破损压缩文件的修复还原

王 刚 彭 华* 唐永旺
(解放军信息工程大学 郑州 450000)

摘 要：数据压缩和解压缩已广泛应用于现代通信和数据传输领域。但是如何解压缩损坏的无损压缩文件仍然是一个挑战。针对在通用编码领域广泛使用的无损数据压缩算法，该文提出一种能够修复误码并解压还原损坏的LZSS文件的有效方法，并给出了理论依据。该方法通过利用编码器留下的残留冗余携带校验信息，在不损失任何压缩性能的情况下，能够修复LZSS压缩数据中的错误。所提方法不需要增加额外比特，也不改变编码规则和数据格式，所以与标准算法完全兼容。即采用具有错误修复能力的LZSS方案压缩的数据，仍然可以通过标准LZSS解码器进行解压。实验结果验证了所提算法的有效性和实用性。

关键词：压缩文件；残留冗余；多重匹配；错误修复
中图分类号：TP301 **文献标识码**：A **文章编号**：1009-5896(2019)08-1831-07
DOI: 10.11999/JEIT180942

Repair and Restoration of Corrupted Compressed Files

WANG Gang PENG Hua TANG Yongwang
(*PLA Information Engineering University*, *Zhengzhou* 450000, *China*)

Abstract: Data compression and decompression are widely used in modern communication and data transmission. However, how to decompress the damaged lossless compressed files is still a challenge. For the lossless data compression algorithm widely used in the general coding field, an effective method is proposed to repair the error and decompress and restore the corrupted LZSS files, and the theoretical basis is given. By using the residual redundancy left by the encoder to carry the check information, the method can repair the errors in LZSS compressed data without loss of any compression performance. The proposed method does not require additional bits or changes in coding rules and data formats, thus it is fully compatible with standard algorithms. That is, the data compressed by LZSS with error repair capability can still be decompressed by standard LZSS decoder. The experimental results verify the validity and practicability of the proposed algorithm.

Key words: Compressed files; Residual redundancy; Multiple matching; Error repair

1 引言

信源编码技术在各种通信系统中都有应用。信源编码的目的是去除冗余，用尽可能少的比特描述信息[1]，因此也称为数据压缩。无损数据压缩方法Lempel-Ziv-Storer-Szymanski(LZSS)及其变种，在众多压缩方案(ZIP, PDF, PNG, OFFICE文档等)中应用广泛，它将文件中未被压缩部分的最长前缀，替换为指向已压缩部分的该相同前缀的一个指针。LZSS的主要缺点是抗误码性能差，即使单个错误也会传播扩散，并在解压过程中产生大量误码[2]。

无损压缩文件的纠错解压方法最初是为了解压从无线信道中截获的损坏压缩文件[3,4]。如果文件在非协作通信中被拦截，则可以使用纠错解压方法修复损坏的压缩数据，因为损坏的压缩文件在这种通信模式下无法重新传输。随着Internet上的数据量急剧增加，无线传感器网络和物联网等新兴应用的出现，对所有无损压缩文件进行备份和重传变得非常困难。因此，损坏的无损压缩文件的错误修复变得非常必要，越来越受到关注。

目前，解决这个问题的主要方案是，增加额外的校验码来保护压缩数据，从而能够在解压过程中进行错误检测和纠正。文献[5]通过在压缩数据中引入3种特殊的位模式，提出了一种压缩数据的误差

收稿日期：2018-10-10；改回日期：2019-02-11；网络出版：2019-02-26
*通信作者： 彭华 phzttyw@126.com
基金项目：国家自然科学基金(61572518, 61501516)
Foundation Items: The National Natural Science Foundation of China (61572518, 61501516)

检测算法。虽然不需要使用额外的位来检测误码，但对损坏的压缩文件中的错误没有提出可行的纠正方法。文献[6]利用压缩编码规则和语法规则，建立了错误比特检测的数学模型，估计了错误比特的粗略范围，利用启发式方法确定错误的准确位置并纠正错误。但必须要有语法规则作为先验信息而且不能纠正替换错误。文献[7]利用随压缩数据一同传递的信源状态和信道状态的信息，在一定误码率条件下对压缩数据的错误进行了纠正。文献[8]针对可变长度信源编码进行分组并为分组码字额外增加了校验码。文献[9]分析了字典编码的错误传播问题，将字典编码规则中的多种模式转换为统一标准的编码，可以提高译码速度并在发生错误的情况下提供更强的健壮性，但没有解决错误的恢复问题。文献[10]通过一元编码对压缩数据中的匹配长度码、标记位等误码敏感部分进行编码，并将同步序列插入到压缩数据中。文献[11]针对压缩数据各部分重要程度的不同，提出了不等误差保护(UEP)方案来检测错误。文献[12]调整了编码器和字典表的构造规则，通过添加冗余位来验证数据。文献[13]使用压缩编码规则来检测损坏的ZIP压缩数据，由于采取了穷举纠错法，为了保证纠错速度，每次只能纠正1 bit。上述这些方法虽然能取得一定的效果，但必须在压缩数据中集成额外的校验码[14]，而不是利用压缩数据自身留下的残留冗余，因此会导致压缩性能下降。更重要的是，这些方法改变了编码规则以及数据格式，导致无法与标准算法兼容，因此实用性打了折扣。

