基于关键简化集合的极化码SSRFSC翻转译码算法

郭锐，刘洋，何美霖，刘兆霆，赵宜楠
（杭州电子科技大学通信工程学院，浙江 杭州 310018）

摘　要：为提高基于简化序列重复节点的快速串行抵消（SSRFSC）译码算法的译码性能，提出了基于关键简化集合（CSS）和简化序列重复（SSR）节点的CSS-SSR-Flip译码算法。所提算法首先选取SSR中最长重复（REP）节点的信息位作为候选比特（CB），从而构造了基于SSR的CSS；然后针对SSR节点的译码特点设计了相应的翻转度量和翻转准则，通过考虑SSR节点中重复序列和源节点对译码的影响来度量CB可靠性，在特定CB翻转时选择次优的重复序列完成SSR节点翻转操作。实验数据表明，当误帧率FER = 10^{-3}时，所提出的CSS-SSR-Flip译码算法与传统的快速简化串行抵消翻转（Fast-SSC-Flip）译码算法相比能够获得超过0.1 dB的性能增益，且具有与新型Fast-SSC-Flip（New-Fast-SSC-Flip）译码算法相同的译码性能；与传统翻转集合大小K相比，CSS大小缩减最多达79.5%，与关键集合（CS）相比集合大小最多可缩减23.1%。

关键词：极化码；快速简化串行抵消；简化序列重复节点；关键简化集合；翻转译码
中图分类号：TN92
文献标志码：A
DOI：10.11959/j.issn.1000–436x.2024171

SSRFSC flipping decoding algorithm based on critical simplification set for polar codes

GUO Rui, LIU Yang, HE Meilin, LIU Zhaoting, ZHAO Yinan
School of Communication Engineering, Hangzhou Dianzi University, Hangzhou 310018, China

Abstract: To improve the decoding performance of the simplified sequence repetition node-based fast successive cancellation (SSRFSC) decoding algorithm, a decoding algorithm (CSS-SSR-Flip) based on critical simplified set (CSS) and simplified sequence repetition (SSR) nodes was designed. Firstly, the information bit of the longest repetition (REP) node in SSR was selected as candidate bit (CB), thus constructing a CSS based on SSR. Corresponding flip metrics and flip criterion were designed according to the decoding characteristics of SSR nodes. Secondly, the CB reliability was measured by considering the influence of repetition sequences and the source of SSR nodes on decoding, and suboptimal repetition sequences were selected to complete SSR node flip operations during specific CB flips. Experimental data shows that at frame error rates of 10^{-3}, CSS-SSR-Flip decoding algorithm is able to obtain more than 0.1 dB performance gain compared to the traditional fast simplified successive cancellation filp (Fast-SSC-Flip) decoding algorithm, and has the same decoding performance as the new Fast-SSC-Flip (New-Fast-SSC-Flip) decoding algorithm. However, compared to the traditional flip set size K, the CSS size of the proposed algorithm is reduced by up to 79.5%, and compared to the critical set (CS), the set size can be reduced by up to 23.1%.

Keywords: polar code, Fast-SSC, SSR node, critical simplified set, flipping decoding

收稿日期：2024–05–10；**修回日期：**2024–08–28
基金项目：国家自然科学基金资助项目（No.62371174）
Foundation Item：The National Natural Science Foundation of China (No.62371174)

0 引言

Arikan[1]提出的极化码是首个能达到二进制输入离散无记忆信道（B-DMC, binary-input discrete memoryless channel）下香农极限的信道编码方案，但在有限的极化码码长下，经典的串行抵消（SC, successive cancellation）译码算法的性能无法满足实际应用场景中高可靠性的要求。为了解决这一问题有学者提出了以SC译码为基础的一系列改进译码算法。其中，串行抵消列表（SCL, successive cancellation list）译码算法选择最佳的L条码字路径进行译码，当其与循环冗余校验（CRC, cyclic redundancy check）码级联时可获得十分优秀的译码性能[2-4]，但同时带来了较高的译码复杂度。

为改善SC的译码性能，一类翻转初始SC译码中由信道噪声引起的首个错误比特的译码算法，即串行抵消翻转（SC-Flip, successive cancellation flip）译码算法应运而生[5]。在初次SC译码失败后，根据对数似然比（LLR, log-likelihood ratio）的绝对值等选择标准计算并得到一组翻转比特的索引集合，并依次选取上述集合中的元素进行额外的SC译码尝试。在中高信噪比时，SC-Flip的平均译码复杂度与同等条件下的SC译码相当，且具有与较小列表大小下SCL译码相近的译码性能。然而文献[6]中表明SC-Flip译码的单比特翻转准则是次优的，其平均译码复杂度和译码性能仍有较大的改进空间。文献[7-9]中提出的动态SC翻转（D-SCF, dynamic SC-Flip）译码算法使用了更合适的度量和准则并且动态更新翻转集合优化候选比特（CB, candidate bit）实现SC的单比特和多比特翻转。上述SC-Flip译码算法在译码性能上接近SCL译码算法，但是在低信噪比时的多次译码迭代导致了较高的译码时延。

Giard等[10]将快速SC译码与SC-Flip结合，提出快速简化SC翻转（Fast-SSC-Flip, fast simplified SC-Flip）译码算法，但是因为单奇偶校验（SPC, single parity-check）节点的翻转导致了一定的性能损失。在此基础上，Zhou等[11]针对SPC节点设计了新的度量与翻转准则，并给出了其余特殊节点的度量和翻转方案，从而提出了新型Fast-SSC-Flip（New-Fast-SSC-Flip）译码算法，在Fast-SSC-Flip的基础上进一步提升了翻转译码性能；该团队还根据节点错误译码概率以及相应信息比特错误译码概率设计了改进的Fast-SSC-Flip（I-Fast-SSC-Flip, improved Fast-SSC-Flip）译码算法，其需要度量译码二叉树中所有特殊节点的错误译码概率，并要进一步根据其中CB的可靠性才能构建翻转集合，导致该方案的译码复杂度随极化码码长、码率的增加而大幅提高。

上述翻转译码算法均将所有信息比特作为翻转候选比特，译码时对叶子节点或者节点内所有信息位的对数似然比的绝对值（|LLR|）进行升序排列以确定CB的可靠性。然而在构造翻转集合时，对CB的|LLR|排序是增加译码复杂度的重要因素，对上述K个信息位的排序会大幅提高集合构造时的复杂度。Zhang等[12]根据极化码的特殊结构对单比特翻转集合进行优化，提出了关键集合（CS, critical set）的概念，通过具体数据仿真可以验证基于CS的SC-Flip译码算法可以达到理想SC-Flip译码的误帧率性能，并且与K个信息位作为CB的传统集合相比，CS中的元素个数较少且可离线构造，这大大提高了翻转比特的识别效率，降低了翻转译码的复杂度。文献[13]利用CS并与优化的LLR阈值结合来构造翻转集合，提高了译码性能但额外的阈值比较在一定程度上提高了翻转译码复杂度。

