中国科学: 信息科学 2021 年 第 51 卷 第 8 期: 1331–1344
SCIENTIA SINICA Informationis
论文

Reed-Solomon 码概率软译码算法增强技术

孙怡宁*, 黄秋, 胡剑浩

电子科技大学通信抗干扰技术国家级重点实验室, 成都 611731
* 通信作者. E-mail: ynsun3842@163.com

收稿日期: 2020–05–07; 修回日期: 2020–09–29; 接受日期: 2020–12–07; 网络出版日期: 2021–08–03

国家重点研发计划项目 (批准号: 2018YFB1801500) 资助项目

摘要 基于概率计算的 Chase 算法可以在保证译码性能的情况下, 大幅度降低软译码的复杂度, 使得 RS 软译码算法得到实际应用. 然而, 概率 Chase 算法的时间复杂度会随着测试码字个数的增加而增长; 在高阶调制下, SSCA 算法产生测试码字时的搜索范围会随着调制阶数的增加呈指数增长, 从而增大了硬件存储开销. 为降低时间复杂度, 本文提出了提前输出算法, 不需要产生所有的测试码字即可实现成功译码. 仿真结果表明该算法可以在逼近原算法误帧率性能的同时将时间复杂度最多降低为原来的近 $1/\tau$ (τ 为测试码字个数). 针对 SSCA 算法存储的压力, 本文通过确定搜索半径辅助的选择方法来减小搜索范围. 仿真结果表明, 提出的 3σ-SSCA 算法仅以少量性能损失为代价可将测试码字单个符号的搜索范围由 q (q 为调制阶数) 个缩小为个位数.

关键词 RS 码, Chase 算法, 概率计算, 提前输出算法, 搜索半径辅助的选择方法

1 引言

Reed-Solomon (RS) 码是由 I. S. Reed 和 G. Solomom 提出的. Berlekamp Massey (BM) [1] 算法和 Euclidean [2] 算法是常见的硬判决译码算法, 并且已在硬件中被广泛地应用, 应用领域包括空间通信、移动通信、军用通信、光纤通信、磁盘阵列及光存储等.

RS 码可以看成是 BCH 码的非二进制形式, 具有优异的纠正随机错误和突发错误的能力. RS 码的译码算法可分为硬判决译码和软判决译码两大类. 硬判决译码理论成熟, 算法简单, 易于硬件实现, 译码时间短, 相比软判决译码其在低复杂度和高速通信等领域具有更广泛的实际应用. 然而, 硬判决译码增益没有软判决高, 因为软判决译码充分利用了信道信息, 使译码器以更大的概率判决出能正确译码的码字. 研究表明, 在加性高斯白噪声 (additive white Gaussian noise, AWGN) 信道中软判决的增益一般比硬判决高 2 ∼ 3 dB; 在衰落信道中可以高于 5 dB.

引用格式: 孙怡宁, 黄秋, 胡剑浩. Reed-Solomon 码概率软译码算法增强技术. 中国科学: 信息科学, 2021, 51: 1331–1344, doi: 10.1360/SSI-2020-0124
Sun Y N, Huang Q, Hu J H. Enhanced stochastic soft decoding algorithm for Reed-Solomon codes (in Chinese). Sci Sin Inform, 2021, 51: 1331–1344, doi: 10.1360/SSI-2020-0124

RS 软判决译码算法面临的主要问题是实现复杂度高, 为了获得接近最大似然 (ML) 译码的性能, 其译码复杂度呈指数增长, 在工程中难以实现. 近年来, 学者们对已有的软译码算法提出了一些改进. 对于 Koetter-Vardy (KV) 算法计算复杂度高的问题, Xing 等 [3] 利用模块最小化 (MM) 插值技术提出了低复杂度 KV 译码算法 (KV-MM). Lee 等 [4] 结合适应性奇偶校验矩阵和已知动态调度译码的概念提出了一种基于置信度传播的双投票残差 BP (double-polling residual BP, DP-RBP) 算法. 为了充分利用 RS 码纠正突发错误的能力, Zhang 等 [5] 结合突发信道特性充分利用信道信息提出了一种新颖的基于突发错误校正算法的有序统计译码算法 (burst-error correcting algorithm-based ordered-statistics decoding, BC-OSD). 为了降低 Chase 算法 [6] 的复杂度, Leroux 等 [7] 提出了基于概率计算的比特级概率 Chase 算法 (bit-wise stochastic Chase algorithm, BSCA). 该算法将接收信号的对数似然比 (Log-likelihood ratio, LLR) 转换为概率, 仅用一个随机数产生器和比较器产生测试码字, 大大简化了 Chase 算法的复杂度. 根据 RS 码的多元性及其在高阶调制下的译码特性, Hossein 提出了一种基于符号的概率 Chase 算法 (symbol-level stochastic Chase algorithm, SSCA) [8]. 与前者不同的是, 其测试码字是直接由符号产生的, 减少了二进制与多进制的转换, 使得概率算法在高阶调制中获得更高的增益. 然而, 上述两种概率译码算法在获得软判决输出时需要产生所有的测试码字, 译码性能会随测试码字个数增加而提高, 但是计算复杂度和译码时延也会随之增加, 针对这一问题, 我们提出了提前输出算法, 不需要产生所有既定数量的测试码字即可输出正确译码结果, 大大降低了计算复杂度和译码时延. 另外, 我们发现在高阶调制下, SSCA 算法的测试码字搜索范围会随着调制阶数呈指数增大, 这在硬件实现中会占用很多存储空间, 基于此, 我们利用高斯分布的 3σ 性质, 提出基于搜索半径的码字选择方法. 在保证误帧率性能损失在可接受范围内的前提下, 我们提出的 3σ 算法将测试码字单个符号的搜索范围降低到个位数, 大大节省了存储空间.

本文结构如下: 第 2 节简要介绍了 RS 码的编译码系统以及基于 Chase 算法的概率译码的基本原理, 第 3 节提出了两种针对概率译码的增强算法, 第 4 节展示了仿真结果并对其进行分析, 第 5 节对本文工作做了总结与展望.

