# 一种交叠的 Shuffled-BP LDPC 译码算法

范亚楠*①② 王丽冲①② 姚秀娟① 孟 新①
①(中国科学院国家空间科学中心 北京 100190)
②(中国科学院大学 北京 100190)

**摘 要**：Shuffled-BP(SBP) 译码算法是一种基于变量节点的串行消息传递译码算法，其收敛速度快于原有的置信度传播译码算法，然而由于实际工程实现中的半并行化处理，其收敛速度和误码性能均有所降低。为了进一步提高 SBP 算法的性能，该文提出一种交叠的 Shuffled-BP(Overlapped Shuffled-BP, OSBP)译码算法。该算法采用若干个相同的子译码器以不同的更新顺序同时进行更新，对于每个变量节点，在每次迭代更新后选取最可靠的信息参与下一次迭代，以此提高迭代的收敛速度。理论分析和仿真实验均表明，在不增加额外存储空间的条件下，OSBP 算法相比于 SBP 算法有着更优的误码性能以及更快的收敛速度。此外，提出的 OSBP 算法对于规则和不规则 LDPC 码均有效。

**关键词**：LDPC 码；收敛速度；译码算法；Shuffled-BP；交叠的 Shuffled-BP
**中图分类号**：TN911.22 **文献标识码**：A **文章编号**：1009-5896(2016)11-2908-08
**DOI**: 10.11999/JEIT151477

# An Overlapped Shuffled-BP LDPC Decoding Algorithm

FAN Yanan①② WANG Lichong①② YAO Xiujuan① MENG Xin①
①(*National Space Science Center*, *Chinese Academy of Sciences*, *Beijing* 100190, *China*)
②(*University of Chinese Academy of Sciences*, *Beijing* 100190, *China*)

**Abstract**: Shuffled-BP (SBP) decoding algorithm is a variable-node-based serial decoding algorithm, which converges faster than the original Belief-Propagation (BP) decoding algorithm. However, due to the semi-parallel processing, there is a decrease in terms of convergence speed and error performance. An Overlapped Shuffled-BP(OSBP) decoding algorithm is proposed to enhance further the performance of the Shuffled-BP algorithm. In this algorithm, more than one sub-decoders are used to execute simultaneously, every sub-decoder has different updating order from each other. Regarding each variable node, the most reliable messages are kept and used for the next iteration, thus a faster convergence can be provided. Both theoretical analysis and simulation results show that, compared with SBP algorithm, OSBP algorithm possesses a better error performance as well as a higher convergence speed and introduces no extra storage requirement. Moreover, the proposed algorithm is effective for both regular and irregular LDPC codes.

**Key words**: Low Density Parity Check (LDPC) code; Convergence speed; Decoding algorithm; Shuffled-BP; Overlapped Shuffled-BP

## 1 引言

LDPC 码以其渐进香农限[1,2]的误码性能而被信道编码界所广泛关注。由于其校验矩阵的稀疏性，其译码算法的空间复杂度较低，并且相比于 Turbo 码有着更低的错误平台。1962 年，Gallager 在其博士论文[3]里提出了置信度传播(Belief-Propagation, BP)LDPC 译码算法，BP 算法是一种基于洪水消息传递机制的译码算法，其收敛速度慢，并且消耗的计算资源和存储资源均非常巨大。为弥补 BP 算法的不足，研究人员开始研究基于串行消息传递机制的译码算法，例如文献[4,5]中提出的分层译码算法以及文献[6,7]中提出的 Shuffled-BP(SBP)译码算法。在该类算法中，消息逐节点依次进行更新和传递，这样使得在一次迭代中刚更新过的消息立刻参与到本次迭代的其他消息的更新中去，因而获得了较 BP 算法近两倍的收敛速度。另外，基于 SISO 的实现方式又节省了大量的存储空间，同时降低了计算复杂度。

收稿日期：2015-12-29；改回日期：2016-06-03；网络出版：2016-07-19
*通信作者：范亚楠 fanyanan_99@163.com
基金项目：中国科学院创新基金(CXJJ14S126)
Foundation Item: CAS Innovation Foundation (CXJJ14S126)

由于 SBP 算法串行更新的缘故，其译码延时比较大。为了降低译码延时，实际工程应用中一般需要进行半并行化处理，如文献[8,9]中所做的那样。然而并行处理后，各组中更新的信息并没有被该组其他节点所利用，导致 SBP 算法的收敛速度和误码性能均有所降低。为了进一步提升SBP算法的性能，文献[10]通过按变量节点度数递减的顺序进行更新，来加快 SBP 算法的收敛速度，但是该方法只适用于非规则 LDPC 码，对于规则 LDPC 码并没有效果；文献[11,12]引入了残差幅度(Residual Magnitude, RM)的概念，每次迭代时选择 RM 最大的节点进行更新，以此来提高SBP算法的收敛速度和误码性能，但是由于该算法每次迭代都要计算 RM，计算复杂度相当巨大，在实际工程中并不适用。

鉴于此，本文针对 SBP 算法提出了一种交叠的 Shuffled-BP(OSBP)算法，该算法充分利用 SBP 算法的特点，将若干个相同的子译码器以不同的顺序同时进行译码。每次迭代后，保留更可靠的变量节点信息参与判决和下一次迭代，进而使得判决更可靠。最终提出的算法在不增加额外存储空间的条件下，使得 SBP 算法的误码性能和收敛速度均有明显的提升，并且该算法对于规则和非规则 LDPC 码均有效。