上述方法在SC-Flip和Fast-SSC-Flip译码算法中验证了其可行性，目前缺少关于序列重复（SR, sequence repetition）节点以及简化序列重复（SSR, simplified sequence repetition）节点相关翻转集合构造和CB翻转度量的研究[14-15]。本文在文献[15]所提出的基于SR节点的快速SC（SRFSC, sequence repetition node-based fast SC）和基于SSR节点的快速SC（SSRFSC, simplified sequence repetition node-based fast SC）译码算法的基础上，根据SSR的译码特点提出了翻转译码关键简化集合（CSS, critical simplified set）和SSR节点的翻转度量和翻转准则，设计了关于集合CSS和SSR节点的翻转译码算法，即CSS-SSR-Flip译码算法。本文提出的CSS可离线构造，具有比CS更小的集合大小，能够减少翻转集合排序复杂度；新的度量和准则能够针对SSR节点更加准确地识别首位信道噪声引起的单比特错误，从而提升译码效率。

1　极化码的SSRFSC译码算法

1.1　极化码编译码介绍

极化码利用信道极化现象，使用K个相对可靠的信道来传输信息，称为信息位，本文用A表示这些信息位索引组成的集合；其余$N-K$个相对不可靠的信道则用来传输提前约定好的已知信息（$N=2^n$为极化码码长），称为冻结位，冻结位索引组成的集合用A_c表示。为了更方便地表示相关信息，本文使用$|Q|$表示集合Q中元素的数量。码长$N=2^n$，信息位长度等于K的极化码，本文使用$P(N,K)$表示，其编码过程为

$$\boldsymbol{x}_0^{N-1}=\boldsymbol{u}_0^{N-1}\boldsymbol{G}_N \tag{1}$$

其中，$\boldsymbol{u}_0^{N-1}=(u_0u_1\cdots u_{N-1})$表示输入的比特序列。

生成矩阵$\boldsymbol{G}_N$由极化码核心矩阵$\boldsymbol{F}_2=\begin{bmatrix}1&0\\1&1\end{bmatrix}$的$n$阶克罗内克积计算而得，即$\boldsymbol{G}_N=\boldsymbol{F}_2^{\otimes n}$。

1.2　极化码SC译码

SC译码算法是极化码最基本的译码算法，在译码第i个信道时需要逐个对前$i-1$个比特进行转移概率的判定。具体而言，使用对数似然比$\boldsymbol{\alpha}[i]$来得到第i个比特u_i的估计值，LLR的表达式为

$$\boldsymbol{\alpha}[i](y_1^N,\hat{u}_1^{i-1})\triangleq \mathrm{lb}\frac{W_N^{(i)}(y_1^N,u_1^{i-1}|0)}{W_N^{(i)}(y_1^N,u_1^{i-1}|1)} \tag{2}$$

其中，$W_N^{(i)}(y_1^N,u_1^{i-1}|1)$和$W_N^{(i)}(y_1^N,u_1^{i-1}|0)$分别为发送1和0时第$i$子信道的转移概率，$u_1^{i-1}$为信源比特，$y_1^N$为输出信号。

通过对LLR进行硬判决，可以得到对应比特的估计值。

$$\hat{u}_i=h(\boldsymbol{\alpha}[i])=\begin{cases}0, & \boldsymbol{\alpha}[i](y_1^N,\hat{u}_1^{i-1})>0\\1, & \boldsymbol{\alpha}[i](y_1^N,\hat{u}_1^{i-1})<0\end{cases} \tag{3}$$

其中，若$i\in A_c$，则不需进行硬判决，该比特为冻结比特，一般将其视为比特0。

在SC译码树中，对数似然比以向量的形式位于根节点处，并向子节点传递。在译码时，对于层级j的某个节点N_j，其LLR向量优先向左子节点传播，如图1所示，得到LLR向量$\boldsymbol{\alpha}_{j-1,\mathrm{L}}=(\boldsymbol{\alpha}_{j-1,\mathrm{L}}[1]\ \boldsymbol{\alpha}_{j-1,\mathrm{L}}[2]\cdots\boldsymbol{\alpha}_{j-1,\mathrm{L}}[2^{j-1}])$，传递的计算过程为

$$\begin{aligned}\boldsymbol{\alpha}_{j-1,\mathrm{L}}[m]=&f(\boldsymbol{\alpha}_j[m],\boldsymbol{\alpha}_j[m+2^{j-1}])=\\&\mathrm{sgn}(\boldsymbol{\alpha}_j[m])\,\mathrm{sgn}(\boldsymbol{\alpha}_j[m+2^{j-1}])\cdot\\&\min(|\boldsymbol{\alpha}_j[m]|,|\boldsymbol{\alpha}_j[m+2^{j-1}]|)\end{aligned} \tag{4}$$

式(4)定义了f函数，N_j的右子节点会接收到相应的LLR向量$\boldsymbol{\alpha}_{j-1,\mathrm{R}}[m]$，其可以通过$g$函数求得，计算过程为

$$\begin{aligned}\boldsymbol{\alpha}_{j-1,\mathrm{R}}[m]=&g(\boldsymbol{\alpha}_j[m],\boldsymbol{\alpha}_j[m+2^{j-1}])=\\&\boldsymbol{\alpha}_j[m+2^{j-1}]+(1-2\boldsymbol{\beta}_{j-1,\mathrm{L}}[m])\boldsymbol{\alpha}_j[m]\end{aligned} \tag{5}$$

其中，$1\leqslant m\leqslant 2^{j-1}$，最终节点$N_j$处的硬判决向量$\boldsymbol{\beta}_j$的计算式为

$$\hat{\boldsymbol{\beta}}_j[m]=\begin{cases}\hat{\boldsymbol{\beta}}_{j-1,\mathrm{L}}[m]\oplus\hat{\boldsymbol{\beta}}_{j-1,\mathrm{R}}[m], & m\leqslant 2^{j-1}\\\hat{\boldsymbol{\beta}}_{j-1,\mathrm{R}}[m-2^{j-1}], & \text{其他}\end{cases} \tag{6}$$

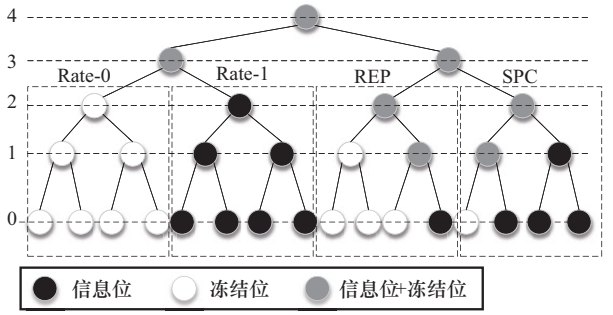

图1　SC译码树和以及特殊节点的二叉树结构

SC译码是一种局部最优的译码算法，未采用真正的最大似然译码算法，因而会导致性能上的损失，但其具有低复杂度的译码优势，其译码复杂度可以表示为$O(N\mathrm{lb}N)$。

1.3　极化码SSRFSC译码

传统SC译码的串行性质导致了每个比特的译码都依赖于先前译码估计的结果从而产生了较高的译码时延。对于某些信息位和冻结位具有特定排列模式的节点，文献[16-17]对其进行了归类，并提出了针对这些节点的高效译码算法，称为Fast-SSC译码算法，从而可以并行计算其返回的硬判决比特向量，基础结构如图1所示。目前有多种Fast-SSC译码的改进算法，文献[14]中提出了新的节点模式，称为SR节点，其中包括了大多现有的特殊节点类型。文献[15]对SR节点进行简化，提出了SSR节点。若令N_j^i表示位于译码树第j层级的第i个节点，假设该节点为SSR节点，其一般结构如图2所示。该类SSR节点除最右侧的源节点（SN, source node）外，其余左侧的所有节点均为码率0（Rate-0）节点或者最长重复（REP, repetition）节点，而源节点仅可能为码率1（Rate-1）节点或者SPC节点。