2 概率 Chase 算法

2.1 可信度矩阵和后验概率向量

RS 码字可以用二进制相移键控 (BPSK) 或者 q 阶调制 (M-PSK 或 M-QAM) 在信道中传输. 在接收机端, 软信息的计算取决于调制方案. 在 2^m 阶有限域 $\mathrm{GF}(2^m)=\{0,1,2,\ldots,2^m-1\}$ 上定义一个 (n,k,d) RS 码, 其中 $m>0$. 该码由 k 符号信息位 $\boldsymbol{U}=(u_0,u_1,\ldots,u_{k-1})$ 通过增加 $n-k$ 个冗余符号编码产生 n 符号长的码字 $\boldsymbol{C}=(c_0,c_1,\ldots,c_{n-1})$, 每个符号可由 m 比特表示 $c_s=(x_0,x_1,\ldots,x_{m-1})_2$, $s=0,1,\ldots,n-1$. 码的最小距离为 d 个符号. 以 BPSK 调制为例, 假设码字 x_i 映射为符号 $x_i'=(-1)^{x_i}$, $i=0,1,\ldots,nm-1$. 这些信号通过噪声方差为 σ^2 的 AWGN 信道到达接收端, 接收信号为 $\boldsymbol{Y}=(Y_0,Y_1,\ldots,Y_{n-1})=(y_0,y_1,\ldots,y_{nm-1})_2$. 在信源等概分布时, 后验概率与先验概率相等, 在概率域和对数似然域每个比特的软信息可以分别表示为

$$p_i=\Pr\{x_i=1|y_i\}=\frac{1}{1+\exp(\frac{2y_i}{\sigma^2})}, \tag{1}$$

$$r_i=\frac{2y_i}{\sigma^2}. \tag{2}$$

利用对数似然域的软信息 r_i 可以得到接收序列每一比特的硬判决:

$$y_i^H = \begin{cases} 1, & \text{if } r_i < 0, \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

其中 $i = 0, 1, \ldots, nm-1$. 当接收到信号 y_i 时, 由式 (1) 可以得到判定该信号发送为 1 的概率, 即 y_i 确定时, $\Pr\{x_i = 1|y_i\}$ 为具体的概率值, 此时由式 (1)∼(3) 可得 $\Pr\{y_i^H = 1\} = \Pr\{x_i = 1|y_i\}$, 所以在二元域有 $\Pr\{y_i^H = 0\} = 1 - \Pr\{y_i^H = 1\}$. 比特后验概率向量 (APP) 为 $\boldsymbol{p} = (p_0, p_1, \ldots, p_{nm-1})$, 那么符号后验概率向量可表示为 $\boldsymbol{P} = (\boldsymbol{P}_0, \boldsymbol{P}_1, \ldots, \boldsymbol{P}_{n-1})$, 其中 $\boldsymbol{P}_i$ 由对应的 m 个比特概率表示 $\boldsymbol{P}_i = (p_{i,0}, p_{i,1}, \ldots, p_{i,m-1})$, 对应的硬判决接收符号的二进制表示为 $\boldsymbol{Y}_i^H = (y_{i,0}^H, y_{i,1}^H, \ldots, y_{i,m-1}^H)_2$. 现在我们可以给每个子向量 $\boldsymbol{P}_i$ 分配一个 Multinoulli 分布, 我们用代数表示 Multinoulli 分布的硬估计 $\boldsymbol{Y}_i^H$ [8], 即后验概率矩阵 $\boldsymbol{\Pi}$ 的每一个元素 π_{ij}:

$$\pi_{ij} = \Pr\left\{y_{i,0}^H = k_0, y_{i,1}^H = k_1, \ldots, y_{i,m-1}^H = k_{m-1}\right\} = \prod_{l=0}^{m-1} \Pr\left\{y_{i,l}^H = k_l\right\} = \prod_{l=0}^{m-1} \Pr\left\{x_{i,l} = k_l|y_{i,l}\right\},$$
$$i = 0, 1, \ldots, n-1, \quad j = 0, 1, \ldots, 2^m - 1, \tag{4}$$

其中 $k_l \in \{0, 1\}$, $l = 0, 1, \ldots, m-1$. 上式表示接收序列中的第 i 个接收符号等于 GF(2^m) 中的第 j 个元素的概率.

一般选择与伽罗华域 (Galois field) 相同阶数的调制阶数, 即 $q = 2^m$. 对于 q 进制调制, 以基于欧氏距离的方法产生 APP 矩阵以简化计算. 设 $s_j \in \mathcal{S}$, 其中 $\mathcal{S}$ 为 q 进制调制的标准星座点集, $j \in \{0, 1, 2, \ldots, q-1\}$. 在发送端, 编码符号 $\boldsymbol{C} = (c_0, c_1, \ldots, c_{n-1})$ 由调制器映射为对应星座点后发送, 通过噪声方差为 σ^2 的 AWGN 信道到达接收器, 接收信号为 $\boldsymbol{Y} = (Y_0, Y_1, \ldots, Y_{n-1})$. 根据接收信号与星座图内所有标准星座点的欧式距离可以计算得到置信度:

$$\hat{\pi}_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\|Y_i - s_j\|^2}{2\sigma^2}\right), \quad i = 0, 1, \ldots, n-1, \ s_j \in \mathcal{S}, \tag{5}$$

归一化后得到 APP 矩阵的每个元素:

$$\pi_{ij} = \frac{\hat{\pi}_{ij}}{\sum_j \hat{\pi}_{ij}}, \tag{6}$$

其中 $\|Y_i - s_j\|$ 为接收符号 Y_i 与星座点 s_j 之间的欧氏距离, π_{ij} 表示接收符号 Y_i 判决为星座点 s_j 的概率.

另外, 某个符号 (m 比特表示) 中的第 k 个比特 b_k 的 LLR 计算如下:

$$L_{b_k} = \log\left(\frac{\sum_{s_j \in \mathcal{S}_0} \exp\left(-\frac{\|Y_i - s_j\|^2}{\sigma^2}\right)}{\sum_{s_j \in \mathcal{S}_1} \exp\left(-\frac{\|Y_i - s_j\|^2}{\sigma^2}\right)}\right), \tag{7}$$

其中, $\mathcal{S}_0$ 为 $b_k = 0$ 的星座点的集合, $\mathcal{S}_1$ 为 $b_k = 1$ 的星座点的集合.

2.2 概率 Chase 算法

比特级 Chase 算法 (bit-wise Chase algorithm, B-CA) [6] 在 1972 年被首次提出, 它是一类利用信道信息的译码方法. B-CA 使用低复杂度的硬判决作为主要模块, 其基本原理是利用信道估计信息产

生不同的测试码字, 得到一组尽可能小的错误模式, 最终选择具有最小软权重的码字作为译码器输出. 其复杂度随最不可靠比特 (least reliable bit, LRB) 数 λ 的增大呈指数级增长, 整个算法的复杂度为 $\vartheta(n^2 2^{\lambda})$. 文献 [6] 中提出了 B-CA 的改进算法, 即符号级 Chase 算法 (symbol-level Chase algorithm, S-CA). 与 B-CA 不同的是, 它利用每个符号的可靠性来产生测试向量. 那么对于 m 位二进制数表示的符号码字来说, λ' 个最不可靠的符号将构造 $q^{\lambda'} = 2^{\lambda' m}$ 个测试向量, 从而其复杂度为 $\vartheta(n^2 2^{\lambda' m})$. 为了降低复杂度, 文献 [9] 提出了两步可信度选择的改进算法, 相比 S-CA 降低了 70% 的复杂度, 但是 Chase 算法的复杂度依然是很高的, 这限制了它的实际应用. 因此, 学者们提出了两种概率算法以降低复杂度.

