基于简化序列重复节点的极化码快速串行抵消译码算法

郭锐，刘洋
（杭州电子科技大学通信工程学院，浙江 杭州 310018）

摘 要：为了进一步降低串行抵消（SC）译码算法的译码时延，在序列重复（SR）节点的基础上，根据 SR 源节点的类型与译码复杂度，对不同类型的拓展类广义奇偶校验（EG-PC）节点进行分解、合并和简化，并使用快速简化串行抵消（Fast-SSC）译码对 Rate-C 节点进行裁剪处理，提出了基于简化 SR 节点的极化码快速 SC 译码算法（SSRFSC）。实验数据表明，在相近的译码性能下（在误帧率为 10^{-3} 时，约有 0.1 dB 的性能损失），与基于 SR 节点的快速 SC（SRFSC）译码算法相比，所提算法的译码时延最多减少了 28%；与 Fast-SSC 译码算法相比，译码时延最多减少了 49%。

关键词：极化码；快速简化串行抵消；简化序列重复节点；译码时延
中图分类号：TN911.22
文献标志码：A
DOI: 10.11959/j.issn.1000−436x.2023088

Simplified sequence repetition nodes-based fast successive cancellation decoding algorithm for polar code

GUO Rui, LIU Yang
School of Communication Engineering, Hangzhou Dianzi University, Hangzhou 310018, China

Abstract: In order to reduce the decoding latency of the successive cancellation (SC) decoding algorithm further, a kind of fast SC decoding algorithm based on simplified sequence repetition (SR) nodes, namely simplified sequence repetition node-based fast SC (SSRFSC), was proposed to optimize decoding latency issues of SC decoding algorithm. Different types of extended class of generalized parity-check (EG-PC) nodes were decomposed, merged and simplified based on the type of SR source node and decoding complexity, and Rate-C node was trimmed using fast simplified successive cancellation (Fast-SSC) decoding. Experimental results show that the decoding latency of the proposed algorithm can be reduced by up to 28% compared to the latest simplified sequence repetition (SRFSC) decoding algorithm when achieving similar decoding performance (approximately 0.1dB performance loss at frame error rate of 10^{-3}). Moreover, compared to the Fast-SSC decoding algorithm, the decoding latency of proposed algorithm can be reduced by up to 49%.

Keywords: polar code, Fast-SSC, simplified SR node, decoding latency

0 引言

极化码是目前唯一一种通过严格的数学证明可以达到二进制离散无记忆信道容量的编码方法[1]。由于具有显式的结构以及较低的编译码复杂度等优点，极化码在最新的 5G 移动通信标准中被批准作为 5G 增强型移动带宽（eMBB, enhanced mobile broadband）控制信道的编码方案[2-3]。虽然串行抵消（SC, successive cancellation）译码算法为极化码提供了一种较低复杂度的译码方案，但 SC 串行译码的特性导致译码时延较高，从而限制了其在低时延通信场景中的应用[4]。因此，解决极化码译码时延的问题受到了广泛关注[5]。

文献[6-7]提出了一种低时延的 SC 译码算法，

收稿日期：2023−01−18；**修回日期：**2023−04−10
基金项目：浙江省重点研发计划基金资助项目（No.2023C03014）
Foundation Item: The Key Research and Development Program of Zhejiang Province (No.2023C03014)

在 SC 译码的每个阶段，针对下一个阶段可能需要的输入信息和中间值，先根据当前的输入信息和中间值，对其所有可能性及可靠性进行预估计，从而减少时延和数据依赖，提高吞吐量，但是需要在硬件结构中增加一些额外的逻辑单元和存储单元。此外，相对于按顺序进行单比特判决的传统 SC 译码，出现了一种在 SC 译码树的中间节点上同时进行多比特判决的改进思路。文献[8-10]采用穷举搜索译码算法进行多比特判决，避免了遍历 SC 译码树时计算中间对数似然比（LLR, log-likelihood ratio）所导致的时延。但是穷举搜索译码算法的复杂性较高，一般只适用于码长较短的极化码。

文献[11]提出了一种简化串行抵消（SSC, simplified successive cancellation）译码算法，使用 Rate-0 和 Rate-1 这 2 种节点对 SC 译码树进行裁剪，可以在不完全遍历 SC 译码树的情况下进行译码，一定程度上降低了译码时延且不会产生译码性能的损失。文献[12]提出了基于重复（REP, repetition）节点和单奇偶校验（SPC, single parity-check）节点的快速简化串行抵消（Fast-SSC, fast simplified successive cancellation）译码算法，大幅降低了译码时延。文献[13]确定了 5 种新节点类型（即 Type-I、Type-II、Type-III、Type-IV 和 Type-V），进一步提高了译码的并行性。然而，上述所有研究都需要为每一类节点设计一个独立的译码方法，这不可避免地增加了算法实现的复杂性。

为此，文献[14]创造性地提出了多节点模式的思想，将 REP、SPC 和 Type-I～Type-V 节点的特点进行拓展和推广，提出了广义 REP（G-REP, generalized repetition）节点和广义奇偶校验（G-PC, generalized parity-check）节点，并给出了该类节点的识别和通用译码规则以进一步减少译码时延。文献[15]分析了短码长极化码中出现最多的 7 种节点，并提出了并行处理这些节点的有效算法。然而，其中一些节点的译码会导致显著的性能损失。文献[16-17]在多节点模式的启发下对文献[14]中的 G-REP 节点进一步推广，以降低 SC 译码树遍历的深度为目的，提出了序列重复（SR, sequence repetition）节点，该类节点通常位于 SC 译码树的更高层级且包含现有的大多数节点类型，具有更通用的冻结位和信息位排列模式；进而提出了基于 SR 节点的快速串行抵消（SRFSC, SR node-based fast SC）译码算法。但是在没有硬件资源限制的条件下，SRFSC 译码算法与文献[13-14]中的方法相比，仅能在一定程度上降低译码时延。

目前，各类最新的极化码快速 SC 译码算法普遍存在的问题是译码复杂度和时延较高，并且缺少一种通用的译码算法以达到低时延高效率译码的目的。多节点模式的思想以及 SR 节点的发现为解决这些问题提供了新的思路。在 SRFSC 译码中，拓展类广义奇偶校验（EG-PC, extended class of generalized parity-check）节点和 Rate-C 节点作为源节点译码过程较复杂，并且 EG-PC 节点在 SC 译码树中有相当高的占比，两者均是 SRFSC 产生译码时延的重要原因。针对这一问题，本文根据不同情况对 EG-PC 节点进行分解、合并或简化，并结合 Fast-SSC 译码算法对 Rate-C 节点进行裁剪处理，提出了简化 SR 节点以及基于简化 SR 节点的快速 SC（SSRFSC, simplified sequence repetition node-based fast SC）译码算法。实验数据表明，与 SRFSC 译码算法相比，SSRFSC 的译码时延最多可减少 28%；与 Fast-SSC 译码算法相比，译码时延最多可减少 49%。