为了有效解决LZSS压缩数据的错误修复问题，针对现有研究成果需要额外增加信息位保护数据且与标准算法无法兼容的不足，本文提出一种与标准LZSS算法保持兼容的新方案，在具有错误修复能力的同时，由于没有改变编码规则和数据格式，仍然可以通过标准LZSS解码器进行解压，并且完全不影响压缩性能。

由于LZSS编码器无法完全去除输入序列的相关性，所以压缩数据流中仍然存在冗余，这些冗余来自编码时指针码字能够从多个匹配指针中进行选择。如果最长前缀有r个匹配，通过在r个指针中选择其一能嵌入$\lfloor \log_2 r \rfloor$个比特，利用这些比特可以进行错误修复。通过对LZSS解码器的广泛测试表明，本文方法与标准LZSS算法完全兼容，且不会影响压缩性能。

本文内容安排如下：第2节介绍LZSS算法的基本原理；第3节对提出的两种纠错解压方法LZSR和LZSRD进行了详细描述；第4节讨论了LZSR算法的理论分析结果；第5节给出了实现过程并对实验结果进行了分析。

2 LZSS算法的基本原理

LZSS算法的基本原理是，在处理过的字符串$Z=(X_1, X_2, \cdots, X_{i-1})$中查找当前编码字符串$S=(X_i, X_{i+1}, \cdots, X_N)$的最长匹配前缀$(X_i, X_{i+1}, \cdots, X_{i+l-1})$，并用指向之前出现的相同前缀的指针进行替换[15]。指针用码字Y_k=(p_k, l_k)表示，其中p_k是当前索引i的最长匹配前缀的位置，l_k是最长匹配前缀的长度。图1所示的是在LZSS算法中，对起始位置为i的序列进行编码时，存在起始位置为j、长度l=6的短语与当前起始位置为i的前缀匹配。

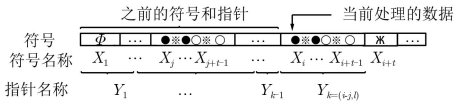

图 1 LZSS算法

设T是一个有限长度符号集A中长度为n的数据，$T_{[i]}(1 \le i \le n)$表示T中的第i个符号。用$T_{[i,\ j]}$作为子串$T_{[i]}T_{[i+1]}\cdots T_{[j]}(1 \le i \le j \le n)$的缩写。$T$的前缀用子串$T_{[1,\ j]}$表示，$T$的后缀用子串$T_{[i,\ n]}$表示。

假设字符串T的前i–1个符号已经在前k–1个短语中解析出来了，即$T_{[1,i-1]}=y_1y_2\cdots y_{k-1}$。为了识别第$k$条短语，LZSS算法查找与$T_{[1,\ i-1]}$的某个子串相匹配的$T_{[i,\ n]}$的最长前缀。如果$T_{[j,\ j+l-1]}, j \le i-l$是与最长前缀匹配的子串，那么有$y_k$=$T_{[j,\ j+l-1]}$。算法给出指针码字$(i-j,\ l)$，然后将当前的位置值由$i$更新为$i+l$。

3 破损LZSS文件的修复还原

信源编码中的残留冗余主要来自两个方面：一是编码过程本身引入的冗余，另一种是编码器忽略信源的分布特性而引入的冗余[16]。本文方法在保证兼容性和不降低压缩率的前提下，利用字典编码的残留冗余，将校验信息嵌入到压缩数据流中以提供保护能力，该信息可用于检测和修复压缩流中的错误。

3.1 LZSR算法

由于LZSS编码器无法完全去除输入序列的相关性，所以压缩数据流中仍然存在残留冗余。对于给定的序列或短语，一般会有不止一个最长的匹配前缀，这意味着存在不止一个匹配指针。通常，算法选择最新的指针[17]，即最小的位置值，但是选择另一个指针不会影响到解压缩过程和结果。如果一个短语的起始位置距离输入序列的开头为i，在序列中存在与从位置i开始的完全匹配的r个最长前

缀，则称该短语具有匹配的多重性r。残留冗余正是来自于编码时指针码字能够从r>1个多重匹配中进行选择，通过从r个指针选项中选择其一可以嵌入$\lfloor \log_2 r \rfloor$个额外比特，这些比特可以用于认证[18]或误码修复。