2.2.1 比特级概率 Chase 算法

2010 年, Leroux 等 [7] 提出的基于概率计算的 Chase 算法取得了重大突破, 称为比特级概率 Chase 算法 (BSCA). 其利用接收序列中每个比特的可靠性基于比特级随机试验产生测试码字, 而不需要确定最不可靠的比特. 其算法如算法 1 所示. 使用阈值 θ 可以防止最可靠的位被翻转. 参数 β 是一个正常数, 通过调整 β 可以提高译码性能. 相比 CA 来说, BSCA 不需要选择最不可靠的比特, 仅需要一个随机数产生器和一个比较器组成的简单概率产生器就可以实现, 因此它具有更低的功耗.

Algorithm 1 Bit-wise stochastic Chase algorithm

Initialization:

Compute p_i according to Eq. (1), where $i = 0, 1, \ldots, nm-1$;
for $0 \leqslant i \leqslant nm-1$ **do**
  if $p_i \leqslant 0.5 - \theta$ **then**
    $p_i = 0$, where $0 \leqslant \theta \leqslant 0.5$;
  else if $p_i \geqslant 0.5 + \theta$ **then**
    $p_i = 1$;
  else
    $p_i = \dfrac{1}{1 + \mathrm{e}^{\beta y_i}}$, where $\beta > 0$;
  end if
end for

Iteration:

for $1 \leqslant t \leqslant \tau$ **do**
  for $0 \leqslant i \leqslant nm-1$ **do**
    Generate a uniformly distributed random value: $\alpha_i \in [0, 1]$;
    $y_i^t = \begin{cases} 0, & \text{if } p_i < \alpha_i; \\ 1, & \text{otherwise;} \end{cases}$
  end for
  Convert the binary vector $(y_0^t, y_1^t, \ldots, y_{nm-1}^t)$ to symbol vector $\boldsymbol{Y}^t = (Y_0^t, Y_1^t, \ldots, Y_{n-1}^t)$ and perform BM-HDD to obtain $\boldsymbol{X}^t = (X_0^t, X_1^t, \ldots, X_{n-1}^t)$, then convert $\boldsymbol{X}^t$ to binary vector $(x_0^t, x_1^t, \ldots, x_{nm-1}^t)$;
  Compute the soft weight of $\boldsymbol{Y}^H \oplus \boldsymbol{X}^t$: $W(\boldsymbol{Y}^H \oplus \boldsymbol{X}^t) = \sum_{i=0}^{nm-1} |p_i - 0.5|(y_i^H \oplus x_i^t)$;
end for

Output: Select the decided word $\boldsymbol{D}$ such that: $\boldsymbol{D} = \boldsymbol{X}^j$, $W(\boldsymbol{Y}^H \oplus \boldsymbol{X}^j) = \min_{t \in \{1,\ldots,\tau\}} W(\boldsymbol{Y}^H \oplus \boldsymbol{X}^t)$.

2.2.2 符号级概率 Chase 算法

许多高数据速率应用通常采用高阶调制, 将纠错码与高阶调制结合在一起. 为了充分利用高阶调制的优势, 在最新的研究中, Hossein 提出了一种新的基于概率计算的 Chase 算法: 符号级概率 Chase

算法 (SSCA) [8]. 它不仅在高阶调制中获得更高的增益, 同样也适用于 BPSK 传输, 简化为比特级版本, 称为 SSBT-SCA (symbol-level search bit-wise-transmission stochastic Chase algorithm). 与 BSCA 不同的是, 测试码字的产生是基于符号的, 那么符号级的测试码字就可以基于概率直接产生, 而不用进行与二进制之间的转化, 简化了算法复杂度.

特别的, 定义一个符号置信度的阈值, 可以加速算法收敛. 令 $j_i^1 = \arg\max_{x_j \in \mathrm{GF}(q)}\{\pi_{ij}\}$ 和 $j_i^2 = \arg\max_{x_j \in \mathrm{GF}(q), j \neq j_i^1}\{\pi_{ij}\}$ 分别表示 APP 矩阵 $\boldsymbol{\Pi}$ 中第 i 行最大值的列索引和次大值的列索引, 这两个索引表示第 i 个符号最有可能的取值位置, 定义符号 c_i 的置信度因子:

$$\gamma_i = \frac{\pi_{ij_i^2}}{\pi_{ij_i^1}}. \tag{8}$$

符号的概率产生过程由 APP 矩阵的累积和矩阵 $\mathcal{AC}_{n\times q}$ 和随机数确定, 如文献 [8] 所述. SSCA 算法如算法 2 所示.

Algorithm 2 Symbol-level stochastic Chase algorithm

Initialization:

Compute $\boldsymbol{Y}^H = [y_0^H, \ldots, y_i^H, \ldots, y_{nm-1}^H]$ according to Eq. (3);
Compute the LLR of each bit: LLR_i according to Eq. (7), where $i = 0, 1, \ldots, nm-1$;
Compute p_i according to Eq. (1), where $i = 0, 1, \ldots, nm-1$;
Compute APP matrix $\boldsymbol{\Pi}_{n\times q}$ according to Eqs. (5) and (6);
Get the cumulative sum matrix $\mathcal{AC}_{n\times q}$ from $\boldsymbol{\Pi}_{n\times q}$ and compute γ_i according to Eq. (8), where $i = 0, 1, \ldots, n-1$;
$t = 1$;

Iteration:

while $t \leqslant \tau$ **do**
  Generate $\boldsymbol{Y}^t$ using Multinoulli distribution;
  Perform BM-HDD on $\boldsymbol{Y}^t = (Y_0^t, Y_1^t, \ldots, Y_{n-1}^t)$ to obtain $\boldsymbol{X}^t = (X_0^t, X_1^t, \ldots, X_{n-1}^t)$, the convert $\boldsymbol{X}^t$ to binary vector $(x_0^t, x_1^t, \ldots, x_{nm-1}^t)$;
  Compute the soft weight of $\boldsymbol{Y}^H \oplus \boldsymbol{X}^t$: $W(\boldsymbol{Y}^H \oplus \boldsymbol{X}^t) = \sum_{i=0}^{nm-1} |p_i - 0.5|(y_i^H \oplus x_i^t)$;
  $t = t + 1$;
end while

Output: Select the decided word $\boldsymbol{D}$ such that: $\boldsymbol{D} = \boldsymbol{X}^j, W(\boldsymbol{Y}^H \oplus \boldsymbol{X}^j) = \min_{t\in\{1,\ldots,\tau\}} W(\boldsymbol{Y}^H \oplus \boldsymbol{X}^t)$.

