DOI:10.13718/j.cnki.xsxb.2020.01.007

一种基于变位与反转变换的动态规划压缩算法①

丁爱芬, 周华君, 石宜金

云南大学旅游文化学院 信息学院,云南 丽江 674199

摘要:针对数据区域的集中化程度和平滑度特征,采用变位标识方法和反转变换策略,降低较大值数据区域的凹凸度,实现了数据无损压缩效率的提升.对比试验结果显示新算法的整体压缩效率优于原数据压缩算法和 huffman 统计编码.算法非常适合大数据环境的磁盘存储、网络传输和应用先进等流式压缩场景.

关 键 词:无损数据压缩;变位标识;反转变换;流式压缩

中图分类号:TP301.6 **文献标志码**:A **文章编号**:1000-5471(2020)01-0037-05

随着计算机应用技术的发展,多媒体技术的日益普及,知识数据呈现出爆炸状态.大规模、高效率的数据通信和存储成为当前网络应用与数据存储两个领域研究的重要课题.数据压缩技术已成为当今数字通信、广播、存储和多媒体娱乐中的一项关键技术方案,压缩算法的编码形式、过程特点和压缩效率直接制约着计算机应用的发展[1-14].按照编码形式有统计编码、预测编码、变换编码、混合编码.按照编码过程有整体编码、流式编码.

虽然有损编码的压缩效率一般为几十到几百倍,但是数据信息存在必然的损失;虽然无损编码数据信息无损失,但压缩效率极低.不同压缩方法的计算机实现存在着资源制约,如统计编码的统计过程需要对数据进行全面计数,如果压缩数据量大,需要消耗大量的内存空间用于统计,同时统计编码需要编码表,不支持流式压缩.

在大数据环境下,很多应用要求数据压缩、传输、解码能够有序分片,逐片立刻可用,且占用空间较少,如流视频应用、大规模图像识别,同时要求数据能够根据需要进行按需、有序、分片压缩存储和读取,如海量图片存储、海量网页存储.海量数据存储问题及其数据量增幅问题已无法通过存储容量的简单扩容解决.海量数据计算问题也对内存容量提出了较高要求,传统的基于数据库和数据仓库的存储严重制约着大数据存储量及其应用场景.大数据环境下的数据压缩技术是解决海量数据存储与数据应用的必要技术,而现今仍采用传统的压缩技术,其中无损数据压缩效率有限,且无法实现分片化、序列化、低时空复杂性需求.同时物联网时代的物联通信环境下,很多设备配置了较少的存储空间,大部分数据需要物联互传,物联通信的片段性、频繁性、海量性,也对分片数据压缩、通信、解压、应用提出了更高的要求.设计一个可以分片压缩、分片解压、空间要求少、算法复杂度低、按需定义压缩比率且能实现流式处理的算法很有必要.

本文提出一种改进的基于变位与反转变换的动态规划压缩算法,实现流式、可分片段的数据无损压缩,通过改进动态规划的最优子结构,利用数据反转操作,实现算法的压缩优化.算法的压缩过程是对数据进行最优分段处理,每个压缩片段可以独立压缩和解压,在压缩过程中不需要一次性读入较多数据,消耗计算机内存资源较少;同时算法过程是流式的,适合序列化和片段化的网络传输和数据存放;再者流式、片段化的压缩方式可以直接进行二次压缩和解压,且压缩和解压的时间和空间复杂度较低,满足即时反复

① 收稿日期:2018-03-24
基金项目:云南省教育厅项目(2016ZDX261).
作者简介:丁爱芬(1985-),女,讲师,硕士,主要从事计算机应用方面的研究.
通信作者:周华君,讲师,硕士.

压缩性质，对较大数据的压缩要求有较好的压缩弹性.