设X的初始部分$X_{[1,\ i-1]}$已经解析过了，对所有0≤m≤r–1，设$\{(p_0, l), (p_1, l), \cdots, (p_m, l), \cdots, (p_{r-1}, l)\}$，$r \geq 1$，为$X_{[i,\ n]}$的最长匹配前缀的所有可能指针，其中$l$>1, $1 \leq p_m \leq i-l$。当r>1时，根据待嵌入数据F中的$d = \lfloor \log_2 r \rfloor$个比特的值来选择$r$个指针中的一个。假设待嵌入数据$F$的前$t$个比特已嵌入到前面的短语中，则编码结果为$(p_{F[t+1,\ t+d]},\ l)$，接着把$X$的当前位置移动到$i$+$l$，并把$t$递增$d$。当存在$r$>1个相同的最长匹配指针时，可以通过指针的合理选择来对额外的比特进行编码。如图2所示，最长匹配前缀的数量r=4，通过从4个匹配指针中选择其一，可嵌入两个比特。额外的比特可以用来表示校验码并进行错误检测和修复。由于选择不同的指针对解压结果没有影响，因此所提算法与标准LZSS解码器完全兼容。

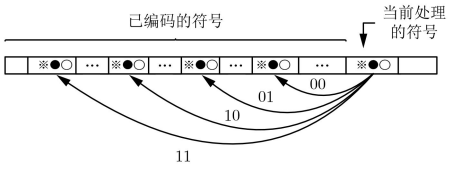

图 2 最长匹配前缀的多重性

确定了LZSS算法的残留冗余后，需要设计利用冗余来修复误码的方法。由于被保护的码字是由字节序列表示的，因此选择Reed-Solomon(RS)码[19]保护数据。RS码是一种广泛应用于数字通信和存储系统中的纠错码，通常表示为RS(a, b)，其中a是包含数据和校验码的分组的大小，b是有效载荷的大小。RS解码器可以纠正分组中的错误数量为e=(a–b)/2。给定一个用s bit表示的码元，RS码的最大分组长度a=2^s–1。因此s=8时，RS码可以用RS(255, 255–2e)表示。

接下来详细介绍如何采用LZSR算法嵌入RS校验码。该方案首先使用标准LZSS算法压缩T，把编码后的压缩数据分成大小为255–2e的分组。从最后一个分组开始，按照逆序依次处理。当处理分组G_i时，编码器首先计算分组G_{i+1}的RS校验码，并利用匹配的多重性将校验码嵌入到分组G_i的指针码字中。第1个分组G_1的校验码保存在压缩文件的开头。LZSR编码器处理压缩数据的操作流程如图3所示。

解压过程按照正向顺序进行。解码器接收指针码字序列，最前面是第1个分组G_1的校验码。首先

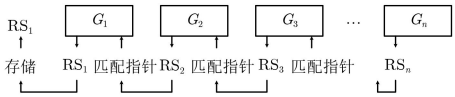

图 3 LZSR编码器(RS$_n$表示分组G_n的校验码)

将输入数据流分成大小为255–2e的分组，然后使用校验码来校正第1个分组G_1，一旦G_1正确，就用LZSR算法对其解压。这不仅重构了原始文件的第1个分组，也恢复了在特定选择的指针码字中存储的比特信息。这些额外的比特被收集起来，成为分组G_2的校验码，由此解码器可以纠正G_2中可能出现的误码，然后解压G_2，并提取出分组G_3的校验码。这个过程一直持续到所有分组都被解压为止。

由于解码器在解压前需要利用RS码进行校验，因此将当前分组的校验码嵌入前一个分组中，就可以随着前一个分组的解压得到当前分组的RS码并对当前分组进行校验。所以，编码器需要以反序处理这些分组。

根据编码数据的冗余，在多重匹配指针中嵌入比特的能力，决定了解码时在分组中能够有效校正的误码e的最大数目。在LZSR算法中，e在所有分组中都是常数，因此它的值受到具有最低冗余的分组的限制。

3.2 LZSRD算法

在LZSR算法中，由于数据不同部分的冗余可能会有很大的不同，因此在所有的编码分组中，如果e取一个常数，那么算法不是最优的。如果数据中有一个分组的冗余度非常低，那么它将决定所有分组中e的最大值。为了更好地利用总体冗余性，可以根据每个分组中冗余比特的可利用性，采用在每个分组之间动态调整的e值。在这种情况下，数据的低冗余部分只会影响到这些部分的错误保护性能和信息的嵌入量，而数据的其余部分可以根据其冗余的可利用性得到更好的保护。因此，动态调整的e的平均值更高，从而更好地抵抗误码影响。