2.2.3 概率 CA 算法纠错能力及复杂度分析

BM 硬判决译码算法的最大纠错能力为 $\lfloor \frac{d-1}{2} \rfloor$. 而 CA 算法 (包括概率 CA 算法) 由于通过翻转一些最不可靠的比特产生了不同的错误模式, 在高信噪比下可能表现出纠正 $d-1$ 个错误符号的能力 [6]. 然而, 要遍历所有的码字并产生对应的错误图案对传统 CA 算法来说复杂度很高, 不具备实际应用价值. 与传统 CA 算法不同的是, 概率 CA 算法不局限于最不可靠比特的子集中. 例如, 如果错误图案有一个错误发生在第 $(\lambda+1)$ 个最不可靠比特上, 由于超出了搜索范围, CA 算法不能纠正它, 但是通过概率 CA 算法这个错误比特可能会被翻转从而可能产生正确的测试码字.

在 B-CA 中, λ 个 LRB 的搜索复杂度为 $\vartheta(\lambda n)$ [7]; 在 BSCA 中这一过程由概率产生, 只有 $\vartheta(n)$ [7] 的复杂度, 相比 B-CA 降低了许多. 在 SSCA 中, SSCA(τ) 的整体复杂度为 $\vartheta(\tau^2 n^2)$ [8], 相比 S-CA 算法复杂度大大降低, 并且译码速度可以通过并行化得到提高.

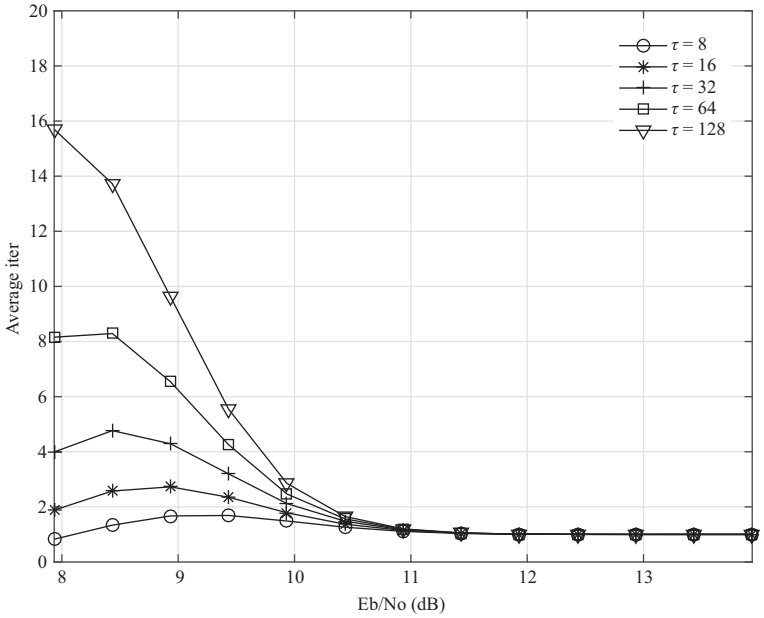

图 1　RS32(31,25) 码在 32-QAM 调制下 SSCA 算法成功译码平均迭代次数

Figure 1　The average number of iterations of successfully decoding of RS32(31,25) with 32-QAM modulation using SSCA algorithm

3　概率 Chase 增强算法

3.1　提前输出算法

由上述算法可知, 概率 Chase 算法产生 τ 组测试码字, 然后选择使得软权重最小的测试码字作为软判决输出, 随着测试码字个数 τ 的增多, 译码性能会有一定程度的改善, 但是运算量和数据存储空间也会随着 τ 的增加而增加. 本小节我们提出了提前输出 (early output, EO) 算法来解决该问题.

研究发现, 由于概率算法根据概率产生测试码字, 在 τ 次迭代内即可产生正确的译码输出, 而不是在 τ 次迭代结束后才能产生正确码字. 同时, 概率算法设置了防止正确码字被误判的阈值, 该参数不仅能够使得误码性能最优, 而且能够将互不相同的测试码字的个数限定在一定范围内. 仿真发现, 在高信噪比下, 基本一次迭代就可以产生能够正确译码的测试码字, 如图 1 所示, 为 RS32(31,25) 码在 32-QAM 调制下使用 SSCA 算法译码, 在产生测试码字的过程中, 高信噪比下平均 1 次迭代就可以得到正确码字, 而其余的迭代次数其实都是冗余的; 即使在中低信噪比下, 对于 $\tau = 128$, 其平均成功迭代次数也可以控制在 20 以内.

基于上述研究, 我们提出在概率译码算法中引入提前输出机制. 将原来的测试码字个数 τ 设为最大迭代次数, 与原概率算法不同的是, 不需要以产生 τ 个测试码字之后计算最小软权重的方式判决译码码字, 而是每产生一组测试码字就进行译码判决; 若译码成功, 则跳出迭代, 否则进入下一次迭代. 我们的判决准则为: BM 硬译码器能够对输入的测试码字成功译码, 并且输出的信息码字可以通过循环冗余校验 (cyclic redundancy check, CRC) 校验. 若输入的测试码字在纠错范围内, BM 译码器可以输出一个满足译码错误概率最小的唯一解, 但是对于有些码型的 RS 码, 在低信噪比下 BM 的误判率较高, 所以根据实际情况选择加入 CRC 校验模块以保证译码性能. 例如对于 RS256(255,239) 码, 只需要占用信息位 8 比特, 即 1 个符号位作为校验位即可满足校验需求. 这样仅以极小的码率牺牲为代价,

用更少的译码尝试和译码时延, 可以获得逼近原算法的译码性能. 该算法我们称为 EO 算法, 如算法 3 所示.

Algorithm 3 Early output stochastic Chase algorithm

Initialization:

Same as BSCA or SSCA;
$t = 1$;

Iteration:

while $t \leqslant \tau$ **do**
  Generate $\boldsymbol{Y}^t$ as BSCA or SSCA;
  Perform BM-HDD on $\mathbf{Y}^t$ to obtain $\boldsymbol{X}^t$;
  if BM decoder returns the flag of decoding successfully and $\boldsymbol{X}^t$ passes the CRC check **then**
    Output $\boldsymbol{X}^t$ as decided decoded word $\boldsymbol{D}_{\text{soft}}$;
    Terminate decoding;
  else
    $t = t + 1$;
  end if
end while

Output: If no satisfactory word $\boldsymbol{D}_{\text{soft}}$ is found, then perform BM-HDD on $\boldsymbol{Y}^H$ to get $\boldsymbol{X}^H$ as output $\boldsymbol{D}_{\text{hard}}$.

