leqslant a\\ a+\left\lfloor\frac{v-a}{\tau}+0.5\right\rfloor\tau, & a<v<b\\ b, & v\geqslant b\end{cases}\tag{10}$$

其中，τ 为每两段之间的间距，且 $\tau=(b-a)/(L-1)$，函数$\lfloor x\rfloor$表示对 x 向下取整。为此，量化向量中的元素 $q_k=a+(k-1)\tau,k=1,2,\cdots,L$。

构建一个完整的查找表可分为以下几步：

(1)令 $Q_{LV}=\{q_1,q_2,\cdots,q_L\}$，$Q_{ID}=\{1,2,\cdots,L\}$分别表示所有量化向量的集合以及每个量化向量中对应元素的索引向量集合，且二者中的每个元素满足一一对应关系，即 $\varphi(q_k)=k,k=1,2,\cdots,L$，则$|Q_{LV}^d|=|Q_{ID}^d|=L^d$。

(2)对每一个校验多胞体 P_d，均构建一个与之相对应的表 LUT_d，每个表中都存储 Q_{LV}^d 中与之相对应的量化向量的欧几里得投影结果。

(3)最后，将 Q_{LV}，Q_{ID}，LUT 中的数据以一定的方式分别存储起来，每输入一个量化向量，便通过 $\varphi(q_k)=k$ 得到与其相对应的索引向量，之后即可找到相对应的表，从表中读取数据进行输出。查找表的结构如表 1 所列。

表 1 查找表的结构

Table 1 Architecture of LUT

No.	索引向量 $\boldsymbol{w}$					量化向量 $\boldsymbol{u}$					投影向量 $\boldsymbol{z}$
	w_1	w_2	$\cdots$	w_{d-1}	w_d	u_1	u_2	$\cdots$	u_{d-1}	u_d	$LUT[i]=(z_1,z_2,\cdots,z_d)$
1	1	1	$\cdots$	1	1	q_1	q_1	$\cdots$	q_1	q_1	$LUT[1]$
2	1	1	$\cdots$	1	2	q_1	q_1	$\cdots$	q_1	q_2	$LUT[2]$
	.	.	$\cdots$	.	.	.	.	$\cdots$	.	.	
	1	1	$\cdots$	1	L	q_1	q_1	$\cdots$	q_1	q_L	
	1	1	$\cdots$	2	2	q_1	q_1	$\cdots$	q_2	q_2	
	.	.	$\cdots$	.	.	.	.	$\cdots$	.	.	
	1	1	$\cdots$	2	L	q_1	q_1	$\cdots$	q_2	q_L	
$\vdots$	.	.	$\cdots$	.	.	.	.	$\cdots$	.	.	$\vdots$
	1	L	$\cdots$	L	L	q_1	q_L	$\cdots$	q_L	q_L	
	2	2	$\cdots$	2	2	q_2	q_2	$\cdots$	q_2	q_2	
	.	.	$\cdots$	.	.	.	.	$\cdots$	.	.	
	2	L	$\cdots$	L	L	q_2	q_L	$\cdots$	q_L	q_L	
	.	.	$\cdots$	.	.	.	.	$\cdots$	.	.	
T	L	L	$\cdots$	L	L	q_L	q_L	$\cdots$	q_L	q_L	$LUT[T]$

Jiao 等在均匀量化的基础上做了更进一步的研究，通过实验发现待投影向量 $\boldsymbol{v}$ 中元素为非均匀分布，进而设计出非均匀量化的方法对待投影向量 $\boldsymbol{v}$ 进行量化。假定量化区间为$[a,b]$，量化段数为 L，为保证算法对所有码字均适用，设 $a+b=1$，且量化向量中的元素关于 0.5 对称，则量化向量为：

$$Q_L=\begin{cases}\{a,q_1,\cdots,q_{k-1},1-q_{k-1},\cdots,1-q_1,b\}, & L\text{ is even}\\ \{a,q_1,\cdots,q_{k-1},0.5,1-q_{k-1},\cdots,1-q_1,b\}, & L\text{ is odd}\end{cases}\tag{11}$$