1 问题提出

对于n个待压缩的数据$\{p_1, p_2, \cdots p_n\}$，其中$0 \leqslant p_x \leqslant 255$，将其分割成$m$个连续段$s_1, s_2, \cdots, s_m$. 其中第$i$个象素段$s_i(1 \leqslant i \leqslant m)$中有$l[i]$个象素，且该段中每个象素都只用$b[i]$bit表示；设$t[i]$表示前$i$个连续段中$p_i$的个数，则$t[i] = \sum_{k=1}^{i-1} l[k]$. 第$i$个象素段$s_i$的元素位置表示为$t[i]+1$至$t[i]+l[i]$，则第$i$个像素段$s_i$的最大数据所占二进制空间为$h_i = |\log(\max_{t[i]+1 \leqslant k \leqslant t[i]+l[i]} p_k + 1)|$，$h_i \leqslant b[i] \leqslant 8$，因此只需要3 bit表示$b[i]$，如果限制$1 \leqslant l[i] \leqslant 255$，则只需要8 bit表示$l[i]$，因此，第$i$个象素段所需的存储空间为$l[i] \times b[i] + 11$ bit. 按此格式存储象素序列$\{p_1, p_2, \cdots p_n\}$，需要$\sum_{i=1}^{m} l[i] \times b[i] + 11m$ bit的存储空间，压缩问题转变成求解不同分组下的最小片段和.

2 动态规划算法求解

采用动态规划算法解决如下：

设$l[i]$，$b[i]$是$\{p_1, p_2, \cdots p_n\}$的最优分段. 显而易见，$l[1]$，$b[1]$是$\{p_1, \cdots, p_l[1]\}$的最优分段，且$l[i]$，$b[i]$是$\{p_l[1]+1 \cdots p_n\}$的最优分段，即图像压缩问题满足最优子结构性质.

设$s[i]$，$1 \leqslant i \leqslant n$，是象素序列$\{p_1, p_2, \cdots p_n\}$的最优分段所需的存储位数，由最优子结构性质易知：

$$s[i] = \max_{l \leqslant k \leqslant min(i, 256)} \{s[i-k] + k \times bmax(i-k+1, i)\} + 11 \tag{1}$$

其中

$$bmax(i, j) = |\log(\max_{i \leqslant k \leqslant j} \{p_k\} + 1)| \tag{2}$$

以此编写的压缩算法所需的计算时间为$O(n)$[15].

通过上述分析，可以看出该算法具有如下特征：

1) 算法适用于片段数据间差距较小的情况；

2) $bmax(i, j)$越小越好；

3) 算法重点是尽可能多地去除多余的0 bit，保留有效的1 bit；

4) k个数据序列的算法压缩效果取决于$bmax(i-k+1, i)$，当k个数据序列的最大值大于128时，$bmax(i-k+1, i) = 8$，该片段的压缩效果基本无变化.

3 改进的动态规划算法

改进的动态规划算法如下：

1) 读取待压缩序列$\{p_1 \cdots p_j\}$，$j < 255$；

2) 定义算法存储片段长度为$tmpsl = Interger.Max$；//$Interger.Max$代表机器表示的最大整数

3) 定义当前位置为$i = 1$，$Reverse = 0$，片段断点位置为$tmpk = i$；

4) 定义断点位置$k(0 \leqslant k \leqslant i)$，初始化$k = i$，初始化$tmpbmax = 8$；

5) 寻找序列片段$\{p_k \cdots p_i\}$最小值$\min x = \min(p_k \cdots p_i)$；

序列最大值$\max x = \max(p_k \cdots p_i)$；//$\{p_k \cdots p_i\}$表示从$k$到$i$的待压缩序列；

6) 求解序列片段$\{p_k \cdots p_i\}$最大值$\max x$所需的二进制位$bmax = \text{binary}(\max x)$，反转后的最大值所需二进制为$\overline{bmax} = \text{binary}(255 - \min x)$；

7) 求解当前序列所需存储空间长度$tmps[i] = \min\{s[i-k] + k \times bmax, s[i-k] + k \times \overline{bmax}\}$；$tmpbmax = \min\{bmax, \overline{bmax}\}$；

8) if $tmps[i] == s[i-k] + k \times \overline{bmax}$ then $Reverse = 1$；

// 当$tmps[i]$来源于$s[i-k] + k \times bmax$，$tmpbmax = bmax$，否则$tmpbmax = \overline{bmax}$，代表需要反转；