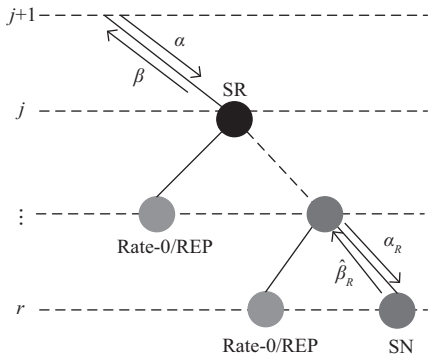

图2　极化码SSR节点结构

SSR节点可用3个参数表示，即SR{SNT,r,$\boldsymbol{v}$}，其中，SNT代表源节点的类型；r表示源节点在译码树中所处的层级；$\boldsymbol{v}$是长度为$j-r$的向量，描述了Rate-0和REP节点的分布情况，$\boldsymbol{v}$中的元素$v[k](r\leqslant k\leqslant j-1)$代表处于SSR译码树第$k$层级的节点是Rate-0还是REP节点，具体取值规则如下

$$v[k]=\begin{cases}0, & \text{Rate-0节点}\\ 1, & \text{REP节点}\end{cases} \tag{7}$$

在译码时首先要定义一组重复序列$\boldsymbol{S}$，由SSR节点的参数向量$\boldsymbol{v}$来确定，根据其中0和1分布的不同，引入变量η表示对应层级中Rate-0节点或者REP节点最后一位的估计值，取值规则可表示为

$$\eta[k]=\begin{cases}0, & v[k]=0\\ 0\text{或}1, & v[k]=1\end{cases} \tag{8}$$

由η可以枚举出REP节点及Rate-0节点所有可能的分布情况，每种情况各对应一组重复序列$\boldsymbol{S}_\lambda$，计算式为

$$\boldsymbol{S}_l=(\eta[j-1],0)\boxplus(\eta[j-2],0)\boxplus\cdots\boxplus(\eta[r],0) \tag{9}$$

其中，$1\leqslant\lambda\leqslant|\boldsymbol{S}_\lambda|$表示不同的重复序列索引，运算符$\boxplus$表示二元域上的克罗内克和运算。不失一般性地假设SSR源节点处于译码树第r阶层的第E个节点位置，在得到重复序列后应根据式(10)计算源节点的LLR向量$\boldsymbol{\alpha}_r^{E,\lambda}$，计算式为

$$\boldsymbol{\alpha}_r^{E,\lambda}[m]=\sum_{k=1}^{2^{j-r}}\boldsymbol{\alpha}_j^i[(k-1)2^r+m](-1)^{\boldsymbol{S}_\lambda[k]} \tag{10}$$

其中，$1\leqslant m\leqslant 2^r$。

根据源节点类型，译码得出源节点所有可能的比特估计值$\hat{\boldsymbol{\beta}}_r^{E,\lambda}[1{:}2^r]$，结合式(10)得到其中最优重复序列的索引，进而求得最大似然（ML, maximum likelihood）解及其对应的最优重复序列，作为源节点最终估计$\hat{\boldsymbol{\beta}}_r^{E,\hat{\lambda}}$。

$$\hat{\lambda}=\underset{1\leqslant\lambda<|\boldsymbol{S}_\lambda|}{\arg\max}\sum_{m=1}^{2^r}\left|\boldsymbol{\alpha}_r^{E,\lambda}[m]\right| \tag{11}$$

其中，$\hat{\lambda}$表示ML解对应的可靠重复序列$\boldsymbol{S}_{\hat{\lambda}}$的索引。最后，由返回的估计向量$\hat{\boldsymbol{\beta}}_r^{E,\hat{\lambda}}$根据对应的重复序列$\boldsymbol{S}_{\hat{\lambda}}$在二元域上推导得到SR节点的比特估计值，计算式为

$$\hat{\boldsymbol{\beta}}_j^i[(k-1)2^{j-r}+1{:}k2^{j-r}]=\hat{\boldsymbol{\beta}}_r^{E,\hat{\lambda}}\oplus\boldsymbol{S}_{\hat{\lambda}}[k] \tag{12}$$

其中，$1\leqslant k\leqslant 2^{j-r}$。

2　关键简化集合

翻转译码可大幅提高译码性能，但极化码中所有的信息位均有可能受到信道噪声影响而产生译码错误，而实际翻转迭代中考虑所有信息比特作为翻转集合依次对其进行翻转译码是不现实的。现有文献对于翻转算法中翻转集合的构建方法一般是通过CB的|LLR|排序或者根据蒙特卡罗仿真得到最易错的CB从而进行构建；此外，Zhang等[12]在基础SC-Flip译码算法的基础上得到了CS的构建方法。本节首先介绍基本翻转译码方案，之后阐述CS的构造方法，并且研究了CS在SSRFSC译码算法下包含首个因信道噪声引起的错误比特的有效性；最后，根据简化SR节点的译码和结构特点对CS进行优化，从而提出了关键简化集合。

2.1　传统翻转译码算法

SC-Flip译码算法的基本思路是在初始SC译码算法未能通过CRC之后，根据每个子信道的|LLR|大小等度量标准对最有可能译码错误的CB进行排序，组成比特翻转集合。重新进行SC译码算法，并在译码到对应比特时进行翻转操作，直至译码结果通过CRC。

Fast-SSC-Flip译码算法的一般译码流程为，在初始Fast-SSC译码算法结束后构建一个大小为K的待翻转集合，该集合中的元素为易错比特的索引，本文称作CB。与SC-Flip译码算法不同的是，集合一般选择初始Fast-SSC译码时各个比特在节点层级对应的|LLR|作为其可靠性的度量。若输出比特未通过CRC，将执行Fast-SSC-Flip译码算法，依次选取翻转集合中的前T个可靠性最低CB作为待翻转比特，在译码至对应比特时，在节点层级执行单比特的翻转译码。

SC-Flip和Fast-SSC-Flip译码算法的重点之一

在于如何构建待翻转集合，此外集合中CB的可靠性度量也将对算法性能和译码时延产生重要影响。为了方便理解，首先定义CB，CB指的是可能被错误估计的位，因此应该在翻转译码中进行修正。对于$P(N, K)$的极化码，为构建待翻转集合，SC-Flip和Fast-SSC-Flip译码算法均选择K个信息位作为CB，这将大大提高翻转集合构建的复杂度，而CS的提出则在一定程度上缓解了上述问题。

2.2　CS的构造

CS的构造是基于SSC译码树的，即用Rate-1和Rate-0节点对SC译码二叉树进行裁剪，Rate-1和Rate-0节点实质上是极化子块，选择裁剪后的SSC译码二叉树中Rate-1节点的首个信息比特作为待翻转比特加入CS中。上述CS构造方案是由传统SC译码的串行特性决定的，SC译码若有译码错误，则其首个错误比特必然属于某个Rate-1极化子块，同时意味着该极化子块之前的子块必定译码正确，串行顺序译码的特定导致了该子块的第一个信息位有极大概率被错误译码。CS构造算法如算法1所示。

