第 37 卷 第 2 期
Vol.37 No.2

计 算 机 工 程
Computer Engineering

2011 年 1 月
January 2011

• 软件技术与数据库 •　　文章编号：1000—3428(2011)02—0040—03　　文献标识码：A　　中图分类号：TP311

RWM & DEWS 数据二次压缩算法研究

周双英，余建桥

(西南大学计算机与信息科学学院，重庆 400715)

摘　要：对柑橘实时水分监控与旱情预警系统中的数据参数进行拟合分析，在探讨 SDT 和 LZW 2 种压缩算法的基础上，提出一种二次压缩算法：第 1 次压缩采用基于 SDT 算法思想的平均值方法；第 2 次压缩采用改进的 LZW 算法实现。对压缩效果进行测试，仿真实验证明，在相同的硬件和软件环境下，该算法能实时、高效地压缩数据。

关键词：实时数据；SDT 算法；LZW 算法；二次压缩算法

Research of Secondary Compression Algorithm of RWM & DEWS Data

ZHOU Shuang-ying, YU Jian-qiao

(College of Computer and Information Science, Southwest University, Chongqing 400715, China)

【Abstract】 Aiming at the data parameters fitting and analysis in Real-time Water Monitoring and Drought Early Warning Systems, and based on the discussion of Swing Door Trengding(SDT) and Lemple-Ziv-Welch(LZW) algorithm, this paper proposes a second compression algorithm. In the first compression, it uses average method based on SDT compression algorithm; in the second compression, it uses the modified LZW algorithm to achieve. Through testing the compression effect, simulation experimental results show that in the same hardware and software environment, the algorithm is capable of compressing data real-time and effectively.

【Key words】 real-time data; Swing Door Trengding(SDT); Lemple-Ziv-Welch(LZW); secondary compression algorithm

DOI: 10.3969/j.issn.1000-3428.2011.02.014

1　概述

在柑橘实时水分监控与旱情预警系统(Real-time Water Monitoring and Drought Early Warning Systems, RWM & DEWS)中，系统参数信息量大，需要进行高效压缩。采用 SDT (Swing Door Trengding)压缩算法，会丢失部分原始数据，并且在一定精度下的压缩率较低；而 LZW(Lemple-Ziv-Welch)算法的字典容量有限，字符串匹配速度慢，压缩过程耗费时间长。针对这些不足，本文提出一种针对 RWM & DEWS 数据特点的二次压缩算法。

2　RWM & DEWS 的数据特征及压缩算法技术

2.1　RWM & DEWS 的数据特征

在 RWM & DEWS 中，每隔一定时间，从田间采集反映柑橘叶片含水量的参数信息(包括土壤张力、树干径流、气象等)，根据收集到的参数信息进行实时计算分析，发出旱情和水涝预警信号，并给出相应的专家意见。

现在的数据压缩理论和技术已经很成熟，在不同的应用领域，针对不同的数据应用特征，可以引用不同的数据压缩技术。在实时领域中，实时数据都有一定的变化规律，应针对这些规律，研究特定的数据压缩算法。在 RWM & DEWS 中，影响含水量的参数很多，其中，温度、湿度、光照、土壤张力、树干径流是关键因素，其变化主要受气候、季节等影响。本文针对中国农业科学院柑桔研究所提供的 2009 年柑橘叶片含水量实时数据中的温度参数进行压缩，温度数据曲线见图 1，其中，采集日期的格式为“月-日-时”。温度参数特点如下：