9) if $tmps[i] < tmpsl$ then $tmpsl = tmps[i]$；$tmpk = k$；$T[i] = Reverse$；

10) if $k = 0$ then 序列的第一次断点位置为$tmpk$，$s[i] = tmp$；$s[i]$片段的断点$l[i]$为$tmpk$；

else

$k--$；转到 5)；

11) if $i < j$ then

$i++$；转到 4)；

else

return s,l,T//s 的第 i 个元素 $s[i]$ 代表从$\{p_1 \cdots p_i\}$ 的最小存储空间，l 的第 i 个元素 $l[i]$ 代表片段 $s[i]$ 的最优断开位置，T 的第 i 个元素 $T[i]$ 代表是否需要反转，为 1 则反转，否则不反转；

12) 根据返回的 $s[i],l[i],T[i]$ 执行压缩.

其中

binary(max$x$) 算法逻辑如下：

1) $len = 1$// 定义 max$x$ 的初始化长度；

2) int tmp = max$x$/2// 开始二进制变换；

3) while $tmp > 0$// 当 tmp 不为 0，表示二进制转化并未完成；

$tmp = (int)(tmp/2)$// 继续对 tmp 进行二进制变换迭代；

$len = len + 1$//max$x$ 的对应长度加 1；

end while// 循环结束；

4) return len.

4　算法压缩对比实验

以 500 Byte 数据量为一个测试数据片段，设计小数值整数不同出现概率下的压缩实验(以压缩一次为例)，实验以 0.02 为概率密度步长(0.02 表示小数值整数出现概率为 0.02，0.04 表示小数值整数出现概率为 0.04)，生成随机序列，对每个步长片段应用改进算法和原算法执行 50 次测试，以 50 次测试的压缩值均值为衡量数据. 对比测试结果如表 1 所示.

表 1　出现小数值整数的不同概率密度下两种压缩算法的压缩均值

V_1	V_2	V_3	V_1	V_2	V_3
0.02	394.00	451.52	0.52	303.68	328.38
0.04	391.98	447.38	0.54	299.92	323.14
0.06	389.10	442.96	0.56	295.92	318.00
0.08	385.46	438.80	0.58	291.58	312.70
0.10	381.90	433.86	0.60	287.70	307.72
0.12	378.22	428.44	0.62	282.70	300.70
0.14	375.54	425.52	0.64	278.44	295.44
0.16	371.34	419.16	0.66	275.84	291.74
0.18	368.30	414.18	0.68	272.06	287.22
0.20	364.10	409.58	0.70	268.02	282.18
0.22	360.90	405.06	0.72	264.02	277.04
0.24	356.70	399.00	0.74	259.84	271.40
0.26	353.10	393.96	0.76	256.08	266.52
0.28	349.18	389.08	0.78	253.20	262.20
0.30	345.18	383.28	0.80	249.40	257.28
0.32	340.32	377.28	0.82	244.66	251.52
0.34	338.88	375.44	0.84	241.30	247.10
0.36	333.38	367.56	0.86	235.38	240.30
0.38	329.70	363.30	0.88	232.94	236.68
0.40	327.22	358.62	0.90	227.22	229.96
0.42	321.76	352.14	0.92	223.04	224.70
0.44	319.16	348.98	0.94	218.36	219.24
0.46	314.40	343.04	0.96	213.04	212.96
0.48	311.04	338.22	0.98	208.70	208.14
0.50	307.72	333.66	1.00	202.10	201.56

表1中，V_1 代表小数值整数的出现概率，V_2 代表反转数据压缩算法的压缩结果长度值(单位为Byte)，V_3 代表非反转数据压缩算法的压缩结果长度值(单位为Byte)，其压缩对比序列见图1. 如图1所示，变位反转压缩算法整体效率优于原压缩算法.