根据上述研究和分析，在LZSR的基础上提出了LZSRD算法。编码过程仍然分两个阶段进行。但在第一阶段，编码器除了压缩输入流之外，还决定每个分组的嵌入能力，这些信息用于确定RS码的参数。输入字符串X首先使用标准LZSS算法进行编码，同时记录每个指针具有的相同最长匹配的数量r，然后根据可用冗余能够嵌入的比特数量，将待嵌入数据划分为不同长度的分组，如图4(a)所示。首先，第一个分组G_1的数据长度为255–2e_1，e_1作为算法的参数输入。根据分组G_{i-1}的r的值，计算分组G_i的校验码的位数，e_i的计算方法如下:

$$e_i = \left\lfloor \left(\sum_{m \in G_{i-1}} \lfloor \log_2 r_m \rfloor \right) \Big/ 16 \right\rfloor \tag{1}$$

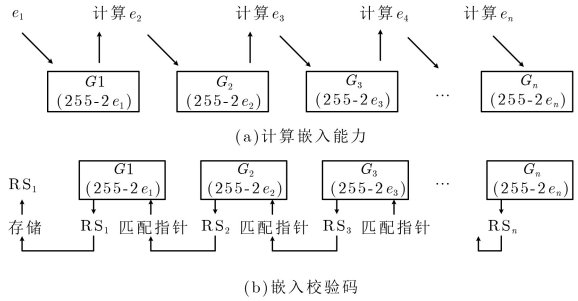

图 4 LZSRD编码器(RS_n表示分组G_n的校验码)

最后得到了数据长度分别为$255-2e_n$的n个分组，把所有数据切分成不同长度的分组后，进行嵌入校验码的过程。在第2阶段中，这些分组数据按照从最后一个到第1个的反序进行处理，每个分组可以嵌入的信息位的数量$2e_i$是不同的。在处理分组G_{i+1}时，编码器读取与分组G_i相关的RS码的参数信息，根据这些信息选择合适的RS编码器并计算分组G_{i+1}的校验位。这些位会被嵌入到分组G_i中。这个过程中，最后一个分组只包含压缩符号，没有其他附加信息。第1个分组G_1的期望误码校正能力e_1是作为输入参数给出的，对于所有其它分组，期望误码校正能力e_i都是根据其前一分组的冗余动态得到的。在LZSRD算法中，第1个分组G_1的校验码添加到编码器输出流的开头。如图4(b)所示。

解压过程与LZSR类似。首先从分组G_{i-1}中提取出嵌入的$2e_i$个校验码，分组G_i使用该校验码确定数据长度并进行误码纠正，然后用LZSR解码器解压出分组G_i的原始数据，并获得下一个分组G_{i+1}的$2e_{i+1}$个校验码，该校验码用来确定分组G_{i+1}的数据长度并校验该分组。这个过程重复执行，直到最后一个分组。

4 LZSR的分析

本节对LZSR算法中匹配的多重性进行了分析，当数据由无记忆信源生成时，根据该分析结果，能够估计最长匹配前缀数量的分布特性。

设$T_{[1,\,n]}$是信源产生的前n个符号，L是在$T_{[1,\,n]}$中找到的$T_{[n+1,\,\infty]}$的最长匹配前缀的长度。用变量W_n表示$T_{[n+1,\,\infty]}$的最长匹配前缀的数量，即$W_n=\sum_{i=1}^{n-L}1\left(T_{[i,i+L-1]}=T_{[n,n+L-1]}\right)$。

设$0<p<1$，$q=1-p$。定义$X(j)$为序列$X_1^{(j)}X_2^{(j)}X_3^{(j)}\cdots$，其中$\{X_i^{(j)}|i,j\in\mathbf{N}\}$是$\{0,1\}$上独立同分布的随机变量集合，且$P\{X_i^{(j)}=0\}=p$。设$l_j^{(n)}=\sup\left\{i\geq 0|X_1^{(j)}\cdots X_i^{(j)}=X_1^{(n+1)}\cdots X_i^{(n+1)}\right\}$。定义$L_n=\max_{j\leq n}l_j^{(n)}$，即$L_n$表示第$(n+1)$个字符串在前$n$个字符串中最长前缀的长度。最后，定义$M_n=\#\left\{j|1\leq j\leq n,l_j^{(n)}=L_n\right\}$，因此$M_n$表示第$(n+1)$个字符串在前$n$个字符串中长度为$L_n$的最长前缀的数量，令$M_0=0$。如果字符串$X(1),X(2),\cdots,X(n)$是$T$的后缀，则$M_n$渐近等价于$W_n$。

$C_{j_1,\cdots,j_k}$是k个字符串$X(j_1),X(j_2),\cdots,X(j_k)$中最长公共前缀的长度，可知$l_j^{(n)}=C_{j,n+1}$。在由$(n+1)$个字符串构造的树中，第$k$个字符串的深度$D_{n+1}(k)$是从树根节点到包含第$k$个字符串的叶节点的路径长度。已知$D_{n+1}(n+1)=\max_{1\leq j\leq n}C_{j,n+1}+1$，可得$L_n=D_{n+1}(n+1)-1$。因此，$M_n=\#\{j|1\leq j\leq n,C_{j,n+1}+1=D_{n+1}(n+1)\}$，即$M_n$表示从新插入分支节点开始的子树的大小。