1　极化码 SC 与 Fast-SSC 译码算法

1.1　极化码编译码基本原理

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；而其余 $N-K$ 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，并为每个比特信道分配了标志 d_m，含义如下

$$d_m=\begin{cases}0, & m\in A_c\\ 1, & \text{其他}\end{cases} \tag{1}$$

其中，m 表示比特信道的索引，码长 $N=2^n$，信息位长度等于 K 的极化码编码过程为

$$\boldsymbol{x}_0^{N-1}=\boldsymbol{u}_0^{N-1}\boldsymbol{G}_N \tag{2}$$

其中，$\boldsymbol{u}_0^{N-1}=(u_0\ u_1\cdots\ u_{N-1})$ 表示输入的比特向量。

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 $\boldsymbol{G}_N=\boldsymbol{F}_2^{\otimes n}$。

SC 译码是一种传统的极化码译码算法。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，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

$$\begin{aligned}\boldsymbol{\alpha}_{j-1,\mathrm{L}}[m]=f(\boldsymbol{\alpha}_j[m],\boldsymbol{\alpha}_j[m+2^{j-1}])=\\ \operatorname{sgn}(\boldsymbol{\alpha}_j[m])\operatorname{sgn}(\boldsymbol{\alpha}_j[m+2^{j-1}])\cdot\\ \min(|\boldsymbol{\alpha}_j[m]|,|\boldsymbol{\alpha}_j[m+2^{j-1}]|)\end{aligned}\tag{3}$$

式(3)定义了 F 函数，N_j 的右子节点会接收到相应的 LLR 向量 $\boldsymbol{\alpha}_{j-1,\mathrm{R}}[m]$，可以通过 G 函数求得，计算过程为

$$\begin{aligned}\boldsymbol{\alpha}_{j-1,\mathrm{R}}[m]=g(\boldsymbol{\alpha}_j[m],\boldsymbol{\alpha}_j[m+2^{j-1}])=\\ \boldsymbol{\alpha}_j[m+2^{j-1}]+(1-2\boldsymbol{\beta}_{j-1,\mathrm{L}}[m])\boldsymbol{\alpha}_j[m]\end{aligned}\tag{4}$$

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

$$\hat{\boldsymbol{\beta}}_j[m]=\begin{cases}\hat{\boldsymbol{\beta}}_{j-1,\mathrm{L}}[m]\oplus\hat{\boldsymbol{\beta}}_{j-1,\mathrm{R}}[m], & m\leqslant 2^{j-1}\\ \hat{\boldsymbol{\beta}}_{j-1,\mathrm{R}}[m-2^{j-1}], & \text{其他}\end{cases}\tag{5}$$

1.2 Fast-SSC 等相关算法

对于某些信息位和冻结位具有特定排列模式的节点，文献[6-7]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，从而可以直接计算其返回的硬判决比特向量，不需要再向译码树更深处遍历。

在 Fast-SSC 中，这些特殊节点可以描述如下。在第 0 层级，Rate-0 节点，每个比特都是冻结位；Rate-1 节点，每个比特都是信息位；REP 节点，除了最后一位是信息位外，其余位置都是冻结位；SPC 节点，除了第一位是冻结位外，其余位置都是信息位。上述 4 种特殊节点的二叉树结构如图 1 所示，其中灰色节点表示既存在信息位又存在冻结位的混合速率节点。文献[13]将其中部分节点合并，在二叉树的更高层引入了 5 种新的节点（Type-I～Type-V）及其相应的译码方法，可以实现比 Fast-SSC 更低的译码时延，部分结构如图 2 和图 3 所示。

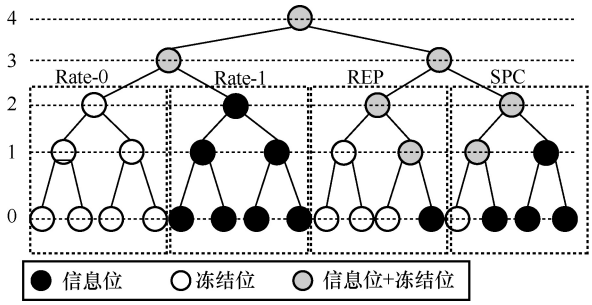


图 1 Fast-SSC 中 4 种特殊节点的二叉树结构

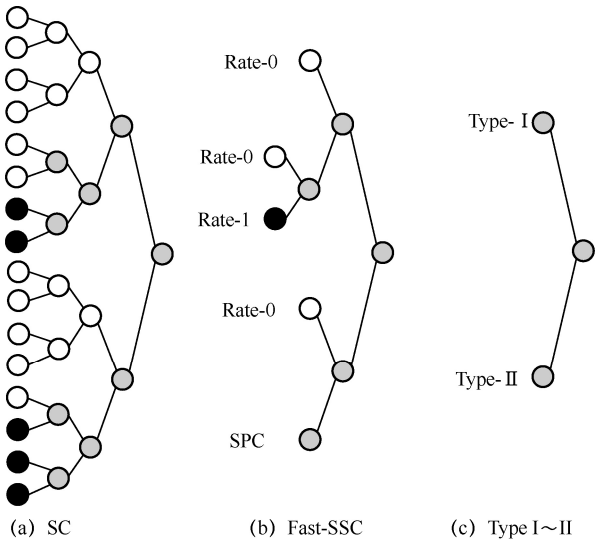


图 2 SC、Fast-SSC 和 Type Ⅰ～Type Ⅱ节点的二叉树结构

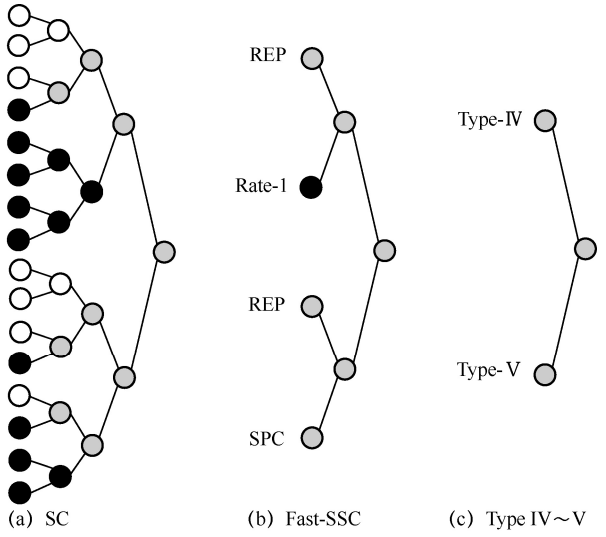


图 3 SC、Fast-SSC 和 Type Ⅳ～Type Ⅴ节点的二叉树结构

文献[14]在此基础上将 REP 和 SPC 节点进行了广义化，提出了广义的奇偶校验（G-PC）节点和广义的序列重复（G-REP）节点，具体形式如下。

1) G-PC 节点：前 2^r 位是冻结位，其余均是信息位。

$$\boldsymbol{p}=[\underbrace{\overbrace{0\cdots0}^{2^r},1\cdots1}_{2^j},1]$$

2) G-REP 节点：最后 2^r 位构成 Rate-C 节点。

$$\boldsymbol{p}=[\underbrace{0,0\cdots0,\overbrace{\mathrm{V}\cdots\mathrm{V}}^{2^r}}_{2^j}]$$

