几种数据压缩算法的比较

曾 玲 饶志宏
(西南交通大学,成都 610031) (信息产业部电子第 30 研究所,成都 610041)

【摘 要】 对几种数据压缩算法进行了总结,并针对实验数据表对其压缩率进行了比较。
【关键词】 数据压缩 压缩率 信源编码

Comparision of Several Kinds of Algorithms for Data Compression

Zeng ling *Rao zhihong*
(Southwest Jiaotong University, Chengdu 610031) (Electronic No. 30 Institute of MII, Chengdu 610041)

【Abstract】 The paper summarizes several kinds of algorithm for data compression and compare their compression ratio according to data – table of experiment.
【Keywords】 data compression, compression ratio, sowce coding

1 引言

数据压缩最初是作为信息论研究中的一个重要课题,在信息论中被称为信源编码。但近年来,数据压缩已不仅限于编码方法的研究,已逐步形成较为独立的体系。它主要研究数据的表示、传输和转换方法,目的是减少数据所占据的存储空间和传输时所需要的时间。

数据压缩主要应用于两个方面:

(1) 传输。通过压缩发送端的原始数据,并在接受端将压缩数据解码恢复,这样可有效地减少传输时间,增加信道带宽。

(2) 存储。在存储时压缩原始数据,而在使用时解压,这将大大增加存储介质的存储量。数据压缩按压缩的失真度分为无损压缩与有损压缩。无损压缩技术主要有哈夫曼(Huffman)编码、算术(Arithmetiic)编码、游程编码(RLE)、LZ 编码。

2 算法描述

2.1 哈夫曼压缩编码

哈夫曼编码属于统计式压缩法,这种方法的主导思想是根据源数据符号发生的概率进行编码。在源数据中出现概率越高的符号,相应码字越短;出现概率越小的符号,其码长越长,从而达到用尽可能少的码符号来表示源数据,达到压缩的效果。该方法需统计、编码两遍扫描源数据流,在已知源数据流中各字符发生的概率情况下使用,即在压缩数据时,遵循固定的源数据符号代码表。在不允许两次扫描源文件的情况下,往往根据先验统计概率或假设构造这张代码表。压缩处理时,若源数据流中的符号与代码表中的符号相匹配,则用与之相对应的较短的代码替代该字符,否则不予处理。显然,这种固定的代码表与源文件的匹配程度决定了该文件的压缩比。后来,人们在此基础上又提出了自适应哈夫曼编码。自适应压缩方法则是根据不同源文件的具体内容,采用完全不同的方法动态地构造代码表,并在构造代码表的同时进行编码压缩[1-2]。

2.2 算术编码

算术编码是基于统计的,无损数据压缩效率最高的方法。其基本思想是将整段要压缩的数据映射到一段实数半封闭范围[0,1]内的某一区段。该区段的范围或宽度等于该段信息概率即是所有使用在该信息内的符号出现概率全部相乘后的概率值。当要

收稿日期:2002 – 06 – 04。
曾 玲:硕士生。目前从事数据处理方面的工作。
饶志宏:工程师。主要研究方向为无线通信。

被编码的信息越来越长时，用来代表该信息的区段就会越来越窄，用来表达这个区段的位就会增加。该方法的缺点也是需两次扫描源数据流[1-2]。(arithmetic 程序采用该算法)

2.3 游程编码

游程编码[1]是针对一些文本数据的特点所设计的，主要是去除文本中的冗余字符或字节中的冗余位，从而达到减少数据文件所占的存储空间的目的。压缩处理的流程类似于空白压缩，区别仅在于要在压缩指示字符之后加上一个字符，用于表明压缩对象。随后是该字符的重复次数。由于该算法是针对文件的某些特点所设计的，所以应用起来具有一定的局限性。为了数据压缩的通用性，一般很少单独采用该方法，主要与其它编码技术配合使用。

2.4 Lz 算法

1977 年，Jacob Ziv 和 Abraham Lempel 提出了被称为 LZ77、LZ78 的基于字典的压缩技术。为了改进 LZ77 的性能，James Storer 和 Thomas Szymanski 于 1982 年提出了 lzss 算法。1984 年，Terry Welch 给出了 LZ78 算法的一种实现技术，称之为 lzw 算法。随后许多研究者又在 LZW 算法的基础上提出了自己的改进方法[1-2]。所有的改进方法可大致分为以下三类。

2.4.1 对字典更新的改进

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，字典中词条越多越长，压缩率将越高。所以加大字典的容量可以提高压缩率，但字典的容量要受计算机内存的限制。如果在通讯中采用 DSP 处理时，字典的容量要受到 DSP 容量的限制。