算法1　CS构造算法

输入　信息位索引集合A

输出　CS

1) 根据索引集合A建立SC译码树；

2) 将SC译码树裁剪为SSC译码树；

3) 取每个Rate-1极化子块的首位信息比特索引，加入CS；

4) 返回翻转CS。

下面考虑以$N = 16$的极化码为例对算法1所示的CS构造流程进行介绍。图3展示了$P(16,12)$，$\frac{E_b}{N_0} = 2.0$ dB条件下构造的极化码SC译码二叉树，信息位为$(u_4 u_6^{16})$，可被裁剪为图4所示的SSC译码二叉树。Rate-1极化子块为A、B、C、D，其中极化子块A、B仅包含1位信息比特，而极化子块C和D分别含有2位和8位信息比特，子块中首个信息比特分别为u_4,u_6,u_7,u_9。根据算法1将每个Rate-1极化子块中的首位信息比特加入CS中得到翻转集合{ 4,6,7,9 }。CS与传统的翻转比特集合大小K相比，大大缩小了CB的检索范围，且该集合可在极化码构造信噪比确定的情况下离线建立，并不会提高翻转译码复杂度。

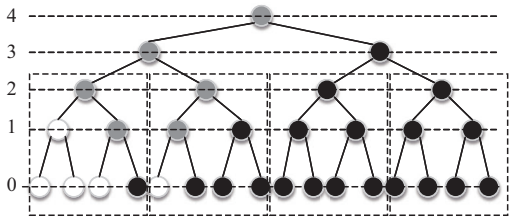

图3　$P(16,12)$极化码SC译码二叉树

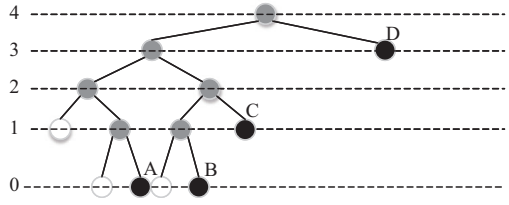

图4　$P(16,12)$极化码SSC译码二叉树

2.3　CS在SSRFSC译码算法下的有效性

已有学者验证了CS在传统SC译码算法以及Fast-SSC译码算法下，包含首位单比特错误的性能与整个信息位集合K相当，本节将CS引入SSRFSC译码算法，验证SSRFSC译码算法在出现单比特错误时，该错误能否被CS精确包含。本节在matlab下利用蒙特卡罗仿真统计首位因信道噪声引起的单比特错误被CS所包含的频率。发送端采用高斯近似法进行可靠性估计，构造信噪比$\mathrm{SNR_d}$为2.5 dB，采用极化码系统编码，其余仿真参数如表1所示。

表1　　**CS准确度仿真参数**

参数	数值
码长N	512, 1 024
码率R	$\frac{1}{2}$
仿真总帧数Frame	10^6

在极化码中，长码长下得到的仿真数据如表2和表3所示，其中，“单比特错误帧数”代表SSRFSC译码算法未通过CRC且产生了单比特错误的总帧数；“CS包含首错帧数”代表根据算法1构造的CS包含上述的首位单比特错误时的总帧数；“准确度”即上述2组数据的比值；“CS大小”表示算法1构造的CS中元素的数量。

从表2和表3可以得出以下结论，极化码码长对CS的准确度几乎没有影响；此外，在中低信噪比下，CS包含首位因信道噪声引起的单比特错误的概率极高，接近100.00%，而且随着信噪比的增加，该概率逐渐增大并趋近于1。上述数据验证了

CS在SSRFSC译码算法下能够有效涵盖首位单比特错误，且CS大小远小于信息比特集合大小K。

表2 $N=512$、$R=\frac{1}{2}$时不同SNR下CS的准确度

信噪比/dB	单比特错误帧数	CS包含首错帧数	准确度	CS大小
1	276 954	275 876	99.61%	
2	97 212	97 187	99.97%	68
3	5 013	5 013	100.00%	

表3 $N=1\,024$、$R=\frac{1}{2}$时不同SNR下CS的准确度

信噪比/dB	单比特错误帧数	CS包含首错帧数	准确度	CS大小
1	279 519	278 766	99.71%	
2	72 082	72 042	99.94%	124
3	1 031	1 031	100.00%	

2.4 CSS的构建

正如前文所述，所有信息位理论上都容易受到信道噪声的影响而产生错误译码。如果最大迭代次数足够大，翻转所有可能的错误位，即$T=K$，那么Fast-SSC-Flip译码算法就可以识别和纠正所有单比特错误并实现理想的简化串行抵消（OA-SSC, oracle-assisted simplified successive cancellation）翻转译码的性能。但实际译码时，需选取可靠性最低的若干比特组成翻转集合进行翻转译码，本文的目标是确保翻转集合具有较低的构造复杂度以及良好的识别单比特错误的有效性。

SC-Flip译码算法的平均迭代次数在中高信噪比时收敛于单次SC译码，Fast-SSC-Flip译码算法同样如此。此外，与SC-Flip译码算法流程相同的步骤在于Fast-SSC-Flip译码算法的单次迭代也需要在译码时进行额外的选取CB构建翻转集合操作。标准Fast-SSC-Flip译码算法通过识别CB中T个最不可靠的决策|LLR|来确定翻转比特；一般需要计算并排序CB的绝对决策|LLR|作为翻转的可靠性度量，选取T个最小决策|LLR|对应的CB构建翻转集合。上述操作在实现时相较于Fast-SSC译码算法会大幅提高译码复杂度以及硬件成本，算法1构建的基于极化码结构的翻转CS能够离线筛选CB，而且CS包含单比特错误的准确性极高，在一定程度上解决了上述问题。但上述CS仅适用于SC-Flip和Fast-SSC-Flip译码算法，为了进一步优化CS，以便其在SSRFSC译码时能准确且高效地识别由信道噪声引起的首位单比特错误，本节提出了基于SSR节点的CSS。

对SSR节点的译码研究中发现，SSR节点的译码关键在于2个部分。其一为SSR源节点能否正确译码；其二为节点中Rate-0和REP节点的数量以及可靠性。其中最重要的又属REP节点，REP的数量决定了SSR译码时译码路径的个数，即重复序列$\boldsymbol{S}$的个数$|\boldsymbol{S}|$。若令N_j^i表示位于译码树第j层级的第i个节点，假设其为SSR节点，那么两者的关系可表示为

$$|\boldsymbol{S}|=2^{n_{\mathrm{REP}}} \tag{13}$$

其中，n_{REP}表示该SSR节点中REP节点的数量。而REP节点的可靠性很大程度上影响了能否选取最优的重复序列$\boldsymbol{S}$，从而实现源节点以及该SSR节点的正确译码；同时源节点的正确译码也影响了能否选取最优的重复序列$\boldsymbol{S}$。