本文以下的结构如下：第 2 节介绍 SBP 译码算法以及该算法的特点；第 3 节在第 2 节的基础上引出了该文提出的交叠的 Shuffled-BP 算法，并详细介绍了子译码器偏移量和更新顺序的选取；第 4 节利用高斯近似的密度进化对 OSBP 算法的收敛性进行了理论分析;第5节分别对于规则和不规则LDPC码给出了相应的仿真结果；第 6 节对全文进行了总结。

## 2　Shuffled-BP(SBP)算法的基本原理

Zhang 和 Fossorier 在文献[6]中提出的 SBP 算法是概念层面的描述，并没有考虑其具体的实现方式。SBP 算法在实际应用中是一种基于 SISO 的迭代译码算法，本文以此方式对 SBP 算法进行描述，以便于解释后续本文的算法，同时这样的描述也可以给实际工程实现提供理论依据。

假设 $\boldsymbol{H}=\{h_{mn}\}$ 为 $M\times N$ 的校验矩阵，$\boldsymbol{c}=(c_1,c_2,\cdots,c_N)$ 为其对应的 LDPC 码编码后的码字，BPSK 调制后该码字映射为 $\boldsymbol{x}=(x_1,x_2,\cdots,x_N)$，其中 $x_n=1-2c_n$，$n=1,2,\cdots,N$。经过 AWGN 信道后，接收端接收到的码字序列为 $\boldsymbol{y}=(y_1,y_2,\cdots,y_N)$，$y_n=x_n+n_n$，其中 $n_n$ 是均值为零方差为 $N_0/2$ 的加性高斯白噪声，则初始信道对数似然比信息为

$$F_n=\ln\left\{\frac{P(x_n=1\mid y_n)}{P(x_n=-1\mid y_n)}\right\}=\frac{4y_n}{N_0} \tag{1}$$

在第 $i$ 次迭代中，令 $r_{mn}(i)$ 和 $q_{mn}(i)$ 分别表示校验节点 $m$ 传递给变量节点 $n$ 的 LLR 信息和变量节点 $n$ 传递给校验节点 $m$ 的 LLR 信息；$Q_n(i)$ 表示变量节点 $n$ 的对数似然比信息，用来做译码的软判决。为便于描述，在这里给出文献[13]中对数似然比的加减运算。

$$l_1\ {}_{[\pm]}\ l_2=\ln\left(\frac{1\pm e^{l_1}e^{l_2}}{e^{l_1}\pm e^{l_2}}\right) \tag{2}$$

$$\sum_i{}_{[\pm]}l_i=2\tanh^{-1}\left\{\left(\prod_i\tanh(l_i/2)\right)^{\pm1}\right\} \tag{3}$$

下面给出 SBP 算法的具体步骤：

初始化　对于任意的 $1\le m\le M$，$1\le n\le N$，若 $h_{mn}=1$，则令 $q_{mn}(0)=F_n$。

第 1 步　在第 $i$ 次迭代中，$i\ge1$，对于 $n=1,2,\cdots,N$，以及 $\forall m\in M(n)$，计算：

$$r_{mn}(i)=R_m\ {}_{[-]}\ q_{mn}(i-1) \tag{4}$$

$R_m$ 为上一列更新后的输出外信息，减去当前列上一次迭代后的信息 $q_{mn}(i-1)$ 便可得到 $r_{mn}(i)$。值得注意的是，在第 1 次迭代过程中更新第 1 列($n=1$)时，$R_m$ 取初始值：

$$R_m=\sum_{n\in N(m)}{}_{[+]}q_{mn}(0) \tag{5}$$

然后利用式(5)所得到的信息去更新 $q_{mn}(i)$ 和 $Q_n(i)$：

$$q_{mn}(i)=F_n+\sum_{m'\in M(n)\backslash m}r_{m'n}(i) \tag{6}$$

$$Q_n(i)=F_n+\sum_{m\in M(n)}r_{mn}(i) \tag{7}$$

最后更新输出外信息 $R_m$：

$$R_m=R_m\ {}_{[-]}\ q_{mn}(i-1)\ {}_{[+]}\ q_{mn}(i) \tag{8}$$

如果 $n<N$，则令 $n=n+1$，返回式(4)继续下一列的更新，此时更新后的 $R_m$ 则被用来更新下一列的变量节点 LLR 信息；若 $n=N$，则本次迭代结束，进入第 2 步。

第 2 步　令 $\hat{\boldsymbol{c}}=\{\hat{c}_n\}$ 为硬判决后的码字，若 $Q_n(i)<0$，则 $\hat{c}_n=1$，否则 $\hat{c}_n=0$。如果 $\boldsymbol{H}\hat{\boldsymbol{c}}^{\mathrm{T}}=\boldsymbol{0}$，或者已达到最大迭代次数，则译码结束，否则返回到第 1 步继续进行迭代。

在实际工程实现中，SBP 算法可以方便地进行并行处理：设并行度为 $g$，变量节点平均分成 $N/g$ 组，每组 $g$ 个节点，每组各节点并行更行，而不同组之间串行更新，此时在第 1 步中，对于 $b=1,2,\cdots,N/g$，每个 $(b-1)g+1\le n\le bg$，以及 $\forall m\in M(n)$，

式(8)应变为

$$R_m = R_{m[-]} \sum_{\substack{n\in N(m)\\(b-1)g+1\le n\le bg}} {}_{[-]}q_{mn}(i-1)_{[+]} \cdot \sum_{\substack{n\in N(m)\\(b-1)g+1\le n\le bg}} {}_{[+]}q_{mn}(i) \tag{9}$$