无论字典多大，在压缩较长数据流时都会被填满，填满后一种方法是不再往字典中加入新词条，以后在字典中找不到匹配的字符串以单个字符的编码输出。这一方法产生的问题是过老的字典不能保证高的压缩率。(lzw 程序采用该算法)。另一种方法是在压缩过程中监视压缩率，当压缩率下降时清除老字典，重新建立新字典。清除老字典又可分为两种，一种是抛弃整个字典，一种是抛弃字典中匹配率较小的节点。由于在建立字典的初期，匹配率较小，所以抛弃整个字典的方法显然不可取，一般采取抛弃部分字典的方法(lzwr 程序采用该算法)。

2.4.2 对字典建立的改进

分析 lzw 算法可知，算法是在边建立字典边压缩数据，在字典建立之初，由于字典中词条较少，可匹配的字符串较少，压缩率不高。因此，如果加快往字典中加入词条就可能提高压缩率。但加快往字典中填入词条将引起字典更快的填满从而导致字典更快的老化，从而使字典重建的次数增加，这将引起压缩率的下降，因此在将词条加入字典时要有所选择，一般将匹配率较大的词条填入字典。在对信源的统计特性一无所知的情况下，只能假设较短的词条其匹配率较大，所以在建立字典时，当 v_node 小于 1/4TABLE_SIZE 时，只允许字长小于某值的词条进入字典(LZWR 程序采用该算法)。因为这具有假设性，所以这种方法不一定适合于所有的信源。另外，在字典中前缀特性的基础上增加辅助前缀，以此加快字典的建立[5]或在前缀特性的基础上增加辅助后缀[9](lzw 程序采用该算法)，但为了避免在还原方收到在字典中还没有定义的编码，算法中不输出上一次输出前新加入字典的词条，因为改进后的算法与 lzw 算法相比在每次输出前可能往字典中加入多条新词条，从而使例外情况复杂化。这将损失一定的压缩率，但消除了还原方处理特殊情况的麻烦。

另外还有一种加快字典建立的方法为：数据压缩的综合字典模型[6]。

2.4.3 对 lzss 算法中匹配长度编码的改进

在 lzss 算法中，无论匹配长度为多少，都分配相同的代码长度。比如：如果用 4bit 来表示匹配长度，则最大匹配长度可达 16 + 2 = 18，但当匹配长度为 3 时，仍采用 4 位二进制数来表示，这显然存在一定的冗余。如果在匹配长度编码中采用变长编码技术，这将减少冗余度，而提高压缩率[7]。

另外一种方法为采用多标志位[8]。在 lzss 算法中采用一位固定的前缀。0 表示紧接其后的是单个字符；1 表示紧接其后的是代码对(匹配位置，匹配长度)。在该方法中，利用输出的多标志位来区别输出的是一个单字符还是一个代码对。

3 实验比较

根据以上算法原理，编写了以下几个程序(其中 lzhuf 程序采用 hufman 与 lzss 混合编码法)，并针对各类数据进行了软件模拟结果，如表 1 所示。

表 1 Buffersize = 2048 时，各种算法压缩率的比较

文件名	原长度	LZSS	LZSSD	LZSSB	LZHUF	ARITHMETIC	LZW	LZWR	ILZW	WINRAR
Cpp. exe	128kB	48. 24%	48. 23%	49. 05%	47. 10%	24. 33%	13. 11%	38. 67%	19. 30%	60. 16
Bcw. exe	1. 3MB	51. 12%	51. 00%	51. 85%	46. 70%	22. 75%	1. 62%	36. 07%	15. 58%	60. 8
Brcc. exe	299kB	55. 20%	54. 64%	55. 59%	54. 52%	28. 88%	11. 65%	44. 10%	8. 06%	68. 83
Rc. exe	55. 7kB	29. 00%	30. 98%	31. 05%	28. 14%	15. 24%	5. 34%	18. 34%	7. 23%	41. 47
Fil1. txt(中文)	7. 33kB	35. 32%	37. 58%	37. 75%	33. 41%	16. 42%	25. 08%	25. 79%	23. 13%	45. 70
Fil5. txt(中文)	17. 1kB	23. 88%	29. 18%	26. 99%	27. 80%	22. 07%	11. 83%	15. 77%	18. 27%	38. 60
Fil4. txt(英文)	15. 0kB	49. 73%	49. 77%	47. 37%	46. 99%	42. 69%	48. 07%	43. 48%	47. 99%	56. 87
Ex3. cpp	466kB	21. 67%	21. 03%	24. 89%	11. 37%	15. 24%	12. 23%	12. 02%	14. 38%	19. 53
Ilzw. cpp	6. 85kB	65. 10%	63. 42%	64. 96%	61. 52%	29. 65%	48. 80%	48. 17%	46. 70%	68. 83
Lzhufcom. cpp	12. 7kB	67. 36%	65. 49%	67. 13%	65. 93%	31. 09%	48. 49%	51. 26%	44. 6%	74. 65
平均压缩率		44. 662%	45. 132%	45. 663%	42. 348%	24. 786%	22. 62%	33. 367%	24. 524%	57. 509%