综合上述分析，可以得到在SSR节点的实际译码中，其中REP节点将对译码估计产生重要影响，但其实际上并未直接参与SSR的译码流程。本文将SSR节点中REP节点的个数与该类REP的信息位是否被CS所包含的数据进行了统计，得到了如表4所示的结果。发现使用算法1构造CS时，由于SSR节点中REP节点的唯一信息位是只有一个比特的Rate-1极化子块，该子块必然会加入到CS中。然而正如前文所述，这些REP节点并未直接参与译码，无法直接获取其决策|LLR|并与CS中的其他待翻转比特一起进行可靠性筛选。

表4 不同码长、码率条件下CS的n_{REP}

码长，码率	SSR中的n_{REP}	CS中的n_{REP}
$N=256,R=\frac{1}{2}$	7	7
$N=512,R=\frac{1}{2}$	10	10
$N=1\,024,R=\frac{1}{4}$	18	18

因此，本文根据SSR的结构特点设计了一种CSS，其构建算法如算法2所示。基本思想在于识别所有SSR节点下REP节点的信息位索引，由于这些REP节点未直接参与SSR译码，无法对这些信息位进行可靠性度量，故将其中的大多数从CS中删除，但仍保留SSR节点中最长REP节点对应的信息位索引，因为仍需考虑REP节点对SSR节

点译码的影响，下文将对如何度量CS中该信息比特的可靠性以及其如何执行对应的翻转译码提供具体解决方案。值得注意的是经过算法2对CS进行简化后，得到的CSS大小有了一定程度的缩减，且该集合仍然能够离线构建。

算法2　CSS构造算法

输入　信息位索引集合A，SSRFSC译码树结构SSR_Struct

输出　CSS

1) 更新节点数量M，节点类型识别标志T_{nd}，创建索引集合$\mathrm{CS}_{\mathrm{REP}}$;

2) CS ← GetCritialSet (I,SSR_Struct);

3)　for i = 1:M do

4)　　if $T_{\mathrm{nd}}(i)$ = SSR且其中存在REP节点then

5)　　根据SSR_Struct (i)，找到非最长REP节点的信息位索引;

6)　　将上述索引加入集合$\mathrm{CS}_{\mathrm{REP}}$;

7)　　end if

8)　end for

9)　for i = 1:$\left|\mathrm{CS}_{\mathrm{REP}}\right|$ do

10)　从CS中删除$\mathrm{CS}_{\mathrm{REP}}(i)$;

11) end for

12) return CSS

3　基于CSS的SSR-Flip译码

CS用于Fast-SSC-Flip译码时，需要根据不同特殊节点的结构在节点层级来实现翻转译码。然而在现有翻转译码算法中，缺少针对SSR节点在节点层级的翻转译码算法。本节将根据前文构建的CSS结合SSR节点的译码特殊性讨论其翻转度量和在节点层级的翻转准则，并提出CSS-SSR-Flip译码算法。

3.1　SSR-Flip译码的翻转度量和翻转准则

如前文所述，令N_j^i表示位于译码树第j层级的第i个节点，$\boldsymbol{\alpha}_j^i[1{:}2^j]$表示根节点处的LLR向量，节点层级CB对应的易错比特翻转度量为ρ；此外，令ς表示翻转译码时节点层级的翻转比特的索引。为了方便说明，需要强调翻转度量的作用在于衡量CB的易错概率，翻转准则的意义在于译码时的翻转操作是基于比特层级还是节点层级以及在该层级如何执行翻转操作。例如传统SC-Flip译码算法的翻转是基于比特层级的翻转，而Fast-SSC-Flip译码算法是基于节点层级的翻转译码，这两类算法在翻转时均要进行单比特翻转。对于本节设计的SSR-Flip译码算法，各种特殊节点的翻转度量和准则如下。

1) Rate-1节点

Rate-1节点进行翻转译码时，在节点层级仅有一位比特进行翻转，该比特的索引为决策LLR最小值对应的索引。译码过程中，首先对节点决策|LLR|进行升序排列，然后找到最小|LLR|即节点层级翻转比特的翻转度量$\rho_{\text{Rate-1}}$。当要对该类节点进行翻转译码时，选取该次译码的最小决策|LLR|所对应的比特进行翻转，翻转度量和翻转索引可通过式(14)、式(15)求得。

$$\rho_{\text{Rate-1}} = \min\left|\boldsymbol{\alpha}_j^i[1{:}2^j]\right| \tag{14}$$

$$\varsigma_{\text{Rate-1}} = \underset{1 \leqslant m \leqslant 2^j}{\arg\min}\left|\boldsymbol{\alpha}_j^i[m]\right| \tag{15}$$

2) REP节点

对于REP节点，需要对其节点层级的所有LLR求和，直接得到该节点唯一一位CB对应的决策LLR幅值作为翻转度量。在翻转译码时，直接对节点层级的所有估计比特进行翻转得到翻转译码结果。

$$\rho_{\text{REP}} = \sum_{m=1}^{2^j}\left|\boldsymbol{\alpha}_j^i[m]\right| \tag{16}$$

$$\varsigma_{\text{REP}} = \forall m, 1 \leqslant m \leqslant 2^j \tag{17}$$

3) SPC节点

SPC节点可以认为是一位冻结比特与2^j-1位信息比特的组合节点，根据其节点结构和节点大小，SPC节点会有j位CB加入CSS中。在译码时，首先将SPC节点最小决策|LLR|对应比特从CB中排除，之后选取j个最小的|LLR|，升序排列依次作为翻转度量。在翻转译码时，根据CS中对应翻转比特的索引信息，对SPC节点层级的比特进行翻转。

若用ε表示SPC节点内的最小决策|LLR|值，m_ε表示其在节点内对应的比特索引，$\mathrm{Inx}_{\min,n}$表示除m_ε外第$n(1 \leqslant n \leqslant j)$个最小|LLR|的索引，则SPC内的$j$个翻转度量和翻转索引计算方法如下。

$$\rho_{\text{SPC}}[n] = \left|\boldsymbol{\alpha}_j^i[\mathrm{Inx}_{\min,n}]\right|, 1 \leqslant n \leqslant j \tag{18}$$

$$\varsigma_{\text{SPC}}[n] = \mathrm{Inx}_{\min,n}, 1 \leqslant n \leqslant j \tag{19}$$

4) SSR节点

SSR节点在翻转迭代译码时，存在重复序列$\boldsymbol{S}$

的选择（即路径选择）以及源节点翻转 2 种翻转情况。首先，重复序列 $\boldsymbol{S}$ 的选择主要受 SR 节点中 REP 节点的影响，SR 节点在路径选择出错时，其最左侧必然存在长度为 2^{j-1} 的低可靠度 REP 节点对译码重复序列 $\boldsymbol{S}$ 的选择产生不利干扰，所以将 SSR 下最长 REP 节点的唯一 CB 保留在 CSS 中；本节提出利用式(20)计算该 REP 节点中 CB 的翻转度量。

$$\rho_{\mathrm{SR}}=\begin{cases}\sum\limits_{k=1}^{2^{j-1}}\boldsymbol{\alpha}_j^i[1+2(k-1)],\ \mathrm{SNT}=\mathrm{Rate}-1\\ \sum\limits_{k=1}^{2^{j-1}}\boldsymbol{\alpha}_j^i[1+2(k-1)]-\mathrm{Parity}\varepsilon_{\mathrm{SNT}},\ \mathrm{SNT}=\mathrm{SPC}\end{cases} \tag{20}$$

