应用于轨迹重构的检测级联二进制字符串中删除的最优码

Serge Kas Hanna 数学与系统分析系，理学院，阿尔托大学，芬兰 电子邮箱：serge.kashanna@aalto.fi

arXiv:2304.09839v1 [cs.IT] 19 Apr 2023

摘要——**考虑将两个或更多字符串 $\mathbf{x}^1, \mathbf{x}^2$、.....级联形成 $\mathbf{x} = \langle \mathbf{x}^1, \mathbf{x}^2, \ldots \rangle$。假设每个级联字符串中至多发生 δ次删除。由于删除会改变字符串的长度，一个根本问题是：为了恢复 $\mathbf{x}^1, \mathbf{x}^2$、..... 的边界，我们需要在 $\mathbf{x}$ 中引入多少冗余？这一边界问题等价于设计能够检测每个级联字符串中删除确切数量的码的问题。在本文中，我们首先推导出逆向结果，给出删除检测码冗余的下界，从而回答上述问题。随后，我们提出一种基于标记的码构造，其冗余在所有删除检测码族中是渐近最优的，并且在所有逐块可译码码中达到精确最优。为展示这类删除检测码的实用性，我们将该码应用于轨迹重构，并设计出一种高效的编码重构方案，仅需常数条迹。**

一、引言

我们研究级联二进制字符串中的删除检测问题。更确切地说，我们考虑如下形式的字符串$\mathbf{x} = \langle \mathbf{x}^1, \mathbf{x}^2, \ldots\ldots, \mathbf{x}^{n/\ell} \rangle \in \mathbb{F}_2^n$，它们隐式地划分为$n/\ell$ 个互不相交的子串$\mathbf{x}^1, \mathbf{x}^2, \ldots\ldots, \mathbf{x}^{n/\ell}$，称为块，每个块的大小为 ℓ。假设$\mathbf{x}$ 经删除操作后变为字符串$\mathbf{y}$。我们希望构造一种编码，使其能够根据观测到的字符串$\mathbf{y}$，检测$\mathbf{x}^1, \mathbf{x}^2, \ldots\ldots, \mathbf{x}^{n/\ell}$各块中发生删除的确切数量。这类编码也可以用来确定$\mathbf{y}$中各块的边界。此外，这些编码还有助于定位删除，并识别信息字符串特定部分中是否发生了删除。因此，它们具有多种潜在应用，例如用于片段编辑信道的编码 [3]–[5], 用于编码迹重构的基于标记的构造 [6], 文件同步 [7]；此外，还可用于检测基因组序列部分中的突变，并在顺序传输中保持同步。

大多数文献都集中研究删除纠正问题，并假设解码器已知码字边界。Levenshtein [8]推导出了基本极限，表明要纠正长度为 n 的二进制字符串中的 δ 最坏情况删除，所需冗余位的最优数量为 $\Theta(\delta \log(n/\delta))$。Levenshtein 还证明，由 Varshamov 和 Tenengolts 构造的码（VT码） [9]能够纠正一次删除，并且其冗余在 n中渐近最优。过去几年里，已有大量关于构造删除纠正码这一经典问题的研究例如 [10]–[16]。 [14]–[16] 中的最新成果给出了冗余量为 $\mathcal{O}(\delta \log n)$的码。近期一些研究还探讨了删除以突发形式发生或局限于码字特定部分的情形，例如 [17]–[21]。

如前所述，文献中的上述研究有一个共同要求：解码器必须知道码字边界，才能成功纠正错误。事实上，很容易证明，若将多个单删除纠错 VT 码的码字拼接起来，并通过一个对每个码字最多删除一比特的信道进行传输，那么解码器就无法确定这些码字的边界。换言之，解码器无法检测每个码字中发生的删除数量。因此，本文研究的问题无法通过拼接属于经典删除纠错码的码字来解决。

文献 [22], [23]此前曾在不同于本文所采用定义的意义下研究删除检测码。文献 [23] 对删除检测码的定义如下。若对于任意$\mathbf{x} \in \mathcal{C}$，从$\mathbf{x}$ 中删除任意 $\delta' \leqslant \delta$ 比特，再在末尾附加任意 δ' 比特，这一过程都不会生成 $\mathcal{C}$中的码字，则称码 $\mathcal{C} \subseteq \mathbb{F}_2^n$ 是δ删除检测码。文献 [22] 中的作者也考虑了类似定义，并重点研究非二进制码。我们的定义与文献 [22], [23] 中的定义主要区别在于：我们要求解码器检测出删除的确切数量，而不仅仅是判断某个块中是否发生了删除。更具体地说，我们的（非正式）定义如下。若存在一个解码器，能够在$\mathbf{x} = \langle \mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{n/\ell} \rangle \in \mathcal{C}$受到最坏情况删除的影响后，确定每个块中发生的确切删除数量，则称码 $\mathcal{C}$ 能够检测$\mathbf{x}$ 各块中至多 δ 次删除，当且仅当满足这一条件。前述两种定义的区别对于本文研究的拼接字符串情形至关重要。具体而言，当存在多个拼接块时，检测出给定块中删除的确切数量，可以使解码器确定该块的边界。

本文的主要贡献和组织结构总结如下。第二节给出了该问题的形式化定义。第三节通过推导删除检测码冗余度的下界，研究了该问题的基本极限。这些界表明，对于

本文的部分研究成果曾发表于2021年和2022年IEEE国际信息论研讨会 [1], [2]。

一个码字 $\mathbf{x} = \langle \mathbf{x}^1, \mathbf{x}^2, \ldots\ldots, \mathbf{x}^{n/\ell} \rangle \in \mathbb{F}_2^n$，若要检测每个大小为 ℓ 的块中最多 δ 次删除，所需的冗余至少为 $2\delta(n/\ell - 1)$ 比特。此外，对于一类称为逐块可译码编码的特定高效可译码编码，所需的冗余至少为 $(2\delta + 1)(n/\ell - 1)$ 比特。在第四节中，我们介绍了一种显式的基于标记的码构造，用于检测每个块中最多 δ 次删除。该码可在 $\mathcal{O}(n)$ 线性时间内完成编码和逐块译码，其冗余为 $(2\delta + 1)(n/\ell - 1)$ 比特。因此，所提出的构造在所有逐块可译码编码中是最优的，并且在所有删除检测码中渐近最优。在第五节中，我们利用该码设计了一种新颖的编码迹重构方案，该方案结合了删除检测码的性质和现有的重构算法，从而能够利用常数数量的迹高效地进行轨迹重构。最后，我们在第六节中讨论了一些尚待解决的问题。

II. 问题陈述与符号约定

我们首先介绍本文中使用的一些记号。令 $[n]$ 表示从 1 到 n （含端点）的整数集合，令 $[i, j]$ 表示从 i 到 j （含端点）的整数集合。分别用 $\mathbf{1}^i$ 和 $\mathbf{0}^j$ 表示由 i 个连续的 1 和 j 个连续的零组成的字符串。对于字符串 $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{F}_2^n$，我们用 $x_i, i = 1, 2, \ldots, n$ 表示 $\mathbf{x}$ 的第 i^{th} 比特。我们将 $\mathbf{x}_{[i,j]} = (x_i, x_{i+1}, \ldots, x_j)$ 写作 $\mathbf{x}$ 的子字符串，其中包含从索引 i 到索引 j 的连续比特。我们用记号 $i \mid j$ 表示 j 能被 i 整除，用 $i \nmid j$ 表示 j 不能被 i 整除。本文中的所有对数均以 2 为底。按照标准记法，$\mathbb{Z}^+ = \{1, 2, \ldots\}$ 表示正整数集合，$\mathbb{Z}_i = \{0, 1, \ldots, i-1\}$ 表示严格小于 $i \in \mathbb{Z}^+$ 的非负整数集合，$\mathbb{F}_2^* = \{0, 1\}^*$ 表示有限二进制字符串集合。

定义1。对于给定的 $\ell \in \mathbb{Z}^+$，满足 $\ell \mid n$ 且 $\ell < n$，$\mathbf{x} \in \mathbb{F}_2^n$ 的 j^{th} 块定义为子字符串

$$\mathbf{x}^j \triangleq \mathbf{x}_{[1+(j-1)\ell, j\ell]} = (x_{1+(j-1)\ell}, x_{2+(j-1)\ell}, \ldots, x_{j\ell}),$$

其中 $j \in [n/\ell]$。

我们使用 $\langle \mathbf{a}, \mathbf{b} \rangle$ 表示两个字符串$\mathbf{a}$ 和$\mathbf{b}$的串接。在定义 2 中给出了能够检测拼接字符串中删除的编码的形式化定义。在本文中，我们重点讨论这些字符串为二进制字符串的情况。

定义 2。令 $\mathcal{C}_\delta(\ell, n) \subseteq \mathbb{F}_2^n$满足 $\delta < \ell \leqslant n/2$ 且$\ell \mid n$，为一个长度为 n 的码，其中包含形如以下形式的码字

$$\mathbf{x} = \langle \mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{n/\ell} \rangle,$$

其中，对于所有 $j \in [n/\ell]$，$\mathbf{x}^j \in \mathbb{F}_2^\ell$。假设$\mathbf{x} \in \mathcal{C}_\delta(\ell, n)$在每个块中受到至多 δ 次删除的影响，从而得到字符串$\mathbf{y} \in \mathbb{F}_2^*$。当且仅当存在一个解码函数时，码 $\mathcal{C}_\delta(\ell, n)$ 被称为每个块最多可检测 δ 次删除，该解码函数

$$\mathrm{Dec}(\mathbf{y}) : \mathbb{F}_2^* \to \mathbb{Z}_{\delta+1}^{n/\ell},$$

该输出每个块中发生的删除的确切数量$\mathbf{x}^j$，适用于所有$\mathbf{x}$ 和 $\mathbf{y}$。

请注意，定义 2 中关于检测每个块的确切删除数量的译码要求，等价于确定观测字符串$\mathbf{y}$中每个块的边界。

例 1. 设$\mathbf{x}$是一个长度为20 的二进制字符串，给定为

$$\mathbf{x} = \overbrace{01\underline{0}11}^{\mathbf{x}^1}\overbrace{01001}^{\mathbf{x}^2}\overbrace{01111}^{\mathbf{x}^3}\overbrace{\underline{1}1101}^{\mathbf{x}^4},$$

其中，$\mathbf{x}$ 由四个长度均为5 的块$\mathbf{x}^1, \mathbf{x}^2$,$\mathbf{x}^3$, 和$\mathbf{x}^4$ 串接而成。假设$\mathbf{x}$ 中带下划线的比特被删除，得到

$$\mathbf{y} = 01110100101111101.$$

若$\mathbf{x} \in \mathcal{C}_1(5,20)$，其中 $\mathcal{C}_1(5,20)$ 是一个满足定义2 的编码，且具有 $\delta = 1$、$\ell = 5$和 $n = 20$，那么解码器的输出应为

$$\mathrm{Dec}(\mathbf{y}) = (1, 0, 1, 1).$$

根据解码器的输出可知，块的边界如下

$$\mathbf{y} = \overbrace{0111}^{block\ 1} | \overbrace{01001}^{block\ 2} | \overbrace{0111}^{block\ 3} | \overbrace{1101}^{block\ 4}.$$

接下来，我们定义一类特殊的删除检测码，称为逐块可译码（定义 3）。这类编码的主要性质是，可以通过每次处理 ℓ比特，以高效的顺序方式进行解码，即无需任何前瞻。

定义 3. 设 $\mathcal{C}_\delta(\ell, n)$ 是一个满足定义2的编码。考虑一个码字$\mathbf{x} \in \mathcal{C}_\delta(\ell, n)$，其每个块中至多有$\delta$ 个比特被删除，得到$\mathbf{y}$。设$\mathbf{y}$中块 j 的起始位置为 α_j，且 $\alpha_1 = 1$。假设已知 α_j，那么编码 $\mathcal{C}_\delta(\ell, n)$ 称为逐块可译码，当且仅当存在一个解码器，只需处理$\mathbf{y}^{[\alpha_j, \alpha_j+\ell-1]}$中的比特，就能输出$\mathbf{x}^j$ 中任意块被删除的确切数量。

按照标准记法，$f(n) = o(g(n))$ 表示 $\lim_{n\to\infty} f(n)/g(n) = 0$；$f(n) = \omega(g(n))$ 表示 $\lim_{n\to\infty} f(n)/g(n) = \infty$；$f(n) = \Omega(g(n))$ 表示f 对于某个常数 $c_1 > 0$，渐近地以下界 $c_1 g(n)$ 为界；$f(n) = \mathcal{O}(g(n))$ 表示 f 对于某个常数 $c_2 > 0$，渐近地以上界 $c_2 g(n)$ 为界；$f(n) = \Theta(g(n))$ 表示 $f(n)/g(n)$渐近地位于某个区间 $[c_1, c_2]$ 内，其中 c_1、$c_2 > 0$ 为常数；而$f(n) = \mathrm{poly}(g(n))$ 表示 $f(n) = g(n)^{\mathcal{O}(1)}$。

