LZW 无损压缩算法的研究与改进

许 霞,马光思,鱼 涛
(西安建筑科技大学 信息与控制工程学院,陕西 西安 710055)

摘 要:研究了数据压缩技术领域中一种较有效的无损压缩算法——LZW。LZW 的原理在于用字典中词条的编码代替被压缩数据中的字符串。因此字典中的词条越长越多,压缩比就越高。加大字典的容量可以提高压缩比。但字典的容量要受到计算机内存的限制,而且其字典也存在被填满的可能。这样当字典不能再加入新词条后,过老的字典就不能保证高的压缩比。为了解决这个问题,设计并实现了一种改进算法;分析了改进算法对复杂度的影响,并选用一些典型文件对改进后的算法进行了应用测试。测试结果表明,改进后的算法具有较好的压缩比和较理想的压缩效率。

关键词:LZW 算法;压缩比;字典;匹配率
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2009)04-0125-03

Research and Improvement on LZW Lossless Compression Algorithm

XU Xia, MA Guang-si, YU Tao
(School of Info. and Control Eng., Xi'an Univ. of Architecture and Tech., Xi'an 710055, China)

Abstract: Researches one kind of effective lossless compression algorithm in the data compression area of technology — LZW. The LZW principle lies in the entry code replaces by the string of character in the packed data. Therefore in the dictionary entry is longer are more, the compression ratio is higher. Enlarges the dictionary capacity to be possible to enhance the compression ratio. But the dictionary capacity is limited by the computer memory, moreover it is possible to fill up the dictionary. After like this when the dictionary can't join the new entry again, the excessively old dictionary can't guarantee the high compression ratio. In view of the LZW algorithm's question, we designed and implemented one kind of improved algorithm; After analyzed the effect of improved algorithm on the complexity, selected some typical documents to test the algorithm on the application. The test result indicated that the improved algorithm has the good compression ratio and the ideal efficiency of compression.

Key words: LZW algorithm; compression ration; dictionary; matching rate

0 引 言

LZW 算法的实质在于用字典中词条的编码代替被压缩数据中的字符串。字典中的词条越长越多,压缩比就越高,所以加大字典的容量可以提高压缩比。但字典容量受计算机内存的限制,且其字典也存在被填满的可能。这样当字典不能再加入新词条后,老字典就不能保证高压缩比。为了解决这个问题,可在压缩时监视压缩比,当压缩比下降时,清除匹配率较小的词条而保留匹配率较大的词条,这样就能在重建字典的同时提高压缩比[1]。

收稿日期:2008-07-08
基金项目:陕西省自然科学基金(2005F38);陕西省教育科研基金(07JK306)
作者简介:许 霞(1983-),女,陕西榆林人,硕士研究生,研究方向为算法设计、框架技术;马光思,教授,研究方向为计算理论、信息安全、Web 技术。

1 LZW 算法简介

LZW 继承了 LZ77 和 LZ78 压缩效果好、速度快的优点,且算法描述易于接受。该算法能在不了解数据统计特性前提下,使压缩比接近已知统计特性时所能达到的压缩比,且易于实现,是目前最常用的算法。

1.1 LZW 算法基本原理

LZW 压缩算法的基本思想是建立一个编码表(转换表)也称串表[2],将输入字符串映射成定长的码字输出,通常码长设为 12bit。把数字图像当作一个一维的比特串,算法在产生输出串的同时动态地更新编码表,这样码表与串表对应产生压缩图像的特殊性质。串表是动态产生的,编码前先初始化,使其包含所有的单字符串。压缩过程中,串表中不断产生新字符串(串表中没有的字符串),存储新字符串时也保存与新串的前缀相应的子码[3]。LZW 算法编码是围绕被称为词典的转换表来完成的,这张转换表用来存放称为前缀的字符序列(缀-字符串表),并为每个表项分配一个

码字,或者叫做序号。

1.2 LZW 算法描述

STEP1 开始时的词典包含所有可能的根(Root),当前前缀 P 为空;

STEP2 当码字流有码字要译时,反复执行 STEP3 和 STEP4;

STEP3 当前字符 C 置字符流中下一个字符;

STEP4 如果缀 - 字符串 $P + C$ 在词典中,则

(1)P 置 $P + C$(用 C 扩展 P);

(2) 把代表当前前缀 P 的码字输出到码字流

否则

(3) 把代表当前前缀 P 的码字输出到码字流;

(4) 把缀 - 字符串 $P + C$ 添加到词典;

(5)P 置 C(现在 P 仅含单字符 C);