其中，$\varepsilon_{\mathrm{SNT}}$ 表示 SSR 节点译码中源节点的最小决策|LLR|，Parity 表示计算其奇偶校验值。在 SSR 节点执行翻转译码时，若翻转比特位是 REP 节点的唯一 CB，本节设计的翻转准则为选取源节点|LLR|之和次大的对应重复序列进行译码，并将这种翻转操作称为 SSR 节点的路径翻转。

其次，针对 SSR 源节点的翻转，在 SSR 节点初次译码时根据其 SNT 长度进行与前文 Rate-1 节点进行相同的翻转度量计算；而当源节点为 SPC 时，设计了新的翻转度量，如式(21)所示，其中 $1\leqslant n\leqslant r$，$1\leqslant m\leqslant 2^r$。

$$\rho_{\mathrm{SR}}=\begin{cases}\min\left|\boldsymbol{\alpha}_r^E[1{:}2^r]\right|, & \mathrm{SNT}=\mathrm{Rate}\text{-}1\\ \left|\boldsymbol{\alpha}_r^E[\mathrm{Inx}_{\min,n}]\right|-\mathrm{Parity}\varepsilon_{\mathrm{SNT}}, & \mathrm{SNT}=\mathrm{SPC}\end{cases} \tag{21}$$

$$\varsigma_{\mathrm{SR}}=\begin{cases}\arg\min\limits_{1\leqslant m\leqslant 2^r}\left|\boldsymbol{\alpha}_r^E[m]\right|, & \mathrm{SNT}=\mathrm{Rate}\text{-}1\\ \mathrm{Inx}_{\min,n}, & \mathrm{SNT}=\mathrm{SPC}\end{cases} \tag{22}$$

3.2 CSS-SSR-Flip 译码算法

基于前文设计的 CSS 以及 SSR-Flip 译码算法下的翻转度量和翻转准则，本文提出了一种关于单比特翻转的基于 CSS 和 SSR 节点的翻转译码算法——CSS-SSR-Flip 译码算法，其具体译码流程如算法 3 所示。首先执行初始的 SSRFSC 译码，若输出结果未通过 CRC 则构建 CSS，并根据翻转度量对其中元素进行升序排列。然后依次选取 CSS 的第 i 个元素执行 SSR-Flip 译码算法，直到翻转译码输出结果通过 CRC。

算法 3 CSS-SSR-Flip 译码算法

输入 信息位索引 A，翻转迭代次数 T，信道接收信息 y，SSRFSC 译码树结构 SSR_Struct

输出 译码估计比特序列 $\hat{u}[1{:}N]$

1) $[\hat{u}[1{:}N],\boldsymbol{\alpha}[1{:}N]]\leftarrow$ SSRFSC(y,I);
2) if 估计序列 $\hat{u}[1{:}N]$ 未通过 CRC then
3) CSS $\leftarrow$ ConstructCSS(I,SSR_Struct)；
4) 根据 $\boldsymbol{\alpha}[1{:}N]$ 对 *CSS* 进行升序排列；
5) for i = 1:T do
6) $[\hat{u}[1{:}N]]\leftarrow$SSR-Flip($\boldsymbol{\alpha}[1{:}N]$, I,SSR_Struct,CSS(i))；
7) if 译码码字通过 CRC then
8) return $\hat{u}[1{:}N]$；
9) else
10) continue；
11) end if
12) end for
13) end if
14) return $\hat{u}[1{:}N]$

第 6 行的 SSR-Flip 译码算法具体流程如算法 4 所示。值得注意的是在 SSR-Flip 译码算法中，若翻转比特位于 SSR 节点的非源节点位置，则在选取 SR 节点的重复序列 $\boldsymbol{S}$ 时，选择次优的重复序列进行译码。

算法 4 SSR-Flip 译码

输入 节点 LLR 向量 $\boldsymbol{\alpha}[1{:}N]$，信息位集合 I，SSRFSC 译码树结构 SSR_Struct，翻转比特信息 CSS(i)

输出 译码估计值 $\boldsymbol{x}(I)$

1) 根据 SSR_Struct 更新节点数量 M，节点类型识别标志 T_{nd}；
2) for j = 1:M do
3) switch the value of $T_{\mathrm{nd}}(j,1)$ do
4) case −1
5) 节点为 Rate-0，执行 Fast-SSC 译码；
6) case 1
7) 节点为 Rate-1，执行 Fast-SSC 译码；
8) if 节点 SSR_Struct(j) 包含 CSS(i) then
9) 执行 Rate-1 节点翻转；
10) end if
11) case 2
12) 节点为 REP，执行 Fast-SSC 译码；
13) if 节点 SSR_Struct(j) 包含 CSS(i) then

14)　　执行REP节点翻转；
15)　end if
16)　case 3
17)　节点为SPC，执行Fast-SSC译码；
18)　if 节点SSR_Struct(j)包含CSS(i) then
19)　　执行SPC节点翻转；
20)　end if
21)　case 4
22)　节点为SSR，执行SSR节点译码；
23)　if 节点SSR_Struct(j)包含CSS(i) then
24)　　if CSS(i,2) = −1路径选择有误 then
25)　　选择次优$\boldsymbol{S}$进行路径翻转译码；
26)　　else
27)　　执行源节点翻转；
28)　　end if
29)　 end if
30) end for
31) return 译码估计序列$\boldsymbol{x}(I)$

4　CSS-SSR-Flip译码算法仿真分析

4.1　性能分析

本节将对极化码CSS-SSR-Flip译码算法的性能进行仿真。通过仿真实验评估本文所提的CSS、翻转度量和翻转准则，并分析相应的译码性能曲线，其中极化码编码方式为系统编码。本节进行对比展示的译码算法有文献[15]提出的SSRFSC译码算法、文献[10]中补偿系数s = 0.5的Fast-SSC-Flip译码算法、文献[11]中所述的New-Fast-SSC-Flip译码算法以及本文设计的CSS-SSR-Flip译码算法。采用加性高斯白噪声（AWGN）信道以及二进制相移键控（BPSK, binary phase shift keying）调制，构造信噪比为2.5 dB，其余参数如表5所示。

在上述条件下得到的详细数据如图5所示。从图5可以得到以下结论：CSS-SSR-Flip译码算法与文献[15]中的SSRFSC译码算法相比，在误帧率（FER, frame error rate）FER = 10^{-2}时有约0.44 dB的性能增益；相较于文献[10]中的Fast-SSC-Flip译码算法，在FER = 10^{-3}时可获得0.12 dB的性能增益；对比文献[11]中的New-Fast-SSC-Flip译码算法，在FER = 10^{-3}的条件下译码性能相同。

码率不变，若将极化码码长改为N = 1 024，

表5　仿真参数

参数	数值
码长N	512，1 024
码率R	$\frac{1}{2}$
仿真总帧数Frame	10^6
CRC生成多项式	$g(x) = x^{16} + x^{15} + x^2 + 1$
文献[10]的翻转补偿系数s	0.5
译码翻转迭代次数T	8