以上各程序皆将字典的大小设置为2048，当字典填满后或停止建立字典使用老字典，或监控压缩率，当压缩率小于某数时立即重建字典。如果在通讯线路中使用，需设置缓冲区，每次从缓冲区读入数据时，都需重建字典。为了模拟这一过程，将上述程序进行改进，每次从文件中读入buffersize 个字符进行处理，当处理完后再读入 buffersize 个字符并重建字典，直到读完文件为止，结果如表 2、表 3 所示。

当采用缓冲区时，增加辅助后缀的 ilzw 算法压缩率反而不如 lzw 算法，这是因为如果上下文相关性较差，匹配长度较小，字典建立快速从而很快填满，而所建立的长字符串在后文的匹配过程中没有采用，所以压缩率下降。

表 2 Buffersize = 1024 时，压缩率比较

文件名	原长度	LZSS	LZSSD	LZSSB	LZW	ILZW
Cpp. exe	128kB	42. 61%	41. 22%	43. 35%	36. 31%	19. 92%
Bcw. exe	1. 3MB	44. 20%	42. 84%	44. 94%	37. 55%	25. 76%
Brcc. exe	299kB	47. 19%	45. 60%	47. 71%	39. 97%	18. 03%
Rc. exe	55. 7kB	23. 37%	22. 49%	24. 98%	18. 47%	15. 32%
Fil1. txt(中文)	7. 33kB	25. 20%	24. 85%	27. 01%	19. 85%	19. 37%
Fil5. txt(中文)	17. 1kB	14. 22%	15. 04%	16. 60%	12. 58%	11. 57%
Fil4. txt(英文)	15. 0kB	29. 81%	35. 43%	32. 94%	31. 68%	34. 28%
Ex3. cpp	466kB	23. 82%	23. 39%	26. 18%	20. 17%	22. 10%
Ilzw. cpp	6. 85kB	51. 69%	50. 08%	52. 08%	36. 27%	38. 29%
Lzhufcom. cpp	12. 7kB	55. 57%	53. 33%	55. 62%	40. 01%	42. 39%
平均压缩率		35. 768%	35. 42%	37. 14%	29. 286%	24. 703%

表 3 Buffersize = 512 时，压缩率比较

文件名	原长度	LZSS	LZSSD	LZSSB	LZW	ILZW
Cpp. exe	128kB	39. 96%	32. 85%	40. 68%	36. 12%	17. 98%
Bcw. exe	1. 3MB	41. 41%	39. 38%	42. 14%	37. 15%	23. 19%
Brcc. exe	299kB	43. 29%	41. 08%	43. 88%	38. 91%	16. 65%
Rc. exe	55. 7kB	21. 39%	19. 14%	22. 77%	19. 07%	13. 50%
Fil1. txt(中文)	7. 33kB	20. 91%	19. 09%	22. 52%	19. 83%	16. 27%
Fil5. txt(中文)	17. 1kB	10. 56%	9. 23%	12. 59%	12. 56%	11. 25%
Fil4. txt(英文)	15. 0kB	24. 28%	28. 67%	27. 31%	29. 32%	29. 03%
Ex3. cpp	466kB	26. 18%	25. 54%	28. 33%	27. 68%	21. 67%
Ilzw. cpp	6. 85kB	44. 72%	42. 77%	45. 41%	34. 69%	33. 50%
Lzhufcom. cpp	12. 7kB	48. 48%	45. 61%	48. 74%	36. 62%	34. 28%
平均压缩率		32. 118%	30. 336%	33. 43%	29. 286%	21. 732%

4 结论

根据实验数据，可知 lzss 系列算法受缓冲区大小影响较大，lzw 算法受缓冲区影响较小，但 lzss 系列算法压缩率比 lzw 算法压缩率高。特别是经过改进的 lzssb 算法比 lzss 算法平均高两个百分比。但 lzss 系列算法程序比 lzw 系列算法程序代码长，所占用的存储容量大，执行时所占用的时间也较长。如对压缩率要求较高时，建议采用 lzssb 程序算法；如果对压缩率要求不是很高，而对程序代码所占用的存储空间有严格要求时，建议使用 lzw 或 ilzw 程序算法 如果不使用缓冲区，对数据流进行实时处理，且对时间开销有严格要求时，建议使用 lzwr 程序算法。