定义指数生成函数

$$\left.\begin{aligned}G(z,u)&=\sum_{n\geq 0}\mathrm{E}\left[u^{M_n}\right]\frac{z^n}{n!}\\W_j(z)&=\sum_{n\geq 0}\mathrm{E}\left[(M_n)^{\underline{j}}\right]\frac{z^n}{n!}\end{aligned}\right\}\tag{2}$$

其中复数$u\in\mathbb{C}$且$j\in\mathbb{N}$。如果$f:\mathbb{C}\to\mathbb{C}$，那么递归关系

$$\begin{aligned}\mathrm{E}[f(M_n)]=&p^n(qf(n)+p\mathrm{E}[f(M_n)])+q^n(pf(n)\\&+q\mathrm{E}[f(M_n)])+\sum_{k=1}^{n-1}\binom{n}{k}p^kq^{n-k}\\&(p\mathrm{E}[f(M_k)]+q\mathrm{E}[f(M_{n-k})])\end{aligned}\tag{3}$$

对于所有$n\in\mathbb{N}$成立。如果$f(0)=0$，那么当$n=0$时递归也成立。为了验证递归，只要考虑当$1\leq j\leq n+1$时$X_1^{(j)}$的可能值。

首先，如果$n\in\mathbb{N}$，那么

$$\begin{aligned}\mathrm{E}\left[u^{M_n}\right]=&p^n\left(qu^n+p\mathrm{E}\left[u^{M_n}\right]\right)+q^n\Big(pu^n\\&+q\mathrm{E}\left[u^{M_n}\right]\Big)+\sum_{k=1}^{n-1}\binom{n}{k}p^kq^{n-k}\\&\left(p\mathrm{E}\left[u^{M_k}\right]+q\mathrm{E}\left[u^{M_{n-k}}\right]\right)\end{aligned}\tag{4}$$

如果$j\in\mathbb{N}$且$n\geq 0$，那么

$$\begin{aligned}\mathrm{E}\left[(M_n)^{\underline{j}}\right]=&p^n\left(qn^{\underline{j}}+p\mathrm{E}\left[(M_n)^{\underline{j}}\right]\right)\\&+q^n\left(pn^{\underline{j}}+q\mathrm{E}\left[(M_n)^{\underline{j}}\right]\right)\\&+\sum_{k=1}^{n-1}\binom{n}{k}p^kq^{n-k}\left(p\mathrm{E}\left[(M_k)^{\underline{j}}\right]\right.\\&\left.+q\mathrm{E}\left[(M_{n-k})^{\underline{j}}\right]\right)\end{aligned}\tag{5}$$

利用Ward等人[20]和Jacquet等人[21]的研究成果可得出这些递归关系的渐近解。

设 $z_k=\dfrac{2kr\pi i}{\ln p}$ 对于所有$k\in\mathbb{Z}$，且$\dfrac{\ln p}{\ln q}=\dfrac{r}{s}\cdot(r,s\in\mathbb{Z})$，有

$$\mathrm{E}\left[(M_n)^{\underline{j}}\right]=\Gamma(j)\frac{q\left(\frac{p}{q}\right)^j+p\left(\frac{q}{p}\right)^j}{h}+\delta_j\left(\log_{\frac{1}{p}}n\right)-\frac{1}{2}+n\left(\frac{\mathrm{d}^2}{\mathrm{d}z^2}\delta_j\left(\log_{\frac{1}{p}}z\right)\right)\Big|_{z=n}+O\left(n^{-2}\right)\tag{6}$$

其中

$$\delta_j(t)=\sum_{k\neq0}-\frac{e^{2kr\pi it}\Gamma(z_k+j)\left(p^jq^{-z_k-j+1}+q^jp^{-z_k-j+1}\right)}{p^{-z_k+1}\ln p+q^{-z_k+1}\ln q}\tag{7}$$

Γ是伽玛函数。式(7)中项$-\dfrac{1}{2}+n\left(\dfrac{\mathrm{d}^2}{\mathrm{d}z^2}\delta_j\left(\log_{\frac{1}{p}}z\right)\right)\Big|_{z=n}$满足$O\left(n^{-1}\right)$。其中，$\delta_j$是一个波动幅度很小的周期函数。如果$\ln p/\ln q$是无理数，那么当$x\to\infty$时$\delta_j(x)\to0$。$M_n$的渐近分布满足：

$$\mathrm{E}\left[u^{M_n}\right]=-\frac{q\ln(1-pu)+p\ln(1-qu)}{h}+\delta\left(\log_{\frac{1}{p}}n,u\right)-\frac{1}{2}+n\left(\frac{\mathrm{d}^2}{\mathrm{d}z^2}\delta_j\left(\log_{\frac{1}{p}}z,u\right)\right)\Big|_{z=n}+O\left(n^{-2}\right)\tag{8}$$