当 $g=1$ 时，就是串行处理的 SBP 算法，当 $g=N$ 时，就变成了原来的 BP 算法。

由于 SBP 算法是逐变量节点依次进行更新的，所以每列更新的时间不同。列更新的时间越晚，则有越多的更新后的信息参与到该列的更新中去，从而得到的变量节点信息更可靠。如果按 $n$=1,2,$\cdots$,$N$ 升序的顺序进行更新，则 $n$ 越大，对应的变量节点信息越可靠，错误率越低；相应地，如果按 $n=N$, $N-1$,$\cdots$,1 降序的顺序进行更新，则 $n$ 越大，对应的变量节点信息越不可靠，错误率越高。

图 1 表示了在信噪比为 3.0 dB 时，通过对 10000 帧(192,96)(3,6)规则 LDPC 码编码数据分别采用升序与降序 SBP 算法进行仿真，得到的不同比特位置的错误比特数。从图中可以看出越晚更新的变量节点，节点的错误比特数越少，其对应的信息越可靠，错误率越低。如果将多个相同的子译码器以不同的节点顺序同时进行迭代，每次迭代后，针对于每一个变量节点在各个子译码器中选取最可靠的信息进入到下一次迭代，从而提高判决的可靠性以及译码器的收敛速度。基于此，本文提出了交叠的 Shuffled-BP 算法。

## 3 交叠的 Shuffled-BP(OSBP)译码算法

正如第 2 节分析的那样，SBP 算法中变量节点信息的可靠性随更新时间增加而提升，为了充分利用可靠度更高的信息，利用多个子译码器以不同的更新顺序同时进行译码。在每次迭代过程中，各子译码器生成并相互之间传递更可靠的信息，从而提高判决的正确率和译码收敛速度，这就是 OSBP 算法的基本思想。

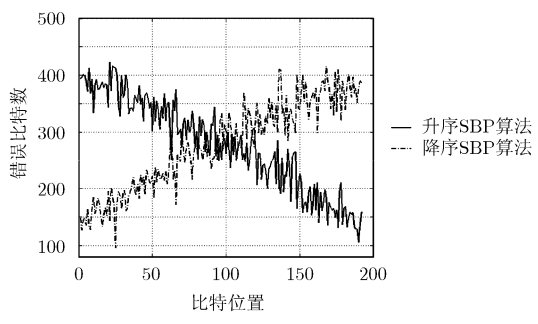



图 1 采用不同更新顺序的 SBP 算法下各比特位置的错误比特数

### 3.1 OSBP 算法

对于 OSBP 算法来说，设其包含的子译码器个数为 $D$，定义 $r_{mn}^d(i)$，$q_{mn}^d(i)$ 和 $Q_n^d(i)$ 分别表示第 $d$ 个子译码器相应的 LLR 信息。每个子译码器的迭代过程与 SBP 算法一致，不同的是每个子译码器列更新的顺序不同，令子译码器 $d$ 的更新顺序为 $S(d,n)$，并设对应于每个比特位置的错误率为 $E(d,n)$。下面给出 OSBP 算法的具体步骤：

初始化　对于每个子译码器 $d\in\{d\,|\,1\le d\le D\}$，若 $h_{m,S(d,n)}=1$，$1\le m\le M$，$1\le n\le N$，则令 $q_{m,S(d,n)}^d(0)=F_{S(d,n)}$。

第 1 步　在第 $i$ 次迭代中，$i\ge 1$，同时对 $D$ 个子译码器进行如下迭代：对于 $n=1,2,\cdots,N$，以及 $\forall m\in M(n)$，计算：

$$r_{m,S(d,n)}^d(i)=R_{m[-]}^d\, q_{m,S(d,n)}^d(i-1) \tag{10}$$

$R_m^d$ 为第 $d$ 个子译码器上一列更新后的输出外信息，减去当前列上一次迭代后的信息 $q_{m,S(d,n)}^d(i-1)$ 便可得到 $r_{m,S(d,n)}^d(i)$。值得注意的是，在第 1 次迭代过程中更新第 1 列( $S(d,1)$ )时，$R_m^d$ 取初始值：

$$R_m^d=\sum_{n\in N(m)} {}_{[+]}q_{mn}^d(0) \tag{11}$$

然后利用式(11)所得到的信息去更新 $q_{m,S(d,n)}^d(i)$ 和 $Q_{S(d,n)}^d(i)$：

$$q_{m,S(d,n)}^d(i)=F_{S(d,n)}+\sum_{m'\in M(S(d,n))\backslash m} r_{m',S(d,n)}^d(i) \tag{12}$$

$$Q_{S(d,n)}^d(i)=F_{S(d,n)}+\sum_{m\in M(S(d,n))} r_{m,S(d,n)}^d(i) \tag{13}$$

最后更新输出外信息 $R_m^d$：

$$R_m^d=R_m^d\ {}_{[-]}\ q_{m,S(d,n)}^d(i-1)\ {}_{[+]}\ q_{m,S(d,n)}^d(i) \tag{14}$$

这里需要注意各信息下标中列的顺序为 $S(d,n)$，即对于 $n=1,2,\cdots,N$，按照 $S(d,n)$ 的顺序对各列依次进行更新。如果 $n<N$，则令 $n=n+1$，返回式(10)继续下一列的更新；若 $n=N$，则本次迭代结束，进入第 2 步。