III. 删除检测码冗余的下界

本节中，我们推导该问题的逆向结果。结果陈述于第三节A，证明见第三节B。

A. 结果

令 $|\mathcal{C}_\delta(\ell,n)|$ 表示码 $\mathcal{C}_\delta(\ell,n)$ 和 $r_\delta(\ell,n)$ 的大小，$\triangleq n-\log|\mathcal{C}_\delta(\ell,n)|$ 表示其冗余。在定理 1 中，我们给出任何能够检测每个块中至多 δ 次删除的码 $\mathcal{C}_\delta(\ell,n)$ 的冗余下界（定义 2）。在定理 2 中，我们具体研究逐块可解码的码（定义 3），并给出这类码的冗余下界。

定理 1. 对于 $\delta,\ell,n\in\mathbb{Z}^+$，满足 $2\delta<\ell\leqslant n/2$ and$\ell\mid n$，任何能够检测每个块中至多 δ 次删除的码 $\mathcal{C}_\delta(\ell,n)\subseteq\mathbb{F}_2^n$ 的冗余 $r_\delta(\ell,n)$ 满足：

(i) If $n/\ell=2$, *then*

$$r_\delta(n/2,n)\geqslant 2\delta;$$

(ii) 若 $n/\ell\geqslant 3$ *and* $2\delta\nmid\ell$，则

$$r_\delta(\ell,n)\geqslant 2\delta(n/\ell-1);$$

(iii) 若 $n/\ell\geqslant 3$ *and* $2\delta\mid\ell$，则

$$r_\delta(\ell,n)\geqslant(2\delta+\varepsilon)(n/\ell-1)-\varepsilon,$$

其中 $0<\varepsilon\leqslant 0.415$ 是由 δ 和 ℓ 确定的函数，定义为

$$\varepsilon_\delta(\ell)=\log\left(\frac{2^{\ell-2\delta}}{2^{\ell-2\delta}-1}\right).$$

定理 2. 对于 $\delta,\ell,n\in\mathbb{Z}^+$，满足 $2\delta<\ell\leqslant n/2$ and$\ell\mid n$，能够检测每个块中至多 δ 次删除的逐块可译码码 $\mathcal{C}_\delta(\ell,n)\subseteq\mathbb{F}_2^n$ 的冗余 $r_\delta(\ell,n)$ 满足

$$r_\delta(\ell,n)\geqslant(2\delta+1)(n/\ell-1).$$

讨论：定理 1 的结果表明，在只有两个拼接字符串，即 $n/\ell=2$ 的情况下，为了检测每个块中最多 δ 次删除，码的冗余必须至少为 2δ 比特。更具体地说，根据下一节中给出的定理 1 的证明可知，第一个块和第二个块中的冗余都必须至少为 δ，因此总冗余至少为 2δ。

在有三个或更多块且 2δ 不能整除 ℓ 的情况下，码的冗余必须至少为 $2\delta(n/\ell-1)$ 比特。证明还表明，码冗余的这一上界由以下块级要求共同构成：*(i)*第一个块和最后一个块的冗余都必须至少为 δ[1]；*(ii)* 其他每个块的冗余必须至少为 2δ。而在 2δ 能整除 ℓ 的情况下，码的冗余必须严格大于 $2\delta(n/\ell-1)$ 比特。

此外，对于逐块可译码的编码，根据定理 2 的证明可知，每个块的冗余必须至少为 $2\delta+1$，但第一个块和最后一个块除外，它们所需的冗余分别至少为 δ 和 $\delta+1$。

[1]块 $\mathbf{x}^1$ 和 $\mathbf{x}^{n/\ell}$ 所需的冗余较少，因为它们只有一个需要保护边界的“相邻”块。

B. 定理1和定理2的证明

定理 1 和定理 2 的证明有一部分是相同的：我们证明，对于任意码字 $\mathbf{x}\in\mathcal{C}_\delta(\ell,n)$，块 $\mathbf{x}^1,\ldots,\mathbf{x}^{n/\ell-1}$ 的最后 δ 比特以及块 $\mathbf{x}^2,\ldots,\mathbf{x}^{n/\ell}$ 的前 δ 比特必须预先确定（固定）。这一必要条件给出了任意码 $\mathcal{C}_\delta(\ell,n)$ 冗余的初步下界，其中 $2\delta<\ell\leqslant n/2$，即

$$r_\delta(\ell,n)\geqslant 2\delta(n/\ell-1). \tag{1}$$

然后，我们证明，如果 $n/\ell\geqslant 3$ and $2\delta\mid\ell$，那么对于每个块 $\mathbf{x}^j$ for$j\in[2,n/\ell-1]$，还必须满足一个额外的必要条件，这将给出

$$r_\delta(\ell,n)\geqslant 2\delta(n/\ell-1)+\varepsilon(n/\ell-2), \tag{2}$$
$$=(2\delta+\varepsilon)(n/\ell-1)-\varepsilon, \tag{3}$$

其中，$\varepsilon>0$ 是一个关于 δ 和 ℓ 的函数，具体形式有待确定。对于定理 2，我们专门考虑定义 3 中定义的逐块可译码编码。我们证明，对于此类编码，式 (1) 中的界可以改进为

$$r_\delta(\ell,n)\geqslant(2\delta+1)(n/\ell-1). \tag{4}$$

我们首先证明，对于每个码字 $\mathbf{x}\in\mathcal{C}_\delta(\ell,n)$，任意块 $\mathbf{x}^j$，$j\in[n/\ell-1]$ δ $x_1^{j+1}\ \ldots\ x_\delta^{j+1}$ 中最后 δ 比特 $x_{\ell-\delta+1}^j,\ldots,x_\ell^j$ 中的每个比特，都必须与块 $\mathbf{x}^{j+1}$的所有比特 ,,不同。假设对于某些 $i_1\in[\ell-\delta+1,\ell]$ 和$i_2\in[\delta]$，有 $x_{i_1}^j=x_{i_2}^{j+1}$。考虑如下删除组合：在$\mathbf{x}^j$中删除最后 $\ell-i_1+1$ 个比特 $x_{i_1}^j,\ldots,x_\ell^j$，并在$\mathbf{x}^{j+1}$ 中删除最前面的 i_2-1 个比特 $x_1^{j+1},\ldots,x_{i_2-1}^{j+1}$（如果 $i_2=1$，则不在 $\mathbf{x}^{j+1}$ 中删除任何比特）。所得字符串具有如下形式。

$$\mathbf{y}_1=(\cdots,x_{i_1-1}^j,x_{i_2}^{j+1},x_{i_2+1}^{j+1},\cdots,x_\delta^{j+1},\cdots,x_\ell^{j+1},\cdots),$$

其中其他所有块均未受删除影响。现在考虑另一种删除组合：在$\mathbf{x}^j$中删除最后 $\ell-i_1$比特 $x_{i_1+1}^j,\ldots,x_\ell^j$，并在$\mathbf{x}^{j+1}$ 中删除最前面的 i_2 比特$x_1^{j+1},\ldots,x_{i_2}^{j+1}$（如果$\mathbf{x}^j$中没有删除任何比特，则为此情形）。所得字符串的形式为

$$\mathbf{y}_2=(\cdots,x_{i_1-1}^j,x_{i_1}^j,x_{i_2+1}^{j+1},\cdots,x_\delta^{j+1},\cdots,x_\ell^{j+1},\cdots),$$

其中其他所有块均未受删除影响。由于$x_{i_1}^j=x_{i_2}^{j+1}$ 根据假设成立，因此我们有$\mathbf{y}_1=\mathbf{y}_2$。然而，$\mathbf{y}_1$ 和$\mathbf{y}_2$ 分别对应于由下式给出的两种不同删除组合（即两种不同的解码器输出）：

$$\mathrm{Dec}(\mathbf{y}_1)=(0,\ldots,0,\underbrace{\ell-i_1+1}_{j},\underbrace{i_2-1}_{j+1},0,\ldots,0)\in\mathbb{Z}_{\delta+1}^{n/\ell},$$
$$\mathrm{Dec}(\mathbf{y}_2)=(0,\ldots,0,\underbrace{\ell-i_1}_{j},\underbrace{i_2}_{j+1},0,\ldots,0)\in\mathbb{Z}_{\delta+1}^{n/\ell}.$$

因此，若对于任意的 $i_1\in[\ell-\delta+1,\ell]$ 和 $i_2\in[\delta]$，都有 $x_{i_1}^j=x_{i_2}^{j+1}$，那么就存在两个不同的解码器输出对应于同一个解码器输入，即 $\mathbf{y}_1=\mathbf{y}_2$，且 $\mathrm{Dec}(\mathbf{y}_1)\neq\mathrm{Dec}(\mathbf{y}_2)$。这与定义 2 中给出的解码函数定义相矛盾。由此可知，以下条件是必要的：

$$x_{i_1}^j\neq x_{i_2}^{j+1},\ \forall\, i_1\in[\ell-\delta+1,\ell],i_2\in[\delta],\text{and } j\in[n/\ell-1]. \tag{8}$$

$$\mathbf{y}_1 = (\cdots, \underbrace{x^j_{\ell-\delta}}_{\ell-\delta}, \underbrace{x^{j+1}_{\delta+1}, \cdots, x^{j+1}_{3\delta}}_{\delta}, \underbrace{x^{j+1}_{3\delta+1}, \cdots, x^{j+1}_{5\delta}}_{\delta}, \cdots\cdots\cdots, \underbrace{x^{j+1}_{\ell-3\delta+1}, \cdots, x^{j+1}_{\ell-\delta}}_{\delta}, \underbrace{x^{j+1}_{\ell-\delta+1}, \cdots, x^{j+2}_{\delta}}_{\delta}, x^{j+2}_{\delta+1}\cdots). \quad (5)$$

$$\mathbf{y}_2 = (\cdots, \underbrace{x^j_{\ell-\delta}}_{\ell-\delta}, \underbrace{x^j_{\ell-\delta+1}, \cdots, x^{j+1}_{\delta}}_{\delta}, \underbrace{x^{j+1}_{\delta+1}, \cdots, x^{j+1}_{3\delta}}_{\delta}, \cdots\cdots\cdots, \underbrace{x^{j+1}_{\ell-3\delta+1}, \cdots, x^{j+1}_{\ell-2\delta}}_{\delta}, \underbrace{x^{j+1}_{\ell-3\delta+1}, \cdots, x^{j+1}_{\ell-\delta}}_{\delta}, x^{j+2}_{\delta+1}\cdots). \quad (6)$$

$$(\underbrace{x^{j+1}_{\delta+1}, \cdots, x^{j+1}_{3\delta}}_{2\delta\text{ 比特}}, \underbrace{x^{j+1}_{3\delta+1}, \cdots x^{j+1}_{5\delta}}_{2\delta\text{ 比特}}, \cdots\cdots\cdots \underbrace{x^{j+1}_{\ell-3\delta+1}, \cdots, x^{j+1}_{\ell-\delta}}_{2\delta\text{ 比特}}) \neq \langle \underbrace{\mathbf{1}^\delta, \mathbf{0}^\delta}_{2\delta\text{ 比特}}, \underbrace{\mathbf{1}^\delta, \mathbf{0}^\delta}_{2\delta\text{ 比特}}, \cdots\cdots\cdots, \underbrace{\mathbf{1}^\delta, \mathbf{0}^\delta}_{2\delta\text{ 比特}} \rangle. \quad (7)$$

由于我们关注的是二进制码，前述约束意味着每个块 $\mathbf{x}^j$, $j \in [n/\ell - 1]$ 的最后 δ 比特必须相同，且必须不同于块 $\mathbf{x}^{j+1}$ 的前 δ 比特，而后者也应相同。也就是说，码 $\mathcal{C}_\delta(\ell, n)$ 的码字只能满足 *(i)* $\mathbf{x}^j_{[\ell-\delta+1,\ell]} = \mathbf{1}^\delta$ 和 $\mathbf{x}^{j+1}_{[1,\delta]} = \mathbf{0}^\delta$; 或 *(ii)* $\mathbf{x}^j_{[\ell-\delta+1,\ell]} = \mathbf{0}^\delta$ 和 $\mathbf{x}^{j+1}_{[1,\delta]} = \mathbf{1}^\delta$。