EO 算法的改进点在于减少了冗余测试码字的产生, 省略了软权重判决过程, 从而降低了计算复杂度和译码时延. 根据平均迭代次数统计, 该改进算法可以使得概率软译码算法的复杂度在中低信噪比下降低为原来的 1/10 ∼ 1/8, 在高信噪比下可降低为原来的 $1/\tau$. 而且, 在 q 进制调制中, EO 算法不需要计算每个比特位的 r_i 和概率 p_i, 因为不需要进行比特异或计算来获得软权重, 所以在 SSCA 算法的初始化中不需要进行算法 2 的 1∼3 步骤, 算法的计算复杂度在 q 进制调制下将进一步降低.

3.2 3σ 搜索半径算法

SSCA 算法在产生 APP 矩阵的过程中需要计算接收信号与星座图内所有标准星座点的距离以获得每个接收符号判决为所有星座点的概率, 对于 GF(q) 域上的 n 符号码长的 RS 码, 使用 q 进制调制需要计算并存储 $n \times q$ 个概率值用于后续运算. 也就是说, 星座图内所有的星座点构成了测试码字每个符号 Y_i 的选择空间 $\mathcal{S}_i$, 这一空间的元素个数 $|\mathcal{S}_i|$ 会随着调制阶数和伽罗华域阶数 (特别在高阶域) 的增加而增加, 其所占用的存储空间也会成比例增加. 因此仅考虑以接收符号为圆心, 以 r 为半径的范围内的星座点作为该符号的选择空间. 其误帧率下限分析如下.

已知 RS 码的纠错能力为 t 个符号, 测试码字每个符号的搜索半径为 r, 则正确码字在搜索半径之内的概率为

$$\varepsilon = \frac{1}{\sqrt{2\pi}\sigma} \int_{Y_i - r}^{Y_i + r} \exp\left(-\frac{|x - Y_i|^2}{2\sigma^2} \right) \mathrm{d}x. \tag{9}$$

下面分两种情况讨论:

(1) 搜索范围内的正确码字一定会被选中, 误码是由于正确码字皆在搜索范围外. 由于 RS 最大可纠正 t 个错误符号, 那么软判决码字出现 e ($1 \leqslant e \leqslant t$) 个错误符号都可以被纠正, 则错 e 个符号有 C_n^e 种组合, 那么可以纠正的错误概率为

$$P_o = \sum_{e=1}^{t} C_n^e (1 - \varepsilon)^e \varepsilon^{n-e}. \tag{10}$$

(2) 误码是由于搜索范围内的正确码字未被选中. 此种情况用欧式距离计算判决为正确码字的概率. 由于搜索范围内的码字与接收信号点的距离不同, 因此只能计算误帧率边界值, 设搜索范围内的正确码字在搜索半径的边界, 即与接收信号点的欧式距离为 r, 则其概率为

$$P_s = \frac{\exp\left(-\frac{r^2}{2\sigma^2}\right)}{\sum_{j\in \mathrm{GF}(q)} \exp\left(-\frac{d_j^2}{2\sigma^2}\right)}, \tag{11}$$

其中 d_j 为接收符号与星座图内标准星座点 s_j 的欧氏距离. 同理, 可以纠正的错误概率为

$$P_{\mathrm{in}} = \sum_{e=1}^{t} C_n^i (1-P_s)^e {P_s}^{n-e}. \tag{12}$$

综上, 改进算法的误帧率下限为

$$\mathrm{FER}_L = 1 - P_o - P_{\mathrm{in}} - \varepsilon^n - {P_s}^n, \tag{13}$$

显然译码性能会随半径增大而提高. 在 AWGN 信道下接收信号服从高斯分布, 根据其数值分布在 $(\mu-3\sigma, \mu+3\sigma)$ 的概率为 0.9973 的性质, 半径为 3σ 即可满足译码需求. 由此我们提出了 3σ 搜索半径辅助算法 (3σ 算法), 该算法对 SSCA 算法的初始化部分进行了改进, 对每个接收符号 Y_i, 以该接收符号为圆心, 以 3σ (σ 为噪声标准差) 为半径圈定搜索范围, 只将进入搜索范围内的标准星座点作为测试码字符号的选择范围, 这些标准星座点构成集合 $\mathcal{S}_i^{3\sigma}$. 在计算 $\mathbf{\Pi}_i$ 以及后续利用 $\mathbf{\Pi}_i$ 产生软判决 Y_i^t 的过程中, 只需在 $\mathcal{S}_i^{3\sigma}$ 进行运算, 而 $\mathcal{S}_i^{3\sigma}$ 外的星座点不予考虑, 从而也减少了一部分运算. 经过仿真测试, 可在保证译码性能的前提下大大减少搜索范围, 测试码字每个符号的选择空间由原来的 $|\mathcal{S}_i| = q = 2^m$ 个缩小为 $|\mathcal{S}_i^{3\sigma}| = 20$ 个以内 (根据信噪比的不同会有所变化), 从而减少了数据存储空间. 由于前述的 EO 算法是对主迭代部分的改进, 两者互不影响因此可以叠加, 基于 EO-SSCA 的 3σ 算法如算法 4 所示.

4 性能仿真分析

我们在 AWGN 信道条件下, 分别在 BPSK 调制和 M-QAM 调制下对 EO 算法进行了仿真实验, 并与 BSCA 和 SSCA 算法进行了比较. 在 M-QAM 调制下对 3σ 算法进行了仿真实验, 并与 SSCA 算法进行了比较. 根据硬件实现和仿真条件, 选择迭代次数为 2 的次幂, τ 取值为 8, 16, 32, 64, 128. 除非有特殊说明, 下文中提到的增益统一为 $\mathrm{FER} = 10^{-4}$ 处获得的增益.

4.1 EO 算法仿真

在 BPSK 调制和 M-QAM 调制下分别对不同 RS 码型进行了 EO-BSCA 和 EO-SSCA 算法的仿真, 译码性能如表 1 所示; 并且与原概率算法进行了对比, 仿真结果如图 2 所示, EO 算法最佳可以得到逼近原算法的译码性能. 但是, EO 算法时间复杂度相比原概率算法降低了许多, BPSK 调制下我们以 BSCA 的计算复杂度为 1 进行归一化, 统计了每个信噪比下不同算法的归一化时间, 如表 2 所示, 高信噪比下 EO(128) 的计算复杂度可降低为 BSCA(128) 的 1/10, 降低为 SSCA(128) 的 $1/2 \sim 1/4$; 低信噪比下也可降低为对应原算法的 $1/2 \sim 1/10$.