(1)数据量大：每隔特定时间间隔采集相关参数信息；

(2)在一定季节内，参数值基本不变或呈慢变化；

(3)在季节交换间，参数值呈现波动变化；

(4)在特殊情况下，参数值有时呈现跳变。



图 1　柑橘叶片温度参数曲线

由于数据量很大，并且在一定时间段内参数值基本保持不变或呈慢变化特性，波动范围也不大，因此开始压缩一般使用有损压缩，最常见的是有损的 SDT 算法。但使用 SDT 算法偏差较大，因此，本文采用基于 SDT 算法思想的平均值方法，既可以有高压缩率，而且偏差不大。在第 1 次压缩后，数据量仍然很大，而且数据在一定阶段内呈现连续性，所以决定采用第 2 次压缩。为了保证一定的精度，第 2 次压缩采用无损压缩，最常见的是无损的 LZW 算法。本文对 LZW 算法做了改进，使压缩效率更好。

2.2　SDT 算法

最常见的有损 SDT 压缩算法[1]是一种直线趋势压缩算法，在规定最大允许误差内，使用一条有起点和终点的直线代替一系列连续数据点，以找到尽可能长的直线趋势。该算

作者简介：周双英(1985－)，女，硕士研究生，主研方向：数据挖掘，实时数据压缩算法；余建桥，教授

收稿日期：2010-07-18　**E-mail：**zhousy01@163.com

法通过减少保留的数据点个数来实现压缩，适合压缩比较平缓的数据。

在 SDT 算法中，用起点和终点的一条直线代替连续的数据点，这样做并未充分考虑中间数据点的情况，有一定的精度损失。在确保一定的精度下，压缩率会比较低。

2.3 LZW 算法

最常用的无损 LZW 压缩算法是基于字典模型的，采用一种先进的串表压缩，首先建立一个字符串表，把每一个第 1 次出现的字符串放入串表中，并用一个数字来表示，这个数字与此字符串在串表中的位置有关，并将这个数字存入压缩文件中，如果这个字符串再次出现，即可用表示它的数字来代替，并将这个数字存入文件中。压缩完成后将串表丢弃。由于只用一个数字来表示串，压缩文件只存储数字，不存储串，从而使数据的压缩效率得到较大的提高。LZW 算法具体实现见文献[2-5]。该算法实现容易，但字典容量有限、字符串匹配速度慢，计算时间长。

3 RWM & DEWS 数据二次压缩算法的实现

3.1 算法原理

先采用基于旋转门压缩算法思想的平均值方法对原始实时数据进行第 1 次压缩，在第 1 次压缩的结果基础上，采用改进的 LZW 算法实现第 2 次压缩[6]。

3.2 第 1 次压缩

本次压缩采用基于 SDT 压缩算法思想的平均值方法，其原理是：将初始采集到的原始数据存放到表中，这些数据是等时间间隔采样得到的。然后将关键数据点读出到数组中，$x0$ 为某一 TL 时间段的第 1 个数据点，xn 为该段的最后一个数据点，TL 为该段数据的采样间隔数或称该段的长度，t 为在某一 TL 时间段内找关键数据点的时间间隔，$m=TL/t$ 为关键点个数。计算每一 TL 时间段内关键点数据的平均值，输出起点、平均值和时间间隔。

平均值方法的流程如图 2 所示。

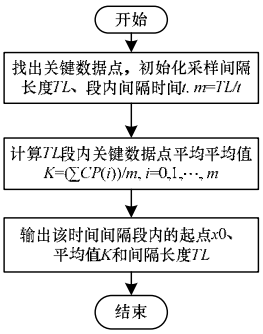


图 2　平均值方法的流程

3.3 第 2 次压缩

针对第 1 次压缩后的数据，本次压缩采用改进的 LZW 算法[4-5]，设计特定的字典模型，将可能出现的单字符初始化到字典中，然后统计单个字符 x 出现的概率 P，设定一个概率阈值 PT，串长 str，如果 $P>PT$, $str=3$，则将该字符 x 对应的串 x、xx、xxx 也同时初始化到字典中以提高算法的执行速度。之所以这样改进初始化，主要考虑到数据在一定时间段内不变或慢变化的特征。