接下来，我们证明，尽管前述结论成立，码 $\mathcal{C}_\delta(\ell, n)$ 不可能包含一对码字 $\mathbf{x}_1$,$\mathbf{x}_2 \in \mathcal{C}_\delta(\ell, n)$，其中 $\mathbf{x}_1$ 满足 *(i)*，且$\mathbf{x}_2$ 满足*(ii)*。为证明这一点，我们假设存在这样一对$\mathbf{x}_1$, $\mathbf{x}_2 \in \mathcal{C}_\delta(\ell, n)$，然后证明在这种情况下，会有两个不同的解码器输出对应于同一个输入字符串$\mathbf{y}$，这与定义 2 中解码函数的定义矛盾。

考虑给定的 $j \in [n/\ell - 1]$，以下两种删除组合。在第一种组合中，从$\mathbf{x}^j_1$的最后 δ 个比特中删除 $\delta - 1$ 个比特，从 $\mathbf{x}^{j+1}_1$的前 δ 个比特中删除比特，而其他块中不删除任何比特。在第二种组合中，删除$\mathbf{x}^j_2$的最后 δ 个比特，从$\mathbf{x}^{j+1}_2$的前 δ 个比特中删除 $\delta - 1$ 个比特，而其他块中不删除任何比特。在这两种组合中，所得字符串为

$$\mathbf{y}_1 = \mathbf{y}_2 = (\cdots, x^j_{\ell-\delta}, 1, x^{j+1}_{\delta+1}, x^{j+1}_{\delta+2}, \cdots, x^{j+1}_{\ell}\cdots),$$

由于 $\mathbf{y}_1$ 和 $\mathbf{y}_2$ 对应于两个不同的解码器输出

$$\mathrm{Dec}(\mathbf{y}_1) = (0, \ldots, 0, \underbrace{\delta - 1}_{j}, \underbrace{\delta}_{j+1}, 0, \ldots, 0) \in \mathbb{Z}^{n/\ell}_{\delta+1},$$

$$\mathrm{Dec}(\mathbf{y}_2) = (0, \ldots, 0, \underbrace{\delta}_{j}, \underbrace{\delta - 1}_{j+1}, 0, \ldots, 0) \in \mathbb{Z}^{n/\ell}_{\delta+1},$$

于是我们有 $\mathbf{y}_1 = \mathbf{y}_2$，其中 $\mathrm{Dec}(\mathbf{y}_1) \neq \mathrm{Dec}(\mathbf{y}_2)$，这与定义 2 矛盾。

因此，我们得出结论：每个块 $\mathbf{x}^j$, $j \in [2, n/\ell - 1]$ 中的前 δ 比特和最后 δ 比特必须预先确定，并且 $\mathbf{x}^1$ 的最后 δ 比特和 $\mathbf{x}^{n/\ell}$ 的前 δ 比特必须预先确定。因此，

$$|\mathcal{C}_\delta(\ell, n)| \leqslant 2^{n - 2\delta(n/\ell - 1)},$$

这给出了式 (1) 中冗余的下界。

鉴于上述约束，接下来我们证明，对于 $n/\ell \geqslant 3$，如果 $2\delta \mid \ell$，则无法达到式 (1) 中的界限，即恰好为 $2\delta(n/\ell - 1)$ 比特的冗余不足以进行解码。

对于 $n/\ell \geqslant 3$ 和给定的 $j \in [n/\ell - 2]$，考虑以下两种删除组合。第一种中，删除$\mathbf{x}^j$中的最后 δ 个比特，删除$\mathbf{x}^{j+1}$中的前 δ 个比特，不删除其他比特。第二种中，删除$\mathbf{x}^{j+1}$中的最后 δ 个比特，删除 $\mathbf{x}^{j+2}$中的前 δ 个比特，并且不删除其他比特。若 $2\delta \mid \ell$，所得字符串$\mathbf{y}_1$ 和$\mathbf{y}_2$分别具有式 (5) 和 (6) 所给出的形式。如前所述，由于这两种删除组合不同，必有$\mathbf{y}_1 \neq \mathbf{y}_2$。不失一般性，假设 $(x^j_{\ell-\delta+1}, \cdots, x^{j+1}_{\delta}) = \langle \mathbf{1}^\delta, \mathbf{0}^\delta \rangle$。在这种情况下，由式 (5) 和 (6) 可知，条件 $\mathbf{y}_1 \neq \mathbf{y}_2$ 等价于式 (7)。结合式 (7) 中的条件，每个$\mathbf{x}^2, \ldots, \mathbf{x}^{n/\ell-1}$ 块可以取值的总数至多为 $2^{\ell-2\delta} - 1$。容易证明，这意味着这些块各自的冗余至少为 $2\delta + \varepsilon_\delta(\ell)$，其中

$$\varepsilon_\delta(\ell) \triangleq \log\left(\frac{2^{\ell-2\delta}}{2^{\ell-2\delta} - 1}\right) > 0.$$

注意到 $\varepsilon_\delta(\ell)$ 是关于 $\ell - 2\delta$ 的递减函数，因此当 $\ell - 2\delta$ 取其可能的最小值时，该函数的值最大。由于 $\ell > 2\delta$ 和 $2\delta \mid \ell$，我们有 $\ell - 2\delta \geqslant 2\delta$。因此，$\ell - 2\delta$ 的可能最小值等于 2，当 $\ell = 4\delta$ 和 $\delta = 1$ 时可取到。因此，$\varepsilon_\delta(\ell) \leqslant \varepsilon_1(4) = 0.415$。结合前面的推导与 (3)，定理 1 的证明得证。

接下来，我们具体考虑逐块解码的码，并证明对于这类码，除了式（8）中的约束外，还必须满足

$$x^{j+1}_{\delta+1} \neq x^j_{i_1}, \forall\ i_1 \in [\ell - \delta + 1, \ell] \text{ and } j \in [n/\ell - 1]. \quad (9)$$

也就是说，式（9）将式（8）中的必要条件扩展到了 $i_2 \in [\delta + 1]$。由式（8）可知，对于所有 $j \in [n/\ell - 1]$，比特 $x^j_{\ell-\delta+1}$、……、x^j_ℓ的取值都相同。不失一般性，假设 $\mathbf{x}^j_{[\ell-\delta+1,\ell]} = \mathbf{1}^\delta$，并设 $x^{j+1}_{\delta+1} = 1$ 的取值与这些比特相同。考虑以下两种删除组合。第一种中，在 $\mathbf{x}^j$的最后 δ 个比特中仅删除一个比特，并在$\mathbf{x}^{j+1}$中删除前 δ 个比特。第二种中，在$\mathbf{x}^j$中不删除任何比特。设 α_j 为$\mathbf{y}$中块 j 的起始位置，并假设解码器已知 α_j 的取值。对于这两种组合，我们有

$$\mathbf{y}_{[\alpha_j, \alpha_j + \ell - 1]} = \langle \mathbf{x}^j_{[\ell-\delta]}, \mathbf{1}^\delta \rangle.$$

因此，解码器仅通过处理$\mathbf{x}^j$ 中的 ℓ 比特，无法确定$\mathbf{y}_{[\alpha_j, \alpha_j + \ell - 1]}$中删除的确切数量（0 或 1）。这与定义 3 中给出的逐块可译码编码的定义相矛盾。因此，条件 (9) 对所有逐块可译码编码都是必要的。该条件使$\mathbf{x}^2, \ldots, \mathbf{x}^{n/\ell}$中的每个块都额外增加 1 比特的冗余。因此，为完成定理 2 的证明，我们在 (1) 的右侧加上 $n/\ell - 1$，得到界 (4)。

IV. 最优删除检测码

本节将说明，定理 2 中的下界可以通过一个基于标记的码构造取到等号；该构造将在第四节B（构造 1）中给出。这个最优码的性质将在第四节A中讨论。

A. 结果

在定理 3 中，我们陈述了关于显式码$\mathcal{D}_\delta(\ell,\ n)$的结果，该码是在第四节B中构造的。定理的证明见第四节E。

定理 3. 对于 δ、ℓ、$n \in \mathbb{Z}^+$，其中 $2\delta < \ell \leqslant n/2$，令

$$\mathbf{x} = \langle \mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{n/\ell} \rangle \in \mathcal{D}_\delta(\ell, n).$$

假设$\mathbf{x}$ 的每个块中至多有 δ 次删除，即$\mathbf{x}^1$、.....、$\mathbf{x}^{n/\ell}$。构造**1**中给出的码 $\mathcal{D}_\delta(\ell, n)$ 可检测每块至多 δ 次删除。该码可在线性时间 $\mathcal{O}(n)$内编码并逐块译码，其冗余为 ($2\delta+1)(n/\ell-1)$ 比特。

讨论： 定理 3 表明，我们在构造**1**中提出的码 $\mathcal{D}_\delta(\ell,\ n)$ 可高效编码和译码，且冗余为 $(2\delta+1)(n/\ell-1)$ 比特。正如我们将在第四节B中所示， $\mathcal{D}_\delta(\ell,\ n)$ 的总冗余由以下部分构成：*(i)* 每块$\mathbf{x}^j$有 $2\delta+1$ 个冗余位，其中 $j \in [2, n/\ell-1]$; *(ii)* $\mathbf{x}^1$有 δ 个冗余位；*(iii)* $\mathbf{x}^{n/\ell}$有 $\delta+1$ 个冗余位。这意味着每块的冗余仅取决于 δ，并且相对于块大小 ℓ 和码字大小 n而言是常数。请注意，如果我们还希望每块能够纠正 δ 次删除，那么每块的冗余需要达到 $\Omega(\delta\log(\ell/\delta))$ [8], ，即至少关于 ℓ呈对数增长。

此外， $\mathcal{D}_\delta(\ell,\ n)$ 是逐块可译码的，因此由定理 2 可知， $\mathcal{D}_\delta(\ell,\ n)$ 在所有逐块可译码的编码中具有最优 冗余。定理 1 给出了任意码 $\mathcal{C}_\delta(\ell,\ n)$ 的冗余下界，其中该码不一定是逐块可译码的，并且每个块最多可检测 δ 次删除。此外，通过将定理 1 中的下界与定理 3 中码的冗余进行比较，可以容易地看出，对于固定数量的块，所提出的码 $\mathcal{D}_\delta(\ell,\ n)$ 在 δ 中所有删除检测码族 $\mathcal{C}_\delta(\ell,\ n)$ 之中具有渐近最优冗余。

B. 码构造

我们提出的用于检测每个块最多 δ 次删除的码由以下构造给出。

构造1 （检测最多 δ 次删除）。 对于所有 δ、ℓ、$n \in \mathbb{Z}^+$，当 $2\delta < \ell \leqslant n/2$ 时，我们定义如下

$$\mathcal{A}_\delta^0(\ell) \triangleq \left\{\mathbf{x} \in \mathbb{F}_2^\ell \;\middle|\; \mathbf{x}_{[1,\delta+1]} = \mathbf{0}^{\delta+1}\right\},$$
$$\mathcal{A}_\delta^1(\ell) \triangleq \left\{\mathbf{x} \in \mathbb{F}_2^\ell \;\middle|\; \mathbf{x}_{[\ell-\delta+1,\ell]} = \mathbf{1}^{\delta}\right\}.$$

删除检测码 $\mathcal{D}_\delta(\ell, n) \subseteq \mathbb{F}_2^n$ 由以下集合给出

$$\left\{ \langle \mathbf{x}^1, \ldots, \mathbf{x}^{n/\ell} \rangle \;\middle|\; \begin{array}{l} \mathbf{x}^1 \in \mathcal{A}_\delta^1(\ell), \\ \mathbf{x}^j \in \mathcal{A}_\delta^1(\ell) \cap \mathcal{A}_\delta^0(\ell), \forall j \in [2, \frac{n}{\ell}-1], \\ \mathbf{x}^{n/\ell} \in \mathcal{A}_\delta^0(\ell). \end{array} \right\}.$$

在证明定理 3 之前，我们将在 IV-C 节介绍解码算法的步骤，并在 IV-D 节给出一个关于 $\delta = 1$ 的示例。由于只需将信息位设置为 $[n]$ 中不受构造 1 限制的位置即可完成编码，因此省略编码算法。

C. 解码

假设 $\mathbf{x} \in \mathcal{D}_\delta(\ell, n)$（其中 $2\delta < \ell \leqslant n/2$）在每个块 $\mathbf{x}^j$ 中至多受到 δ 次删除的影响，对于所有 $j \in [n/\ell]$，从而得到 $\mathbf{y} \in \mathbb{F}_2^*$。解码器的输入是 $\mathbf{y}$，输出为（$\delta_1, \delta_2, \ldots,$ $\delta_{n/\ell}$） $\in \mathbb{Z}_{\delta+1}^{n/\ell}$，其中 $\delta_j \leqslant \delta$ 表示块 j中发生的删除次数。