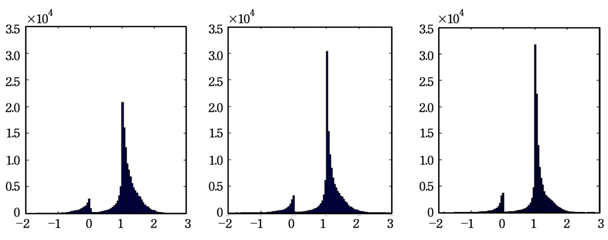
之后，给定间距 τ，则 $T=(b-a)/\tau$，$T_0=\lceil T/2\rceil$，$K=\lfloor L/2\rfloor$。由式(11)可知，量化向量中的元素关于 0.5 对称，因此，仅需对$(a,0.5)$区间中的元素进行优化挑选，即从 $A=\{a+k\tau:k=1,2,\cdots,T_0-1\}$中找出 $K-1$ 个元素，使量化向量进行欧几里得投影后得到的结果与原始向量进行投影后得到的结果之间的均方误差最小。

通过实验发现，该方法相较于之前的均匀量化，在译码性能上获得了较大的提升，但由于该方法的计算复杂度太高，导致在量化段数较多的情况下难以得到实现。例如，当我们考虑在量化区间$[-1.50,2.50]$中进行段数为 9 的分段，且令间距 $\tau=0.02$ 时，$T_0=100$，$K=4$，其需要进行 $C_{99}^3=\frac{99\times98\times97}{3\times2\times1}=156\,849$ 次的优化挑选。若要进一步提升精度，令 $\tau=0.01$，则需要进行 $C_{199}^3=\frac{199\times198\times197}{3\times2\times1}=1\,293\,699$ 次优化挑选，为此很难通过该方法得出结果。同样地，当选取段数更多时，算法的计算复杂度也会随着段数数目的增多呈现指数性增长，从而难以得出结果。同时，该方法选取最优的标准是量化向量与原始向量投影结果之间的均方误差最小，但事实上，均方误差最小并不意味着译码性能最好，因此该方法的衡量标准也存在着一定的局限性。

3 量化算法的优化

针对文献[15]中存在的问题，本节通过探究待投影向量中元素的分布情况，提出一种新的非均匀量化方法，以降低文献[15]中所采用的量化方法的复杂度(在量化过程中不受量化段数数目、精度等外在因素的限制)，并且在译码性能上获得一定的提升。前文已提到过，量化方案是由待投影向量中元素的分布情况来决定的，为此，本节在文献[15]的基础上分别探究两种不同码字的待投影向量中元素的分布情况。其中，对于 WIMAX(576,288)(码率为 0.5)，我们记为码字 C_1；对于 WIMAX(480,240)(码率为 0.5)，我们记为码字 C_2。

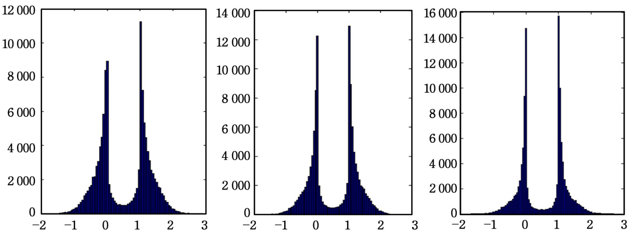
如图 1 所示，对于码字 C_2，(a)，(b)，(c)分别表示其在不同信噪比下待投影向量中元素的分布情况，可以看出，待投影向量中的元素大多集中在 0,1 附近，且在区间$[a,1]$中，元素的分布为依次增加，而在区间$[1,b]$中，元素的分布为依次减少。当信噪比逐渐增大时，待投影向量中的元素也越来越向 0,1 方向收敛。同样地，如图 2 所示，对于码字 C_2，其待投影向量中的元素有着相同的分布情况。