其中，V 代表该位为信息位或者冻结位。对于 G-PC 节点，译码时将视其为多个独立的 SPC 节点进行并行译码；而 G-REP 节点的译码取决于 Rate-C 节点返回

的估计值向量 $\boldsymbol{\beta}_r$，该类节点最终估计值的计算式为

$$\hat{\boldsymbol{\beta}}_j = (\overbrace{\hat{\boldsymbol{\beta}}_r \cdots \hat{\boldsymbol{\beta}}_r}^{2^{j-r}}) \tag{6}$$

2　SRFSC 译码

SR 节点具有比上述节点更一般化的冻结位和信息位排列模式，该类节点的二叉树结构如图 4 所示。在译码树中，SR 节点除了最右侧某个层级为 r 的节点可以是任意类型的 Rate-C 节点以外，其余任意层级为 i $(r \leqslant i \leqslant j)$ 的节点只能是 Rate-0 或 REP 节点。

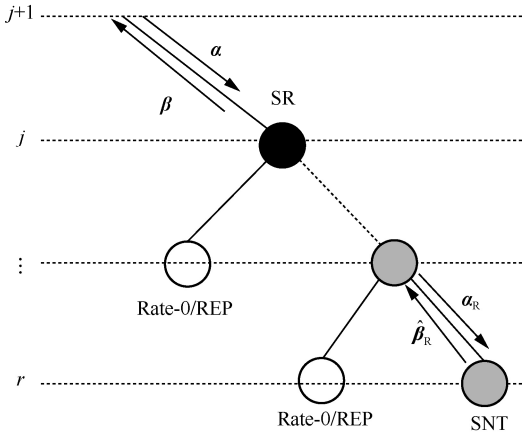


图 4　SR 节点的二叉树结构

SR 节点一般用 3 个参数做进一步的描述，即 SR$\{$SNT$,r,v\}$。其中，SNT 代表源节点的类型，包括 Rate-C、EG-PC 等节点，EG-PC 节点的结构特点是最左侧的子节点为 Rate-0 或者 REP，其余子节点均为 Rate-1。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，表示 Rate-0 和 REP 节点的分布情况；向量 $\boldsymbol{v}$ 中的某个元素 $v[k]$ $(r \leqslant k \leqslant j-1)$ 代表 SR 中处于译码树第 k 层级的节点是 Rate-0 还是 REP 节点，具体取值规则如下

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

SR 节点在译码时首先要定义一组重复序列 $\boldsymbol{S}$，主要由 SR 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 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 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

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

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。若 SR 源节点处于译码树第 r 层级的第 E 个节点位置，在译码时首先应根据式(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，译码得出源节点的比特估计值 $\hat{\boldsymbol{\beta}}_r^{E,\lambda}[1:2^r]$，利用式(11)得到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 $\hat{\boldsymbol{\beta}}_r^{E,\hat{\lambda}}$。

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

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，$1 \leqslant k \leqslant 2^{j-r}$。

3　基于简化 SR 节点的快速 SC 译码算法

本节提出了简化 SR 节点及相应的快速 SC 译码算法。首先，针对 SR 节点的源节点，统计分析了其在多种码率、码长下出现的类型及频率。然后，对于译码时延较高的源节点，如 EG-PC 和 Rate-C 节点，提供了相应的优化方法，并由此提出了 SSRFSC 译码算法。

3.1　源节点的统计

在 SR 节点中，源节点有 4 种类型：Rate-C、Rate-0、Rate-1 和 EG-PC。本文采用高斯近似的方法来选择信息位，并对不同长度和码率的极化码进行了大量实验。实验结果显示，在 SR 的源节点中，EG-PC 节点占据了较大比例，本文提出了分解、合并和简化 3 种方法来对不同情况下的 EG-PC 节点进行优化，具体如表 1 所示。

表 1 **各种码长码率条件下 EG-PC 节点可优化比率**

可优化类型	N=256				N=1 024			
	$R=\frac{1}{8}$	$R=\frac{1}{4}$	$R=\frac{1}{2}$	$R=\frac{3}{4}$	$R=\frac{1}{8}$	$R=\frac{1}{4}$	$R=\frac{1}{2}$	$R=\frac{3}{4}$
分解	14.3%	0	6.3%	0	0	6.1%	4.4%	6.8%
合并	0	18.2%	6.3%	16.7%	15.8%	6.1%	8.9%	6.8%
简化	28.6%	36.4%	25%	25%	31.6%	33.1%	31.1%	20.5%
合计	42.9%	54.6%	37.6%	41.7%	47.4%	45.3%	44.4%	34.1%

EG-PC 节点作为源节点时译码过程较复杂，在译码时计算约束条件以及使用瓦格纳译码需要消耗 2 个时间步数，不利于降低译码时延。

对于 Rate-C 节点，本文用符号 $N_r^{i(\mathrm{RC})}$ 表示，其中，r 代表该节点在 SC 译码树中所在的阶层，i 代表节点在当前阶层中的位置，RC 代表该节点属于 Rate-C 节点。由于其没有特别典型的节点结构，需要对其直接进行 SC 译码，译码所需时间步数为 $2^{r+1}-2$，可见当 Rate-C 节点处于译码树较高层级时，将产生大量的译码时延。

根据以上的统计数据和分析可以得出以下结论：EG-PC 节点的占比较高，且 EG-PC 和 Rate-C 节点的译码过程较复杂，是 SRFSC 译码算法中产生译码时延的重要原因。

3.2 SSRFSC 译码算法

经过上文的分析可知，对 EG-PC 节点直接进行译码会导致较大的译码时延。为了解决这一问题，本文将 EG-PC 节点的译码分为 3 种情况并进行分类处理。第一种情况，EG-PC 自身作为一个 SR 节点存在，可以拆分为新的 SR 节点以及一个或多个 Rate-1 节点进行译码；第二种情况，EG-PC 作为源节点存在，此时可对其进行合并处理；第三种情况，其最左侧仅有一位冻结位，此时 EG-PC 节点转化为 SPC 节点，无须计算奇偶校验约束，直接执行 SPC 节点的译码即可。此外，本文将 Rate-C 节点的译码与 Fast-SSC 相结合。基于上述译码操作，提出了简化的 SR 节点以及 SSRFSC 译码算法。

1) EG-PC 节点的分解。如图 5 所示，少数情况下，EG-PC 节点本身作为 SR 节点存在，若符合该结构，将其分解为长度为 2^{r+1} bit 的 SR 节点 $N_{r+1}^{i(\mathrm{SR})}$ 和 $j-r-1$ 个 Rate-1 节点 $N^{(R1)}$，其中 SR 节点 $N_{r+1}^{i(\mathrm{SR})}$ 的源节点也是 Rate-1 节点。对于该类节点本文给出了如下的译码算法，EG-PC 节点的 LLR 向量为 $\boldsymbol{\alpha}_j^i$，拆分后 SR 节点的 LLR 计算式为

$$\boldsymbol{\alpha}_{r+1}^1[m]=\sum_{k=1}^{2^{j-r+1}}\boldsymbol{\alpha}_j^i[m+(k-1)2^{j-r-1}] \tag{13}$$