第 2 步　对于每个变量节点 $1\le n\le N$，令 $q_{mn}(i)=q_{mn}^{dt_n}(i)$，$Q_n(i)=Q_n^{dt_n}(i)$，其中 $dt_n$ 表示使第 $n$ 个比特错误率最低的子译码器编号，即

$$dt_n=\arg\min_{1\le d\le D}\{E(d,n)\} \tag{15}$$

需要说明的是，对于实际给定的各子译码器的更新顺序 $S(d,n)$ 以及错误率 $E(d,n)$，$dt_n$ 可以事先计算出来，并不需要在每次迭代过程中都进行计算。

然后对于每个子译码器 $d$，令

$$R_m^d=\sum_{n\in N(m)} {}_{[+]}q_{mn}(i),\ q_{mn}^d(i)=q_{mn}(i) \tag{16}$$

第 3 步　令 $\hat{\boldsymbol{c}}=\{\hat{c}_n\}$ 为硬判决后的码字，若

$Q_n(i)<0$，则 $\hat{c}_n=1$，否则 $\hat{c}_n=0$。如果 $\boldsymbol{H}\hat{\boldsymbol{c}}^{\mathrm{T}}=\boldsymbol{0}$，或者已达到最大迭代次数，则译码结束，否则返回到第 1 步继续进行迭代。

从第 2 步可以看出，不同于 SBP 算法，OSBP 算法在每次迭代后只保留了最可靠的变量节点信息参与下一次迭代，所以得到了更好的性能。另外在每次迭代后，只需存储最可靠的信息 $q_{mn}(i)$，而 $Q_n(i)$ 可以方便地由 $q_{mn}(i)$ 得出，并不需要存储每个子译码器的变量节点信息 $q_{mn}^{d}(i)$，因此相比于 SBP 算法而言，并不需要额外的存储空间。

每个子译码器以不同的更新顺序同时进行迭代，每个变量节点都被各子译码器在不同的时间进行更新。正如 3.2 节里图 2 将要展示的那样，各子译码器的更新交叠在一起，相互之间传递更可靠的信息，故这里将该算法称之为交叠的 Shuffled-BP 算法。

与 SBP 算法一样，在实际工程实现中 OSBP 算法也可以方便地进行并行处理：设并行度为 $g$，变量节点平均分成 $N/g$ 组，每组 $g$ 个节点，对于每个子译码器而言，每组各节点并行更新，而不同组之间串行更新，此时在第 1 步中，当 $b=1,2,\cdots,N/g$ 时，对于每个子译码器 $d\in\{d\,|\,1\le d\le D\}$，每个 $(b-1)g+1\le n\le bg$，以及 $\forall m\in M(n)$，式(14)应变为

$$R_m^d=R_{m[-]}^d\sum_{\substack{S(d,n)\in N(m)\\(b-1)g+1\le n\le bg}}{}_{[-]}q_{m,S(d,n)}^d(i-1)_{[+]}\cdot\sum_{\substack{S(d,n)\in N(m)\\(b-1)g+1\le n\le bg}}{}_{[+]}q_{m,S(d,n)}^d(i)\tag{17}$$

### 3.2 最大化译码性能的更新顺序选取

每个子译码器 $d$ 的更新顺序由两个参数决定：偏移量和更新方向，其中偏移量是指译码器开始更新的位置与位置 1(最左端的列)之间的偏移；更新方向分为前向更新与后向更新，前向更新是指向右循环递增地进行更新，后向更新是指向左循环递减地更新，设 $\mathrm{offset}(d)$ 表示子译码器 $d$ 的偏移量，对于前向和后向更新，当 $n=1,2,\cdots,N$ 时，其更新顺序分别为

$$\mathrm{forward}(d)=[\mathrm{offset}(d)+n-1]\bmod N+1\tag{18}$$

$$\mathrm{backward}(d)=[\mathrm{offset}(d)-n+1]\bmod N+1\tag{19}$$

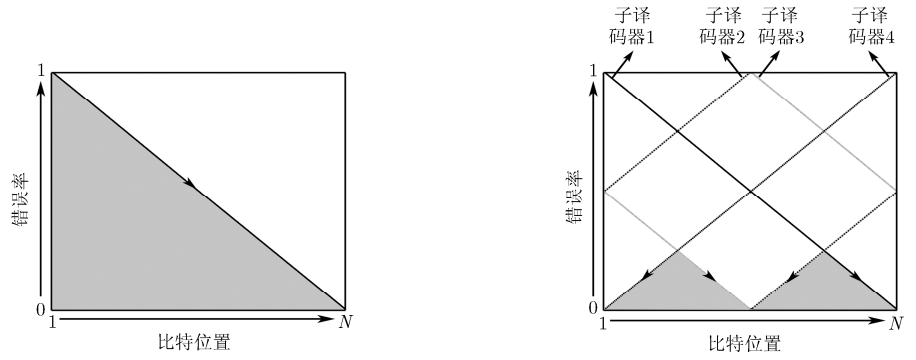
根据第 2 节的分析可知，越晚被更新的节点对应的错误率越低，所以不失一般性，这里将前向与后向更新顺序的错误率表示为如式(20)，式(21)的一种线性关系：

$$\mathrm{errforward}(d)=\{[\mathrm{offset}(d)-n]\bmod N\}/N\tag{20}$$

$$\mathrm{errbackward}(d)=\{[N-\mathrm{offset}(d)+n-2]\bmod N\}/N\tag{21}$$

由此可见，各个子译码器的更新顺序直接影响 OSBP 算法的性能，为了最大化译码性能，将 $N$ 平均分成 $2^k$ 份，$k\ge 0$，并沿着变量节点等间隔地布置子译码器，每个等间隔点前后分别布置两个子译码器，子译码器的个数 $D=2^{k+1}$。在等间隔点 $(2N/D)l$ 处，$0\le l\le D/2-1$，两个子译码器的偏移量分别为 $(2N/D)l$ 和 $[(2N/D)l-1]\bmod N$，对应的更新方向分别为前向和后向更新，然后利用式(18)～式(21)就可以得出各子译码器的更新顺序 $S(d,n)$ 和错误率 $E(d,n)$。