解码按逐块方式进行，因此在解码块 j 之前，我们知道前面的 $j-1$个块已被正确解码。因此，在解码完前 $j-1$ 个块后，解码器知道块 j 在 $\mathbf{y}$ 中的正确起始位置。设块$j \in [n/\ell]$ 在 $\mathbf{y}$ 中的起始位置为 α_j，其中 $\alpha_1 = 1$；并且 $\forall j \in [n/\ell-1]$ 令

$$\mathbf{s}^j \triangleq \mathbf{y}_{[\alpha_j+\ell-\delta, \alpha_j+\ell-1]}.$$

若要解码块 j，解码器会从左到右扫描$\mathbf{s}^j$ 中的比特，查找第一个 0（如果存在）。如果$\mathbf{s}^j$ 中没有零，即$\mathbf{s}^j = \mathbf{1}^\delta$，则解码器判定块 j中没有发生删除（$\delta_j = 0$），并将块 $j+1$ 的起始位置设为 $\alpha_{j+1} = \alpha_j + \ell$。否则，若$\mathbf{s}^j$ 中的第一个 0 位于位置 β_j，且 $1 \leqslant \beta_j \leqslant \delta$，则解码器判定块 j中发生了 $\delta_j = \delta - \beta_j + 1$ 次删除，并将块 $j+1$ 的起始位置设为 $\alpha_{j+1} = \alpha_j + \ell - \delta_j$。解码器对每个块重复此过程，直到前 $n/\ell-1$ 个块解码完毕。最后，解码器根据最后一个块在$\mathbf{y}$ 中的起始位置 $\alpha_{n/\ell}$检查其长度，并相应地输出 $\delta_{n/\ell}$。注意，此解码器满足定义 3，因为$\mathbf{s}^j$中最后一比特的索引 $\alpha_j + \ell - 1$与块 j的起始位置相距 ℓ个位置 α_j。

注 1. 通过在由构造 1 限制的位置将所有零翻转为一、将所有一翻转为零，可以得到一个与 $\mathcal{D}_\delta(\ell, n)$ 等价的码。上述解码算法也可以作相应修改。然而，该解码算法不能用于组合上述两种构造中的码字所得的码。

注 2. 在我们的错误模型中，我们假设每个块中最多存在 δ 个最坏情况删除。在此模型下，我们在第 IV-E 节证明上述解码方案的正确性。请注意，如果任意给定块中的删除数量超过 δ，则该块以及逐块解码器随后处理的所有其他块都将解码失败。

D. $\delta = 1$ 的示例

接下来，我们给出一个适用于 $\delta = 1, \ell = 5,$ 和$n = 20$ 的码示例。考虑如下给出的码字 $\mathbf{x} \in \mathcal{D}_1$(5,20)：

$$\mathbf{x} = \overbrace{10\underline{1}01}^{\mathbf{x}^1}\ \overbrace{00111}^{\mathbf{x}^2}\ \overbrace{0001\underline{1}}^{\mathbf{x}^3}\ \overbrace{\underline{0}0100}^{\mathbf{x}^4}.$$

假设 $\mathbf{x}$ 中带下划线的比特被删除，结果为

$$\mathbf{y} = 10010011100010100.$$

为了确定第一个块中的删除数量，解码器首先检查$\mathbf{s}^1 = y_5 = 0$，由此可知 $\beta_1 = 1$，因此判定$\mathbf{x}^1$中发生了 $\delta_1 = 1$ 次删除。因此，将块 2 的起始位置设为 $\alpha_2 = 5$。然后，由于 $\mathbf{s}^2 = y_9 = 1$，解码器判定$\mathbf{x}^2$, 中没有发生删除（$\delta_2 = 0$），并将块 3 的起始位置设为 $\alpha_3 = 10$。类似地，我们有$\mathbf{s}^3 = y_{14} = 0$，这意味着$\mathbf{x}^3$中发生了 $\delta_3 = 1$ 次删除，$\beta_3 = 1$, 并且$\alpha_4 = 14$。现在，由于最后一个块还剩下 $\ell - 1 = 4$ 比特，解码器判定$\mathbf{x}^4$中发生了 $\delta_4 = 1$ 次删除。

E. 定理3的证明

码 $\mathcal{D}_\delta(\ell, n)$ 的冗余度取决于构造1中固定的比特数量，即 $(2\delta + 1)(n/\ell - 1)$。编码只需将信息位设置为 $[n]$ 中未受构造1限制的位置即可。因此，编码和解码算法的复杂度均为 $\mathcal{O}(n)$，因为它们只需对比特进行一次遍历，并执行常数时间操作。接下来，我们证明解码算法的正确性。

考虑一个码字 $\mathbf{x} \in \mathcal{D}_\delta(\ell, n)$，其每个块至多受到 δ 次删除的影响，从而得到字符串 $\mathbf{y}$。由于解码是逐块进行的，因此只需证明当我们有 $n/\ell = 2$ 个串联块时，算法的正确性即可。为了证明码 $\mathcal{D}_\delta(\ell, n)$ 能够检测每个块中至多 δ 次删除的情况，我们对 $\delta \in \mathbb{Z}^+$ 使用归纳法，其中 $2\delta < \ell$ 和 $n = 2\ell$。

基础情形：我们首先证明 $\delta = 1$ 的正确性。该码字由以下内容给出。

$$\mathbf{x} = \langle \mathbf{x}^1, \mathbf{x}^2 \rangle = (x_1, \ldots, x_{n/2}, x_{n/2+1}, \ldots, x_n) \in \mathcal{D}_1(n/2, n).$$

要解码第一个块，解码器观测到 $\mathbf{s}^1 = y_{n/2}$。现在考虑两种情况：*(i)* $\mathbf{x}^1$中未发生删除；以及*(ii)* $\mathbf{x}^1$中有一个比特被删除。在第一种情况下，根据构造1，我们始终有 $\mathbf{s}^1 = y_{n/2} = 1$。因此，根据 IV-C 节中描述的解码算法，解码器可以正确判定第一个块中未发生删除。现在考虑上述第二种情况。根据码构造，$\mathbf{x}^2$ 的前两个比特的值均为 0。因此，容易看出，对于$\mathbf{x}^1$, 中的任意一次删除以及$\mathbf{x}^2$, 中的任意一次删除，我们始终有 $\mathbf{s}^1 = y_{n/2} = 0$。因此，解码器总能正确检测到$\mathbf{x}^1$中的一次删除。第二个块中的删除数量随后根据第二个块在$\mathbf{y}$ 中的起始位置以及$\mathbf{y}$ 中尚未解码的比特数量确定。至此，$\delta = 1$的证明完成。

归纳步骤：我们假设码 $\mathcal{D}_{\delta-1}(n/2, n)$ 能检测每个块中多达 $\delta-1$ 次删除，并证明 $\mathcal{D}_\delta(n/2, n)$ 能检测每个块中多达 δ 次删除。根据构造 1，容易看出，若 $\mathbf{x} \in \mathcal{D}_\delta(n/2, n)$，则 $\mathbf{x} \in \mathcal{D}_{\delta-1}(n/2, n)$，因而 $\mathcal{D}_\delta(n/2, n) \subset \mathcal{D}_{\delta-1}(n/2, n)$。因此，由归纳假设可知，$\mathcal{D}_\delta(n/2, n)$ 能检测每个块中多达$\delta - 1$ 次删除。接下来，我们证明 $\mathcal{D}_\delta(n/2, n)$ 可以每个块也能检测恰好 δ 次删除。假设在 $\mathbf{x}^1$ 中恰好发生了 δ 次删除。为了解码第一个块，解码器按照 IV-C 节中的说明，从左到右扫描 $\mathbf{s}^1 = \mathbf{y}^{[\ell-\delta+1,\ell]}$ 中的比特，寻找第一个 0。根据码构造可知，对于 $\mathbf{x}^1$，中的任意 δ 次删除，$\mathbf{s}^1$ 的首比特始终为 0，即 $y_{\ell-\delta+1} = 0$。为此，注意到：*(i)* 对于 $\mathbf{x}^1$，中的任意 δ 次删除，$\mathbf{y}$ 中属于第二个块的首比特会向左移动 δ 个位置；以及 *(ii)* 由于 $\mathbf{x}^{[n/2+1,n/2+\delta+1]} = \mathbf{0}^{\delta+1}$，且我们假设 $\mathbf{x}^2$，中至多发生 δ 次删除，因此 $\mathbf{y}$ 中属于第二个块的首比特始终为 0。因此，对于 $\mathbf{x}^1$, 中的任意 δ 次删除，以及 $\mathbf{x}^2$, 中不超过 δ 次删除，均有 $y_{\ell-\delta+1} = 0$。根据解码算法，在这种情况下，解码器会判定 $\mathbf{x}^1$ 中发生了 δ 次删除。此外，根据第二个块在 $\mathbf{y}$ 中的起始位置，以及 $\mathbf{y}$ 中尚待解码的比特数，即可确定第二个块中的删除次数。

V. 在轨迹重构中的应用

本节介绍删除检测码的一种可能应用：将构造1中的基于标记的编码应用于轨迹重构。

A. 问题陈述与相关工作

在轨迹重构中，目标是给定序列 $\mathbf{x} \in \mathbb{F}_2^n$ 的随机迹，重构未知序列；其中每条迹均由一个随机信道独立生成，作为该信道的输出。我们关注删除信道的情形，即$\mathbf{x}$ 中的每一位都以概率 p 独立地被删除。目标是设计高效算法，仅利用少量迹即可重构$\mathbf{x}$ 。已有大量文献研究$\mathbf{x}$ 未编码时的轨迹重构问题，例如 [24]–[35]。

轨迹重构问题自然产生于DNA存储的数据检索阶段。具体而言，当前最先进的DNA测序方法通常会输出所存储数据的若干个错误副本。这促使近年来一系列工作研究编码设置下的轨迹重构问题 [6], [36]–[39], ：在观察序列 $\mathbf{x}$ 的多条迹之前，允许先对其进行编码[2]。

数据编码是提高存储系统可靠性的一种自然且广为人知的策略。由于我们关注删除信道，所考虑的基准是经典的删除纠错码（例如 [10], [12]–[16]），它只需一个迹即可重构存储的数据。使用删除纠错码纠正长度为 n 的序列中的 d 次删除所需的最优冗余（即存储开销）为 $\Theta(d \log(n/d))$ [8]。此外，文献中现有的大多数删除码都具有复杂的编码和解码算法。因此，编码迹重构的目标是利用多个迹的存在——这在 DNA存储等应用中很自然——来设计满足以下条件的码：*(i)* 冗余增长速度严格

[2]在 DNA存储中编码数据，可以通过采用 DNA 合成策略来实现，这些策略能够对任意数字信息进行编码，并将其存储在 DNA 中 [40]–[42]。

慢于删除纠错码的最优冗余；和/或 *(ii)* 能够利用少量迹，通过简单高效的重构算法完成重构。

Abroshan 等人 在 [37]中研究了编码迹重构，其中每条迹受到 d 次删除的影响， d 相对于序列长度 n固定，且 d 次删除的位置均匀随机。 [37] 中所用方法的核心思想是拼接一系列 Varshamov-Tenengolts (VT) [9] 码字，这些码字是单删除纠错码，在 n中具有渐近最优冗余；此外，还具有高效的线性时间编码和解码算法。拼接的 VT 码字数量选为一个严格大于 d 的常数，使得每个 VT 码字中的平均删除数严格小于一。随后，使用一种高效的重建算法，通过在迹中找到某个 VT 码字的无删除副本，或使用 VT译码器解码一次删除，来重构每个 VT 码字。由于每个码字的长度是 n的一个常数比例，且 VT 码的冗余关于分组长度呈对数增长，因此不难看出， [37] 中的码的冗余为 $\Theta(\log n)$。 [6] 和 [36] 中的工作研究了删除概率 $p = \epsilon$ 固定时的轨迹重构编码，即平均而言会删除线性数量的比特。对于这一情形， [6]中的作者引入了码率为$1 - \mathcal{O}(\ ^1\log n)$的二进制码，这些码可以从 $\log^c n$ 条迹中高效重构，其中 $c > 1$。 [36] 中的作者证明了码率为 $1 - \mathcal{O}(\ ^1\log n)$的二进制码存在，并且可以从 $\exp\big((\log \log\ n)^{1/3}\big)$ 条迹中重构。

——

B. 贡献

我们首先关注仅受删除影响的二进制序列的编码迹重构，以此作为设计能够抵御 DNA 存储中各种错误类型的编码的第一步。我们考虑随机删除模型，其中每个位都以概率 p 独立删除。我们关注如下情形： $p = k/n^\alpha$，其中 $k > 1$ 和 $\alpha \in (0.5, 1]$ 是常数。我们使用第四节B（构造1）中引入的删除检测码，设计一种用于轨迹重构的新型编码 $\mathcal{C}' \subseteq \mathbb{F}_2^n$ 。我们的贡献如下。