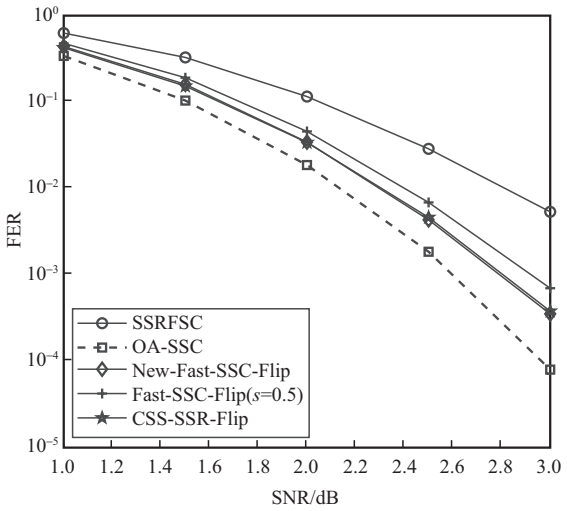

图5　N = 512、$R = \frac{1}{2}$时各算法译码性能对比

得到上述各种译码算法的性能曲线如图6所示。由图6可以得到如下结论：CSS-SSR-Flip译码算法与前文所提的SSRFSC译码算法相比，在FER = 10^{-2}时有约0.33 dB的性能增益；对比文献[10]中的Fast-SSC-Flip译码算法，在FER = 10^{-3}时有接近0.07 dB的性能增益；对比New-Fast-SSC-Flip译码算法，在FER = 10^{-3}的条件下译码性能相同。结合图5和图6可知，CSS-SSR-Flip译码算法可以获得与New-Fast-SSC-Flip译码算法相近的译码性能，且相较于SSRFSC译码算法和Fast-SSC-Flip译码算法可以获得一定的性能提升。

为进一步探究不同情况下CSS-SSR-Flip译码算法的译码性能，下面分析不同码率条件下上述译码算法的仿真数据。因此，本文在码长N = 512的基础上，将极化码码率设置为$R = \frac{1}{4}$、$R = \frac{3}{4}$，得到各译码算法的译码性能曲线如图7、图8所示。

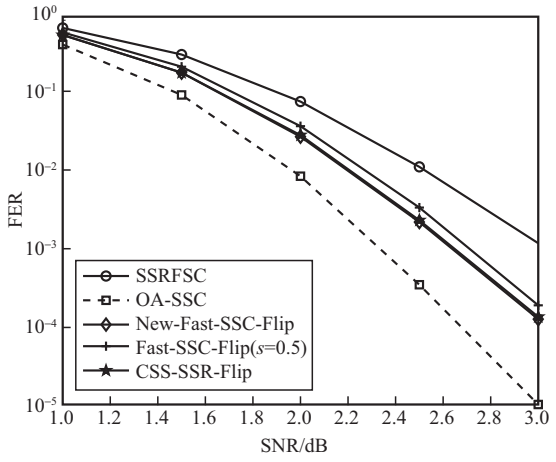

图6　$N = 1\,024$、$R = \frac{1}{2}$时各算法译码性能对比

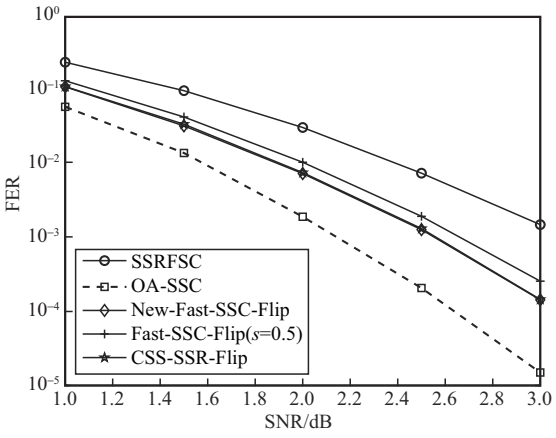

图7　$N = 512$、$R = \frac{1}{4}$时各算法译码性能对比

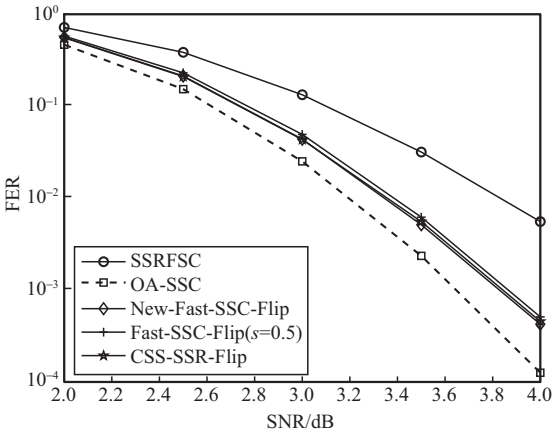

图8　$N = 512$、$R = \frac{3}{4}$时各算法译码性能对比

4.2　复杂度分析

本节将对文献[14]中提出的Fast-SSC译码算法和前文提及的SSRFSC译码算法、Fast-SSC-Flip译码算法、New-Fast-SSC-Flip译码算法以及本文提出的CSS-SSR-Flip译码算法的译码复杂度进行分析对比。翻转译码算法的译码复杂度与平均翻转迭代次数呈正相关，如果初次翻转译码迭代的译码结果未能通过CRC，则将选取翻转集合中的元素，在进行下一次迭代译码时进行翻转，采用平均归一化迭代次数（ANI, average normalized number of iteration）来对Fast-SSC译码算法相关翻转译码的平均译码复杂度进行度量。平均归一化迭代次数的定义式为

$$\mathrm{ANI} = \frac{\sum_{i=1}^{F} \mathrm{Ir}_i}{F} \tag{23}$$

其中，F是译码总帧数，Ir_i表示第i帧翻转译码的迭代次数，$\mathrm{Ir}_i \in [1, T+1]$；随着信噪比的增加，每次翻转译码的平均归一化迭代次数将趋近于1。在与图5~图8同等的条件下，得到如图9和图10所示的各种翻转译码的ANI曲线。

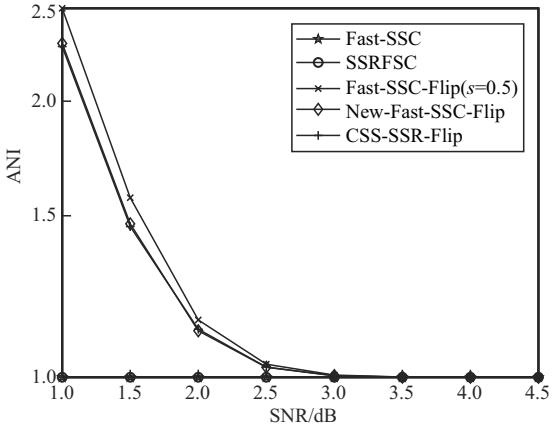

图9　$N = 512$、$R = \frac{1}{4}$时各算法译码性能对比

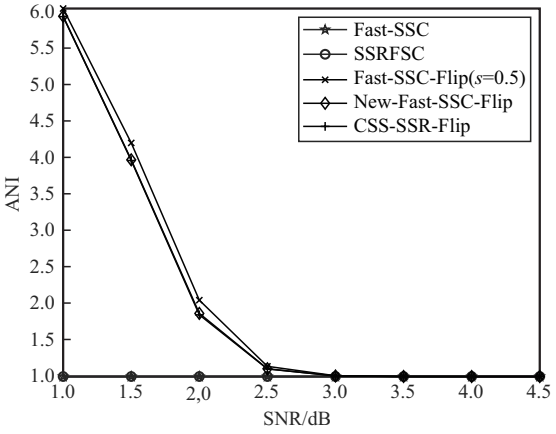

图10　$N = 1\,024$、$R = \frac{1}{2}$时各算法译码性能对比

从图9和图10中可以得到，随着信噪比的增大，3种译码算法的ANI逐渐降低并且最终趋近于1。与文献[10]提出的Fast-SSC-Flip译码算法相比，CSS-SSR-Flip译码算法能降低在较大程度上降低平均译码复杂度。在高、中、低信噪比下，CSS-SSR-Flip译码算法相较于文献[11]提出的New-Fast-SSC-Flip译码算法均具有相同的平均译码复杂度。