其中，$1\leqslant m\leqslant 2^{r+1}$。利用 SR 节点的 LLR 向量，根据式(14)容易得到源节点的 LLR，计算式为

$$\boldsymbol{\alpha}_r^{2,\lambda}[m]=\sum_{k=1}^{2^r}\boldsymbol{\alpha}_{r+1}^1[m+2(k-1)](-1)^{\boldsymbol{S}_\lambda[k]} \tag{14}$$

其中，$1\leqslant\lambda\leqslant|\boldsymbol{S}_\lambda|$，$1\leqslant m\leqslant 2^r$，$1\leqslant k\leqslant 2$ 且 $\boldsymbol{S}_\lambda=(\eta_r,0)$。源节点在译码时利用 $\hat{\boldsymbol{\beta}}_r^{2,\lambda}[m]=h(\boldsymbol{\alpha}_r^{2,\lambda}[m])$ 进行硬判决，然后选取可靠性最高的重复序列 $\boldsymbol{S}_{\hat{\lambda}}$ 与相应的源节点估计比特 $\hat{\boldsymbol{\beta}}_r^{2,\hat{\lambda}}$，得到最优的 SR 估计值 $\hat{\boldsymbol{\beta}}_{r+1}^1$，计算式为

$$\hat{\lambda}=\arg\max_{1\leqslant\lambda<|\boldsymbol{S}_\lambda|}\sum_{m=1}^{2^r}\left|\boldsymbol{\alpha}_r^{E,\lambda}[m]\right| \tag{15}$$

$$\hat{\boldsymbol{\beta}}_{r+1}^1[2(k-1)+1:2k]=\hat{\boldsymbol{\beta}}_r^{2,\hat{\lambda}}\oplus\boldsymbol{S}_{\hat{\lambda}}[k] \tag{16}$$

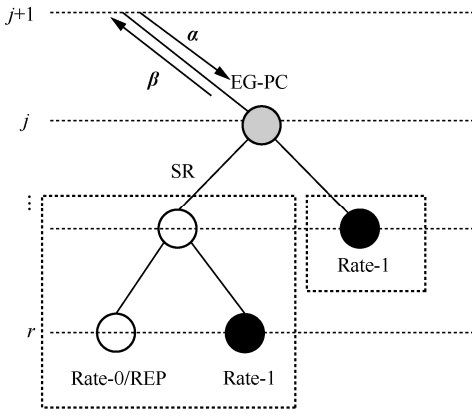


图 5 EG-PC 节点的拆分

根据其余 Rate-1 节点的判决结果 $\hat{\boldsymbol{\beta}}_c^{(R1)}$，向上层节点返回比特估计值，直至得到 EG-PC 节点的估计值 $\hat{\boldsymbol{\beta}}_j^1$，推导过程为

$$\begin{aligned}\hat{\boldsymbol{\beta}}_{r+2}^{1} &= (\hat{\boldsymbol{\beta}}_{r+1}^{1} \oplus \hat{\boldsymbol{\beta}}_{r+1}^{2(R1)} \quad \hat{\boldsymbol{\beta}}_{r+1}^{2(R1)})\\ \hat{\boldsymbol{\beta}}_{r+3}^{1} &= (\hat{\boldsymbol{\beta}}_{r+2}^{1} \oplus \hat{\boldsymbol{\beta}}_{r+2}^{2(R1)} \quad \hat{\boldsymbol{\beta}}_{r+2}^{2(R1)})\\ &\vdots\\ \hat{\boldsymbol{\beta}}_{j}^{1} &= (\hat{\boldsymbol{\beta}}_{j-1}^{1} \oplus \hat{\boldsymbol{\beta}}_{j-1}^{2(R1)} \quad \hat{\boldsymbol{\beta}}_{j-1}^{2(R1)})\end{aligned} \tag{17}$$

其中，$r-1 \leqslant c \leqslant j-1$。

2) EG-PC 节点的合并。在某些情况下，译码树中的 EG-PC 是 SR 的源节点。如图 6 所示，若某个处于译码树 r 阶层的第 E 个节点为 EG-PC 节点，则其可以表示为 $N_r^{E(\text{EG-PC})}$，此时可将 EG-PC 节点转化为两部分，从而组合成为一个新的 SR 节点，该 SR 的源节点变为了 Rate-1。对此时的 SR 节点进行译码，本文给出了如下的译码流程。首先，由 SR 节点的 LLR 向量 $\boldsymbol{\alpha}_j^i[m]$ 可以求出 Rate-1 的 LLR 向量 $\boldsymbol{\alpha}_r^{2,\lambda}[m]$，计算式为

$$\boldsymbol{\alpha}_{r-1}^{2,\lambda}[m] = \sum_{k=1}^{2^{j-r+1}} \boldsymbol{\alpha}_j^i[m+(k-1)2^{j-r-1}](-1)^{\boldsymbol{S}_\lambda[k]} \tag{18}$$

$$\boldsymbol{S}_\lambda = (\eta[j-1],0) \boxplus (\eta[j-2],0) \boxplus \cdots \boxplus (\eta[r-1],0) \tag{19}$$

其中，$1 \leqslant m \leqslant 2^{r-1}$，$1 \leqslant \lambda \leqslant |\boldsymbol{S}_\lambda|$，$1 \leqslant k \leqslant 2^{j-r+1}$。进而直接进行硬判决完成 Rate-1 节点的估计。源节点为 Rate-1 而非 EG-PC，节省了后者计算校验约束和比特反转等译码步骤。后续根据式(15)得到最优的重复序列 $\boldsymbol{S}_\lambda$ 求得 SR 的估计值 $\hat{\boldsymbol{\beta}}_j^i$，计算式为

$$\hat{\boldsymbol{\beta}}_j^i[2^{j-r+1}(k-1)+1:k2^{j-r+1}] = \hat{\boldsymbol{\beta}}_{r-1}^{2,\hat{\lambda}} \oplus \boldsymbol{S}_{\hat{\lambda}}[k] \tag{20}$$

显然，当 EG-PC 节点符合上述结构时，可以将其分解后再组合，且在实际操作过程中，此类情况出现的频率相对较高。

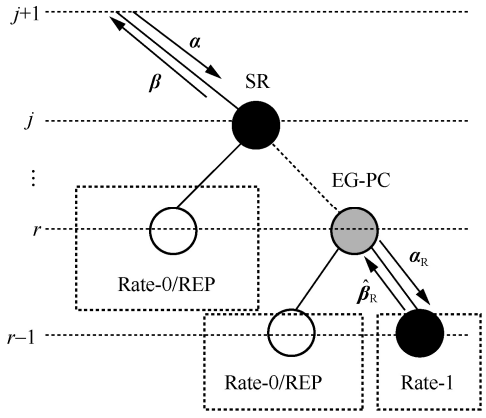


图 6　EG-PC 节点的合并

3) EG-PC 节点的简化。如图 7 所示，除上述情况外，当 EG-PC 节点左侧有仅有一位冻结位时，EG-PC 则转化为 SPC 节点，这类情况的译码流程与常规的 EG-PC 节点类似，但 SPC 节省了计算奇偶校验约束的步骤。译码时根据式(21)得到源节点的 LLR 向量

$$\boldsymbol{\alpha}_r^{2,\lambda}[m] = \sum_{k=1}^{2^{j-r}} \boldsymbol{\alpha}_j^i[m+(k-1)2^{j-r}](-1)^{\boldsymbol{S}_\lambda[k]} \tag{21}$$