- 对于所有 $\alpha \in (0.5, 1]$，我们的码的冗余为$o(n^{1-\alpha}\log n)$，即其增长速度严格低于最优删除纠错码的冗余，后者为$\Theta(n^{1-\alpha}\log n)$，适用于 $d = \Theta(np) = \Theta(n^{1-\alpha})$。此外，对于平均删除数量固定这一特定情形，即 $\alpha = 1$ 和 $p = k/n$，我们证明，我们的码能够高效地重构任意冗余为 $\omega(1)$ 的码字 $\mathbf{x} \in \mathcal{C}'$ 。这一情形类似于每条迹中的删除数量固定的情形，该情形已在 [37]中得到研究。对于这一情形，与 [37]中提出的码的 $\Theta(\log n)$ 冗余相比，我们的码具有更低的冗余。

- 我们提出的重建算法以线性时间运行 $\mathcal{O}(n)$。

- 我们以重构错误概率的上界形式，为所提出的码的性能提供理论保证。该上界表明，对于所有$\alpha \in (0.5, 1]$，以及任意码字$\mathbf{x} \in \mathcal{C}'$，当迹的数量为常数 $t = \mathcal{O}(1)$时，重构错误概率会随着 n 的增长而渐近趋于零。

- 我们通过编辑距离误差来评估所提出码的数值性能，该指标已被近期多项相关工作采用作为性能度量，例如 [43]–[45]。我们将所提出码的性能与 [37] 中的码以及 Bitwise Majority Alignment（BMA）重建算法的一个简单编码版本 [27] 进行比较。仿真结果表明，在迹数量较少的情况下，所提出的码具有较低的编辑距离误差，且性能优于 [37] 中的码以及 BMA 算法的简单编码版本。

C. 符号约定

我们考虑如下删除模型。对于给定的输入序列 $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{F}_2^n$、删除概率 $p = k/n^\alpha$，其中 $k > 1$ 和 $\alpha \in (0.5, 1]$ 为常数，以及整数 t，信道返回 $\mathbf{x}$ 的 t 条迹。$\mathbf{x}$ 的每条迹记为 $\mathbf{y}_i \in \mathbb{F}_2^*,\ i = 1, \ldots, t$，其获取方式是将 $\mathbf{x}$ 通过删除信道发送，删除概率为 p；即，删除信道以概率 p独立地删除 $\mathbf{x}$ 的每个比特 $x_i\ \ i = 1, \ldots, n$，并按顺序输出包含 $\mathbf{x}$ 中所有未被删除比特的子序列。t 条迹 $\mathbf{y}1, \ldots, \mathbf{y}_t$相互独立同分布，均为输入 $\mathbf{x}$ 时删除信道的输出。重建算法的输入为 t条迹 $\mathbf{y}1, \ldots, \mathbf{y}_t$、码字长度 n以及信道参数 k 和 α。令 $\hat{\mathbf{x}} \in \mathbb{F}_2^n$ 表示重建算法的输出。重构错误的概率定义为 $P_e \triangleq \Pr(\hat{\mathbf{x}} \neq \mathbf{x})$，编辑距离误差定义为随机变量 $L_d \triangleq \mathrm{Lev}(\mathbf{x}, \hat{\mathbf{x}})$，其中 $\mathrm{Lev}(.,.)$ 表示两个序列之间的列文斯坦距离 [8] ，而 P_e 和 L_d 中的随机性来自删除过程。

D. 预备知识

接下来，我们概述我们的码构造，并解释所使用技术背后的主要思想。我们的正式结果见第 V-E 节。

1) 码构造概述：我们的码构造的总体思路如下。我们通过拼接 n/ℓ 个块 $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{n/\ell}$（定义1）来构造一个码字 $\mathbf{x} \in \mathbb{F}_2^n$，每个块的长度均为 ℓ，其中 $\ell \in \mathbb{Z}^+$ 是该码的一个参数[3]。我们构造 $\mathbf{x} = \langle \mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{n/\ell} \rangle$ 中的每个块，使其满足以下两个性质。首先，这些块满足**游程长度受限**约束，即每个连续的0或1的游程长度均不超过某个最大值。其次，每个块的开头和结尾都有少量固定比特，这些固定比特对应于构造1所指定的标记。

引入标记的目标是让解码器恢复块的边界。确定这些

[3]为简化起见，我们假设 ℓ 能整除 n ，并将在第V-E节中取消这一假设。

块边界使我们能够获得对应于每个块的迹。因此，我们可以将迹重构问题细分为更小的子问题。这些子问题之所以“更容易”，是因为在迹的数量相同的情况下，每个块的预期删除数少于 $\mathbf{x}$ 中的删除总数。这提高了正确重构更多块的概率，从而产生较低的编辑距离误差，正如我们在第 V-F 节中所展示的那样。此外，标记还可以防止潜在的重构错误传播到码字的所有部分。为理解这种细分背后的直觉，我们给出以下示例。

考虑一种朴素的轨迹重构算法，它尝试在各条迹中找到一个无删除副本，以此重构序列 $\mathbf{x}$ 。在删除概率较低的未编码设置中，我们预计每条迹中只会出现少量删除。我们还预计，迹中的许多部分（即块）都不会发生删除。因此，如果我们使用一种删除检测码（定义 2），使每个块的边界都能被恢复，那么我们就更有可能通过在各条迹中分别找到每个块的一个无删除副本，来重构 $\mathbf{x} = \langle \mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^{n/\ell} \rangle$ 。

例如，考虑这样一种情形：解码器拥有$t = 2$ 条长度为 n的二进制序列 $\mathbf{x}$ 的迹。假设第一条迹 $\mathbf{y}_1$ 在位置i_1发生了一次删除，其中 $i_1 < n/2$；第二条迹 $\mathbf{y}_2$ 在位置 i_2发生了一次删除，其中 $i_2 > n/2$。因此，解码器拥有

$$\mathbf{y}_1 = (x_1, ..., x_{i_1-1}, x_{i_1+1}, ..., x_{n/2}, ..., x_n),$$
$$\mathbf{y}_2 = (x_1, ..., x_{n/2}, ..., x_{i_2-1}, x_{i_2+1}, ..., x_n).$$

若$\mathbf{x}$ 未经过编码，使用朴素算法将无法成功重构，因为两个迹中都没有$\mathbf{x}$ 的无删除副本。现在假设$\mathbf{x} \in \mathcal{C}_1(n/2, n)$，其中$\mathcal{C}_1(n/2, n)$是一个符合定义 2 的码，即能够检测两个大小为$\ell = n/2$的块中各至多一次删除。这样，便可成功重构$\mathbf{x}$ ，因为只要知道各块的边界，就能从第二个迹中获得第一个块的无删除副本，并从第一个迹中获得第二个块的无删除副本。请注意，一般而言，我们可能无法找到每个块的无删除副本。不过，我们预计这种划分仍能让我们找到大多数块的无删除副本，从而相比未编码设置降低编辑距离误差。

受前述示例启发，为了在解码器处重构$\mathbf{x}$ ，我们提出以下方法。解码器通过 t 个独立删除信道获得$\mathbf{x}$ 的 t 条独立迹。为了重构$\mathbf{x}$，解码器首先使用底层的删除检测码恢复每条迹中的块边界，如第四节中所述。接着，解码器使用逐位多数对齐（BMA）算法（下文将对此进行说明），根据各块对应的迹重构每个块。最后，将恢复出的块按顺序拼接起来，以重构$\mathbf{x}$。

2）逐比特多数对齐：逐比特多数对齐（BMA）算法最初于 [27] 作为一种重构未编码二进制序列$\mathbf{x} \in \mathbb{F}_2^n$的算法提出从 t 条迹中重构。在我们的方法中，我们首先确定每个大小为 ℓ 的块的边界，然后在每个块的层面应用 BMA 重构算法。因此，该算法的输入是一个 $t \times \ell$ 二进制矩阵，其中包含对应于某个块的t 条迹。由于删除可能导致某些接收到的迹长度小于 ℓ ，因此将算法输入中的迹补齐至长度 ℓ ，补齐时使用一种特殊字符（不是 0 或 1）。该算法的核心思想是采用多数表决方式，根据多数迹中的比特值重构每个比特。具体而言，每条迹都设置一个指针，初始指向最左侧的比特；根据多数指针的投票结果，确定重构序列中的比特值。随后，将投出多数票的迹所对应的指针向右移动一位，而其他指针则保持在原来的比特位置。因此，重建过程从左至右扫描各条迹中的比特，并且在算法的不同状态下，各指针可能指向不同的比特位置。最后，算法输出一个长度为 ℓ比特的重构序列。

令 $q(j)$ 为与迹 j,$j = 1, \ldots, t$ 对应的指针，其中 $q(j) = i$, $i = 1, \ldots, \ell$，表示与迹 j 对应的指针正指向位置 i处的比特。详细算法见算法1。

算法1： 按位多数对齐

输入： 一个大小为 $t \times \ell$ 的二进制矩阵 T
输出： 一个二进制序列 y 大小为 ℓ

```
设令 q(j) ← 对所有 j = 1, . . . , t;
对于 i ← 1 到 ℓ，执行
    令 b为所有 j在 T(j, q(j)) 上的众数;
    y(i) ← b;
    对于 j ← 1 到 t，执行
        如果 q(j) == b 则
            q(j) ← q(j) + 1;
        end
    end
结束
```

[27] 中的作者证明，对于满足 $p = \mathcal{O}(1/\sqrt{\ell})$ 的删除概率，BMA 算法（算法1）能够以高概率[4], 从 $t = \mathcal{O}(\ell \log \ell)$ 条迹中重构任意长度为 ℓ 的（未编码）二进制序列。对于某些包含较长连续0或1的序列，BMA 算法的性能较差 [27]。例如，考虑一个以一段较长的连续0开头的序列。由于删除操作，该段连续0中观测到的0的数量在不同的迹之间有所不同。从左向右扫描时，在这段连续0结束之前的某个时刻，大多数迹可能会投票选择1。这将导致该段连续0被拆分，从而造成错误的重构。要成功重构这类具有较长连续段的序列，需要的迹数量显著多于.....

[4]重构错误的概率在 ℓ 中渐近趋于零，其中该概率是对删除过程的随机性取平均所得。

游程受限 序列。事实上， [27] 中的一项中间结果表明，对于游程受限 序列，若任意游程的长度至多为$\sqrt{\ell}$，则 BMA 算法所需的迹数为$t = \mathcal{O}(1)$。然而，在 [27],所考虑的未编码最坏情况轨迹重构中，对于任意 二进制序列，BMA 算法仍需要 $t = \mathcal{O}(\ell \log \ell)$ 条迹，才能使重构错误的概率（对删除过程的随机性取值）渐近趋于零。本节中，我们的一种方法是利用编码轨迹重构，通过将码限制为形如$\mathbf{x} = \langle \mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{n/\ell} \rangle \in \mathbb{F}_2^n$的码字，从而减少所需的迹数，其中大小为 ℓ 的任意块中的游程长度均不超过 $\sqrt{\ell}$。

注 3. 一种直接的方法是对输入码字$\mathbf{x} \in \mathbb{F}_2^n$ 加以限制，使其仅包含最长连续段不超过 $\sqrt{n}$ 的序列，从而得到 *BMA* 算法的编码版本。请注意，文献中尚未讨论过这种简单变体的 *BMA* 重构算法；但根据 *[27],* 中的一个中间结果，很容易看出，这种 *BMA* 算法的编码版本只需要常数条迹。然而，我们在第 V-F 节中的结果表明，我们提出的码与这种简单的 *BMA* 算法编码版本在编辑距离误差方面存在显著差异。这凸显了引入标记这一步的重要性，它使问题能够被划分为更小的子问题。

E. 理论结果

令 $\mathcal{L}^n(f(n)) \subseteq \mathbb{F}_2^n$ 为长度为 n 的限长游程二进制序列集合，其中序列$\mathbf{x} \in \mathcal{L}^n(f(n))$中任意 0 或 1 的连续段长度至多为 f ({v14)。