(a)$SNR=2.5$ dB (b)$SNR=3.0$ dB (c)$SNR=3.5$ dB

图 1 码字 C_1 的待投影向量元素分布图

Fig. 1 Distribution of projected vector for C_1



(a)SNR=2.5 dB　(b)SNR=3.0 dB　(c)SNR=3.5 dB

图 2　码字 C_2 的待投影向量元素分布图

Fig. 2　Distribution of projected vector for C_2

根据图 1、图 2 所示的元素分布情况,可进行如下分段。

(1)当量化向量中的元素属于区间$[a,0]$或$[1,b]$时,量化向量中的元素在区间$[1,b]$上的分布情况呈现为由密到稀(属于区间$[a,0]$中的元素与属于区间$[1,b]$中的元素关于 0.5 对称),即有较多的元素集中在 1 附近,当逐渐远离 1 时,该处集中的元素也越来越少。因此,对该区间的元素进行量化分段时,在 1 附近,分段间距应设置得较小一些;远离 1 的地方,量化间距应设置得较大一些。

(2)当量化向量中的元素属于区间(0,1)时,待投影向量中的元素在区间(0,1)上的分布情况并未表现出明显的规律。当待投影向量中存在一个元素 $v_i=0.5$ 时,投影向量中与之相对应的元素有较大可能在 0.5 附近,由 $\boldsymbol{P}_j\boldsymbol{x}=\boldsymbol{z}_j$ 可知,向量 $\boldsymbol{x}$ 中与之相对应的元素亦在 0.5 附近,由于此时对码字取值存在较大的不确定性,为减小量化引入的误差,在 0.5 附近应将量化间距设置得较小。

为得出最优量化方案,本文通过构造映射函数来实现上述量化分段原则。由上述量化分段原则可知,有诸多函数满足该量化分段规律,例如三次函数、指数函数等。由于需用差分进化方法对函数中的参数进行优化,所选取的函数应满足结构简单、计算复杂度较低以及参数较少等特性。因此,综合衡量后本文设计如式(12)所示的函数作为量化映射关系:

$$f(x)=me^{nx},x=1,2,\cdots \tag{12}$$

由文献[14]可知,量化向量中的元素关于 0.5 对称,因此令量化向量为:

$$\boldsymbol{Q}_L=\begin{cases}\{1-q_k,\cdots,1-q_1,q_1,\cdots,q_k\}, & \text{if } L \text{ is even}\\ \{1-q_{k-1},\cdots,1-q_1,0.5,q_1,\cdots,q_k\}, & \text{if } L \text{ is odd}\end{cases} \tag{13}$$

其中,$K=\lfloor L/2 \rfloor$。为方便起见,令 $Q_l=\{q_1,q_2,\cdots,q_k\}$,且 Q_l 中的每个元素 $q_k=me^{nk},k=1,2,\cdots,k$,之后对待投影向量 $\boldsymbol{v}$ 中的每个元素进行判决赋值,即可得到量化后的向量。算法 1 的步骤如下所示。

算法 1　非均匀量化算法

Step 1　输入待投影向量 $\mathbf{v}$,初始化量化段数 L,映射函数参数 m,n,$K=\lfloor L/2 \rfloor$以及量化向量

$$\mathbf{Q}_L=\begin{cases}\{1-q_k,\cdots,1-q_1,q_1,\cdots,q_k\}, & \text{if L is even}\\ \{1-q_{k-1},\cdots,1-q_1,0.5,q_1,\cdots,q_k\}, & \text{if L is odd}\end{cases}$$

Step 2　产生向量 $\mathbf{q}_k$,且 $\mathbf{q}_k=me^{nk}$,其中 $1\leqslant k\leqslant K$。

Step 3　更新向量 $\mathbf{v}$,其中 $j\in J,i\in N_c(j)$。

$$v_{j\to i}=\begin{cases}1-q_k, & v_{j\to i}\leqslant 1-q_k\\ v_{j\to i}, & 1-q_k<v_{j\to i}\leqslant q_k\\ q_k, & v_{j\to i}>q_k\end{cases}$$