其中

$$\delta(t,u)=\sum_{k\neq0}-\frac{e^{2kr\pi it}\Gamma(z_k)\left(q(1-pu)^{-z_k}+p(1-qu)^{-z_k}-p^{-z_k+1}-q^{-z_k+1}\right)}{p^{-z_k+1}\ln p+q^{-z_k+1}\ln q}\tag{9}$$

Γ为伽玛函数。由此可得到

$$\mathrm{E}\left[u^{M_n}\right]=\sum_{j=1}^{\infty}\left[\frac{p^jq+q^jp}{jh}+\sum_{k\neq0}-\frac{e^{2kr\pi i\log_{\frac{1}{p}}n}\Gamma(z_k)(p^jq+q^jp)(z_k)^{\underline{j}}}{j!\left(p^{-z_k+1}\ln p+q^{-z_k+1}\ln q\right)}\right]u^j+O\left(n^{-1}\right)\tag{10}$$

则

$$P(M_n=j)=\frac{p^jq+q^jp}{jh}+\sum_{k\neq0}-\frac{e^{2kr\pi i\log_{\frac{1}{p}}n}\Gamma(z_k)\left(p^jq+q^jp\right)(z_k)^{\underline{j}}}{j!\left(p^{-z_k+1}\ln p+q^{-z_k+1}\ln q\right)}+O\left(n^{-1}\right)\tag{11}$$

如果$\ln p/\ln q$是无理数且u是固定的，那么当$x\to\infty$时$\delta(x,u)\to0$，得到$P(M_n=j)=\dfrac{p^jq+q^jp}{jh}$。所以，完全可以利用LZSS压缩时匹配的多重性进行错误的修复。

5　实验及性能分析

坎特伯雷语料库(Canterbury Corpus)[22]和卡尔加里语料库(Calgary Corpus)[23]是一组文件的集合，用于无损数据压缩算法的基准测试，本文使用这两个语料库进行实验。标准LZSS编码器，使用长度为32 kB的滑动窗口，匹配短语的最大长度为256。我们使用一台2.60 GHz CPU和8 GB RAM内存的台式电脑评估本文方法的性能。

5.1　LZSS的嵌入能力

由于本文方法需要利用冗余携带校验码对数据进行保护，所以首先对LZSS压缩数据的嵌入能力进行测试。图5记录了用LZSS压缩数据时，坎特伯雷语料库和卡尔加里语料库中，所有文件的最长匹配短语的数量r，随文件长度变化的平均值。分析图5可以发现，r的值渐近收敛于某个常数。这是因为LZSS属于滑动窗口式字典编码，在压缩初始阶段，由于字典中条目不多，所以最长匹配短语的数量较少。随着压缩的进行，字典逐渐填满，最长匹配短语的数量也逐渐增加并趋于稳定。

图6给出了坎特伯雷语料库和卡尔加里语料库中的文件，被压缩成LZSS文件时，可以在压缩数据中嵌入的比特数量的平均值。图6表明，能够在文件F中嵌入的比特数量随着文件的长度$|F|$线性增长。由于长为255–2e的分组数据，需要2e个校验位来纠正e个错误，只要压缩数据的嵌入率不小于$2e/(255-2e)$就能实现纠正e个错误。实验结果表明，当采用LZSS压缩文件，最长匹配短语的数量趋于稳定时，每个数据分组中，能够满足e=2甚至更大值时需要的嵌入率。

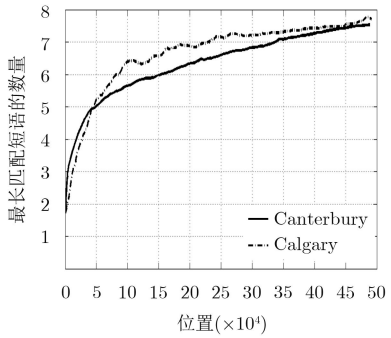

图 5 最长匹配短语数量的平均值与文件长度的关系

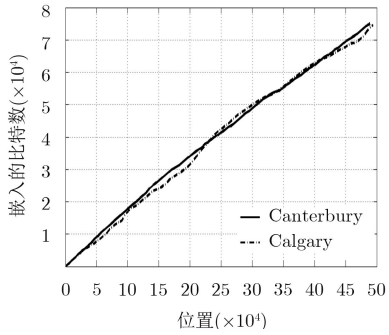

图 6 嵌入的比特数量与文件长度的关系

5.2 LZSRD的错误修复能力

本小节将比较本文提出的LZSRD方法和文献[6]的错误修复能力。文献[6]方法是利用压缩编码规则和语法规则，确定错误的位置并修复错误。本文方法是利用LZSS编码器留下的残留冗余携带校验信息，发现并修复错误。