由于在 BPSK 调制下 SSCA 需要进行二进制与多进制的转换, 增加了计算复杂度, 理论和仿真证明, 在 BPSK 调制下 BSCA 算法的计算复杂度比 SSCA 算法低, 如表 2 所示, 特别是随着调制阶数和

表 1 不同 RS 码概率算法相比 BM-HD 误帧率增益表 ($\tau = 128/\mathrm{FER} = 10^{-4}$) (dB)

Table 1 FER gain of different stochastic algorithms for different RS codes compared with BM-HD ($\tau = 128/\mathrm{FER} = 10^{-4}$) (dB)

RS codes	RS16(11,5)	RS16(15,9)	RS32(31,25)
BSCA(128)/BPSK	2.77	2.375	1.825
EO-BSCA(128)/BPSK	3.16	2.375	1.795
SSCA(128)/BPSK	2.87	2.42	1.44
EO-SSCA(128)/BPSK	3.086	2.44	1.4
SSCA(128)/M-QAM	1.01(16-QAM)	0.37(16-QAM)	0.948(32-QAM)
EO-SSCA(128)/M-QAM	1.08(16-QAM)	0.52(16-QAM)	0.943(32-QAM)

Algorithm 4 3σ algorithm based on EO-SSCA

Initialization:

Compute the Euclidean distance: $d_{ij} = \|Y_i - s_j\|^2$, $i = 0, 1, \ldots, n-1$, $j = 0, 1, \ldots, q-1$;
for $0 \leqslant i \leqslant n-1$ **do**
  Find the constellation $s_j \in \mathcal{S}_i$ which satisfies $d_{ij} \leqslant 3\sigma$ to form the search set $\mathcal{S}_i^{3\sigma}$;
  Compute the row vector of APP matrix $\boldsymbol{\Pi}_i$ according to Eqs. (5) and (6) in $\mathcal{S}_i^{3\sigma}$;
  Compute $\mathcal{AC}_i$ according to $\boldsymbol{\Pi}_i$ and obtain γ_i according to Eq. (8) and $\mathcal{AC}_i$;
end for
$t = 1$;

Iteration:

while $t \leqslant \tau$ **do**
  Generate $\boldsymbol{Y}^t$ as SSCA;
  Perform BM-HDD on $\boldsymbol{Y}^t$ to obtain $\boldsymbol{X}^t$;
  if BM decoder returns the flag of decoding successfully and $\boldsymbol{X}^t$ passes the CRC check **then**
    Output $\boldsymbol{X}^t$ as decided decoded word $\boldsymbol{D}_{\mathrm{soft}}$;
    Terminate decoding;
  else
    $t = t + 1$;
  end if
end while

Output: If no satisfactory word $\boldsymbol{D}_{\mathrm{soft}}$ is found, then perform BM-HDD on $\boldsymbol{Y}^H$ to get $\boldsymbol{X}^H$ as output $\boldsymbol{D}_{\mathrm{hard}}$.

伽罗华域阶数的增加, BSCA 算法 (包括 EO-BSCA) 的优势更加明显. 同时, 原概率算法的计算复杂度还受到码长影响, 相同伽罗华域下码长的缩短似乎能够减少原概率算法 BSCA 和 SSCA 之间的计算复杂度差距. 对于缩短码 RS16(11,5) 来说 BSCA 和 SSCA 的计算速度几乎一样; 但是 EO 算法并不会受到此影响, EO-BSCA 的计算复杂度仍然远低于 EO-SSCA, 依然能够体现出 BSCA 算法在 BPSK 调制下的低计算复杂度的优势.

在高阶调制下, BSCA 算法在 BPSK 下的计算复杂度和 SSCA 在一个数量级, 但是 SSCA 减少了二进制与多进制间的转换, 这在硬件实现中更加方便. 同样, EO 算法时间复杂度相比原概率算法降低了许多, 以 SSCA 的计算复杂度为 1 进行归一化, 如图 3 所示, 对于 RS32(31,25) 码, 在低信噪比下 EO-SSCA(128) 降低了近 50% 的复杂度, 而在高信噪比下降低了 90%; 对于 RS256(255, 239) 码, 在低信噪比下 EO-SSCA(128) 降低了近 20% 的复杂度, 而在高信噪比下降低了 80%. 另外, 我们以 RS16(15,9) 码为例, 分别在 BPSK 条件下对 EO-BSCA(128), EO-SSCA(128) 和 16-QAM 条件下对 EO-SSCA(128) 统计了 10^5 帧数据的迭代次数, 并拟合为高斯分布, 如图 4 所示, 图中红色、黑色和

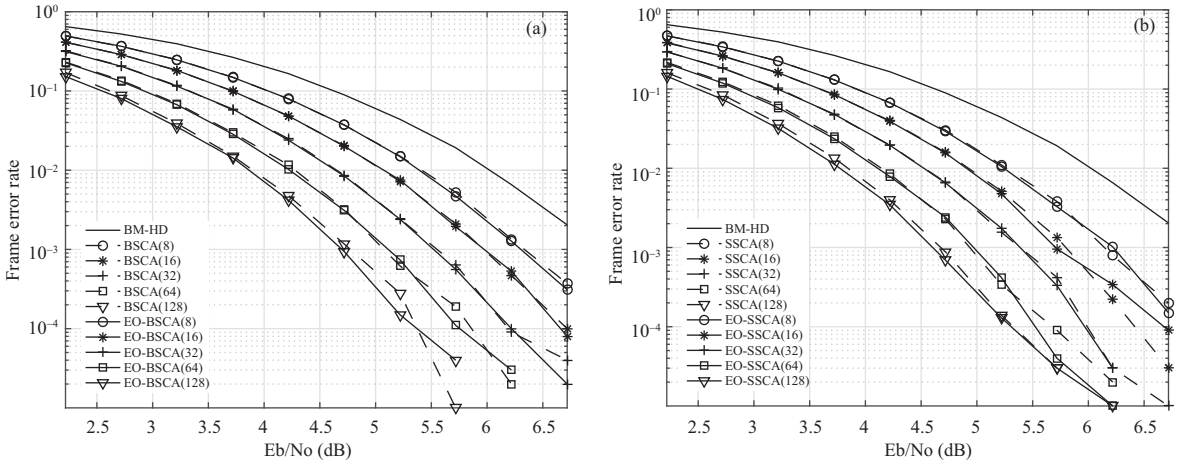

图 2　BPSK 调制下 RS16(15,9) 码 EO-BSCA 与 (a) BSCA 和 (b) SSCA 的误帧率比较

Figure 2　FER of RS16 (15,9) code with BPSK modulation using EO-BSCA, compared with BSCA (a) and SSCA (b)

表 2　BPSK 调制下不同算法高信噪比下归一化时间复杂度对比表 (FER = 10^{-4})

Table 2　Comparison of normalized time complexity of different algorithms in high Eb/No with BPSK modulation (FER = 10^{-4})

RS codes	BSCA(128)	EO-BSCA(128)	SSCA(128)	EO- SSCA(128)
RS16(11,5)	1	0.07	1	0.25
RS16(15,9)	1	0.1	1.188	0.426
RS32(31,25)	1	0.1	1.625	0.875