Step 4　若 $\mathbf{Q}_L(t)\leqslant v_{j\to i}\leqslant \mathbf{Q}_L(t+1)$,则更新向量

$$u_{j\to i}=\begin{cases}Q_L(t), & v_{j\to i}-Q_L(t)<Q_L(t+1)-v_{j\to i}\\ Q_L(t+1), & v_{j\to i}-Q_L(t)\geqslant Q_L(t+1)-v_{j\to i}\end{cases}$$

其中,$1\leqslant t\leqslant L,j\in J,i\in N_c(j)$。

Step 5　输出向量 $\mathbf{u}$,算法结束。

由算法 1 可以看出,与文献[14]所采用的方法相比,本文所采用的方法对量化的段数没有限制,即不会存在计算复杂度过大的情况。之后,即可把该算法代入 ADMM-LP 的模型中,并对参数进行优化。算法 2 给出了完整的译码流程。

算法 2　基于非均匀量化的 ADMM-L2 译码算法

Step 1　输入向量 $\mathbf{x}$,初始化向量 $\mathbf{z}_{j\to i}^0=0.5,\boldsymbol{\lambda}_{j\to i}^0=0,\mathbf{w}^0=0$,迭代次数 k=0,以及最大迭代次数 iter_max;其中 $j\in J,i\in N_c(j)$。

Step 2　向量 $\mathbf{x}$ 的更新:

$$\mathbf{x}_{i\to j}^{k+1}\prod_{[0,1]}\left[\frac{1}{d_{v_i}-2\alpha/\mu}\left(\sum_{j\in N_v(i)}(z_j^k-\lambda_j^k)-(\gamma_i+\alpha)/\mu\right)\right]$$

Step 3　向量 $\mathbf{v}$ 更新:$v_{j\to i}^{k+1}=P_j x_{i\to j}^{k+1}+\lambda_{i\to j}^{t\to k}/\mu$。

Step 4　量化:$u_{j\to i}^{k+1}=Q(v_{j\to i}^{k+1})$

Step 5　向量 $\mathbf{z}$ 更新:$z^{k+1}=LUT$。

Step 6　向量 $\boldsymbol{\lambda}$ 的更新:$\lambda_{j\to i}^{k+1}=\lambda_{j\to i}^{k}+\mu(P_j\, x_{i\to j}^{k+1}-z_{j\to i}^{k+1})$。

Step 7　若译码正确,$HC_i=0$;或者迭代次数达到 iter_max,则算法结束,输出 x;否则返回 Step 2。

由算法 2 可知,本文选用 $g(x)=-\alpha(x-0.5)^2$ 作为惩罚函数,需要优化的参数有惩罚参数 μ,α,以及量化函数中的参数 m 和 n。为此,我们采用文献[18]中的差分进化法(Differential Evolution,DE)对参数进行优化,即可得出最优的分段方式和最优的惩罚参数。本文中 DE 算法的最大迭代次数 $G_{\max}=60$,每一代种群中个体的数目为 100,其中参数 μ 的取值范围设为 $0<\mu<5$,α 的取值范围设为 $0<\alpha<5$,而对于量化函数中的参数 m,n,如图 1、图 2 所示,待投影向量中元素的分布基本集中在区间$[-2,3]$之间。因此,为节省优化参数的时间,可将 m 的取值范围设为 $0<m<2$,n 的取值范围设为 $0<n<2$。通过 DE 算法找出多组误码率表现最好的参数向量,并求取均值。

4　实验结果

本文实验平台的处理器为 Intel Core I7-4790,其主频为 3.60 GHz,内存为 12 GB,操作系统为 Win7 64 位操作系统,软件开发环境为 VS2013。对两种不同的工业标准码字进行实验,分别为码字 C_1:WIMAX(576,288)(码率为 0.5),码字 C_2:WIMAX(480,240)(码率为 0.5)。

在仿真实验中,Case1 和 Case3 为采用文献[15]中非均匀量化方法分别进行段数为 8,9 段量化后的实验结果;Case5,7,9 则为采用文献[14]中的均匀量化方法分别进行 11,15,28 段量化后的实验结果;Case2,4,6,8 则为采用本节所提出的非均匀量化方法分别进行 8,9,11,15 段量化后的实验结果。