STEP5 结束。

实现 LZW 算法时,按照上述流程进行。压缩开始,从原文件读入第一个字符作为基码;输入字符,并将此字符作为查找匹配串的前缀,再读入一个字符作为扩展,在表中查找由前缀和扩展符组成的字符串是否存在。若存在,则整串作为前缀,再读入一个字符作为扩展符,重复上述过程。若不存在,则形成新串的变换码,并输出此变换码,作为该新串字符的压缩码。将字符串的扩展符作为下一字符串的前缀,重复以上步骤,继续读入字符,重复上述查找、填表、输出操作,直至文件末尾[4]。

1.3 对 LZW 算法进一步分析

就其外部特征来说,LZW 压缩技术对于可预测性不大的数据具有较好的处理效果,常用于 GIF 格式的图像压缩,其平均压缩比在 2:1 以上,最高压缩比可达到 3:1;对于数据流中连续出现的字节和字串,LZW 具有很高的压缩比;LZW 除了用于图像数据处理以外,还被用于文本程序等数据压缩领域;LZW 是无损压缩,文件在压缩前和解压缩后完全一样,不会有一位的差错,完整地保持了原文件的特点;LZW 压缩技术有很多变体。例如常见的 ARC、PKARC、PKZIP 等高压缩程序;对于任意宽度和像素位长度的图像,都具有稳定的压缩过程。压缩和解压缩速度较快;对机器硬件要求不高,在 Intel80386 计算机上即可进行压缩和解压缩。

从其内部原理来说,LZW 压缩算法是基于字典的 LZ 系列算法的改进型,其精义在于它不断地读取字符串,如果在字典中找到匹配,那么继续下去,一直读到字典里没有为止,然后把它加入字典。初始状态的字典并非为空,字典初始状态具有 256 个项,所以第一步的时候要初始化,使词典包含所有可能的根,这样这个算法就能循环起来。字典大小是有限制的,字典满了以后,复位字典再从头开始。LZW 的精华就在这里,为什么可以复位字典?因为 LZW 压缩算法不需要放一个大块头字典到它的压缩数据里,在解压缩的时候,可以根据输入重新生成字典。

传统 LZW 算法压缩的原理在于用字典中词条的编码代替被压缩数据中的字符串。因此字典中的词条越长越多,压缩比就越高。加大字典的容量可以提高压缩比。但字典的容量要受到计算机内存的限制,而且其字典也存在被填满的可能。这样当字典不能再加入新词条后,过老的字典就不能保证高的压缩比[5]。为了解决这个问题,在压缩时必须监视压缩比,当压缩比下降时,清除匹配概率较小的词条而保留匹配概率较大词条。这样在重建字典的同时又可提高压缩比。

2 LZW 算法的改进

2.1 改进思路

针对原算法其字典存在被填满的可能,而且字典的容量要受到计算机内存限制的不足,采用改进的阈值判断操作和字典填满之后淘汰匹配率小的词条的方法。

当字典填满时,输入一定长比特数据流,用现有字典对其进行压缩,然后判断这个被压缩的比特数据流的压缩比(压缩比 $=\frac{\text{输入流的大小}}{\text{输出流的大小}}$)是否大于所指定的阈值,如所得到的压缩比大于所指定的阈值,则继续前面的操作,进行压缩、判断;如所得到的压缩比小于所指定的阈值,则进行删除字典中多余词条的操作。

删除字典中多余的词条,首先在字典中设置清除标志,当字典填满后,在进行阈值判断时,对每一个码字(Code Word)的使用情况计数。当进行阈值判断所得到的压缩比小于指定的阈值时,发出清除标志,这时根据字典中每一个码字使用的计数值大小进行排序,对字典进行相应的重构,然后删除排序靠后的若干项。这样做不仅解决了字典被填满和计算机内存不足的问题,而且在一定程度上避免了多余的计算,提高了算法的压缩比。

2.2 LZW 改进算法描述

STEP1 初始化,使开始词典包含所有可能的根(Root),当前前缀 P 置空;

STEP2 当码字流有码字要译时,反复执行 STEP3,STEP4,STEP5 和 STEP6;

STEP3 读入字符数据流中的下一个字符 C;

STEP4 如果字符串 $P + C$ 在当前词典中,

(1)P 置 $P + C$(用字符 C 扩展 P);

(2) 把代表当前前缀 P 的码字输出到码字流

否则

(3) 输出表示 P 的码字到编码数据流;

(4) 把字符串 $P+C$ 添加到词典;

(5)P 置 C(此时 P 仅含一个字符 C);

STEP5 若字典未满,则执行 STEP3;