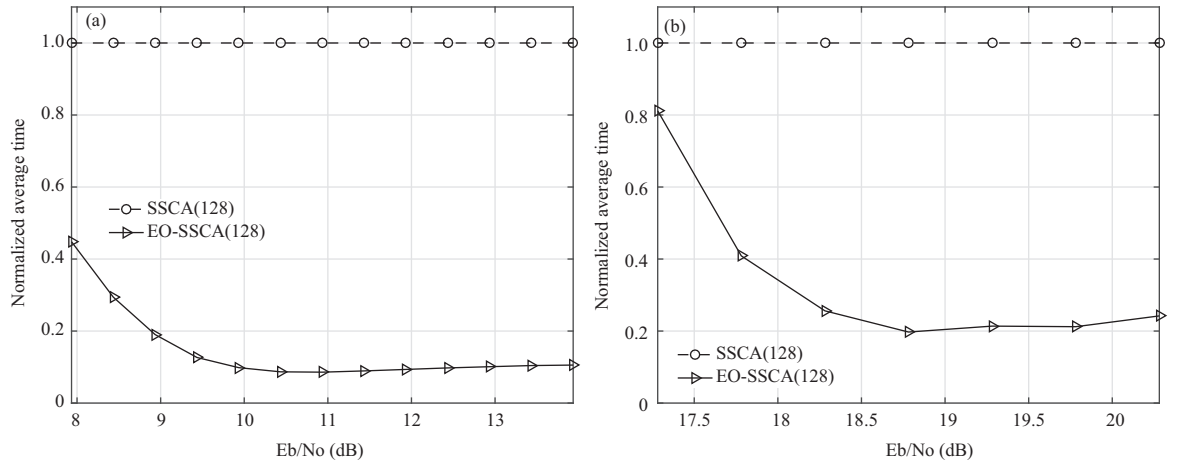

图 3　(a) 32-QAM 调制下 RS32(31,25) 码和 (b) 256-QAM 调制下 RS256(255,239) 码 EO-SSCA(128) 与 SSCA(128) 归一化时间复杂度对比图

Figure 3　Comparison of normalized time complexity of (a) RS32(31,25) between EO-SSCA(128) and SSCA(128) with 32-QAM modulation and (b) RS256(255,239) between EO-SSCA(128) and SSCA(128) with 256-QAM modulation

蓝色的五角星为高信噪比下对应算法的迭代次数的最大值, 均小于设定最大迭代次数. 可以看出, EO 算法在高信噪比下迭代次数分布曲线的均值和方差均较小, 说明达到原算法最大迭代的概率极低, 特

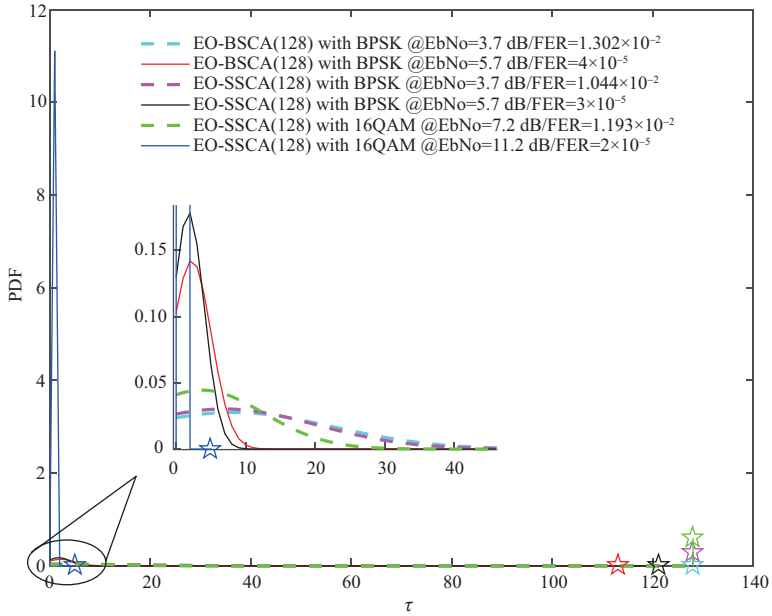

图 4 RS16(15,9) 码 EO-BSCA(128) 和 EO-SSCA(128) 的迭代次数分布曲线

Figure 4 Distribution curve of iteration number of EO-BSCA(128) and EO-SSCA(128) for RS16(15,9)

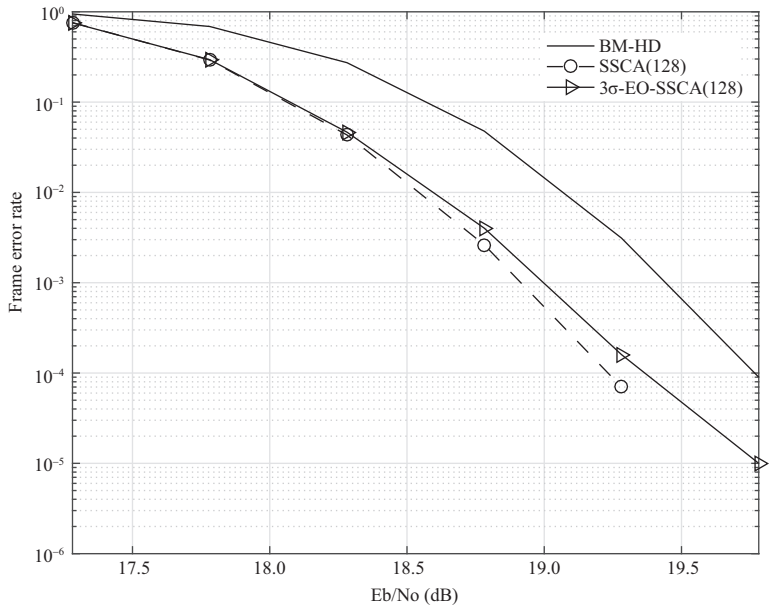

图 5 256-QAM 调制下 RS256(255,239) 码 3σ-EO-SSCA 与 SSCA 误帧率

Figure 5 FER of RS256(255,239) using 3σ-EO-SSCA and SSCA with 256-QAM modulation

别在高阶调制下迭代次数最大值远低于设定的最大迭代次数. 而原算法必须计算全部迭代, 由算法的计算复杂度和时延与迭代次数相关可知 EO 算法在高信噪比下可以降低原算法的最大计算复杂度和时延.