其中，$1 \leqslant m \leqslant 2^r$，$1 \leqslant k \leqslant 2^{j-r}$。由于源节点是 SPC，对其译码步骤较简单，通过硬判决得到判决值 $\hat{\boldsymbol{\beta}}_r^{2,\lambda}$ 后判断其是否符合初始奇偶校验，最终的估计结果计算式为

$$\hat{\boldsymbol{\beta}}_r^{2,\lambda} = \begin{cases} \hat{\boldsymbol{\beta}}_r^{2,\lambda}[m] \oplus \zeta^\lambda, & m = \varepsilon^\lambda \\ \hat{\boldsymbol{\beta}}_r^{2,\lambda}[m], & \text{其他} \end{cases} \tag{22}$$

其中，ζ^λ 为硬判决结果的总和，ε^λ 为 SPC 中最小 LLR 对应的比特索引，ζ^λ 和 ε^λ 的计算式分别为

$$\zeta^\lambda = \bigoplus_{m=1}^{2^r} \hat{\boldsymbol{\beta}}_r^{2,\lambda}[m] \tag{23}$$

$$\varepsilon^\lambda = \arg\min_{1 \leqslant m < 2^r} |\boldsymbol{\alpha}_r^{2,\lambda}[m]| \tag{24}$$

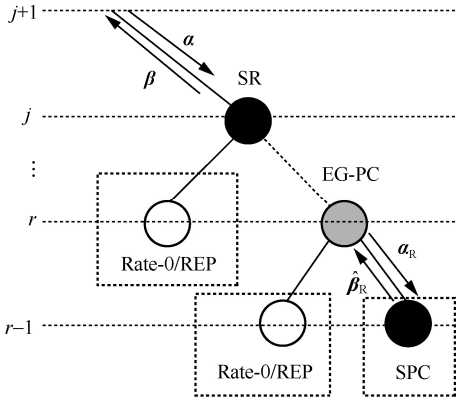


图 7　EG-PC 节点的简化

根据式(15)得到最可靠的重复序列后，SR 的译码估计值为

$$\hat{\boldsymbol{\beta}}_j^i[2^{j-r}(k-1)+1:k2^{j-r}] = \hat{\boldsymbol{\beta}}_r^{2,\hat{\lambda}} \oplus \boldsymbol{S}_{\hat{\lambda}}[k] \tag{25}$$

为方便说明，本文以 N=1 024，R=$\frac{1}{2}$的极化码为例，若在译码树中存在某个序列重复节点 SR(EG-PC,2,(0,1))。该 SR 节点 $N_3^{1(\text{SR})}$ 位于二叉树的第 3 层级且源节点是处于第 2 层级的 EG-PC 节点 $N_2^{2(\text{EG-PC})}$。此时的节点结构符合第三种情况：EG-PC 最左侧的节点仅有一位冻结位，直接将其转化为 SPC 节点进行译码。

由于 SR 的参数向量 $\boldsymbol{v}$=(1)中只有一个 1（SR 只有一个 REP 节点），那么对应的重复序列 $\boldsymbol{S}_\lambda$ 共有 2^1 个，即 $|\boldsymbol{S}_\lambda|=2,\ \lambda\in\{1,2\}$。根据式(8)容易获得 $\eta[2]=0$ 或 $\eta[2]=1$，结合式(9)得到 SR 节点对应的重复序列 $\boldsymbol{S}_\lambda$，即

$$\boldsymbol{S}_1=(0,0)$$
$$\boldsymbol{S}_2=(1,0)$$

若取 $\boldsymbol{\alpha}_3^1=(5.9\ 11.7\ 3.3\ 2.0\ -9.1\ -6.2\ -8.2\ -12.1)$（为方便说明取小数点后一位）表示 SR 节点 $N_3^{1(\mathrm{SR})}$ 接收的 LLR 向量，$\boldsymbol{\alpha}_2^{2,\lambda}$ 表示源节点对应于重复序列 $\boldsymbol{S}_\lambda$ 的 LLR 向量。为便于理解，本文提出了 4 个步骤来有效地进行译码。

1) 根据式(21)，源节点的 LLR 向量 $\boldsymbol{\alpha}_2^{2,\lambda}$ 计算结果如下

$$\boldsymbol{\alpha}_2^{2,1}=(-3.2\quad 5.5\quad -4.9\quad -10.2)$$
$$\boldsymbol{\alpha}_2^{2,2}=(-14.9\quad -17.9\quad -11.5\quad -14.1)$$

2) 对于每组 LLR 向量，根据源节点的类型（此处为 SPC）和式(22)对其进行译码得到源节点返回的估计比特 $\hat{\boldsymbol{\beta}}_2^{2,\lambda}$

$$\hat{\boldsymbol{\beta}}_2^{2,1}=(0\ 0\ 1\ 1)$$
$$\hat{\boldsymbol{\beta}}_2^{2,2}=(1\ 1\ 1\ 1)$$

3) 根据式(15)选取可靠性最高的重复序列 $\boldsymbol{S}_{\hat{\lambda}}$（$\hat{\lambda}=2$）

$$\hat{\lambda}=\arg\max_{0\leqslant\lambda<|\boldsymbol{S}_\lambda|}\sum_{m=1}^{2^r}|\boldsymbol{\alpha}_2^{2,\lambda}[m]|=2$$

4) 根据最优的重复序列及其对应源节点的估计比特 $\hat{\boldsymbol{\beta}}_2^{2,2}$，结合式(25)完成最终 SR 节点的比特估计 $\hat{\boldsymbol{\beta}}_3^{1(\mathrm{SR})}$

$$\hat{\boldsymbol{\beta}}_3^{1(\mathrm{SR})}=\hat{\boldsymbol{\beta}}_2^{2,2}\oplus\boldsymbol{S}_2[k]=(0\ 0\ 0\ 0\ 1\ 1\ 1\ 1)$$

为了简化 EG-PC 节点的复杂译码步骤，本文对其进行了分解、合并和简化等操作，使 SR 的源节点只剩下 Rate-1 和 SPC 两类简单节点。然而，SRFSC 译码中的 Rate-C 节点仍然需要用 SC 译码算法处理，导致译码时延过高。因此，本文放弃了将 Rate-C 作为源节点的做法，而是借鉴 Fast-SSC 译码算法中的四类节点裁剪方法，对非 SR 节点进行优化。最终结果如表 2 所示。Rate-C 节点经过裁剪后只有 3 种类型（缺少 Rate-0），这是由于几乎所有的 Rate-0 节点已经被合并在 SR 中。从表 2 中可以看出，随着码长和码率的增加，Rate-C 节点可以被裁剪为更多的特殊节点，从而节省更多的译码时延。