图 2 表示了一次迭代中 SBP 算法与 OSBP 各子译码器更新过程的示意图：纵轴表示错误率，横轴表示节点位置。斜线上的箭头指示更新的方向，按箭头所指的方向进行更新，各斜线所对应横坐标的位置即为各子译码器更新的顺序。图中设子译码器的个数 $D=4$，在 OSBP 算法在每次迭代过程中，当 $n>3N/4$ 时，各子译码器开始存储当前的信息 $q_{m,S(d,n)}^d(i)$，即对应于图 2(b)中的阴影部分；而 SBP 算法在每次迭代中存储的信息对应于图 2(a)中的阴影部分，可见 OSBP 算法在每次迭代后错误率更低。




(a) SBP算法译码器更新过程　　(b) $D$=4时OSBP算法各子译码器更新过程

图 2　SBP 算法与 OSBP 各子译码器更新过程对比示意图

## 4 基于密度进化的收敛性分析

本节将采用高斯近似的密度进化理论，针对SBP和OSBP译码算法推导了其对应的消息均值进化，并基于该理论定量地分析了SBP算法和OSBP算法的收敛性。

密度进化(DE)是分析基于消息传递的迭代译码算法性能有效的理论工具，如果信道和译码算法均满足对称性条件，那么误码率与所传输的码字无关，这样可以假设传输码字均为0，以大大减化DE算法的复杂度。然而即便如此，由于需要精确计算节点消息的概率密度，DE算法的计算量仍然非常巨大，文献[14]提出了高斯近似的方法，将多维节点消息的DE简化为1维均值的进化，大大降低了DE的计算量，并且由其计算出来的译码门限与精确的DE所得到的结果仅仅相差0.3%~1.2%，保证了分析的有效性。

不同于BP译码算法，在SBP算法下，节点的概率密度与比特位置有关。令 $v_l^{(i)}$ 和 $u_l^{(i)}$ 分别为第 $i$ 次迭代中对应于第 $l$ 个比特位置的变量消息密度和校验消息密度的均值，则对于 $(dv,dc)$ 规则LDPC码来说(其中 $dv$ 与 $dc$ 分别为变量节点与校验节点的度数)，第 $i$ 次迭代时变量消息的均值进化与BP译码算法下的一致，表示为

$$v_l^{(i)} = v_0 + (dv-1)u_l^{(i)} \tag{22}$$

其中，变量消息的初始均值 $v_0 = 4/N_0$。由式(8)可以看出，校验消息均值 $u_l^{(i)}$ 不仅与更新后的 $v_{l'}^{(i)}, l'<l$ 有关，同时也依赖于更新前的 $v_{l'}^{(i-1)}, l'>l$，为了避免计算所有 $v_{l'}^{(i)}$ 和 $v_{l'}^{(i-1)}$ 可能的组合，这里分别对更新前后的变量消息均值取平均，令 $v_{l'}^{(i)}, l'<l$ 的均值为

$$\overline{v_{l'<l}^{(i)}} = \frac{1}{l-1}\sum_{l'=1}^{l-1} v_{l'}^{(i)} \tag{23}$$

同理，令 $v_{l'}^{(i-1)}, l'>l$ 的均值为

$$\overline{v_{l'>l}^{(i-1)}} = \frac{1}{N-l}\sum_{l'=l+1}^{N} v_{l'}^{(i-1)} \tag{24}$$

参与校验消息均值进化的变量消息均值的所有可能的组合数为

$$C_A(dc) = \binom{N-1}{dc-1} \tag{25}$$

而在所有的这些组合当中，有 $j$ 个更新后的变量消息均值和 $dc-1-j$ 个更新前的变量消息均值的组合数为

$$C(j,dc) = \binom{l-1}{j}\binom{N-l}{dc-1-j} \tag{26}$$

那么在SBP译码算法下，校验消息均值的进化可以表示为

$$u_l^{(i)} = \sum_{j=0}^{dc-1} \frac{C(j,dc)}{C_A(dc)} \phi^{-1}\left(1-\left[1-\phi\left(\overline{v_{l'<l}^{(i)}}\right)\right]^j \cdot\left[1-\phi\left(\overline{v_{l'>l}^{(i-1)}}\right)\right]^{dc-1-j}\right) \tag{27}$$

其中，$\phi(x)$ 可以通过式(28)近似计算得到[14]：

$$\phi(x)=\begin{cases}1, & x=0\\ \exp\left(-0.4527x^{0.86}+0.0218\right), & 0<x<10\\ \sqrt{\dfrac{\pi}{x}}\left(1-\dfrac{10}{7x}\right)\exp\left(-\dfrac{x}{4}\right), & x\ge 10\end{cases} \tag{28}$$