在测试错误修复能力时，把坎特伯雷语料库和卡尔加里语料库中的所有文件压缩成LZSS文件。通过引入随机分布在整个压缩数据中的不同数量的错误，来测试对错误的修复能力。实验时，向LZSS文件注入不同数量的错误进行测试，随着压缩数据中错误数量(用误比特率(BER)表示)的变化，对于每个文件每次确定的错误数量，进行100次具有不同随机分布错误的实验，分别使用LZSRD和文献[6]中的方法修复错误并解压文件，统计正确修复错误并成功解压文件的概率的平均值。

两种方法实验结果的比较显示在图7中。实线为本文方法成功解压的概率均值，虚线为文献[6]方法成功解压的概率均值。从图7可以看出，在BER=6×10^{-4}时，文献[6]几乎无法正确解压文件，而LZSRD能够成功解压的概率约为0.9。因此，本文方法的错误修复能力远高于文献[6]。这是因为文献[6]的方法，无法发现不会破坏字典结构的替换错误，而且只有当错误比特间的间隔大于错误检测延迟时，错误比特才可能被纠正。随着BER的增加，一旦间隔小于错误检测延迟，则无法纠正错误比特。因此

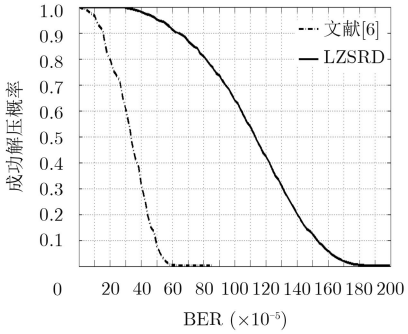

图 7 错误修复能力的比较

文献[6]的纠错能力较低。而本文方法只要满足在每个分组数据中的错误数不超过校验能力，就能修复错误并成功解压。

5.3 LZSRD的实用性

把坎特伯雷语料库和卡尔加里语料库中的所有文件压缩成LZSS文件。每个LZSS文件中添加一些均匀分布的错误比特，以生成损坏的LZSS文件。在BER=10^{-5}时，分别使用LZSRD和文献[6]中的方法修复错误并解压文件，统计正确修复错误并成功解压文件的概率的平均值。

测试结果显示在图8中。实线为本文方法成功解压的概率均值，虚线为文献[6]方法成功解压的概率均值。如图8所示，随着压缩文件长度的增加，文献[6]的纠错能力呈指数曲线下降。这是因为，如果LZSS文件中错误比特的位置是随机的，则文献[6]的错误修复率必定小于1。由于其错误模式每次最多只能纠正1 bit，因此，当LZSS文件的长度无限时，文献[6]的方法最终会失败。而本文方法采用的RS码具有稳定的错误修复能力，纠错率只会受到误码率的影响，几乎不会受到文件长度的影响。

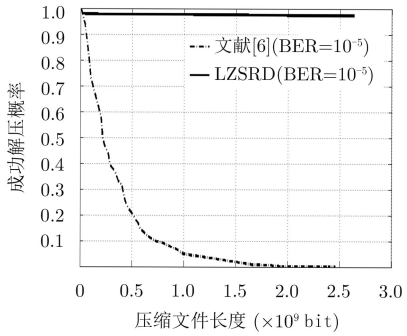

图 8 纠错率与压缩文件长度的关系

6 结束语

LZSS是通用压缩领域应用极为广泛的无损数据压缩方法，但解压过程引发的错误传播扩散限制了其应用。为了有效解决LZSS压缩数据的错误修复问题，现有方法需要增加额外的信息位保护数据，这导致了压缩性能的下降且与标准算法无法兼容。针对这些不足，本文提出了与标准LZSS算法保持兼容的新方案，由于没有改变编码规则和数据格式，仍然可以通过标准LZSS解码器进行解压。新方案不需要额外增加任何比特，在具有错误修复能力的同时，完全不影响压缩性能。

参 考 文 献

[1] DRMOTA M and SZPANKOWSKI W. Redundancy of lossless data compression for known sources by analytic methods[J]. *Foundations and Trends in Communications*

and Information Theory, 2016, 13(4): 277–417. doi: 10.1561/0100000090.

[2] DAS S, BULL D M, and WHATMOUGH P N. Error-resilient design techniques for reliable and dependable computing[J]. *IEEE Transactions on Device and Materials Reliability*, 2015, 15(1): 24–34. doi: 10.1109/TDMR.2015.2389038.

[3] MENGHWAR G D and MECKLENBRAUKER C F. Cooperative versus non-cooperative communications[C]. The 2nd International Conference on Computer, Control and Communication, Karachi, Pakistan, 2009: 1–3.

[4] HAMSCHIN B M, FERGUSON J D, and GRABBE M T. Interception of multiple low-power linear frequency modulated continuous wave signals[J]. *IEEE Transactions on Aerospace and Electronic Systems*, 2017, 53(2): 789–804. doi: 10.1109/TAES.2017.2665140.