其中,对于 C_1,Case1,3,5,7,9 中的参数直接选用文献[14-15]中的参数,Case2,3,4,6,8 中的参数则使用差分进化法优化得出;对于码字 C_2,Case1,3,5 中的参数以及分段情况采用文献[15]中的方法优化得出,其他实例则使用差分进化法得出相关参数,并进行分段。具体数据如表 2 所列。另外,在本文实验中,信道模型均采用加性高斯白噪声信道,最大迭

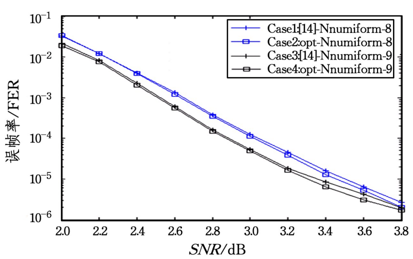
代次数 $iter\_max$=100。同时,在误帧率的实验中,保证在每一个信噪比(dB)的条件下至少收集 100 个错误帧;在平均迭代次数的实验中,保证在每一个 dB 的条件下至少收集 10000 帧的迭代次数总和。在实验结果中,[·]-$uniform$-y 和[·]-$nonuniform$-y 和分别表示该段结果为采用[·]中的均匀/非均匀量化方法进行 y 段量化后得出的实验结果。而 opt-$nonuniform$-y 则表示使用本文所提出的优化量化算法进行 y 段的量化后得出的实验结果。

表 2 码字 C_1 和 C_2 的参数列表

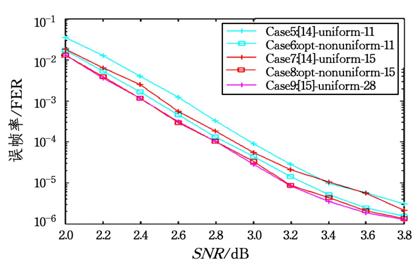
Table 2 Parameter lists of C_1 and C_2

码字	Case	μ	α	m	n	Q_L	L
C_1	1	5.400	3.800	—	—	{−1.500 −0.420 0.020 0.340 0.660 0.980 1.420 2.500}	8
	2	5.900	4.200	0.475	0.414	{−1.488 −0.645 −0.087 0.281 0.719 1.087 1.645 2.488}	
	3	5.900	4.200	—	—	{−1.500 −0.480 −0.080 0.160 0.500 0.840 1.080 1.480 2.50}	9
	4	5.600	4.500	0.610	0.356	{−1.533 −0.775 −0.243 0.129 0.5 0.871 1.243 1.775 2.533}	
	5	4.300	3.100	—	—	{−1.500:0.400:2.500}	11
	6	5.100	3.600	0.598	0.294	{−1.601 −0.938 −0.445 −0.077 0.198 0.5 0.802 1.077 1.445 1.938 2.601}	
	7	4.000	2.700	—	—	{−1.6:0.3:2.60}	15
	8	5.200	3.900	0.598	0.210	{−1.601 −1.108 −0.709 −0.385 −0.123 0.1999 0.262 0.5 0.738 0.9101 1.123 1.385 1.709 2.108 2.601}	
	9	4.000	2.700	—	—	{−2.200:0.200:3.200}	28
C_2	1	5.700	4.300	—	—	{−1.50 −0.580 −0.140 0.160 0.840 1.140 1.580 2.500}	
	2	5.700	4.500	0.45	0.428	{−1.493 −0.625 −0.059 0.310 0.690 1.059 1.625 2.493}	8
	3	5.500	4.1	—	—	{−1.500 −0.680 −0.160 0.240 0.500 0.760 1.160 1.680 2.500}	
	4	5.600	4.300	0.591	0.360	{−1.494 −0.740 −0.214 0.153 0.500 0.847 1.214 1.740 2.494}	9
	5	5.400	4.500	—	—	{−1.500:0.400:2.500}	
	6	5.600	4.500	0.542	0.308	{−1.528 −0.858 −0.365 −0.003 0.263 0.5 0.737 1.003 1.365 1.858 2.528}	11
	7	5.100	4.000	—	—	{−1.600:0.300:2.600}	
	8	4.800	3.700	0.589	0.211	{−1.580 −1.089 −0.692 −0.370 −0.109 0.102 0.273 0.5 0.727 0.898 1.109 1.370 1.692 2.089 2.580}	15
	9	4.500	3.300	—	—	{−2.200:0.200:3.200}	28