构造 2 (轨迹重构码)。 考虑给定的删除概率 $p = k/n^\alpha < 1$, 其中 $k > 1$ 和$\alpha \in (0.5, 1]$ 为常数。该码由如下形式的码字组成：$\mathbf{x} = \langle \mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{\lceil n/\ell \rceil} \rangle$，其中每个块的长度设为 $\ell = \lfloor 1/p \rfloor$。令 $\delta \in \mathbb{Z}^+$，其中 $\delta > 1$，为一个码参数，表示每个块 $\mathbf{x}^m$,$m \in \{1, \ldots, \lceil n/\ell \rceil\}$中可检测删除数量的严格上界，即该码每块最多可检测 $\delta - 1$次删除。对于 $\ell = \lfloor 1/p \rfloor > \delta^2$，我们定义该码。

$$\mathcal{C}'_\delta(n) \triangleq \left\{ \mathbf{x} \in \mathbb{F}_2^n \mid \mathbf{x} \in \mathcal{D}_{\delta-1}(\lfloor 1/p \rfloor, n) \cap \mathcal{L}^n(\sqrt{\lfloor 1/p \rfloor}) \right\},$$

其中， $\mathcal{D}$ 是构造 1 中定义的删除检测码。

序列$\mathbf{x} \in \mathcal{C}'_\delta(n)$的重建算法步骤如下。

1) 基于 $\mathcal{D}_{\delta-1}(\lfloor 1/p \rfloor, n)$ 的解码算法（IV-C 节），恢复所有 t 条迹中的块边界。 2) 应用 BMA 算法（算法1）重建每个块。 3) 拼接重建后的块以获得$\mathbf{x}$。

定理 4。 对于任意删除概率 $p = k/n^\alpha$ （满足 $0 < p < 0.5$），其中 $k > 1$ 和 $\alpha \in (0.5, 1]$ 为常数，码 $\mathcal{C}'_\delta(n) \subseteq \mathbb{F}_2^n$ （构造 2）可以从 $t = \mathcal{O}(1)$ 线性时间内的迹 $\mathcal{O}(n)$。该码的冗余$r_{\mathcal{C}'}(\delta)$ 满足

$$(kn^{1-\alpha} - 1)(2\delta - 1) + o(1) \leqslant r_{\mathcal{C}'}(\delta) < 2kn^{1-\alpha}(2\delta - 1) + o(1).$$

令 $p(n) = \mathcal{O}(\text{poly}(n))$，对于 $\delta = \lceil \delta^*(p(n)) \rceil = o(\log n)$ 其中

$$\delta^*(p(n)) = \frac{2 \ln \left(\sqrt{e} n^{1-\alpha} p(n) \right)}{W\left(2e \ln \left(\sqrt{e} n^{1-\alpha} p(n) \right) \right)},$$

其中， $W(.)$ 为 *Lambert* W 函数，任意码字 $\mathbf{x} \in \mathcal{C}'_\delta(n)$ 的重构错误概率满足

$$P_e < \mathcal{O}\left(n^{1-2\alpha} + \frac{1}{p(n)} \right).$$

推论 5. 对于任意删除概率 $p = k/n^\alpha$，满足$0 < p < 0.5$, 其中 $k > 1$ 和 $\alpha \in (0.5, 1)$ 为常数，构造 2 中定义的码可以从$\mathcal{O}(1)$ 条迹中在线性时间 $\mathcal{O}(n)$内重构，冗余为

$$r_{\mathcal{C}} = \Theta\left(n^{1-\alpha} \frac{\log n}{\log \log n} \right),$$

以及满足以下条件的重构错误概率

$$P_e < \mathcal{O}(n^{1-2\alpha}),$$

对于任意码字。

推论 6。 对于任意删除概率 $p = k/n$ *with*$0 < p < 0.5$, 其中 $k > 1$ 为常数，由构造2 定义的码可以从 $\mathcal{O}(1)$ 条迹中以线性时间 $\mathcal{O}(n)$完成重构，并且对于任意冗余 $r_{\mathcal{C}} = \omega(1)$，重构错误概率在 n 中渐近趋于零。

讨论： 定理 4 表明，由构造 2 定义的码可以从常数数量的迹中高效重构。此外，该结果用码参数表示了冗余和重构错误概率，并展示了这两个码属性之间的关系（权衡）。推论 5 和推论 6 均由定理 4 推出。推论 5 表明，对于 $\alpha \in (0.5, 1)$，冗余为

$$r_{\mathcal{C}} = \Theta\left(n^{1-\alpha} \frac{\log n}{\log \log n} \right) = o(n^{1-\alpha} \log n),$$

并且重构错误概率在 n 中以次数为 $1 - 2\alpha$ 的多项式渐近趋于零。注意，上述冗余的增长严格慢于删除纠正码的最优冗余（具有 $t = 1$条迹）。此外，重构算法的复杂度关于 n 为线性。推论 6 表明，对于 $\alpha = 1$，使用常数数量的迹和任意 $\omega(1)$冗余，均可实现渐近趋于零的重构错误概率。

F. 仿真结果

本节通过编辑距离误差 $L_d = \text{Lev}(\hat{\mathbf{x}}, \mathbf{x})$ [43]–[45], 评估我们在 Construction2 中定义的码 $C'_\delta(n)$ 的数值性能。我们给出仿真结果，将我们码的归一化编辑距离误差 $\text{Lev}(\hat{\mathbf{x}}, \mathbf{x})/n$ 与以下方案进行比较：*(i)* 针对多个 t， n 值的 BMA 算法编码版本（注释 3 ），以及

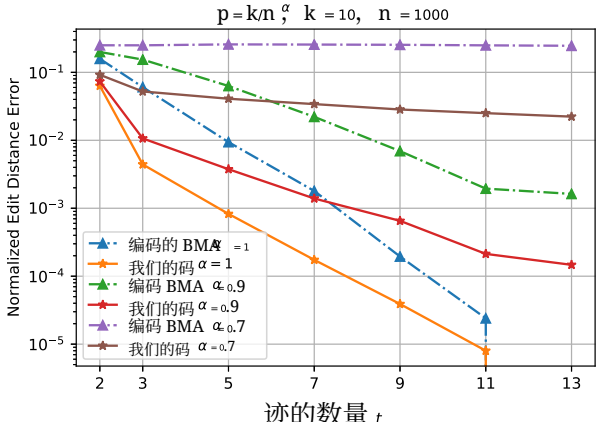

图 1：归一化编辑距离误差随迹的数量 t 的变化，码字长度为 $n = 1000$，删除概率为 $p = k/n^{\alpha}$，其中 $k = 10$ 和 $\alpha \in \{0.7, 0.9, 1\}$。

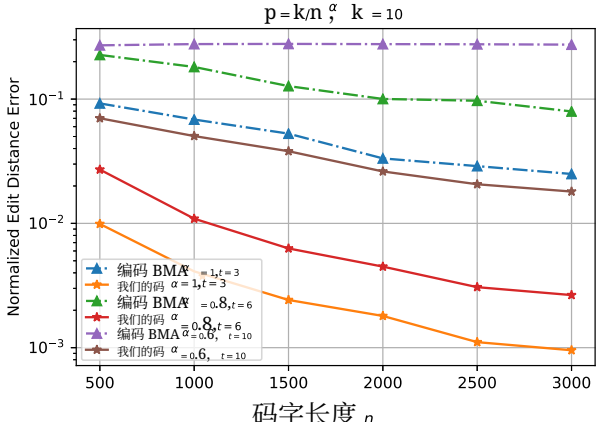

图 2：归一化编辑距离误差随码字长度 n 的变化，删除概率为 $p = k/n^{\alpha}$，其中$k = 10$ 和 $(\alpha, t) \in \{(1, 3), (0.8, 6), (0.6, 10)\}$。

$\alpha \in (0.5, 1]$；以及*(ii)* 针对 $\alpha = 1$ 和多个 t 值的 [37] 中的码。

仿真结果如图 1、2 和 3 所示。编辑距离误差取 10^3 次仿真的平均值，再除以分组长度 n进行归一化。每次运行中，均匀随机选取一个长度为 n 的码字，并对其应用随机删除过程，以获得 t 条迹。对于所有仿真，码 $C'_{\delta}(n)$ 的参数 δ 均设为 $\delta = 3$ 。总体而言，结果表明，与 BMA 的简单编码版本（注释 3）相比，我们的码的编辑距离误差要低得多。这证实了我们的直觉：引入标记并将重构 $\mathbf{x}$ 的问题划分为更小的子问题十分重要。具体而言，由于删除数量较少，子问题更容易解决；而标记可防止潜在的重构错误传播至整个码字，从而降低编辑距离误差。例如，我们在图 2 中观察到，对于 $n = 3000$，我们的码的归一化编辑距离误差为 $\approx 10^{-3}$，而编码后的

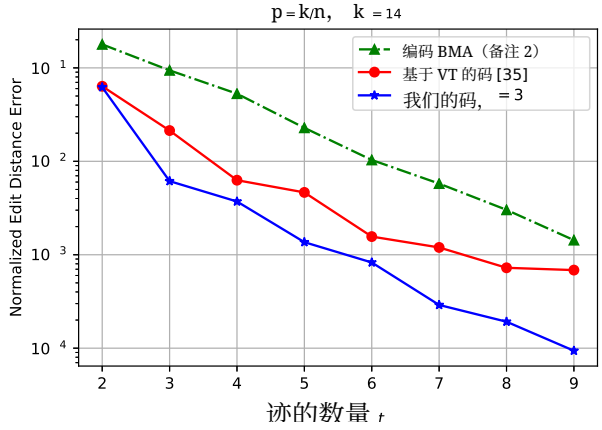

图 3：归一化编辑距离误差随迹的数量 t 的变化，码字长度为 $n = 994$，删除概率为 $p = k/n^{\alpha}$，其中 $k = 14$ 和 $\alpha = 1$。

BMA 的归一化编辑距离误差为 。请注意，这种改进是以较低的码率为代价的；不过，通过改变参数 δ 和 ℓ，可以实现不同的权衡。

在图 3 中，我们还比较了我们的码与 [37] 中基于 VT 的码在平均每个片段的删除数为常数（即 $\alpha = 1$）的情形下的性能。 [37] 中的码通过串接 $n/\ell = 17$ 个大小为 $\ell = 59$ 的 VT 码字构造而成，码率为 ≈ 0.9。我们的码的参数设为 $\delta = 3$、 $n = 994$ 和 $\ell = 71$，码率为 ≈ 0.934。结果表明，与 [37]，中的码相比，我们的码在码率更高的情况下具有更低的编辑距离误差。

VI. 结论

本文研究了在拼接二进制字符串中构造能够检测最坏情况删除确切数量的编码问题。首先，我们通过推导删除检测码冗余的下界，确定了该问题的基本极限。随后，我们提出了一种最优的基于标记的构造，对于逐块可译码的码，该构造以相等的冗余达到下界。我们还证明，在所有删除检测码族中，就删除数量而言，该构造的冗余是渐近最优的。我们将这种基于标记的构造应用于编码轨迹重构，并评估了由此带来的增益。仍有一些开放问题有待研究，包括如何为检测删除与插入的组合构造码，以及确定其基本极限。未来研究的另一个有趣方向是，研究针对同时引入删除和插入的随机信道的编码轨迹重构。