STEP6 如果压缩比小于指定阈值,则清除匹配率小的词条,否则,返回 STEP1;

STEP7 结束。

3 算法分析

LZW 算法大量依赖快速的字典查找,若直接检索字典,代码执行速度慢,其时间复杂度为 $O(n^2)$[6]。引入哈希表能有效提高字符串表的检索效率及整体执行效率。哈希表的容量与字典的容量均为表中不定长元素的代码数组,用于存放哈希值相同的字符串代码。实际上,较好的哈希函数产生的重复值极少,这样检索字符串所需比较的次数将大幅减少,时间复杂度几乎接近于 $O(n)$,从而有效提高了代码的执行效率。但是,由于字典的容量有限,当字典被填满时,压缩比就会降低。为了提高压缩比,就要重新对字典进行初始化,即重新构造哈希表。这样将会加大内存开销,也难以保证新构造哈希表中词条的匹配率。

改进的 LZW 算法删除了匹配率小的词条,并根据字典中每个码字的使用计数值大小进行排序。一方面释放了大量无效内存;另一方面,对有序哈希表可以采取二分查找,其时间复杂度降为 $O(\log n)$。由此可见,改进算法不会影响经典算法的复杂度。

4 应用测试及性能分析

为了测试改进后算法的有效性,笔者选择了几个不同类型的文件,分别用经典 LZW 和改进 LZW 对其压缩,记录压缩前文件的大小,压缩后文件的大小,压缩比及压缩时间。应用测试结果分别见表 1 和表 2。

表 1　经典 LZW 算法测试结果

文件类型	压缩前大小(kb)	压缩后大小(kb)	压缩比(%)	所用时间
.doc	740.0	355.2	21:10	1.5 秒
.ppt	1031.0	690.7	15:10	1.8 秒
.bmp	786.1	188.7	42:10	2.7 秒
.jpg	1888.6	1885.6	1:1	1.7 秒
.wmv	6451.1	6386.5	1:1	3.7 秒
.rar	8312.2	8316.1	1:1	3.8 秒

比较表 1 与表 2 的测试结果可见:

表 2　改进 LZW 算法测试结果

文件类型	压缩前大小(kb)	压缩后大小(kb)	压缩比(%)	所用时间
.doc	740.0	309.2	24:10	0.8 秒
.ppt	1031.0	580.9	18:10	1.1 秒
.bmp	786.1	117.4	45:10	1.9 秒
.jpg	1888.6	1885.6	1:1	1.0 秒
.wmv	6451.1	6269.8	1:1	2.9 秒
.rar	8312.2	8316.1	1:1	3.1 秒

(1)改进后的 LZW 算法依然保留着经典 LZW 算法的本质特点,即压缩.doc,.ppt 等文件时,压缩比较高,压缩效果好;压缩图像类文件.bmp(16 位) 时,压缩比较高,压缩效果好,而压缩.jpg 和.wmv 时,压缩比较低,因为.JPG、.WMV 等文件支持高级别压缩,一般都自带压缩,因此压缩比偏低,与预期结果一致;对于已经压缩过的文件,如 rar 文件再次进行压缩时,其压缩比极低,因为经过压缩的文件又对其进行第二次压缩时其冗余很少。

(2)对于几类文件的压缩,改进的 LZW 算法不仅压缩比较高,且压缩用时也较短,由此验证了改进算法的正确性和有效性。

5 结束语

LZW 压缩算法是一种有效的数据压缩算法。文中在认真研究 LZW 算法工作原理的基础上,设计了一种改进算法。改进算法实现了压缩过程中自动释放内存,测试结果令人满意,性能良好,使压缩过程更合理,压缩效果更理想。

参考文献:

[1] 方世强.文本压缩技术综述[J].工业工程,2002,5(2):16-17.

[2] 杨国梁,张光年.无损 LZW 压缩算法及实现[J].首都师范大学学报,2004,25(专集):12-14.

[3] Nelson M.数据压缩技术原理与范例[M].贾启东译.北京:希望出版社,1995.

[4] Hayashj S,Kubo J I,Yamazato T,et al.A new source coding method based on LZW adopting the least recently used deletion heuristic[C]//IEEE Pacific Rim Conference On Communication.Victoria:Computers and Signal Processing,1993:190-193.

[5] Cho Gyoun-yon,Cho Dong-ho.A study on the efficient multiplexer[C]//IEEE International Conference on Communucations.New York:IEEE,1995:18-22.

[6] 王俊蛟.数据无损压缩算法的 C++实现[J].电脑编程技巧与维护,2001(1):27-27.