参考文献

1 袁玫, 袁文. 数据压缩技术及其应用. 北京: 电子工业出版社 1994
2 骆新, 陈睿. 数据压缩实用技术. 北京: 学苑出版社, 1993
3 Ziv J, Lempel A. A Universal Algorithm for Sequential Data Compression. IEEE Transcations on Information Theory, 1977; (3): 337—343
4 Ziv J, Lempel A. Compression of Individual Sequences via Variable_Rate Coding. IEEE Transcations on Information Theory, 1978; (5): 530—536
5 刘方. 一种数据无损压缩技术的研究. 南京航空航天大学学报 1995; (2): 804—809
6 谭兆信. 数据压缩的综合字典模型. 计算机工程与设计, 1997; (2): 11—15
7 王忠效, 姜丹. 关于 Lempel_ziv 77 压缩算法及其实现的研究. 计算机研究与发展 1996; (5): 329—340
8 于洪斌, 马俊光. 多标志位的 LZ 数据压缩算法. 微型机与应用, 1998; (6): 8—22
9 张桂玲, 王正光. 一种实时处理的无失真数据压缩技术, 遥测遥控, 1994; (8): 9—13

(上接第 6 页)

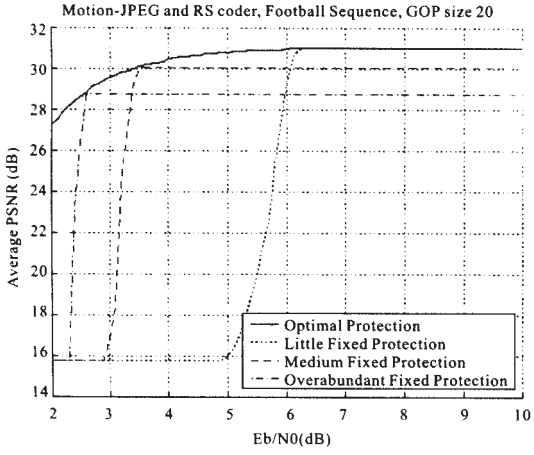

图 4 采用运动 JPEG 视频信源编码器和 RS 信道编码器的 JSCM 系统性能仿真曲线

图 4 中，虚线所示为采用了固定的信源压缩码率和信道保护码率的传输系统性能曲线，分为较低信道保护、中等信道保护和较高信道保护三种情况；而实线为本文提出的信源压缩和信道保护码率联合优化的 JSCM 系统性能曲线。由图中可以看出，JSCM 系统的性能曲线高于所有的固定码率参数的系统性能曲线，并且总是能获得最高的端到端 PSNR (Peak Signal – to – Noise Rate, PSNR) 值。实质上，JSCM 系统性能曲线是所有固定码率系统的性能曲线的凸包，其上的每一点对应于在某一特定信道 SNR 值下的一个最优信源和信道编码器参数配置。

由以上仿真结果可见，这里提出的 JSCM 通用视频传输系统能获得比传统的固定编码传输码率的系统更高的整体性能增益，且该系统可应用于多种信源 – 信道编码器对组合，具有很好的普遍性和通用性。

参考文献

1 Ngan K N, Chi W Y, Keng T T. Video coding for wireless communication systems. New York: Basel, 2001
2 Azami S B Z, Duhamel P, Rioul O. Joint source – channel coding: Panorama of methods. In Proceedings of CNES workshop on Data Compression, 1996: 1232 ~ 1254
3 李天昊, 余松煜, 殷志明. 基于联合信源信道编码的视频流传输. 通信技术, 2002; (1): 9 ~ 126

(上接第 8 页)

实现。对于序列，该算法仅需 14 次移位和 31 次加法操作；而浮点 DCT 需要 13 次浮点乘法和 29 次加法。用 C6201 实现 DCT 时，binDCT 比浮点 DCT 快 47 倍，比改进的整数化 DCT 快 2 – 3 倍。表 1 给出了在 C6201 评估板上实现矩阵变换时，三种算法消耗的时钟数。而且由于 BinDCT 是整数对整数的变换，它能精确地实现无损变换。另外，binDCT 和 DCT 非常接近，专门针对 DCT 的视觉量化矩阵和编码策略无需做复杂的修改也能应用于 binDCT。

表 1 DCT 与 binDCT 编码时钟数比较

	浮点 DCT	整数化 DCT	binDCT – C
时钟数	61942	3390	1317

binDCT 也有不足之处。它只是 DCT 的近似，并不是真正的 DCT，它不具备 DCT 的正交特性，是一种双正交变换，也就是说，它解相关和集中能量的能力不如 DCT，变换之后的数据还存在比较大的相关性，在变换编码后进行量化时，小的分量比较多，对压缩比和图像的 PSNR 都有一定的影响，我们应该权衡考虑编码速度、编码质量和压缩比，选择合适的变换编码算法。

参考文献

1 Wintz P A. Transform picture coding. Proc. IEEE, 1972; 60 July: 809 – 820
2 Chen W H, Smith C H, Fralick S C. A Fast Computational Algorithm for the Discrete Cosine Transform. IEEE Trans, 1977; COM – 25 (Sept): 1004 – 1009