致谢

作者谨向审阅本文的匿名审稿人致谢。他们富有见地的评论和宝贵的反馈极大地提升了本文的质量。

参考文献

[1] S. Kas Hanna 和 R. Bitar，《在串接字符串中以最优冗余检测删除和插入》，载于 2021年IEEE国际信息论研讨会 (ISIT)，2021年，第1985–1990页。[2] S. Kas Hanna，《具有渐消删除概率的多信道轨迹重构编码》，载于 2022年IEEE国际信息论研讨会 (ISIT)，2022年，第360–365页。[3] X. Jiao、H. Liu、J. Mu、H. Han 和 Y.-C. He，《分段编辑信道的前缀 Varshamov-Tenengolts 码》，《IEEE通信汇刊》，第70卷，第3期，第1535–1545页，2022年。[4] M. Abroshan、R. Venkataramanan 和 A. Guill´en i F`abregas，《分段编辑信道编码》，《IEEE信息论汇刊》，第64卷，第4期，第3086–3098页，2018年。[5] Z. Liu 和 M. Mitzenmacher，《具有分段错误的删除与插入信道编码》，《IEEE信息论汇刊》，第56卷，第1期，第224–232页，2010年。[6] M. Cheraghchi、R. Gabrys、O. Milenkovic 和 J. Ribeiro，《编码迹重构》，《IEEE信息论汇刊》，第66卷，第10期，第6084–6103页，2020年。[7] R. Venkataramanan、V. Narasimha Swamy 和 K. Ramchandran，《用于从删除、插入和替换中实现同步的低复杂度交互式算法》，《IEEE信息论汇刊》，第61卷，第10期，第5670–5689页，2015年。[8] V. I. Levenshtein，《能够纠正删除、插入和反转的二进制码》，载于《苏联物理学文献》，第10卷，1966年，第707–710页。[9] R. Varshamov 和 G. Tenengol’ts，《单个非对称错误纠错码》，《自动化与遥测》，第26卷，第2期，第286–290页，1965年。[10] J. Brakensiek、V. Guruswami 和 S. Zbarsky，《用于纠正多重删除的高效低冗余码》，《IEEE信息论汇刊》，第64卷，第5期，第3403–3410页，2018年5月。[11] S. Kas Hanna 和 S. El Rouayheb，《用于删除和同步的猜测与校验码》，载于2017年IEEE国际信息论研讨会 (ISIT)，2017年6月，第2693–2697页。[12] ——，《用于删除、插入和同步的猜测与校验码》，《IEEE信息论汇刊》，第65卷，第1期，第3–15页，2019年1月。[13] B. Haeupler，《最优文档交换及用于插入和删除的新码》，载于2019年IEEE第60届计算机科学基础年会 (FOCS)，2019年，第334–347页。[14] K. Cheng、Z. Jin、X. Li 和 K. Wu，《确定性文档交换协议及近乎最优的编辑错误二进制码》，载于2018年IEEE第59届计算机科学基础年会 (FOCS)，2018年，第200–211页。[15] J. Sima 和 J. Bruck，《关于最优 k重删除纠错码》，《IEEE信息论汇刊》，第67卷，3375，2021 6，3360 no. pp. – 。[16] J. Sima、R. Gabrys 和 J. Bruck，《最优系统型 t重删除纠错码》，载于2020年IEEE国际信息论研讨会 (ISIT)，2020年，第769–774页。[17] C. Schoeny、A. Wachter-Zeh、R. Gabrys 和 E. Yaakobi，《纠正突发删除或插入的编码》，《IEEE信息论汇刊》，第63卷，第4期，第1971–1985页，2017年。[18] S. Kas Hanna 和 S. El Rouayheb，《使用猜测校验码纠正突发和局部删除》，载于2017年第55届Allerton通信、控制与计算会议 (Allerton)，2017年10月，第9–16页。[19] ——，《用于纠正局部删除的编码》，《IEEE信息论汇刊》，第67卷，第4期，第2206–2216页，2021年。[20] A. Lenz 和 N. Polyanskii，《纠正可变长度突发删除的最优编码》，载于2020年IEEE国际信息论研讨会 (ISIT)，2020年，第757–762页。[21] R. Bitar、S. Kas Hanna、N. Polyanskii 和 I. Vorobyev，《纠正局部删除的最优编码》，载于2021年IEEE国际信息论研讨会 (ISIT)，2021年，第1991–1996页。[22] S. Konstantinidis、S. Perron 和 L. A. Wilcox-O’Hearn，《一种检测编码消息中同步错误的简单方法》，《IEEE信息论汇刊》，第49卷，1363，2003 5，1355 no. pp. – 。[23] F. Palunˇci´c、K. A. S. Abdel-Ghaffar 和 H. C. Ferreira，《插入/删除检测码与边界问题》，《IEEE信息论汇刊》，第59卷，第9期，第5935–5943页，2013年。[24] V. I. Levenshtein，《从最少数量的失真模式中重构对象》，载于《科学院院刊》，第354卷，第5期。俄罗斯科学院

, 1997, 第593–596页。

[25] IEEE信息论汇刊，序列的高效重构，第2–22页，2001年。[26] V. I. Levenshtein，《从子序列或超序列高效重构序列》，《组合理论杂志》A辑，第93卷，第2期，第310–332页，2001年。 [在线]。可访问：https://www.sciencedirect.com/science/article/pii/S0097316500930814

[27] T. Batu、S. Kannan、S. Khanna 和 A. McGregor，《从随机迹重构字符串》，载于《第十五届ACM-SIAM离散算法年度研讨会论文集》，丛书：SODA ’04。美国：工业与应用数学学会，2004年，第910–918页。[28] T. Holenstein、M. Mitzenmacher、R. Panigrahy 和 U. Wieder，《具有恒定删除概率的迹重构及相关结果》，载于 SODA，第8卷，2008年，第389–398页。[29] A. McGregor、E. Price 和 S. Vorotnikova，《重新审视迹重构》，载于《欧洲算法研讨会》。Springer，2014年，第689–700页。[30] A. De、R. O’Donnell 和 R. A. Servedio，《用于迹重构的最优均值算法》，载于《第49届ACM SIGACT计算理论年度研讨会论文集》，2017年，第1047–1056页。[31] F. Nazarov 和 Y. Peres，《使用 exp (o (n1/3)) 个样本进行迹重构》，载于《第49届ACM SIGACT计算理论年度研讨会论文集》，2017年，第1042–1046页。[32] L. Hartung、N. Holden 和 Y. Peres，《具有变化删除概率的迹重构》，载于《2018年第十五届分析算法与组合学研讨会论文集（ANALCO）》。SIAM，2018年，第54–61页。[33] Y. Peres 和 A. Zhai，《删除信道的平均情形重构：少量多项式级迹即可满足需求》，载于《2017年IEEE第58届计算机科学基础年度研讨会（FOCS）》。IEEE，2017年，第228–239页。[34] N. Holden、R. Pemantle 和 Y. Peres，《随机字符串和任意删除概率下的次多项式迹重构》，arXiv预印本 arXiv:1801.04783，2018年。[35] X. Chen、A. De、C. H. Lee、R. A. Servedio 和 S. Sinha，《低删除率范围内的多项式时间迹重构》，arXiv预印本 arXiv:2012.02844，2020年。[36] J. Brakensiek、R. Li 和 B. Spang，《使用恒定数量的迹进行编码迹重构》，载于《 2020 年IEEE第61届计算机科学基础年度研讨会（FOCS）》，2020年，第482–493页。[37] M. Abroshan、R. Venkataramanan、L. Dolecek 和 A. G. i F`abregas，《具有多个迹的删除信道编码》，载于《2019年IEEE国际信息论研讨会（ISIT）》，2019年，第1372–1376页。[38] S. R. Srinivasavaradhan、S. Gopi、H. D. Pfister 和 S. Yekhanin，《 Trellis BMA：用于DNA存储的IDS信道编码迹重构》，载于《2021年IEEE国际信息论研讨会（ISIT）》。IEEE，2021年，第2453–2458页。[39] I. Maarouf、A. Lenz、L. Welter、A. Wachter-Zeh、E. Rosnes 和 A. G. i Amat，《用于读取 DNA 序列多次读数的级联编码》，《 IEEE信息论汇刊》，第69卷，927，2023 2，910第。期，第–页。[40] G. M. Church、Y. Gao 和 S. Kosuri，《DNA中的下一代数字信息存储》，《科学》，第337卷，1628，2012 6102，1628第。期，第–页。[41] N. Goldman、P. Bertone、S. Chen、C. Dessimoz、E. M. LeProust、B. Sipos 和 E. Birney，《迈向合成DNA中实用、高容量、低维护的信息存储》，《自然》，第494卷，第7435期，第77–80页，2013年。[42] S. H. T. Yazdi、R. Gabrys 和 O. Milenkovic，《便携式且无差错的基于DNA的数据存储》，《科学报告》，第7卷，第1期，第1–6页，2017年。[43] S. R. Srinivasavaradhan、M. Du、S. Diggavi 和 C. Fragouli，《关于多个删除信道上的最大似然重构》，载于《2018年IEEE国际信息论研讨会（ISIT）》，2018年，第436–440页。[44] D. Bar-Lev、Y. Gershon、O. Sabary 和 E. Yaakobi，《针对t-删除信道上最优期望归一化距离的解码》，载于《2021年IEEE国际信息论研讨会（ISIT）》，2021年，第1847–1852页。[45] S. Davies、M. Z. R´acz、B. G. Schiffer 和 C. Rashtchian，《近似迹重构：算法》，载于《2021年IEEE国际信息论研讨会（ISIT）》，2021年，第2525–2530页。[46] M. Mitzenmacher 和 E. Upfal，《概率与计算：算法和数据分析中的随机化与概率技术》。剑桥大学出版社，2017年。[47] A. Hoorfar 和 M. Hassani，《Lambert W 函数和超幂函数的不等式》，《纯粹与应用数学不等式杂志》，2008，第9卷，第2期，第5–9页。

附录 A 定理 4 的证明

首先，我们根据码参数 δ 推导构造 2 中定义的码 $\mathcal{C}'_\delta(n)$ 的冗余表达式，记为 $r_{\mathcal{C}'}(\delta)$，其中 $\delta = o(\log n)$。注意，定理 3 假设 $\ell \mid n$；然而，如果 $\ell \nmid n$，则块的总数将为 $\lceil n/\ell \rceil$，其中最后一块的长度严格小于 ℓ。在这种情况下，根据构造 1，我们仍需要在最后一块 $\mathbf{x}^{\lceil n/\ell \rceil}$ 中添加 $\delta+1$ 个冗余位，以检测块 $\mathbf{x}^{\lceil n/\ell \rceil - 1}$ 中删除的确切数量。因此，根据定理 3，$\mathcal{D}_{\delta-1}(\lfloor 1/p \rfloor,\ n)$ 对应的冗余记为 $r_{\mathcal{D}}(\delta)$，其值为

$$r_{\mathcal{D}}(\delta) = (2\delta - 1)\left(\left\lceil \frac{n}{\ell} \right\rceil - 1\right). \tag{10}$$

以下论断通过应用关于向下取整和向上取整函数的标准界，给出了块数量的下界和上界。

断言 7. 对于 $n > 1$ 和 $p < 1/2$，我们有

$$kn^{1-\alpha} - 1 \leqslant \left\lceil \frac{n}{\ell} \right\rceil - 1 < 2kn^{1-\alpha}, \tag{11}$$

$$kn^{1-\alpha} \leqslant \left\lceil \frac{n}{\ell} \right\rceil < (2k+1)n^{1-\alpha}. \tag{12}$$

我们暂且假设该结论成立，并将在附录 D 中给出证明。将式 (11) 中的界应用于式 (10) 中 $r_{\mathcal{D}}(\delta)$ 的表达式，可得

$$(kn^{1-\alpha} - 1)(2\delta - 1) \leqslant r_{\mathcal{D}}(\delta) < 2kn^{1-\alpha}(2\delta - 1). \tag{13}$$

接下来，我们证明，构造 2 中的游程受限约束引入了额外的冗余，但该冗余在 n 中渐近趋于零，即 $r_{\mathcal{C}'}(\delta) = r_{\mathcal{D}}(\delta) + o(1)$。

令 $X \in \mathbb{F}_2^n$ 为一个随机变量，表示从长度为 n 的二进制序列中均匀随机选取的一个序列。令 $L(X)\colon \mathbb{F}_2^n \to [n]$ 表示 X 中最长连续段的长度。回顾一下，$\mathcal{L}^n(\sqrt{\ell})$ 表示满足以下条件的长度为 n 的游程长度受限二进制序列集合：对于序列 $\mathbf{x} \in \mathcal{L}^n(\sqrt{\ell})$，其中任意一段连续的 0 或 1 的长度均不超过 $\sqrt{\ell}$。我们定义以下概率。

$$P_1 \triangleq P(X \in \mathcal{D}_{\delta-1}(\ell, n), X \notin \mathcal{L}^n(\sqrt{\ell})), \tag{14}$$

$$P_2 \triangleq P(X \in \mathcal{D}_{\delta-1}(\ell, n), X \in \mathcal{L}^n(\sqrt{\ell})), \tag{15}$$

其中 $\ell = \lfloor 1/p \rfloor = \lfloor n^\alpha / k \rfloor$。注意，

$$P_2 = P(X \in \mathcal{C}'_\delta(n)) = P(X \in \mathcal{D}_{\delta-1}(\ell, n)) - P_1. \tag{16}$$

首先，我们希望计算 P_1 的一个上界。为简洁起见，以下我们将用 $\mathcal{D}_{\delta-1}$ 代替 $\mathcal{D}_{\delta-1}(\ell, n)$。

$$P_1 = P(X \in \mathcal{D}_{\delta-1}, L(X) > \sqrt{\ell}), \tag{17}$$

$$= P(L(X) \geqslant \sqrt{\ell} + 1 \,\big|\, X \in \mathcal{D}_{\delta-1}) P(X \in \mathcal{D}_{\delta-1}), \tag{18}$$

$$\leqslant n \cdot \frac{1}{2^{\sqrt{\ell}+1-\delta}} \cdot \frac{1}{2^{r_{\mathcal{D}}(\delta)}}, \tag{19}$$

$$= 2^{-r_{\mathcal{D}}(\delta)} \cdot \frac{n}{2^{\sqrt{\ell}+1-\delta}}. \tag{20}$$