[5] KWON B, GONG M, and LEE S. Novel error detection algorithm for LZSS compressed data[J]. *IEEE Access*, 2017, 5: 8940–8947. doi: 10.1109/ACCESS.2017.2704900.

[6] WANG Digang, ZHAO Xiaoqun, and SUN Qingquan. Novel fault-tolerant decompression method of corrupted Huffman files[J]. *Wireless Personal Communications*, 2018, 102(4): 2555–2574. doi: 10.1007/s11277-018-5277-5.

[7] KOSTINA V, POLYANSKIY Y, and VERDú S. Variable-length compression allowing errors[J]. *IEEE Transactions on Information Theory*, 2015, 61(8): 4316–4330. doi: 10.1109/TIT.2015.2438831.

[8] ZHANG Jie, YANG Enhui, and KIEFFER J C. A universal grammar-based code for lossless compression of binary trees[J]. *IEEE Transactions on Information Theory*, 2014, 60(3): 1373–1386. doi: 10.1109/TIT.2013.2295392.

[9] KLEIN S T and SHAPIRA D. Practical fixed length Lempel-Ziv coding[J]. *Discrete Applied Mathematics*, 2014, 163: 326–333. doi: 10.1016/j.dam.2013.08.022.

[10] KITAKAMI M and KAWASAKI T. Burst error recovery method for LZSS coding[J]. *IEICE Transactions on Information and Systems*, 2009, E92.D(12): 2439–2444. doi: 10.1587/transinf.e92.d.2439.

[11] PEREIRA Z C, PELLENZ M E, SOUZA R D, *et al*. Unequal error protection for LZSS compressed data using Reed-Solomon codes[J]. *IET Communications*, 2007, 1(4): 612–617. doi: 10.1049/iet-com:20060530.

[12] LAKHANI G. Reducing coding redundancy in LZW[J]. *Information Sciences*, 2006, 176(10): 1417–1434. doi: 10.1016/j.ins.2005.03.007.

[13] PARK B, SAVOLDI A, GUBIAN P, *et al*. Recovery of damaged compressed files for digital forensic purposes[C]. 2008 International Conference on Multimedia and Ubiquitous Engineering, Busan, South Korea, 2008: 365–372. doi: 10.1109/MUE.2008.49.

[14] KOSTINA V, POLYANSKIY Y, and VERD S. Joint source-channel coding with feedback[J]. *IEEE Transactions on Information Theory*, 2017, 63(6): 3502–3515. doi: 10.1109/TIT.2017.2674667.

[15] KEMPA D and KOSOLOBOV D. LZ-end parsing in compressed space[C]. 2017 Data Compression Conference, Snowbird, USA, 2017: 350–359.

[16] 徐金甫, 刘露, 李伟, 等. 一种基于阵列配置加速比模型的无损压缩算法[J]. 电子与信息学报, 2018, 40(6): 1492–1498. doi: 10.11999/JEIT170900.

XU Jinfu, LIU Lu, LI Wei, *et al*. A new lossless compression algorithm based on array configuration speedup model[J]. *Journal of Electronics & Information Technology*, 2018, 40(6): 1492–1498. doi: 10.11999/JEIT170900.

[17] DO H H, JANSSON J, SADAKANE K, *et al*. Fast relative Lempel-Ziv self-index for similar sequences[J]. *Theoretical Computer Science*, 2014, 532: 14–30. doi: 10.1016/j.tcs.2013.07.024.

[18] ATALLAH M J and LONARDI S. Augmenting LZ-77 with authentication and integrity assurance capabilities[J]. *Concurrency and Computation: Practice and Experience*, 2004, 16(11): 1063–1076. doi: 10.1002/cpe.804.

[19] REED I S and SOLOMON G. Polynomial codes over certain finite fields[J]. *Journal of the Society for Industrial and Applied Mathematics*, 1960, 8(2): 300–304. doi: 10.1137/0108018.

[20] WARD M D and SZPANKOWSKI W. Analysis of a randomized selection algorithm motivated by the LZ'77 scheme[C]. The 1st Workshop on Analytic Algorithmics and Combinatorics, New Orleans, USA, 2004: 153–160.

[21] JACQUET P and SZPANKOWSKI W. Analytical depoissonization and its applications[J]. *Theoretical Computer Science*, 1998, 201(1/2): 1–62. doi: 10.1016/S0304-3975(97)00167-9.

[22] The Canterbury corpus[EB/OL]. http://corpus.canterbury.ac.nz/descriptions/#cantrbry, 2018.

[23] The Calgary corpus[EB/OL]. http://corpus.canterbury.ac.nz/descriptions/#calgary, 2018.

王　刚：男，1981年生，副教授，研究方向为信号分析、信息处理、模式识别.

彭　华：男，1973年生，教授，研究方向为通信信号处理、软件无线电.

唐永旺：男，1981年生，讲师，研究方向为信息处理、协议分析.