图 3 表示 C_1 在不同分段情况下得到的误帧率性能。如图 3(a)所示,对于 C_1,与文献[14]中的方法相比,本文所设计的方法在低信噪比下的译码性能基本相同,但在高信噪比下的性能有着一定的提升,如 Case3 和 Case4 所示,Case4 在 3.4dB 的条件下,误帧率可达到 6×10^{-6},Case3 则需要在 3.45dB左右才能达到同样的性能。从图3(b)中亦可看出,本文所选方法的受用面更广,可用于段数较多的场合,如 Case6 和 Case8 所示。



(a)

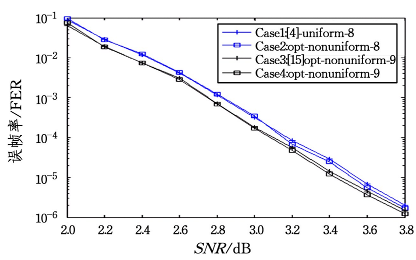


(b)

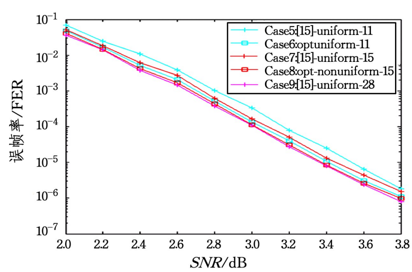
图 3 码字 C_1 的误帧率

Fig. 3 Frame error rate of C_1

图 4 表示码字 C_2 在不同分段情况下的误帧率性能。在低信噪比条件下本文方法与文献[15]中的方法的译码性能基本相同,但在高信噪比下本文方法码字 C_1 有将近 0.05dB 的性能提升。



(a)



(b)

图 4 码字 C_2 的误帧率

Fig. 4 Frame error rate of C_2

图 5 给出了码字 C_1 和 C_2 在不同分段情况下得到的平均迭代次数。可以看出,对于码字 C_1 和 C_2,采用本文所提算法的平均迭代次数略高于文献[15]中所使用的方法。如图 5 中 Case3 和 Case4 所示,在信噪比为 3.0dB 的条件下,Case3 的

平均迭代次数为 14.5 左右,Case4 的平均迭代次数则为 15.1 左右。

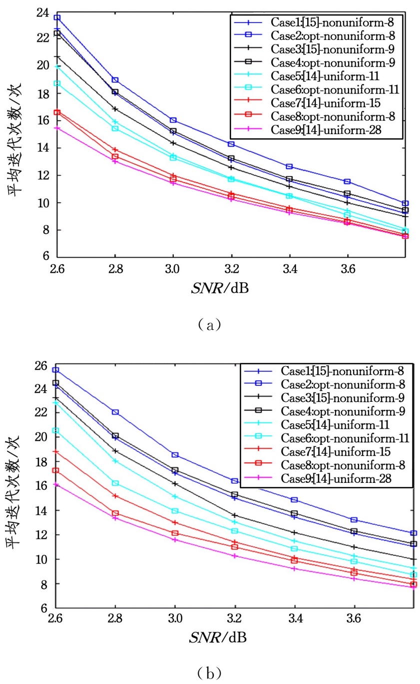


图 5　码字 C_1 和 C_2 的平均迭代次数

Fig. 5　Average number of iterations of C_1 and C_2

结束语　本文通过探究待投影向量中元素的分布情况,并使用函数形式来表示元素的分布规律,设计出一种非均匀量化方法。与文献[15]中的方法相比,本文所提方法不受量化段数、精度等的影响。由实验结果可知,相较于文献[14]中的方法,本文所提方法对不同的码字在高信噪比下均能实现 0.05 dB 左右的性能增益。但与此同时,本文所提量化方法也并非最佳量化方案,今后可对该问题做更多的研究,从而找出更优的量化方法。

参 考 文 献

[1] FELDMAN J. Decoding error-correcting codes via linear programming[D]. Cambridge: Massachusetts Institute of Technology, 2003.

