·开发研究与工程应用· 文章编号:1000-3428(2017)03-0299-05 文献标志码:A 中图分类号:TP301.6

无损压缩算法 LZW 前缀编码优化及应用

鄢海舟[1],胥布工[1],石东江[2],郑伟德[3]

(1. 华南理工大学 自动化科学与工程学院,广州 510641;2. 华能国际电力股份有限公司湖南分公司,长沙 410002;
3. 华能湖南岳阳发电有限责任公司,湖南 岳阳 414002)

摘 要: 利用 LZW 算法进行数据压缩,当字典长度为 l 时,前缀在区间[0,$l-1$]中,因而只能通过⌈lb$l$⌉ bit 对前缀进行编码,使区间[l,$2^{\lceil lb l\rceil}-1$]内的数据不能被充分利用,造成冗余现象。针对该问题,提出一种前缀映射编码的改进压缩算法。对满足条件的前缀不直接编码输出,而是将其映射到区间[l,$2^{\lceil lb l\rceil}-1$],此时编码位数并未减少但却隐含一个标志位信息,标志下一个前缀编码用更少的比特位来编码。与原 LZW 算法相比,改进算法不增加计算量和存储空间,并且通用性好。英文文本数据压缩仿真和残差数据压缩应用结果均验证了该算法的有效性。

关键词: LZW 算法;前缀映射;编码;无损压缩;标志位;去冗余

中文引用格式: 鄢海舟,胥布工,石东江,等. 无损压缩算法 LZW 前缀编码优化及应用[J]. 计算机工程,2017,43(3):299-303.

英文引用格式: Yan Haizhou,Xu Bugong,Shi Dongjiang,et al. Prefix Encoding Optimization and Application of Lossless Compression Algorithm LZW[J]. Computer Engineering,2017,43(3):299-303.

Prefix Encoding Optimization and Application of Lossless Compression Algorithm LZW

YAN Haizhou[1],XU Bugong[1],SHI Dongjiang[2],ZHENG Weide[3]

(1. School of Automation Science and Engineering,South China University of Technology,Guangzhou 510641,China;
2. Hunan Branch,Huaneng Power International Co.,Inc.,Changsha 410002,China;
3. Huaneng Hunan Yueyang Power Generation Co.,Ltd.,Yueyang,Hunan 414002,China)

【Abstract】 LZW algorithm can not make full use of the length of dictionary for prefix encoding when compressing data and has redundancy. When considering the length of the dictionary is l, the prefix can be located within [0,$l-1$] only, and the LZW algorithm uses ⌈lb$l$⌉ bit to encode the prefix. Therefore, the data of the interval [l,$2^{\lceil lb l\rceil}-1$] is not fully used. Aiming at this problem, this paper proposes an improved compression algorithm of prefix mapping encoding. It does not encode and output the prefix which meets the condition, but maps it to the interval [l,$2^{\lceil lb l\rceil}-1$]. In this case, the number of prefix encoding is not decreased but implies the information that the next prefix will be encoded with less bits. Compared with LZW agorithm, the improved algorithm has advantages that it does not increase the amount of calculation and memory space, and can be used more generally in data compression. The effectiveness of the improved algorithm is verified when tested with English text data in simulation experiments and applied into residual data compression.

【Key words】 LZW algorithm;prefix mapping;encoding;lossless compression;flags;redundancy elimination

DOI: 10.3969/j.issn.1000-3428.2017.03.050

0 概述

随着信息时代的到来,各领域需要分析、传送和储存的数据也与日俱增。对数据进行压缩后再传送与储存,既可以保证传送的速度,也可以有效节省储存空间,这使得无损压缩算法十分有用[1]。早期的无损压缩算法主要是基于统计方法[2-4],直到 1977 年,以色列学者 Ziv J 和 Lempel A 利用数据流中重复数据可用短码编码的思想,提出了基于字典的压缩算法,称为 LZ77 压缩算法[5]。次年,两人对