表 2　不同 $|\boldsymbol{S}|$ 下的 SR 节点数量和 Rate-C 节点裁剪结果

N	R	SR						非 SR		
		$\|\boldsymbol{S}\|$=1	2	4	8	16	合计	Rate-1	REP	SPC
256	1/8	1	1	1	0	0	3	1	2	1
	1/4	1	3	0	1	1	5	2	0	4
	1/2	2	2	1	1	0	6	3	2	5
	3/4	1	2	2	0	0	5	2	0	5
1 024	1/8	2	3	2	1	1	9	1	3	5
	1/4	4	5	5	1	0	15	3	6	9
	1/2	5	10	3	2	0	20	7	3	15
	3/4	6	6	2	1	0	15	13	4	12

综合上述对 EG-PC 与 Rate-C 节点快速译码的改进思路，本文提出了 SSRFSC 译码算法。首先 EG-PC 节点与 Rate-C 节点不再作为 SR 源节点进行译码，而是仅考虑源节点为 Rate-1 节点及 SPC 的 SR 节点结构，称其为简化的 SR 节点，并提出了译码算法 1 对该类节点进行译码，对于简化的 SR 节点以外的节点，则直接使用 Fast-SSC 译码算法进行译码。所提 SSRFSC 译码算法如算法 2 所示。

算法 1　简化的 SR 节点译码

输入　节点 LLR 向量，重复序列 $\boldsymbol{S}$

输出　节点估计值 $\hat{\boldsymbol{\beta}}_j^i[1:2^j]$

1)计算源节点 LLR $\boldsymbol{\alpha}_r^{E,\lambda}$ ($\lambda\in[1,|\boldsymbol{S}_\lambda|]$)
2) for ($\lambda=1;\ \lambda\leqslant|\boldsymbol{S}|;\ \lambda++$) do
3)　if SNT = Rate-1 then
4)　$\hat{\boldsymbol{\beta}}_r^{E,\lambda}[m]=h(\boldsymbol{\alpha}_r^{E,\lambda}[m]),m\in[1,2^r]$
5)　else if SNT = SPC then
6)　$\hat{\boldsymbol{\beta}}_r^{E,\lambda}[m]=h(\boldsymbol{\alpha}_r^{E,\lambda}[m]),m\in[1,2^r]$
7)　根据 $\boldsymbol{\alpha}_r^{E,\lambda}[m]$ 和 $\boldsymbol{\beta}_r^{E,\lambda}[m]$ 比特序列进行奇偶校验和比特翻转
8)　end if
9) end for
10)选取最优的候选比特估计值索引 $\hat{\lambda}$

11) $\hat{\lambda}=\arg\max\limits_{\lambda\in[1,|\boldsymbol{S}_\lambda|]}\sum\limits_{m=1}^{2^r}|\boldsymbol{\alpha}_r^{E,\lambda}[m]|$

12)得到节点估计值并向父节点传递

算法 2　SSRFSC 译码算法

输入　节点 LLR 向量，信息位索引集合 I

输出　译码输出比特序列 $\boldsymbol{x}(I)$

1) 初始化节点数量 M，节点类型识别标志 T_{nd}
2) 更新 M 和 T_{nd}，更新 LLR
3) for ($i=1$; $i\leqslant M$; $i++$) do
4)　switch the value of $T_{\text{nd}}(i,1)$ do
5)　case −1
6)　节点为 Rate-0 执行 Fast-SSC 译码
7)　case 1
8)　if $T_{\text{nd}}(i,2)=-1$ 该节点属于 SR 节点 then
9)　执行简化的 SR 节点译码
10)　else
11)　节点为 Rate-1，执行 Fast-SSC 译码
12)　end if
13)　case 2
14)　节点为 REP，执行 Fast-SSC 译码
15)　case 3
16)　if $T_{\text{nd}}(i,2)=-1$ 该节点属于 SR 节点 then
17)　执行简化的 SR 节点译码
18)　else
19)　节点为 SPC，执行 Fast-SSC 译码
20)　end if
21)　end case
22)　end switch
23) end for
24) 返回译码估计序列 $\boldsymbol{x}(I)$

4　译码性能与时延分析

本节主要对 SSRFSC 译码算法的译码性能与时延进行分析，并将其与 Fast-SSC 以及 SRFSC 译码算法进行比较。

4.1　译码性能分析

图 8 和图 9 展示了 N={256,1 024}、$R=\left\{\frac{1}{8},\frac{1}{4},\frac{1}{2}\right\}$、帧数为$10^6$ 时，Fast-SSC、文献[14]

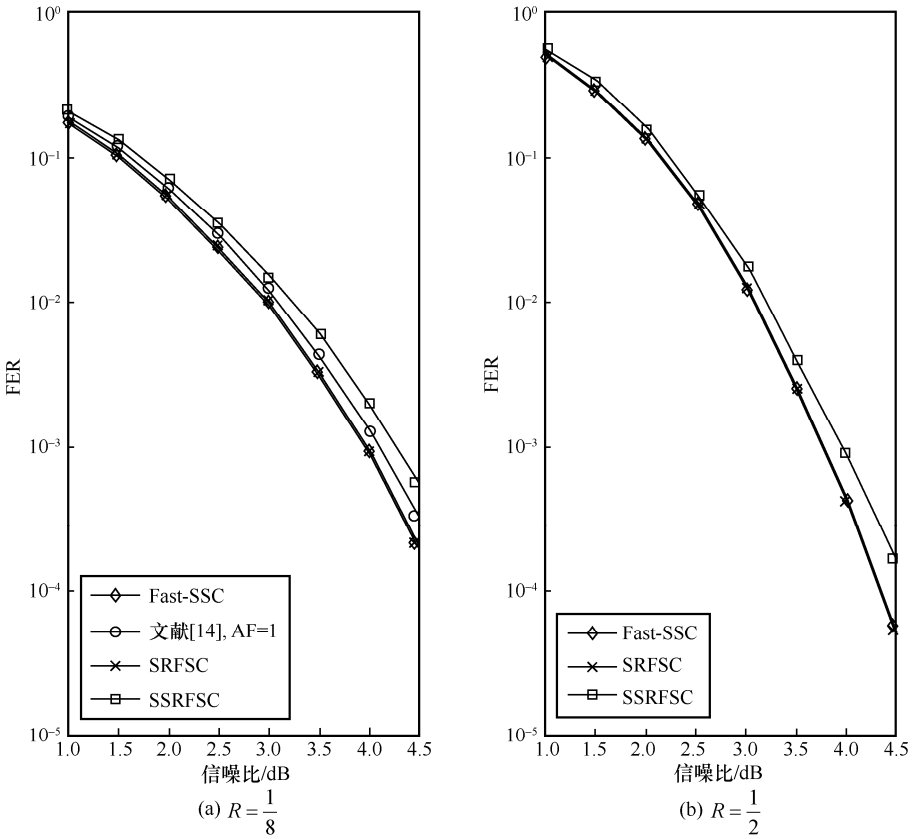
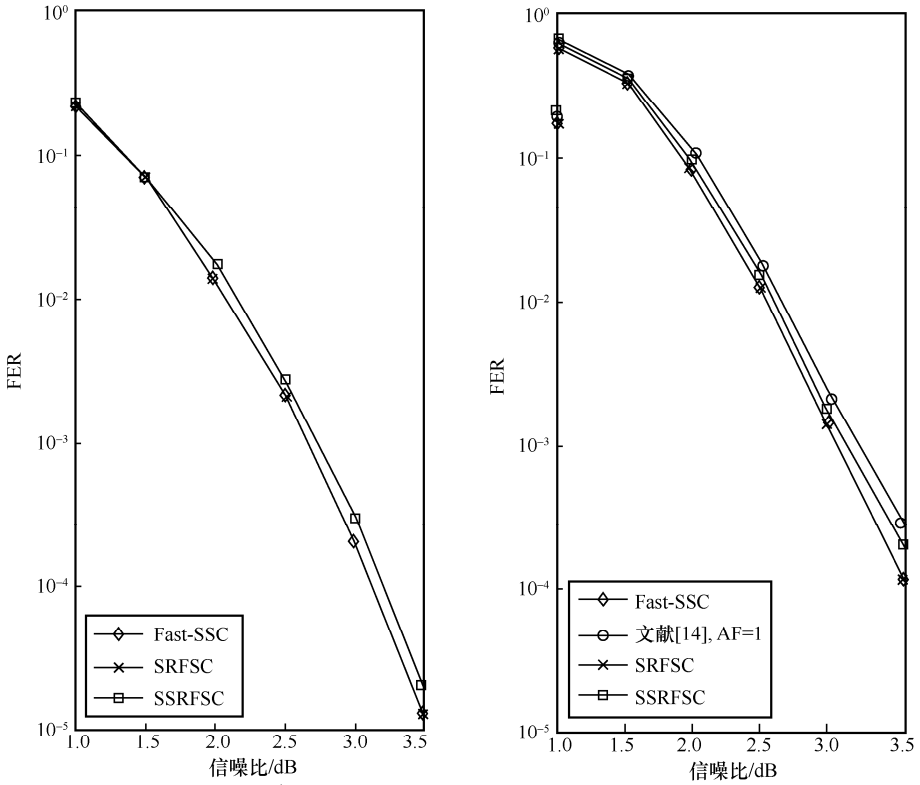