[2] FELDMAN J, WAINWRIGHT M J, KARGER D R. Using linear programming to decode binary linear codes[J]. IEEE Transactions on Information Theory, 2005, 51(3): 954-972.

[3] BURSHTEIN D, GOLDENBERG I. Improved linear programming decoding of LDPC codes and bounds on the minimum and fractional distance[J]. IEEE Transactions on Information Theory, 2011, 57(11): 7386-7402.

[4] FELDMAN J, MALKIN T, SERVEDIO R A, et al. LP decoding corrects a constant fraction of errors[J]. IEEE Transactions on Information Theory, 2007, 53(1): 82-89.

[5] BARMAN S, LIU X S, DRAPER S C, et al. Decomposition methods for large scale LP decoding[J]. IEEE Transactions on Information Theory, 2013, 59(12): 7870-7886.

[6] JIAO X P, WEI H Y, MU J J. Improved ADMM penalized decoder for irregular low-density parity-check codes[J]. IEEE Communication Letters, 2015, 19(6): 913-916.

[7] LIU X S, DRAPER S C. The ADMM Penalized Decoder for LDPC Codes[J]. IEEE Transactions on Information Theory, 2016, 62(6): 2966-2984.

[8] WANG B, MU J J, JIAO X P, et al. Improved Penalty Functions of ADMM Decoder for LDPC Codes[J]. IEEE Communication Letters, 2016, 21(99): 234-237.

[9] ZHANG X, SIEGEL P H. Efficient iterative LP decoding of LDPC codes with alternating direction method of multipliers[C]// IEEE International Symposium on Information Theory. IEEE, 2013: 1501-1505.

[10] WEI H, BANIHASHEMI A H. An Iterative Check Polytope Projection Algorithm for ADMM-Based LP Decoding of LDPC Codes[J]. IEEE Communications Letters, 2017, PP(99): 1-1.

[11] ZHANG G, HEUSDENS R, KLEIJN W. Large scale LP decoding with low complexity[J]. IEEE Communications Letters, 2013, 17(11): 2152-2155.

[12] WASSON M, DRAPER S C. Hardware based projection onto the parity polytope and probability simplex[C]// Asilomar Conference on Signals, Systems and Computers. IEEE, 2015: 1015-1020.

[13] WEI H Y, JIAO X P, MU J J. Reduced-complexity linear programming decoding based on ADMM for LDPC codes[J]. IEEE Communication Letters, 2015, 19(6): 909-912.

[14] JIAO X P, MU J J, HE Y C, et al. Efficient ADMM Decoding of LDPC Codes Using Look-Up Tables[J]. IEEE Transactions on Communications, 2017, 19(4): 271-285.

[15] JIAO X, HE Y C, MU J, et al. Memory-Reduced Lookup Tables for Efficient ADMM Decoding of LDPC Codes[J]. IEEE Signal Processing Letters, 2018, 25(99): 1-1.

[16] DEBBABI I, GAL B L, KHOUJA N, et al. Fast Converging ADMM Penalized Algorithm for LDPC Decoding[J]. IEEE Communication Letters, 2016, 20(4): 648-651.

[17] DEBBABI I, GAL B L, KHOUJA N, et al. Comparison of different schedulings for the ADMM based LDPC decoding[C]// International Symposium on Turbo Codes & Iterative Information Processing. IEEE, 2016: 51-55.

[18] STORN R, PRICE K. Differential Evolution-A Simple and Efficient Heuristic for global Optimization over Continuous Spaces [J]. Journal of Global Optimization, 1997, 11(4): 341-359.