根据第 $i$ 次迭代后的变量消息的概率分布 $N(v_l^{(i)}, 2v_l^{(i)})$，得到错误率为

$$Pe^{(i)} = \mathrm{erfc}\left(\frac{1}{N}\sum_{l=1}^{N} v_l^{(i)} \Bigg/ \left(2\sqrt{\frac{1}{N}\sum_{l=1}^{N} v_l^{(i)}}\right)\right) \tag{29}$$

由SBP译码算法下的消息均值进化规则可以方便地得到OSBP算法的消息均值进化规则。由式(10)~式(16)可以看出，OSBP算法中变量消息均值的进化与SBP算法的一致，如式(22)所示，不同之处在于校验消息均值的进化。由于在OSBP算法中，每次迭代时只保留了对应每个比特位置最可靠子译码器的输出外信息，所以变量消息的均值进化需要加入如式(30)过程：

$$v^{(i)}_{l+\frac{N}{D}(d-1)} = v^{(i)}_{(-1)^{d \bmod 2} l + \frac{N}{D} d \bmod 2 + \frac{D-1}{D}N}, \quad 1 \le l \le \frac{N}{D}, \quad 1 \le d \le D \tag{30}$$

对于度分布为 $\lambda(x)=\sum_{k=1}^{dv}\lambda_k x^{k-1}$ 和 $\rho(x)=\sum_{k=1}^{dc}\rho_k x^{k-1}$ 的非规则LDPC码，也可以容易地得到SBP算法与OSBP算法下的消息均值进化。在SBP算法下，第 $i$ 次迭代过程中度数为 $k$ 的变量消息均值进化可以表示为

$$v_{l,k}^{(i)} = v_0 + (k-1)u_l^{(i)} \tag{31}$$

此时，$v_l^{(i)}$ 变为变量消息均值对于不同度数的平均：

$$v_l^{(i)} = \sum_{k=1}^{dv} \lambda_k v_{l,k}^{(i)} \tag{32}$$

令

$$\overline{\psi_{l'<l}^{(i)}} = \frac{1}{l-1}\sum_{l'=1}^{l-1}\sum_{k=1}^{dv}\lambda_k\phi\left(v_{l',k}^{(i)}\right) \tag{33}$$

$$\overline{\psi_{l'>l}^{(i-1)}} = \frac{1}{N-l}\sum_{l'=l+1}^{N}\sum_{k=1}^{dv}\lambda_k\phi\left(v_{l',k}^{(i-1)}\right) \tag{34}$$

那么校验消息的均值进化可以表示为

$$u_l^{(i)}=\sum_{k=1}^{dc}\rho_k\sum_{j=0}^{k-1}\frac{C(j,k)}{C_A(k)}\phi^{-1}\left(1-\left[1-\overline{\psi_{l'<l}^{(i)}}\right]^j\cdot\left[1-\overline{\psi_{l'>l}^{(i-1)}}\right]^{k-1-j}\right)\tag{35}$$

这里 $u_l^{(i)}$ 为校验消息均值对于不同度数的平均。只需要在此进化过程中加入式(30)即可得到在OSBP 算法下的消息均值进化。

图 3 描绘了通过密度进化分析所得到的SBP 算法和 OSBP 算法在不同迭代次数下的误比特率。图 3(a)采用码率为 1/2 的(3,6)规则 LDPC 码，信噪比为 1.2 dB；图 3(b)采用码率为 1/2 的非规则 LDPC 码，最大变量节点度数为 5，其度分布与文献[15]中的一致，信噪比为 0.9 dB。由图 3 可知，无论对于规则还是非规则 LDPC 码，OSBP 算法的收敛速度均快于 SBP 算法，并且子译码器个数 $D$ 越大，OSBP 算法的收敛速度越快；在相同的迭代次数下，OSBP 算法的误比特率低于 SBP 算法，并且误比特率随 $D$ 的增大而减小。

密度进化分析的前提是码长无限长、校验矩阵中的'1'随机分布，故其得到的具体的迭代次数对于实际有限长的 LDPC 码来说是一个相对值，并不能反映实际所需的迭代次数。为了得到子译码器个数 $D$ 与译码延时的关系，这里将 OSBP 算法的迭代次数相对于 SBP 算法下的迭代次数进行归一化：令 $I_D$ 表示子译码器个数为 $D$ 时 OSBP 算法所需要的迭代次数，$I_{\mathrm{SBP}}$ 表示 SBP 算法所需要的迭代次数，则 OSBP 算法的译码延时为

$$T_D=\frac{I_D}{I_{\mathrm{SBP}}}T_{\mathrm{SBP}}\tag{36}$$

其中 $T_{\mathrm{SBP}}$ 为 SBP 算法的译码延时。对于特定的误码率，$I_D$ 与 $I_{\mathrm{SBP}}$ 的比值可以由之前密度进化分析得到的误码曲线图中看出，例如图 3，如此就可以间接地得到子译码器个数 $D$ 与译码延时的关系。由图 3 可知，$D$ 越大，$I_D$ 与 $I_{\mathrm{SBP}}$ 的比值就越小，因而 OSBP 算法相对于 SBP 算法的译码延时就越小。