图 8　N=256, R=$\frac{1}{8}$ 和 $\frac{1}{2}$ 时各算法的误帧率曲线



(a) $R=\frac{1}{4}$　　(b) $R=\frac{1}{2}$

图 9　N=1 024, $R=\frac{1}{4}$ 和 $\frac{1}{2}$ 时各算法的误帧率曲线

中的广义 Fast-SSC（AF=1）、SRFSC 以及 SSRFSC 的误帧率（FER）曲线（其中信道可靠度按高斯近似方法选取），其中 AF=1 表示一位额外冻结位。从图 8 和图 9 中可以看出，与 Fast-SSC 和 SRFSC（两者误码性能接近）译码算法相比，SSRFSC 译码算法只有轻微的性能损失，而且始终在可以接受的范围内；随着信噪比的增加，误帧率性能变化趋势几乎固定不变。轻微的性能损失一方面是由于 SR 节点自身的结构特性，一旦其源节点的译码出现错误，必将导致该 SR 节点返回的估计值发生错误，从而导致严重的错误传递；另一方面，当对源节点进行译码时，存在选择错误候选路径的情况，该类错误通常会出现在源节点为 SPC 时，进而影响译码性能。

4.2　译码时延分析

本文根据各个节点译码所需的时间步数来分析译码时延，并遵循文献[11,14]中的假设，即不考虑硬件资源限制，所有可并行化的指令操作均可在一个时间步数内完成；硬判决操作和位操作可以立即执行；节点校验和实数加减均消耗一个时间步数。

SSRFSC 译码算法的译码时延主要分为 2 个部分进行分析。一是简化的 SR 节点部分，表 3 展示了本文所提算法中简化 SR 节点译码时各个步骤所需时间步数。源节点译码时，若为 Rate-1 节点，则不消耗时间步数，若为 SPC 节点，则消耗一个或 2 个时间步数；路径选择时，若候选路径仅为一条，即 $|\boldsymbol{S}|=1$，则不需要路径选择，不消耗时间步数；当 $|\boldsymbol{S}|>1$ 时，路径选择消耗 2 个时间步数。此外，步骤 3 的路径选择可以与步骤 2 的源节点译码及后续运算并行执行。对于源节点为 SPC 的 SR 节点，分析其译码时延得到时间步数为

$$T_{\text{SR-SPC}}=1+\max\{\varphi,\gamma-1\} \tag{26}$$

其中，当该 SPC 节点不需要翻转操作时，φ 取值为一个时间步数；反之，翻转|LLR|最低位额外消耗一个时间步数，φ 取值为 2 个时间步数。

表 3　　简化 SR 节点译码的时间步数

译码步骤	消耗的时间步数
计算源节点 LLR	1
源节点译码	0（Rate-1）
	1 或 2（SPC）
路径选择	0 或 2

γ 的取值与 $|\boldsymbol{S}|$ 大小相关，由于第三步的候选路径选择可以与第二步的源节点译码以及后续的 G 函数并行执行，因此第三步取值为 $\gamma-1$。类似地，对于源节点为 Rate-1 的 SR 节点，译码时间步数为

$$T_{\text{SR-R1}}=1+\max\{0,\gamma-1\} \tag{27}$$

在 SSRFSC 译码中，简化的 SR 节点只包含上述 2 种情况，那么容易得出该类节点需要消耗的译码时延为

$$\sum_{n_{1\leqslant\varphi\leqslant 2}} T_{\text{SR-SPC}}+\sum_{n_{|S|}} T_{\text{SR-R1}} \tag{28}$$

其中，$n_{1\leqslant\varphi\leqslant 2}$ 表示 SPC 节点中 φ 取 1～2 个时间步数时 2 种情况下的节点数量。除去简化的 SR 节点，其余节点只需要分析 SPC、REP 两类节点的译码时延即可（Rate-0、Rate-1 的译码可立即执行），消耗的时间步数为

$$T_{\text{Non-SR}}(n_{\text{SPC}}+n_{\text{REP}}) \tag{29}$$

其中，$T_{\text{Non-SR}}$ 在节点为 SPC 时，取一个或 2 个时间步数；当节点为 REP 时取固定值，为 2 个时间步数，因为 REP 节点的 LLR 求和消耗一个时间步数，节点校验另外消耗一个时间步数。那么所提算法的总体译码时延可表示为

$$\sum_{n_{1\leqslant\varphi\leqslant 2}} T_{\text{SR-SPC}}+\sum_{n_{|S|}} T_{\text{SR-R1}}+T_{\text{Non-SR}}(n_{\text{SPC}}+n_{\text{REP}}) \tag{30}$$

本文考虑了 2 种码长 N={256,1 024}和 3 种码率 $R=\left\{\frac{1}{8},\frac{1}{2},\frac{3}{4}\right\}$，并在加性白高斯噪声（AWGN）信道中采用二进制相移键控（BPSK）调制。选取高斯近似的方法确定可靠比特信道。在译码树中，SR 节点的长度对应于其所处的级别。位于译码树较高级别且具有较大 $|\boldsymbol{S}|$ 的 SR 节点对降低总体时延的贡献更大。例如，当 N=1 024，$R=\frac{1}{2}$ 时，SR 节点数量较多且 $|\boldsymbol{S}|>1$ 比例较大，这将节省更多的译码时延，因为这些节点在译码过程中可以利用更高程度的并行性。