基金项目: 国家自然科学基金-广东联合基金重点项目(U1401253)。

作者简介: 鄢海舟(1993—),男,硕士研究生,主研方向为电力数据去噪与压缩;胥布工(通信作者),教授、博士;石东江、郑伟德,高级工程师。

收稿日期: 2016-06-14 **修回日期:** 2016-07-20 **E-mail:** yan_haizhou@126.com

算法进行改进,用“前缀,字符”在字典中匹配代替了原来在窗口中查找相同字符,新算法减少了“前缀,字符”的比较数目,称之为 LZ78 算法[6]。之后,对 LZ78 算法的改进越来越多,LZW 算法[7]就是其中一个变种。当数据重复度大时,LZW 算法压缩比优于基于统计的压缩算法。相对基于统计的压缩算法,基于字典的压缩算法在运算速度上更具优势,因为前者需要扫描 2 遍原数据。LZW 压缩算法在压缩及解压数据时,动态生成一个字典,用于保存已压缩的历史数据,当“前缀,字符”可在字典中查找到时,则继续读入待编码数据,直到“前缀,字符”无法在字典中匹配时,对前缀进行编码输出以达到压缩的目的。LZW 压缩算法在压缩时间及压缩性能上的优势使其应用范围十分广泛,如文献[8]使用压缩感知及 LZW 编码来压缩心电图信号。

本文分析 LZW 压缩算法基本原理,在其基础上提出改进算法,并通过英文文本数据的仿真及残差数据的压缩应用验证该算法的有效性。

1 LZW 算法分析

LZW 算法的基本思想是依据已编码字符建立一个字典(或称位串表),再将未编码字串映射为串表的索引编码输出。具体过程如下:

步骤 1 将待编码数据中所有可能出现的字符置于字典头部,同时将前缀 p 置空。

步骤 2 当待编码数据中还有数据需编码时,转步骤 3;否则,转步骤 5。

步骤 3 读入待编码数据中下一个字符 c,若字符串 $p+c$ 在字典中,则将前缀 p 置为 c;否则将字符串 $p+c$ 加入字典中,输出前缀 p 编码,且令 $p=c$。

步骤 4 转步骤 2。

步骤 5 结束编码。

重复上述步骤,即可对输入数据流进行编码,但还需考虑一个重要问题——字典的长度。事实上,许多 LZW 改进算法就是针对字典进行改进的。因为当字典长度一定时,提高字符串 $p+c$ 匹配的比例能很大程度上提高压缩比。一般地,对由256 个字符组成的字符流编码时,可设置字典为12 bit。原 LZW 算法所采用的是定长码字编码,即每次输出前缀用固定的 12 bit 编码,显然这样输出流中有许多冗余。目前,在使用 LZW 算法时,一般都根据前缀所处的阶段用变长码字编码。以标准码长为 12 bit 的 LZW 压缩算法为例,该算法编码表可以容纳4 096 个码字。整个编码表分成 5 个部分:0 ~ 255,256 ~ 511,512 ~ 1 023,1 024 ~ 2 047,2 048 ~ 4 095,每一部分输出的代码字长度分别为 8 bit,9 bit,10 bit,11 bit,12 bit。若使用该字典来压缩一串由 256 个字符构成的数据,则字典初始化时0 ~ 255 被数据流中根字符占据,因此,编码从 258 开始(可设置 256 为重建字典,257 为结束标志),不断加入新串到字典中。若字典长度在第 2 部分则用9 bit编码前缀,若字典长度在第 3 部分则用 10 bit 编码前缀,以此类推。本文中提出的改进算法也是在此基础上进行改进的。

由以上分析知,LZW 压缩算法能够压缩的原因是用 k bit 字典索引去编码一个或多个字符。k bit 编码的字符越多则压缩效果越好。因此,许多提高字典匹配的改进方法被提出。文献[9]实现并改进了 LZW 算法,使其适用于中文压缩。文献[10]考虑到使用固定长度字典不能适应待编码数据分布规律发生变化的情况,提出一种自适应变长字典的改进方法,即当压缩比下降到一定程度时即重建字典。此方法对分布规律多变的数据有一定效果,但对较平稳数据效果不好,因为重建字典时容易使压缩比下降。文献[11-12]则分别从去除根字符所占用的空间、去除输出机制的冗余性出发进行考虑。两者均在不同程度上提高了压缩比。文献[12]算法的通用性较好,但因其引入了双索引机制,故需要额外存储空间。文献[13]使用一种双串表机制来提高匹配字典的概率。文献[14]先使用 BWT 算法,再使用 LZW 算法进行压缩以提高字典匹配率,从而提高压缩比。文献[15]采用 hash 查找法,从字典的存储角度出发来加快匹配字典的速度。上述改进方法能在不同程度上提高压缩比,但都有不足之处,或是增加了内存和计算量,或是只针对某类数据性能有所提高。