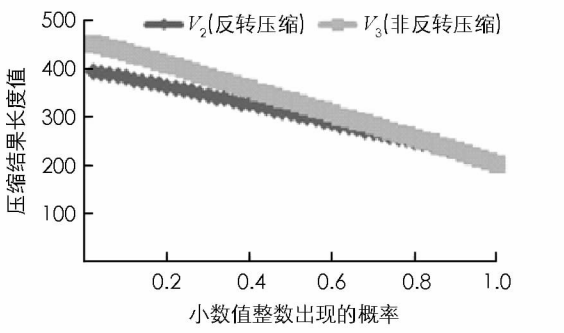

图1 小数值整数的不同概率密度下两种算法压缩效率对比图

由于每个压缩片段的最大值不超过255，所以两种算法的时间复杂度均为 $O(n)$，空间复杂度均为 $O(1)$. 算法比较简单，仅通过变位与反转的方法减少了算法的复杂性，使用性广；算法具有无损压缩性，算法的使用过程信息具有可逆性，可通过12位的信息位直接进行还原；算法压缩所需要的时间较少，算法支持流式压缩，在使用中可以根据需要利用多线程直接对前一个压缩片段结果进行压缩，在网络传输过程中可以根据需要边压缩、边传递、边使用，这对流式数据应用是很有益处的.

将基于变位与反转变换的动态规划压缩算法与基于统计的编码算法进行对比(选取huffman算法). 以500 Byte数据量为一个测试数据片段，设计在不同数据分布场景下的实验. 以$\lceil 255/20 \rceil$为步长，以$\lceil 255/20 \rceil \times i$为小数据区块和大数据区块出现频率，对序列进行压缩，其中 i 代表循环次数，同时也对数据随机均匀分布的情况进行实验. 实验结果对比图分别如图2—4所示.

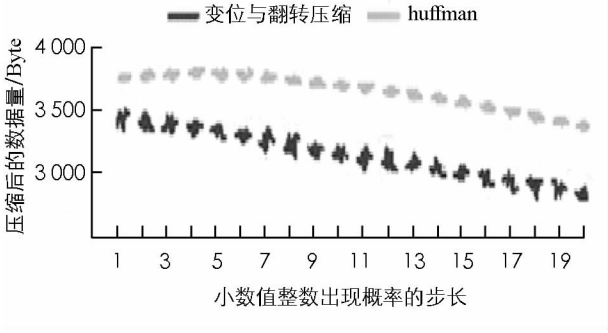

图2 小数据块下压缩效果对比图

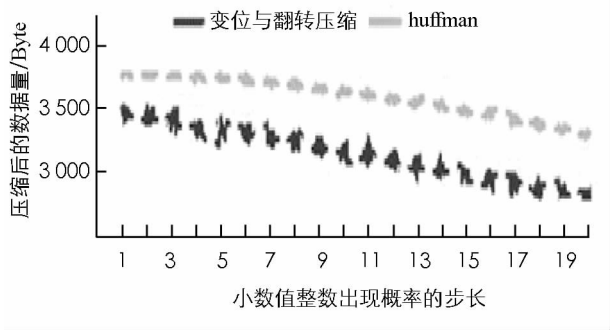

图3 大数据块下压缩效果对比图

如图2,3所示，在小数据块和大数据块两种情况下基于标志的反转变位压缩算法明显优于基于统计的huffman算法，只有在数据随机均匀分布的情况下huffman算法才稍优于基于变位与反转的压缩算法，但这种情况下，两种算法的压缩效率都不高.

综上，基于变位与反转的压缩算法比原移位压缩算法效果更好，在存在数据区块特征的数据序列中明显优于基于统计的huffman算法，在随机均匀分布的数据序列中稍逊于huffman编码.