注意，式 (19) 是由要求至少一个长度为 $\sqrt{\ell}+1$ 的窗口包含一个游程，并应用并集界得到的。因此，

$$P_2 = P(X \in \mathcal{D}_{\delta-1}) - P_1, \tag{21}$$

$$\geqslant 2^{-r_{\mathcal{D}}(\delta)} - 2^{-r_{\mathcal{D}}(\delta)} \cdot \frac{n}{2^{\sqrt{\ell}+1-\delta}}, \tag{22}$$

$$= 2^{-r_{\mathcal{D}}(\delta)} \left[1 - \frac{n}{2^{\sqrt{\ell}+1-\delta}}\right]. \tag{23}$$

$$\tag{24}$$

因此，

$$|C'_\delta(n)| = 2^n \cdot P_2 \geqslant 2^{n - r_{\mathcal{D}}(\delta)} \left[1 - \frac{n}{2^{\sqrt{\ell}+1-\delta}}\right]. \tag{25}$$

因此，我们得到冗余的以下上界。

$$r_{\mathcal{C}'}(\delta) \leqslant r_{\mathcal{D}}(\delta) + \log\left[1 - \frac{n}{2^{\sqrt{\ell}+1-\delta}}\right]. \tag{26}$$

由于 $\ell = \lfloor 1/p \rfloor = \lfloor n^\alpha / k \rfloor$，其中 $k \geqslant 1$ 且 $\alpha \in (0.5, 1])$ 为常数，因此我们有 $\ell = \Theta(n^\alpha)$。此外，由于 $\delta = o(\log n)$，可得

$$\lim_{n \to +\infty} \log\left[1 - \frac{n}{2^{\sqrt{\ell}+1-\delta}}\right] = \log(1) = 0. \tag{27}$$

因此，

$$r_{\mathcal{C}'}(\delta) \leqslant r_{\mathcal{D}}(\delta) + o(1). \tag{28}$$

此外，$r_{\mathcal{C}}(\delta) \geqslant r_{\mathcal{D}}(\delta)$ 是一个平凡的下界。因此，我们有

$$r_{\mathcal{C}'}(\delta) = r_{\mathcal{D}}(\delta) + o(1). \tag{29}$$

结合式 (29) 和式 (13) 的结果，我们完成定理 4 中所给码的冗余证明。

接下来，我们推导错误概率的上界，即重构不成功的概率。定义以下事件。令 D 表示检测删除并确定所有块的边界这一过程对所有块和所有 t 迹均成功的事件。令 B 表示 BMA 算法（算法1）成功重构 $\mathbf{x} = \langle \mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{\lceil n/\ell \rceil} \rangle$ 的事件，也就是说，成功重构所有块 $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{\lceil n/\ell \rceil}$。令 P_e 表示构造 2 对应的错误概率。为推导上界，我们假设若事件 D 未发生，则重构不成功。因此，我们可以写出

$$P_e \leqslant P(D^c) + P(B^c, D) \tag{30}$$

$$= P(D^c) + P(B^c | D) P(D) \tag{31}$$

$$\leqslant P(D^c) + P(B^c | D). \tag{32}$$

回顾一下，在构造 2 中我们设定了 $\ell = \lfloor 1/p \rfloor = \lfloor n^\alpha / k \rfloor$，因此 $p = \Theta(1/\ell)$。如果对所有迹而言，确定块边界的过程都成功，那么由 [27] 可知，对于每个块，BMA 算法的错误概率为[5] $\mathcal{O}(1/\ell)$，其中 $p = \Theta(1/\ell)$，迹的数量为常数，且序列中不存在长度大于 $\ell = \lfloor 1/p \rfloor = \lfloor n^\alpha / k \rfloor$ 的游程

[5]概率 $\mathcal{O}(1/\ell)$ 上界中的常数在 [27] 中未明确给出。

因此，对 $\lceil n/\ell \rceil = \lceil n/\lfloor 1/p \rfloor \rceil$ 个块应用并集界可得

$$P(B^c|D) \leqslant \left\lceil \frac{n}{\ell} \right\rceil \mathcal{O}\left(\frac{1}{\ell}\right) \tag{33}$$
$$< (2k+1)n^{1-\alpha}\mathcal{O}\left(n^{-\alpha}\right) \tag{34}$$
$$= \mathcal{O}\left(n^{1-2\alpha}\right). \tag{35}$$

由于 $\alpha \in (0.5, 1]$，从 (35) 可以很容易看出，$P(B^c|D)$ 在 n 中渐近趋于零。因此，在解码器利用分隔符比特成功恢复块边界的情况下，重构以高概率成功，其中概率是针对删除过程的随机性而言的。

接下来，我们推导 $P(D^c)$ 的一个上界。令 Y_m，$m \in \{1, \ldots, \lceil n/\ell \rceil\}$ 表示一个随机变量，该变量表示经过删除概率为 p 的删除信道传输 $\mathbf{x}$ 次后，块 m 中被删除的确切数量。注意，$Y_m \sim$ 服从二项分布 Binomial(ℓ，p)。根据构造可知，当且仅当 $Y_m < \delta$ 时，给定块的边界才能被成功恢复，其中 $\delta = o(\log n)$ 是一个码参数。令 D_j，$j \in [t]$ 表示成功确定迹 j 中所有块边界的事件。因此，通过对 $\lceil n/\ell \rceil$ 个块应用并集界，我们得到

$$P(D_j^c) \leqslant \left\lceil \frac{n}{\ell} \right\rceil P(Y_m \geqslant \delta). \tag{36}$$

接下来，我们通过对二项分布随机变量应用以下 Chernoff 界，对 $P(Y_m \geqslant \delta)$ 给出上界。

命题 8 ([46])。设 Y 为一个二项分布随机变量，其参数为 $E[Y] = \mu$。对于任意 $\delta > 0$，

$$P(Y \geqslant (1+\delta)\mu) \leqslant \exp\left[-\mu\left((1+\delta)\ln(1+\delta) - \delta\right)\right].$$

由于 $E[Y_m] = \ell p = \lfloor 1/p \rfloor p$、$p \leqslant 1/2$ 和 $\lfloor 1/p \rfloor \geqslant 1/p - 1$，因此

$$E[Y_m] = \left\lfloor \frac{1}{p} \right\rfloor p \geqslant \left(\frac{1}{p} - 1\right) p = 1 - p \geqslant \frac{1}{2}. \tag{37}$$

因此，对二项分布随机变量 Y_m 应用命题 8，可得

$$P(Y_m \geqslant \delta) \leqslant P(Y_m \geqslant \delta E[Y_m]) \tag{38}$$
$$\leqslant e^{-E[Y_m](\delta \ln \delta - \delta + 1)} \tag{39}$$
$$\leqslant e^{-\frac{1}{2}(\delta \ln \delta - \delta + 1)}. \tag{40}$$

因此，将 (40) 代入 (36)，可得

$$P(D_j^c) \leqslant \left\lceil \frac{n}{\ell} \right\rceil e^{-\frac{1}{2}(\delta \ln \delta - \delta + 1)} \tag{41}$$
$$< (2k+1)n^{1-\alpha}e^{-\frac{1}{2}(\delta \ln \delta - \delta + 1)}, . \tag{42}$$

Let

$$\delta^*\left(p(n)\right) \triangleq \frac{2\ln\left(\sqrt{e}n^{1-\alpha}p(n)\right)}{W\left(2e\ln\left(\sqrt{e}n^{1-\alpha}p(n)\right)\right)}, \tag{43}$$

其中，$W(z)$ 是 Lambert W 函数，其定义为关于变量 w 的方程 $we^w = z$ 的解。

通过在 (42) 中将 δ 替换为 $\delta^*(p(n))$，可以很容易地得到

$$P(D_j^c) < \frac{2k+1}{p(n)}. \tag{44}$$

由于根据构造，δ 是一个整数，我们令 $\delta = \lceil \delta^*(p(n)) \rceil$，则 (44) 中的界仍然成立。此外，由于我们考虑了常数个迹 $t = \mathcal{O}(1)$，因此对这 t 个迹应用并集界，可得

$$P(D^c) \leqslant \sum_{j=1}^{t} P(D_j^c) < t \cdot \frac{2k+1}{p(n)} = \mathcal{O}\left(\frac{1}{p(n)}\right). \tag{45}$$

因此，将 (35) 和 (45) 代入 (32)，可得

$$P_e < \mathcal{O}\left(n^{1-2\alpha} + \frac{1}{p(n)}\right). \tag{46}$$

此外，删除检测算法的时间复杂度为 n 的线性阶（定理 3），而 BMA 算法的复杂度与输入矩阵 $t \times \ell$ 的大小成线性关系，因为在重建过程中，每条迹都会被扫描一次。因此，与构造 2 对应的重建算法的复杂度为 n 的线性阶，由此完成定理 4 的证明。

附录 B 推论 5 的证明

对于 $\alpha \in (0.5, 1)$，推论 5 的结果可由定理 4 得出：将 $p(n) = n^{2\alpha-1}$ 代入 δ^* 的表达式以及错误概率 P_e 的上界中；此外，还需应用 Lambert W 函数的标准界。对于 $p(n) = n^{2\alpha-1}$，有

$$\delta^* = \frac{2\ln\left(\sqrt{e}n^{\alpha}\right)}{W\left(2e\ln\left(\sqrt{e}n^{\alpha}\right)\right)}, \tag{47}$$

$$P_e < \mathcal{O}\left(n^{1-2\alpha} + \frac{1}{n^{2\alpha-1}}\right) = \mathcal{O}\left(n^{1-2\alpha}\right), \tag{48}$$

其中，式（48）中的最后一个等号是由迹的数量 t 为常数这一事实得出的。由于 α 是常数，我们有

$$2\ln\left(\sqrt{e}n^{\alpha}\right) = \ln e + 2\alpha \ln n = \Theta(\log n).$$

此外，文献 [47] 中给出了关于 Lambert W 函数的以下界。对于每个 $x \geqslant e$，

$$\ln x - \ln\ln x \leqslant W(x) \leqslant \ln x - \frac{1}{2}\ln\ln x. \tag{49}$$

由该界可得 $W(x) = \Theta(\ln x)$，因此

$$W\left(2e\ln\left(\sqrt{e}n^{\alpha}\right)\right) = W(\Theta(\log n)) = \Theta(\log\log n). \tag{50}$$

因此，我们得到 $\delta^* = \Theta(\log n/\log\log n)$。最后，根据定理 4 中关于 $r_{\mathcal{C}'}(\delta^*)$ 的界，我们可以推导出$r_{\mathcal{C}'} = \Theta(n^{1-\alpha}\log n/\log\log n)$。

附录 C 推论 6 的证明

推论 6 中的结果可由定理 4 在 $\alpha = 1$ 的情况下推出。对于 $\alpha = 1$，我们有

$$\delta^* = \frac{2\ln\left(\sqrt{e}p(n)\right)}{W\left(2e\ln\left(\sqrt{e}p(n)\right)\right)}, \tag{51}$$

$$P_e < \mathcal{O}\left(\frac{1}{n} + \frac{1}{p(n)}\right), \tag{52}$$

且 $r_{\mathcal{C}'}(\delta^*) = \Theta(\delta^*)$。因此，容易看出，对于 $p(n) = \omega(1)$，当 n 趋于无穷时，错误概率渐近趋于零，其中 $r_{\mathcal{C}'} = \omega(1)$。

Appendix D
Proof of Claim 7

对于 $x \in \mathbb{R}$，有

$$x - 1 < \lfloor x \rfloor \leqslant x \tag{53}$$

$$x \leqslant \lceil x \rceil < x + 1. \tag{54}$$

因此，

$$\left\lceil \frac{n}{\ell} \right\rceil - 1 = \left\lceil \frac{n}{\lfloor 1/p \rfloor} \right\rceil - 1 \tag{55}$$

$$< \left\lceil \frac{n}{\frac{1}{p} - 1} \right\rceil - 1 \tag{56}$$

$$< \frac{n}{\frac{1}{p} - 1} \tag{57}$$

$$= \frac{np}{1-p} \tag{58}$$

$$< 2np \tag{59}$$

$$= 2kn^{1-\alpha}, \tag{60}$$

其中，(59) 根据 $p < 1/2$ 得出。因此，

$$\left\lceil \frac{n}{\ell} \right\rceil < 2k^{1-\alpha} + 1 \leqslant (2k+1)n^{1-\alpha}, \tag{61}$$

其中，最后一个不等式在 $n \geqslant 1$ 时成立，并在 $n = 1$ 时取等号。此外，

$$\left\lceil \frac{n}{\ell} \right\rceil = \left\lceil \frac{n}{\lfloor 1/p \rfloor} \right\rceil \geqslant \lceil np \rceil \geqslant np = kn^{1-\alpha}. \tag{62}$$