2 LZW 算法改进

以上提高压缩比的改进方法可分为 2 类:一类是基于字典的改进方法;另一类则是从前缀编码去冗余角度进行考虑。因 LZW 算法先进行字典“前缀,字符”的匹配再输出前缀编码,故两类方法可以结合使用,但因每类方法在计算或内存方面有所增加,故结合使用需牺牲这两方面的性能。而数据压缩的本质是去冗余的过程[4,16],因此,下文试图提出一种不增加计算、内存等额外开销的改进算法。该算法不增加计算量和储存,只需在原算法基础上进行简单的判断即可实现。

2.1 前缀映射编码

使用 LZW 对前缀编码一般采用变长编码,即当字典长度 l 在 256 ~ 511 之间时采用 $k=9$ bit 编码,在 512 ~ 1 023 时采用 $k=10$ bit 编码,在 1 024 ~ 2 047时采用 $k=11$ bit 编码。那么,当字典长度 l 为非 2 的整数次幂时,区间 $[l,2^k-1]$ 的数据将永远不可能被编码。由于被编码的前缀在区间 $[0,l-1]$

内具有很大的随机性,前缀在区间[0,$2^{k-2}-1$]时可用($k-2$) bit进行编码,但此时需要一位标志位标志着前缀用($k-2$) bit 进行编码,而映射到区间[l,2^k-1]的前缀编码比特不变,但具有标志作用,因此当前缀可用更少位编码时,可查看其前一个前缀是否可映射,若可以,则此前缀用指定位数进行编码,否则正常编码。

若要将[0,$l-1$]的数据映射到[l,2^k-1],因$l\in[2^{k-1}+1,2^k]$,故[l,2^k-1]区间长度必然小于[0,$l-1$]区间的长度。因此,并不能进行一一映射。本文考虑一种简单的映射关系:即将区间[0,$l-1$]的前(2^k-l)个元素映射到区间[l,2^k-1],其他元素不进行映射。这样映射函数是一个简单的线性函数:

$$p_c=f(p_r,l)=p_r+l \tag{1}$$

其中,p_r和p_c分别为映射前及映射后前缀的值;l为字典长度。这样则只对在区间[0,2^k-l-1]的前缀进行映射。解码时,一旦发现前缀比当前字典长,将其减去字典长度即可。

考虑如下前缀进行映射编码:<…,1 024(l=1 025),260(l=1 026),511(l=1 027),1 007(l=1 028),128(l=1 029),528(l=1 030),268(l=1 031),…>。若采用原始的 LZW 对前缀输出编码,则每个前缀均用 k=11 bit 进行编码,共需 7×11=77 bit 编码;若采用映射前缀编码,则:第 1 个前缀 1 024 不能映射,因映射后的值为 2 049>2 047(11 bit),第 2 个前缀映射为 260+1 026=1 286(11 bit)同时标志第 3 个前缀 511 采用 11−2=9 bit 进行编码,第 4 个前缀映射为 1 007+1 028=2 035(11 bit),同时标志第 5 个前缀 128 采用 11−2=9 bit 进行编码,第6 个前缀映射为 528+1 030=1 558(11 bit),同时标志第 7 个前缀 268 采用 11−2=9 bit 进行编码。此时,映射编码所用的比特位为:11+11+9+11+9+11+9=71 bit,节约空间为(77−71)/77=7.8% 。

2.2　改进算法的实现

当“前缀,串”在字典中不存在时,LZW 算法将其加入字典,并输出前缀编码。此时后缀变前缀,编码下一个字符。为利用区间[l,2^k-1]上的数据,每次都对 2 个待编码的前缀进行编码。设待编码的 2 个前缀为 p_{pre}和 p_{now},且当前字典长度为 l 时,用$k=\lceil \mathrm{lb}l \rceil$ bit 对其进行编码。若前缀 $p_{now}\in[0,2^{k-2}-1]$时,即可以用($k-2$) bit 进行编码,则将 p_{pre}映射到区间[l,2^k-1],再用 k bit 编码。采用如式(1)所示的线性映射函数。由于映射关系是确定的,因此解码时一旦发现前缀比当前字典长,则将此前缀映射回区间[0,$l-1$]即可。此时表明下一个码字长度为($k-2$)bit。此时对其中一个数据减少了 2 bit 编码。本文改进算法编码过程如下:

步骤 1　设置 p_{pre}为空,用于存储前一次编码。

步骤 2　按照原 LZW 算法编码字符得到待编码前缀 p_{now}。

步骤 3　当 p_{pre}为空时,令 $p_{pre}=p_{now}$,转步骤 5;否则转步骤 4。

步骤 4　当 $p_{now}<2^{k-2}$ 且 p_{pre}可映射编码时,将 p_{pre}映射编码为 k bit,同时将 p_{now}编码为($k-2$) bit,设置 p_{pre}为空;否则将 p_{pre}编码为 k bit 输出,令 $p_{pre}=p_{now}$,转步骤 5。

步骤 5　若未结束编码,转步骤 2,否则,结束编码。

在解码时,因字典长度是已知的,当码元大于字典长度时,由式(1)可知 $p_r=p_c-l$,故由码元 p_c减去字典长度 l 即可得出此前缀在字典中的位置。这时标志着下一个码元将变短两位。

需指出的是,在对前缀进行映射时,采用了如式(1)所示的简单函数,亦可采用如式(2)所示的线性映射。采用线性映射对随机性较大的前缀具有略好的效果。

$$p_c=(p_r+(A-1))/A+l \tag{2}$$

当满足 $p_r=(n-1)A+l,n=1,2,\cdots$时,按式(2)映射,否则不进行映射。其中,$A=\lceil L/(2^k-l)\rceil-1$,$k=\mathrm{lb}l$($l$ 为字典长度)。另外,当前缀可映射时,则标志下一个前缀可用更少的位编码。可以证明,当前缀分布均匀时,标志下一个前缀用($k-2$) bit 编码可获得最优压缩比。如何选择更少的比特位来编码此前缀可获得最佳的性能提高不易确定,但可以肯定的是,只要选择更少的比特位编码此前缀,必可提高压缩比。一般地,可用式(3)来指定此前缀编码。对具有 256 个字符的数据,当字典选择 12 位时,压缩比较大时可选择 $a=1,b=2,A=10$;压缩比较小时可按式(4)来选择,如图片数据。对十六进制数据可选取 $a=1,b=3,A\geqslant b$,对二进制数据可选取 $a=b=1,A\geqslant b$,亦可对前缀差分再进行映射编码。

$$k_{next}=\begin{cases}k-a,k\leqslant A\\k-b,k>A\end{cases} \tag{3}$$

$$k_{next}=8 \tag{4}$$

其中,k 和 k_{next}分别表示当前前缀编码所需比特数及将当前前缀映射后标志下一前缀编码所需比特数;a,b,A 为参数。

本文改进算法采用将前缀映射到另一区间的方法来显示标志位,因此,该方法对任何数据都可在不同程度上提高压缩比,只需判断前缀是否可映射及下一前缀是否可用更少的比特位编码,并且无需额外储存,也不会增加计算量。该方法可与其他

基于字典的改进的方法相结合,进一步提高压缩比。

3 实验仿真与应用

3.1 文本数据压缩仿真实验

为验证本文改进算法的有效性,使用 C 编程,采用 hash 查找法查找字典。字典长度设置为 4 096(12 bit)项。由于英文文本数据具有 256 种字符,因此初始化字典后字典长度为 258(其中,256 切换字典,257 结束编码)。仿真压缩文件 1 ~6 分别为哈利波特、呼啸山庄、双城记、傲慢与偏见、巴黎圣母院、麦田里的守望者的英文文本。仿真曲线如图 1 和图 2所示,具体数据比较如表 1 所示。