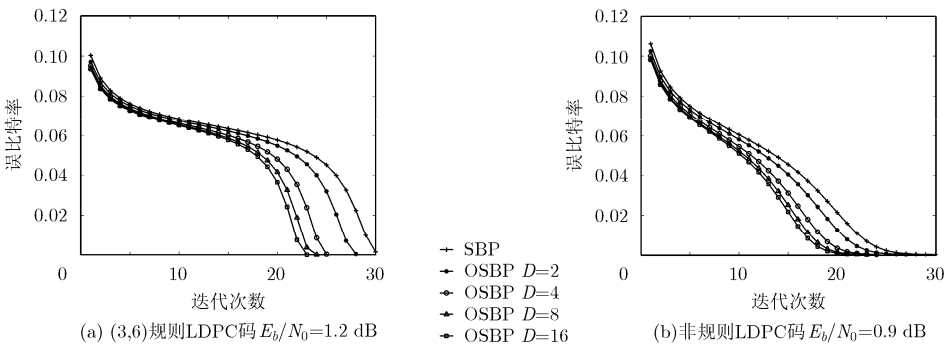
## 5　仿真结果及分析

为了验证本文所提出 OSBP 算法的性能，本文以两种码率为 1/2 的 LDPC 码为例：(1)文献[16]中准循环构造算法构造的环长为 8、码长为 192 的(3,6)规则 LDPC 码，这里称之为 C1 码；(2)文献[15]中变量节点度分布为 $\lambda(x)$=0.32660$x$+0.11960$x^2$+0.18393$x^3$+0.36988$x^4$，并采用文献[17]中改进的 PEG 算法构造的码长为 1024 的非规则 LDPC 码，这里称之为 C2 码。在 AWGN 和 BPSK 调制的条件下，采用蒙特卡洛仿方法，对 OSBP 算法与 SBP 算法的性能做了对比仿真实验。

图 4 分别表示了 C1 码和 C2 码在采用 OSBP 算法和 SBP 算法下的误比特率。其中 C1 码的信噪比为 3.0 dB; C2 码的信噪比为 2.0 dB; OSBP 算法的子译码器个数分别取 2, 4, 8。由图 4 可知，在相同的迭代次数和信噪比的条件下，本文算法的误比特率相比于 SBP 算法降低了一半以上，并且当 $D$ 增加时，误比特率会进一步有小幅度的降低，且 $D$ 越大，降低的幅度越小。

图 5 分别表示了 C1 码和 C2 码在采用两种译码算法下的误码曲线。其中 OSBP 算法的子译码器个数为 4。由图 5 可以看出在相同的最大迭代次数下，相比于 SBP 算法，本文提出的 OSBP 算法有着更优的误码性能，最大迭代次数越少，这种优势越明显。对于 C1 码而言，在误比特率等于$10^{-3}$的条件下，误码性能提升了大约 0.3 dB。对于 C1 码和 C2 码来说，当最大迭代次数分别大于 5 次和 10 次的条件下，本文提出的 OSBP 算法仍然保持有略微的优势。

图 6 表示当子译码器个数 $D$ 分别为 2, 4, 8 时的归一化平均迭代次数：设 OSBP 算法的平均迭代次数为 $I_{\mathrm{OSBP}}$，SBP 算法的平均迭代次数为 $I_{\mathrm{SBP}}$，则归一化平均迭代次数可以表示为 $I_{\mathrm{OSBP}}/I_{\mathrm{SBP}}$，该值越小，则收敛速度越快。译码时设最大迭代次数为 10 次，由图 6 可以看出，相比于 SBP 算法，本文提出的算法有着更快的收敛速度，并且 $D$ 越大，收敛




(a) (3,6)规则LDPC码 $E_b/N_0$=1.2 dB　　(b)非规则LDPC码 $E_b/N_0$=0.9 dB

图 3　通过密度进化分析得到的不同迭代次数下的误比特率

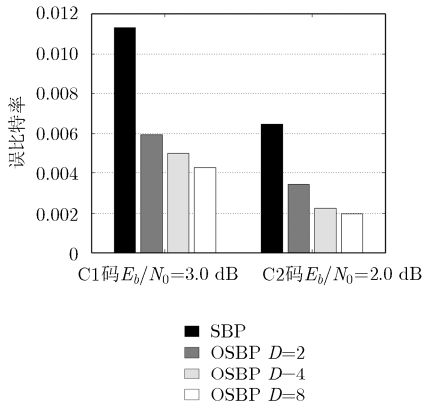



图 4 OSBP 算法与 SBP 算法在特定信噪比下的误比特率

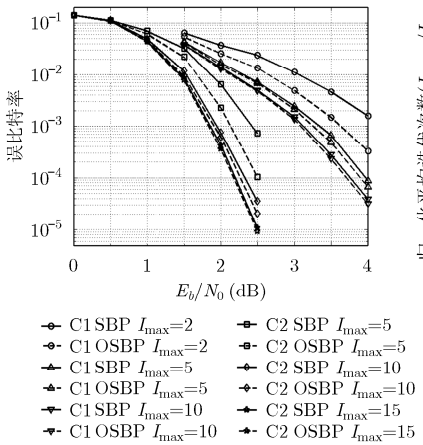



图 5 OSBP 算法与 SBP 算法误码性能的比较仿真

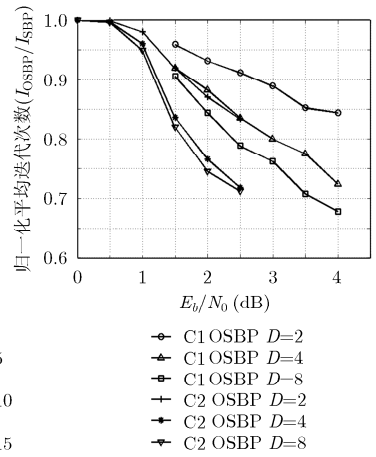



图 6 OSBP 算法与 SBP 算法平均迭代次数的比较仿真

速度越快。具体地讲，对于 C1 码来说，在信噪比为 4.0 dB 的条件下，OSBP 算法的收敛速度比 SBP 算法提高了约 32%；对于 C2 码来说，在信噪比为 2.5 dB 的条件下，OSBP 算法的收敛速度比 SBP 算法提高了约 29%。另外在信噪比区间 (1.5 ∼ 2.5 dB) 下，通过对比两种不同码长 LDPC 码，发现在 $D$ 相同的条件下码长越长 OSBP 算法的收敛速度越快。