表 3 M-QAM 调制下对应 GF(M) 域上 RS 码单个符号的搜索范围内最大码字个数统计表

Table 3 Statistics table of the maximum number of codewords within the search range of a single symbol for RS codes in GF(M) with M-QAM modulation

RS codes@EbN0 (dB)/FER	SSCA(128)	3σ-EO-SSCA(128)
RS256(255,239)@17.28 dB/FER=0.7540	256	2
RS256(255,239)@20.28 dB/FER=2.0e−6	256	1
RS512(30,18)@17.22 dB/FER=0.6852	512	9
RS512(30,18)@21.22 dB/FER=1.0e−5	512	4
RS1024(30,18)@20.22 dB/FER=0.5786	1024	9
RS1024(30,18)@24.72 dB/FER=1.1e−5	1024	4

4.2 3σ 算法仿真

我们以 RS256(255,239) 为例, 进行基于 EO-SSCA 的 3σ 算法的仿真, 如图 5 所示, 3σ-EO-SSCA(128) 与 SSCA(128) 相比有 0.1 dB 左右的损失, 这一损失在可接受范围内; 而且通过仿真试验, 码型不同可能不会有性能损失甚至有少量增益. 在 SSCA 中, 测试码字每个符号的搜索范围都为 256 个星座点, 一帧数据有 255 个符号, 则需要 255×256 个存储空间, 而在我们提出的 3σ 算法中, 如表 3 所示, 即使在低信噪比下每个符号的搜索空间最多有 2 个星座点, 因此只需要 255×2 个存储空间, 为原算法的 1/128. 对于更高阶调制, 如 512-QAM 和 1024-QAM, 仿真结果表明对于减少单个符号的搜索空间仍有显著效果, 如表 3 所示. 另外我们对短码和中短码也进行了测试, 发现搜索范围最大不会超过 20 个星座点. 从而 3σ 算法可以大大减少测试码字的搜索空间, 节省存储空间.

5 结论

我们提出了两种改进的 RS 概率译码算法, 其中 EO 算法可以显著降低原概率算法的计算复杂度和译码时延, 3σ 算法对 SSCA 算法进行改进, 在高阶调制下可以显著降低存储空间. 我们对不同码长和伽罗华域的 RS 码进行了仿真试验, 并且与原概率算法 BSCA 和 SSCA 进行了综合比较. BPSK 调制方式下, 除非考虑突发错误, 否则优先选择 EO-BSCA 算法, 因为在发生突发错误下, SSCA 更有优势 [8]; 在 M-QAM 调制下, 优先考虑 EO-SSCA 算法; 我们提出的 3σ 搜索半径辅助算法对高阶调制下的 SSCA 算法进行了改进, 通过划定 3σ 半径将测试码字每个符号的搜索范围由 q (q 为调制阶数) 个缩小为个位数, 从而可以显著节省存储空间. 在未来我们将继续研究本文提出的算法在衰落信道环境中和级联译码条件下的性能.

参考文献

1 Massey J. Shift-register synthesis and BCH decoding. IEEE Trans Inform Theory, 1969, 15: 122–127

2 Sugiyama Y, Kasahara M, Hirasawa S, et al. A method for solving key equation for decoding Goppa codes. Inform Control, 1975, 27: 87–99

3 Xing J Y, Chen L, Bossert M. Low-complexity koetter-vardy decoding of reed-solomon codes using module minimization. In: Proceedings of IEEE International Conference on Communications (ICC), Shanghai, 2019

4 Lee H C, Wu J H, Wang C H, et al. An iterative soft-decision decoding algorithm for Reed-Solomon codes. In: Proceedings of IEEE International Symposium on Information Theory (ISIT), Aachen, 2017. 2775–2779

5 Zhang W, Wang S Y, Luo H W, et al. High-performance soft decision algorithm for bursty channel decoding of Reed-Solomon codes. IEEE Commun Lett, 2019, 23: 1141–1144

6 Chase D. Class of algorithms for decoding block codes with channel measurement information. IEEE Trans Inform Theory, 2003, 18: 170–182
7 Leroux C, Hemati S, Mannor S, et al. Stochastic chase decoding of Reed-Solomon codes. IEEE Commun Letter, 2010, 14: 863–865
8 Mani H, Hemati S. Symbol-level stochastic chase decoding of Reed-Solomon and BCH codes. IEEE Trans Commun, 2019, 67: 5241–5252
9 Chu S I, Chen Y H, Chiu Y C. Fast chase algorithms for decoding Reed-Solomon codes. In: Proceedings of International Symposium on Next-Generation Electronics (ISNE), Kwei-Shan, 2014

Enhanced stochastic soft decoding algorithm for Reed-Solomon codes

Yining SUN*, Qiu HUANG & Jianhao HU

National Key Laboratory of Science and Technology on Communication, University of Electronic Science and Technology of China, Chengdu 611731*, China*
* Corresponding author. E-mail: ynsun3842@163.com

Abstract The Chase algorithm based on stochastic computing can significantly reduce the complexity of soft decoding while ensuring the decoding performance, so that the soft decoding algorithm of Reed-Solomon codes is practically applied. However, the time complexity of the stochastic Chase algorithm will increase as the number of test test-vectors increases. In addition, under high-order modulation, the search range of generating test-vectors in the symbol-level stochastic Chase algorithm (SSCA) will increase exponentially by the order of modulation, thus increasing the hardware storage consumption. In order to reduce the time complexity, this paper proposes an early output algorithm, which does not require all test-vectors to achieve successful decoding. Simulation results show that the proposed algorithm can approach the original stochastic decoding performance while reducing the time complexity to nearly $1/\tau$ (τ is the number of test-vectors). In view of the storage issue in the SSCA, this paper reduces the search range by search-radius-determination assisted selection method. Simulation results indicate that the proposed 3σ-SSCA can reduce the search range of a single symbol of a test-vector from q (q is the order of modulation) to less than ten at the cost of a small performance loss.

Keywords RS codes, the chase algorithm, stochastic computing, early output algorithm, search-radius-determination assisted selection method

Yining SUN was born in 1996. She received her B.S. degree from the Hefei University of Technology, Hefei, in 2018. Currently, she is pursuing her M.S. degree at the University of Electronic Science and Technology of China. Her research interests include stochastic computing and error correction coding.

Qiu HUANG was born in 1995. He received his B.S. degree from the University of Electronic Science and Technology of China, Chengdu, in 2018. Currently, he is pursuing his M.S. degree at the University of Electronic Science and Technology of China. His research interests mainly include stochastic computing, error correction decoding and communication integrated circuits and system.

Jianhao HU received his B.E. and Ph.D. degrees in communication systems from the University of Electronic Science and Technology of China (UESTC) in 1993 and 1999 respectively. He joined the City University of Hong Kong from 1999 to 2000 as a postal doctor. From 2000 to 2004, he served as a senior system engineer at the 3G research center of the University of Hong Kong. He has been a professor of National Key Laboratory of Communication of UESTC since 2005. His research interests include high-speed low-power DSP technology with VLSI, NoC, wireless communications and software radio.