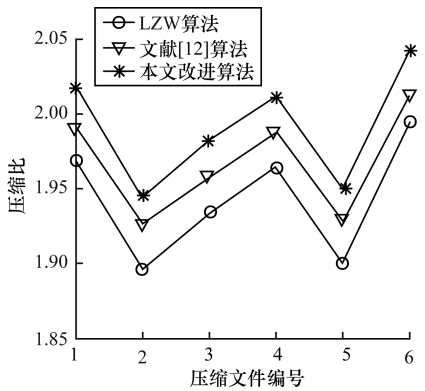

图 1 文本压缩比对比 1

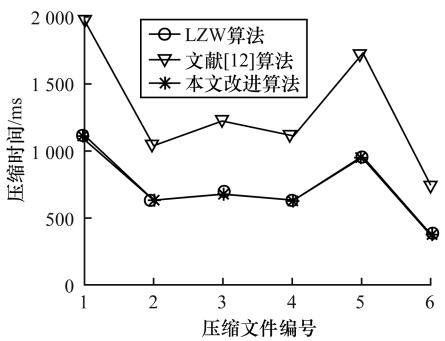

图 2 文本压缩时间对比

表 1 改进算法与原算法压缩性能对比 ms

压缩文件编号	LZW 算法		文献[12]算法		本文改进算法	
	压缩比	压缩时间/ms	压缩比	压缩时间/ms	压缩比	压缩时间/ms
1	1.969	1 094	1.991	1 986	2.018	1 110
2	1.896	629	1.926	1 032	1.945	630
3	1.933	688	1.958	1 234	1.981	688
4	1.964	628	1.988	1 109	2.013	630
5	1.900	954	1.927	1 719	1.950	955
6	1.994	386	2.012	729	2.042	386

图 1、图 2 和表 1 反映了 3 种算法性能的差别,由表 1 可知,文献[12]算法较原算法压缩比提高 1.2% 但压缩时间却大大高于原算法,而本文改进算法较原算法具有 2.5% 的提高,且本文改进算法几乎不增加压缩时间。究其原因,文献[12]算法采用了双索引机制,使得每编码一次前缀需要搜索第二索引,且该算法使用了算术编码,增加了压缩开销。另外文献[12]算法需占用一个标志位用于节点是否拥有第二索引号,占用了额外内存。若用此内存用于增加字典长度结合本文改进算法,压缩比提高较为显著。

本文算法只是改进了前缀编码机制,对字典未作任何处理,由于改进方法不增加内存,且不增加计算,因此,可与基于字典的改进方法相结合,以进一步提高压缩比。本文算法与文献[11]结合仿真结果如图 3 所示。由仿真结果知,在未增加压缩开销的情况下,将文献[11]算法结合本文算法,压缩比较文献[11]算法可提高 1.5% 。

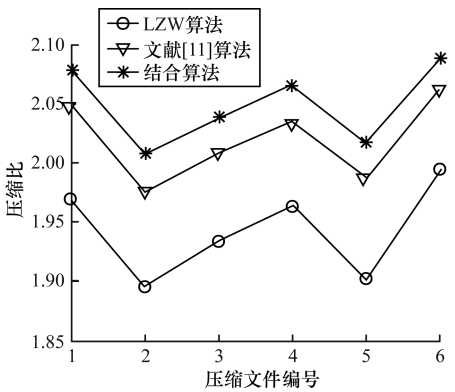

图 3 文本压缩比对比 2

3.2 残差数据压缩应用

对电厂 600 mW 辅机设备的#5 机组的 $IP=5$ 的风机使用 S930 设备采集其震动数据,按 1 s 间隔,以 25.6 kHz/16 bit 采集数据。对采集的辅机震动数据进行无损压缩,先经 FFT 变换,再对频率幅值进行阈值滤波,约保留 5% 左右主要频率。最后阈值滤波后的数据与原数据相比会有较小的残差数,如图 4 所示,该图表示一个文件中的数据值。使用 LZW 压缩算法对残差数据进行压缩,设置字典长度为 65 536(16 bit)项。考虑到所要压缩的数据集中于 20 以下,因此,初始化字典时将后缀初始化为对 10 取余后的值(后缀% 10),以提高字典利用率。压缩数据所需比特对比结果如图 5 所示,其中纵坐标表示压缩此数据平均每个字符需要的比特位。由图 5 可知,文献[11]并不能保证压缩比完全优于原算法,