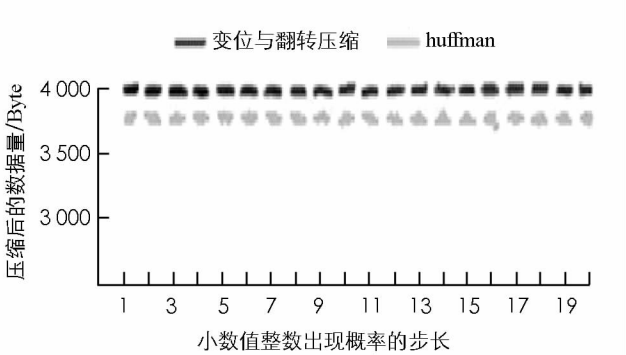

图4 数据随机均匀分布情况下压缩效果对比图

5 结 论

通过对图像压缩算法的分析，提出一种基于标识的变位与反转变换的动态规划无损压缩算法，算法改进了原图像压缩算法的最优子结构存储和表示方法，以变位和反转变化方法构造新的动态规划最优子结构. 经过试验对比，新的图像数据压缩算法总体上优于原算法，在极大值出现较频繁的压缩片段，压缩效率明显优于原图像数据压缩算法. 算法压缩一次的时间复杂度为 $O(n)$，空间复杂度为 $O(1)$，适合大规模数据的压缩. 算法具有分段压缩和流式拼接特征，适

合网络环境下数据的分段压缩和快速片段解压拼接的流媒体应用. 算法的压缩率动态可调，适合网速受限条件下的大规模流媒体传输和磁盘存储.

参考文献：

[1] 王　超，王　浩，王　伟，等. 基于优化 ROI 的医学图像分割与压缩方法研究 [J]. 重庆邮电大学学报(自然科学版)，2015，27(2)：279-284.
[2] 张　波，刘郁林，常博文，等. 线性回归的分布式压缩采样算法 [J]. 重庆邮电大学学报(自然科学版)，2014，26(2)：207-213.
[3] 周晓恺. 一个数据无损压缩算法研究[D]. 武汉：华中科技大学，2015.
[4] 蔡　明，乔文孝，鞠晓东，等. 一种新的数据无损压缩编码方法[J]. 电子与信息学报，2014，36(4)：1008-1012.
[5] 刘　杰，易茂祥，朱　勇. 采用字典词条衍生模式的测试数据压缩 [J]. 电子与信息学报，2012，34(1)：231-235.
[6] 王林泓，王　平. 融合降采样的复基带脉冲压缩算法研究 [J]. 西南大学学报(自然科学版)，2016，38(2)：162-168.
[7] 朱晓波，周廷刚，曾　波，等. 顾及空间邻接关系的多级河流线状矢量数据并行压缩算法 [J]. 西南大学学报(自然科学版)，2017，39(2)：100-106.
[8] 李传伟，慕德俊，李安宗，等. 随钻声波测井数据实时压缩算法 [J]. 西南石油大学学报(自然科学版)，2008，30(5)：81-84，9.
[9] 吴国清，陈　虹. 一种科学数据无损压缩方法 [J]. 计算机工程与应用，2006，42(5)：172-175.
[10] 刘　杰，徐三子. 用于编码压缩的测试位重组算法 [J]. 计算机工程，2010，36(21)：19-21.
[11] 胡学龙，江新炼，周　琳，等. 一种改进的无损压缩数字音频编码器 [J]. 微电子学与计算机，2003，20(7)：23-25.
[12] 杜庆峰，周晓玮，谢　涛，等. 大规模向量式有限元行为数据压缩模型及算法 [J]. 同济大学学报(自然科学版)，2014，42(11)：1711-1717.
[13] 李宗霖，周晓伟，张　柳. 向量式有限元面单元行为数据并行压缩研究 [J]. 电脑编程技巧与维护，2015(4)：69-70，90.
[14] 姜正吉. 基于 KLT 变换的光谱图像压缩[D]. 西安：西安电子科技大学，2013.
[15] 王晓东. 计算机算法设计与分析[M]. 4 版. 北京：电子工业出版社，2012：75-86.

A Dynamic Programming Compression Algorithm Based On Variable Identification And Reverse Transformation

DING Ai-fen，　ZHOU Hua-jun，　SHI Yi-jin

Information School，Tourism and Culture College of Yunnan University，Lijiang Yunnan 674199，China

Abstract：Aiming at the concentration degree and smoothness of data region，using variable identification method and reverse transformation strategy to reduces the smoothness of large value data regions，the efficiency of the Lossless data compression algorithm is enhanced. The experimental results of comparison tests show that the compression efficiency of the new algorithm is better than the original data compression algorithm and the Huffman statistical coding. the algorithm is very suitable for flow compression such as disk storage，network transmission and application in big data environment.

Key words：lossless data compression；variable identification；reverse transformation；flow compression

责任编辑　张　枸