## 6 结束语

本文为了进一步提升 Shuffled-BP(SBP)算法的性能提出了一种交叠的 Shuffled-BP(OSBP)迭代译码算法，通过充分利用 SBP 算法中变量节点信息的可靠性随更新时间增加而提升的特点，采用若干个相同的子译码器同时交叠地对 LDPC 码进行更新迭代，相互之间生成并传递更可靠的信息，从而进一步提高 SBP 算法的收敛速度和误码性能。基于密度进化理论对提出的算法进行了分析，证明了 OSBP 算法的误码性能和收敛速度均优于 SBP 算法。仿真实验表明，本文的算法对于规则 LDPC 码和非规则 LDPC 码均有效，在不增加额外存储空间的条件下，OSBP 算法的误码性能优于 SBP，并且其误比特率随着子译码器个数的增加而降低，尤其在中高信噪比区间下，误码性能较 SBP 算法提升约 1 倍。除此之外，相比于 SBP 算法，OSBP 算法有着更快的收敛速度，并且其收敛速度随着码长和子译码器个数的增加而增加，所以在实际工程应用中，OSBP 算法可以带来更大的编码增益和更低的译码延时。

## 参 考 文 献

[1] SHANNON C E. A mathematical theory of communication[J]. *ACM SIGMOBILE Mobile Computing and Communications Review*, 2001, 5(1): 3–55.

[2] MACKAY D J C and NEAL R M. Near Shannon limit performance of low density parity check codes[J]. *Electronics Letters*, 1996, 32(18): 1645–1646.

[3] GALLAGER R G. Low-density parity-check codes[J]. *IRE Transactions on Information Theory*, 1962, 8(1): 21–28.

[4] HOCEVAR, D. A reduced complexity decoder architecture via layered decoding of LDPC codes[C]. IEEE Workshop on Signal Processing Systems (SIPS), Austin, TX, USA, 2004: 107–112.

[5] ZHANG Xinmiao and TAI Ying. High-speed multi-block-row layered decoding for Quasi-cyclic LDPC codes[C]. IEEE Global Conference on Signal and Information Processing (GlobalSIP), Atlanta, GA, USA, 2014: 11–14.

[6] ZHANG J and FOSSORIER M. Shuffled belief propagation decoding[C]. Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 2002, 1: 8–15.

[7] WU Sheng, JIANG Xiaobo, and NIE Zhenghua. Alternate iteration of shuffled belief propagation decoding[C]. International Conference on Communications and Mobile Computing (CMC), Shenzhen, China, 2010, 2: 278–281.

[8] LAOUINI N, BEN Hadj Slama L, and BOUALLEGUE A. An optimized min-sum variable node layering for LDPC decoding[C]. International Conference on Multimedia Computing and Systems (ICMCS), Marrakech, Morocco, 2014: 794–799.

[9] SUN Yang and CAVALLARO J R. VLSI architecture for layered decoding of QC-LDPC codes with high circulant weight[J]. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2013, 21(10): 1960–1964.

[10] ASLAM C A, GUAN Yongliang, and CAI Kui. Improving the belief-propagation convergence of irregular LDPC codes

using column-weight based scheduling[J]. *IEEE Communications Letters*, 2015, 19(8): 1283–1286.

[11] LIU Xingcheng, ZHANG Yuanbin, and RU Cui. Variable-node-based dynamic scheduling strategy for belief-propagation decoding of LDPC codes[J]. *IEEE Communications Letters*, 2015, 19(2): 147–150.

[12] LI Jia, YANG Gaigai, and ZHAO Zhiqiang. An improved-performance decoding algorithm of LDPC codes for layered decoding[C]. IEEE International Conference on Communication Problem-Solving (ICCP), Beijing, China, 2014: 318–321.

[13] HAGENAUER J, OFFER E, and PAPKE L. Iterative decoding of binary block and convolutional codes[J]. *IEEE Transactions on Information Theory*, 1996, 42(2): 429–445.

[14] CHUNG Saeyoung, 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.

[15] RICHARDSON T J, SHOKROLLAHI M A, and URBANKE R L. Design of capacity-approaching irregular low-density parity-check codes[J]. *IEEE Transactions on Information Theory*, 2001, 47(2): 619–637.

[16] ZHANG Yi and DA Xinyu. Construction of girth-eight QC-LDPC codes from arithmetic progression sequence with large column weight[J]. *Electronics Letters*, 2015, 51(16): 1257–1259.

[17] JIANG Xueqin, XIA Xianggen, and LEE Moonho. Efficient progressive edge-growth algorithm based on Chinese remainder rheorem[J]. *IEEE Transactions on Communications*, 2014, 62(2): 442–451.

范亚楠：男，1991 年生，博士生，研究方向为星间通信系统中的信道编译码技术.

王丽冲：女，1990 年生，硕士生，研究方向为分布式卫星组网建模与仿真.

姚秀娟：女，1977 年生，研究员，研究方向为卫星组网通信与模式识别技术.