而本文改进算法提高了约 7.5% ,得到了低于数据熵的良好结果。

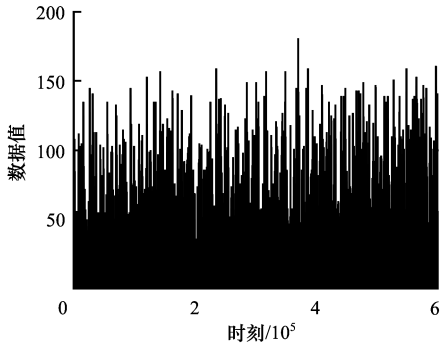

图 4　残差数据

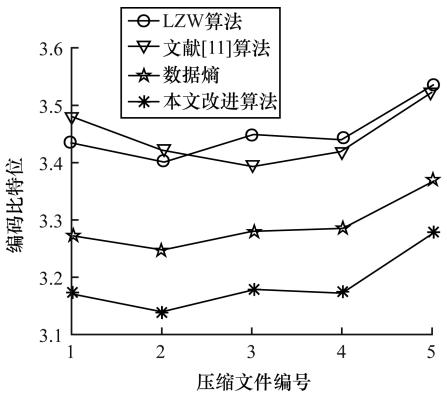

图 5　压缩数据所需比特对比

仿真与应用结果表明,本文算法较原 LZW 算法性能有所提高,且该算法可与基于字典的改进方法相结合,几乎不增加计算量及内存,适用性好。若结合基于字典的改进方法使用,性能提升更明显。

4　结束语

本文通过分析 LZW 算法的原理,利用其编码前缀时存在的冗余数据,设计将前缀映射为编码的改进算法。与原 LZW 算法相比,该算法几乎不增加计算量和存储空间,而且算法通用性好。由于利用编码输出的冗余来提高算法性能,因此基于字典的改进 LZW 算法也可进一步应用本文算法达到更好的效果。同时,对于二进制数据、图像数据、文本数据和绝对值较小的整数(即残差数据),该算法均可不同程度地提高压缩比。英文文本数据压缩实验与辅机设备残差数据的应用结果验证了本文算法的有效性。

参考文献

[1] 解瑞云,海本斋. 基于自适应霍夫曼和 Golomb-Rice 混合编码的 WSN 无损压缩算法[J]. 计算机工程,2016,42(7):86-93.

[2] Huffman D A. A Method for the Construction of Minimum-redundancy Codes[J]. Proceedings of the IRE, 1952, 40(9):1098-1001.

[3] Rissanen J. A Universal Data Compression System[J]. IEEE Transactions on Information Theory, 1983, 29(5): 656-664.

[4] Shannon C E. A Mathematical Theory of Communication[J]. The Bell System Technical Journal, 1948, 27(3):379-423.

[5] Ziv J, Lempel A. A Universal Algorithm for Sequential Data Compression[J]. IEEE Transactions on Information Theory, 1977, 23(3):337-443.

[6] Ziv J, Lempel A. Compression of Individual Sequences via Variable-rate Coding[J]. IEEE Transactions on Information Theory, 1978, 24(5):530-536.

[7] Welcht A. A Technique for High-performance Data Compression[J]. Computer, 1984, 17(6):8-19.

[8] Pratap U, Sunkaria R K. ECG Compression Using Compressed Sensing with Lempel-Ziv-Welch Technique[C]// Proceedings of the 1st International Conference on Next Generation Computing Technologies. Washington D. C., USA:IEEE Press, 2015:863-867.

[9] 王　平. LZW 无损压缩算法的实现与研究[J]. 计算机工程,2002,28(7):98-99.

[10] Chai Z, Chen W. An Adaptive LZW Compression Algorithm Using Changeable Maximum-code-length[C]//Proceedings of the 4th International Conference on Computer and Information Technology. Washington D. C., USA: IEEE Press, 2004:1175-1180.

[11] Nandi U, Mandal J K. A Compression Technique Based on Optimality of LZW Code(OLZW)[C]//Proceedings of the 3rd International Conference on Computer and Communication Technology. Washington D. C., USA: IEEE Press, 2012:166-170.

[12] Lakhani G. Reducing Coding Redundancy in LZW[J]. Information Sciences, 2006, 176(10):1417-1434.

[13] 吴宇新,余松煜. 对 LZW 算法的改进及其在图象无损压缩中的应用[J]. 上海交通大学学报,1998,32(9):110-113.

[14] Li Bin, Ni Guiqiang, Luo Jianxin, et al. BWT-based Data Preprocessing for LZW[C]//Proceedings of the 1st International Conference on Multimedia and Signal Processing. Washington D. C., USA: IEEE Press, 2011: 37-40.

[15] 张凤林,刘思峰. LZW ~*:一个改进的 LZW 数据压缩算法[J]. 小型微型计算机系统,2006,27(10):1897-1899.

[16] Sayood K. 数据压缩导论[M]. 4 版. 贾洪峰,译. 北京:人民邮电出版社,2014.

编辑　金胡考