译码翻转集合的构建也是影响平均译码复杂度的重要因素，本文设计的CSS翻转集合与传统的翻转集合以及文献[12]提出的CS相比具有明显的低平均译码复杂度优势。Fast-SSC-Flip译码算法的翻转比特集合以K个信息位作为CB，CB排序复杂度为$O(K\,\mathrm{lb}\,K)$；CS缩小了翻转集合大小；本文的CSS则在CS的基础上进一步降低了翻转集合大小从而提高了CB的排序复杂度，具体数据如表6所示。从表6中可以得出，CSS在多种码长码率情况下均具有比传统翻转集合和CS更小的元素数量，与New-Fast-SSC-Flip译码算法所采用的传统翻转集合相比，翻转集合大小最多减少79.5%，并且具有离线构造的优势；相对于CS，CSS的大小最多可降低23.1%。

表6　传统翻转集合、CS、CSS大小对比

码长	码率	传统翻转集合	CS	CSS
$N=256$	$R=\frac{1}{4}$	64	26	20
	$R=\frac{1}{2}$	128	39	32
$N=512$	$R=\frac{1}{4}$	128	51	43
	$R=\frac{1}{2}$	256	68	60
$N=1\,024$	$R=\frac{1}{4}$	256	87	72
	$R=\frac{1}{2}$	512	124	105

5　结束语

本文在SSRFSC译码算法的基础上设计了CSS和SSR节点的翻转度量和翻转准则,并提出了基于CSS和SSR的翻转译码算法，即CSS-SSR-Flip译码算法。实验数据表明，当误帧率FER = 10^{-3}时，CSS-SSR-Flip译码算法与传统的Fast-SSC-Flip译码算法相比能够获得约0.1 dB的性能增益，具有与New-Fast-SSC-Flip译码算法相同的译码性能；提出的CSS可离线构造，并具有比CS更小的集合大小，以便在SSR节点中更加准确地识别首位信道噪声引起的比特错误。实验结果表明，相较于New-Fast-SSC-Flip译码算法，CSS-SSR-Flip译码算法能在译码性能没有损失且复杂度不变的前提下缩减翻转集合的大小最多达79.5%，与CS相比最多缩减比例为23.1%，从而降低了翻转集合构造的复杂度。

参考文献：

[1] ARIKAN E. Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels[J]. IEEE Transactions on Information Theory, 2009, 55(7): 3051-3073.

[2] TAL I, VARDY A. List decoding of polar codes[J]. IEEE Transactions on Information Theory, 2015, 61(5): 2213-2226.

[3] NIU K, CHEN K. CRC-aided decoding of polar codes[J]. IEEE Communications Letters, 2012, 16(10): 1668-1671.

[4] LI B, SHEN H, TSE D. An adaptive successive cancellation list decoder for polar codes with cyclic redundancy check[J]. IEEE Communications Letters, 2012, 16(12): 2044-2047.

[5] AFISIADIS O, BALATSOUKAS-STIMMING A, BURG A. A low-complexity improved successive cancellation decoder for polar codes[C]// Proceedings of the 2014 48th Asilomar Conference on Signals, Systems and Computers. Piscataway: IEEE Press, 2014: 2116-2120.

[6] CONDO C, ERCAN F, GROSS W J. Improved successive cancellation flip decoding of polar codes based on error distribution[C]//Proceedings of the 2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW). Piscataway: IEEE Press, 2018: 19-24.

[7] CHANDESRIS L, SAVIN V, DECLERCQ D. Dynamic-SCFlip decoding of polar codes[J]. IEEE Transactions on Communications, 2018, 66 (6): 2333-2345.

[8] ERCAN F, TONNELLIER T, DOAN N, et al. Simplified dynamic SC-flip polar decoding[C]//Proceedings of the ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Piscataway: IEEE Press, 2020: 1733-1737.

[9] ZHANG X T, LIU Y Z, CHEN S P. BER evaluation based SCFlip algorithm for polar codes decoding[J]. IEEE Access, 2003, 8: 3042-3054.

[10] GIARD P, BURG A. Fast-SSC-Flip decoding of polar codes[J]. arXiv Preprint, arXiv: 1712.00256, 2017.

[11] ZHOU Y C, LIN J, WANG Z F. A new Fast-SSC-Flip decoding of polar codes[C]//Proceedings of the ICC 2019 - 2019 IEEE International Conference on Communications (ICC). Piscataway: IEEE Press, 2019: 1-6.

[12] ZHANG Z Y, QIN K J, ZHANG L, et al. Progressive bit-flipping de-

coding of polar codes: a critical-set based tree search approach[J]. IEEE Access, 2018, 6: 57738-57750.

[13] ERCAN F, GROSS W J. Fast thresholded SC-Flip decoding of polar codes[C]//Proceedings of the ICC 2020 - 2020 IEEE International Conference on Communications (ICC). Piscataway: IEEE Press, 2020: 1-7.

[14] ZHENG H T, ALI HASHEMI S, BALATSOUKAS-STIMMING A, et al. Threshold-based fast successive-cancellation decoding of polar codes[J]. IEEE Transactions on Communications, 2021, 69(6): 3541-3555.

[15] 郭锐, 刘洋. 基于简化序列重复节点的极化码快速串行抵消译码算法[J]. 通信学报, 2023, 44(5): 158-168.
GUO R, LIU Y. Simplified sequence repetition nodes-based fast successive cancellation decoding algorithm for polar code[J]. Journal on Communications, 2023, 44(5): 158-168.

[16] ZHENG H T, BALATSOUKAS-STIMMING A, CAO Z Z, et al. Implementation of a high-throughput Fast-SSC polar decoder with sequence repetition node[C]//Proceedings of the 2020 IEEE Workshop on Signal Processing Systems (SiPS). Piscataway: IEEE Press, 2020: 1-6.

[作者简介]

郭锐（1980– ），男，湖北十堰人，博士，杭州电子科技大学副教授、硕士生导师，主要研究方向为无线通信、信道编码。

刘洋（1997– ），男，河南信阳人，杭州电子科技大学硕士生，主要研究方向为信道编码。

何美霖（1986– ），女，湖南衡阳人，博士，杭州电子科技大学讲师、硕士生导师，主要研究方向为认知无线通信、无线通信、多码率编码。

刘兆霆（1975– ），男，江西广丰人，博士，杭州电子科技大学副教授、硕士生导师，主要研究方向为无信通信、雷达信号处理。

赵宜楠（1977– ），男，黑龙江哈尔滨人，博士，杭州电子科技大学教授、博士生导师，主要研究方向为认知无线通信、信号处理。