图 10 和图 11 展示了 N=256 和 N=1 024 时不同译码算法的平均译码时间步数。值得注意的是，这些实验数据均是在符合本节假设的条件下得到的。从图 10 和图 11 中可以看出，与传统的 Fast-SSC 译码算法相比，SSRFSC 译码算法可以节省 15%～30%的时间步数；特别是在 N=1 024，$R=\frac{3}{4}$ 的情况下，所提算法减少的时间步数高达 49%，这是此时译码树中简化 SR 节点的数量较多且多数位于译码树的较高层级所导致的；相对于文献[14]中提出的广义 Fast-SSC 译码（AF=1），SSRFSC 译码算法需要的时间步数明显优于文献[14]方法，平均译码时延最多可降低 28%。

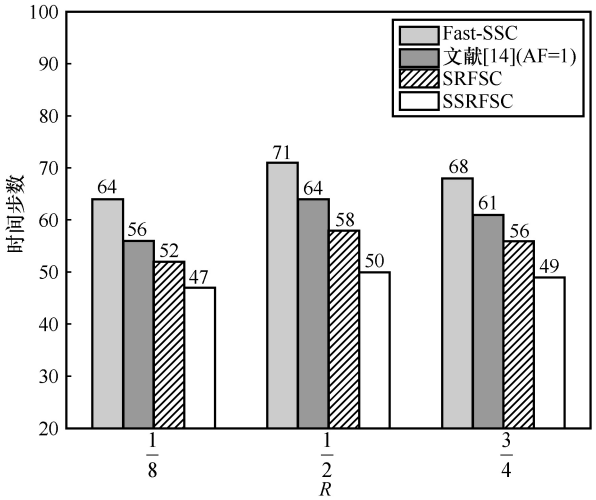


图 10　N=256 时不同译码算法的平均译码步数

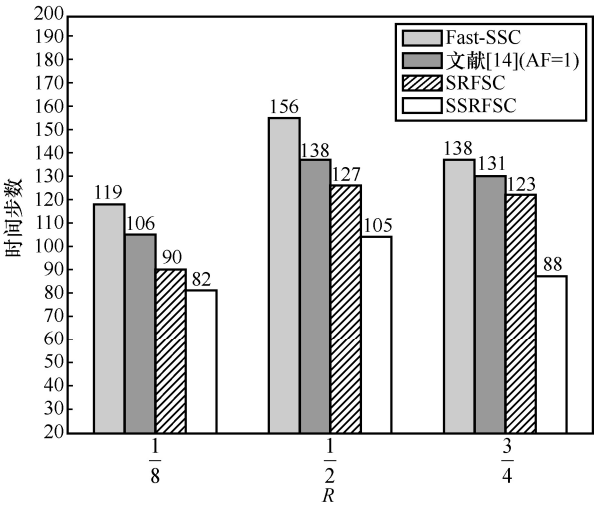


图 11　N=1 024 时不同译码算法的平均译码时间步数

5　结束语

本文提出了一种基于简化 SR 节点的 SSRFSC 译码算法，能够对各种信息位和冻结位的排列模式进行快速译码。在现有 SR 节点的基础上对其进行改进，舍弃了源节点为 EG-PC 和 Rate-C 的情况，将某些出现频率较高的 EG-PC 节点进行分解、合并或简化，从而定义了简化的 SR 节点，该类节点的源节点仅包括 Rate-1 和 SPC 这 2 种情况，但仍包含大多数现有的节点类型；对于 SR 节点以外的节

点，舍弃了传统的 SC 的译码方法，在译码时与 Fast-SSC 算法结合。在译码性能几乎没有损失的情况下，SSRFSC 译码算法与 Fast-SSC 相比译码时延最多可减少 49%；与 SRFSC 译码算法相比，译码时延最多可减少 28%。

参考文献：

[1] ARIKAN E. Channel polarization: a method for constructing capacity-achieving codes[C]//Proceedings of 2008 IEEE International Symposium on Information Theory. Piscataway: IEEE Press, 2008: 1173-1177.

[2] 3GPP TSG RAN WG1. Chairman's notes of agenda item 7.1.5 channel coding and modulation[EB]. 2016.

[3] 3GPP. 5G;NR;multiplexing and channel coding: TS 38.212[S]. 2018.

[4] GAMAGE H, RAJATHEVA N, LATVA-AHO M. Channel coding for enhanced mobile broadband communication in 5G systems[C]//Proceedings of 2017 European Conference on Networks and Communications (EuCNC). Piscataway: IEEE Press, 2017: 1-6.

[5] NIU K, CHEN K, LIN J R, et al. Polar codes: primary concepts and practical decoding algorithms[J]. IEEE Communications Magazine, 2014, 52(7): 192-203.

[6] ZHANG C, YUAN B, PARHI K K. Reduced-latency SC polar decoder architectures[C]//Proceedings of 2012 IEEE International Conference on Communications (ICC). Piscataway: IEEE Press, 2012: 3471-3475.

[7] YUAN B, PARHI K K. Low-latency successive-cancellation polar decoder architectures using 2-bit decoding[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2014, 61(4): 1241-1254.

[8] YUAN B, PARHI K K. Reduced-latency LLR-based SC list decoder for polar codes[C]//Proceedings of the 25th Great Lakes Symposium on VLSI. New York: ACM Press, 2015: 107-110.

[9] YUAN B, PARHI K K. Low-latency successive-cancellation list decoders for polar codes with multibit decision[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2015, 23(10): 2268-2280.

[10] HUSMANN C, NIKOLAOU P C, NIKITOPOULOS K. Reduced latency ML polar decoding via multiple sphere-decoding tree searches[J]. IEEE Transactions on Vehicular Technology, 2018, 67(2): 1835-1839.

[11] ALAMDAR-YAZDI A, KSCHISCHANG F R. A simplified successive-cancellation decoder for polar codes[J]. IEEE Communications Letters, 2011, 15(12): 1378-1380.

[12] SARKIS G, GIARD P, VARDY A, et al. Fast polar decoders: algorithm and implementation[J]. IEEE Journal on Selected Areas in Communications, 2014, 32(5): 946-957.

[13] HANIF M, ARDAKANI M. Fast successive-cancellation decoding of polar codes: identification and decoding of new nodes[J]. IEEE Communications Letters, 2017, 21(11): 2360-2363.

[14] CONDO C, BIOGLIO V, LAND I. Generalized fast decoding of polar codes[C]//Proceedings of IEEE Global Communications Conference (GLOBECOM). Piscataway: IEEE Press, 2019: 1-6.

[15] GAMAGE H, RANASINGHE V, RAJATHEVA N, et al. Low latency decoder for short blocklength polar codes[C]//Proceedings of 2020 European Conference on Networks and Communications (EuCNC). Piscataway: IEEE Press, 2020: 305-310.

[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 2020 IEEE Workshop on Signal Processing Systems (SiPS). Piscataway: IEEE Press, 2020: 1-6.

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

[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，主要研究方向为认知无线通信、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。