改进的 LZW 算法编码流程如下：

(1)设定 PT、串长 str 阈值(扫描、统计、统计单字符概率)。

(2)初始化字典，根据统计到的单个字符 a 出现的概率 $P(a)$，如果 $P(a)>PT$，则将该字符 a 对应的串长 aa、aaa 和所有可能出现的单个字符都同时初始化到字典中并分别赋予相应码值(即码字 CW)。

(3)从输入字符流(CharStream)中读入第 1 个字符作为前缀串($Prefix$)。

(4)读入下一个字符 C：

1)如果：没有字符 C 表示输入已穷尽，则输出前缀串 $Prefix$ 对应的码字 CW，输出结束；

2)如果 $Prefix+C$ 已在串表中，则 $Prefix+C$->$Prefix$；goto (4)；

3)如果 $Prefix+C$ 不在串表中(尚未定义)：则 CW for $Prefix$->输出；$Prefix+C$ ->串表，C->$Prefix$；goto(4)。

3.4 LZW 的解码流程

LZW 的解码流程如下：

(1)将发端传送的 LZW 压缩代码作为译码的依据。LZW 编码器的输出代码由压缩代码及单个字符所组成的串表构成。在初始化时，串表中包括所有可能的单个字符和相应的串长，其余的代码值按顺序分配给新的字符串。

(2)从输入的码字流读第 1 个码字，把其对应的字符 C 输出，并把当前的码字当旧码(Old Code)保存起来。

(3)读入下一个字符码字 CW：

1)如果没有码字 CW 表示输入已穷尽，则结束；

2)如果 CW 已存于表中，则 S for CW->输出，Old Code 对应的字符串 OS->$Prefix$，该前缀与 S 的最后一个字符 C(从右往左)$Prefix+C$->串表，CW->Old Code：goto(3)；

3)如果 CW 不在串表中(尚未定义)：则码字 Old Code 对应的字符串 OS->$Prefix$，该前缀与 OS 的最后一个字符 C(从右往左)$Prefix+C$->串表，$Prefix+C$ ->输出，CW->Old Code；goto(3)。

4 实验与分析

本节针对 SDT 算法、平均值方法、LZW 算法、改进的 LZW 算法以及本文提出的二次压缩算法，对数据进行拟合和特征分析[7]。实验采用 Visual C++平台，使用 C 语言编写实现，由于硬件、软件环境以及编程风格均相同，因此即使计算的结果与现已实现的压缩算法结果有一定差别，仍然可以使用该结果来比较证明所提出的二次压缩算法的高效性。程序输出各种算法的压缩时间和压缩比。压缩比计算公式为：$C=N/N'$，其中，N 为压缩前数据容量；N'为压缩后数据容量。

设 CT 为压缩时间，表示压缩 N 个数据所用的时间。设 UCT 为解压时间，表示解压原始数据容量为 N 时所用时间。由于该系统数据的压缩与解压存在相应正关系，设 $UCT=CT$。λ 为压缩系数，b 表示压缩比与压缩解压时间的制约关系。压缩算法性能指标计算公式为：

$$XN= XN=\lambda\times\frac{C}{(CT+UCT)\times b}\times N$$

其中，λ、b 为系数因子，由于仿真只看关键因素，因此 λ 取值为 1，b 取值为 1 000。针对具体的情况决定压缩比和压缩时间哪个更重要，系数因子自定设置，在这里的设置使两者相对公平。

4.1 平均值方法与 SDT 算法仿真比较

分别利用平均值方法和原始的 SDT 算法对中国农业科学院柑桔研究所提供的 2009 年反映柑橘叶片含水量的 74 880 点实时数据进行压缩，压缩结果如表 1 所示。可以看出，用这 2 种算法进行第 1 次压缩后压缩比相同；在压缩时间上，平均值方法要稍微大一点，压缩性能相差不大。

表 1 SDT 算法与平均值方法的性能比较

压缩类型	原始文件/点	压缩后文件/点	压缩时间/s	压缩比
SDT	74 880	4 820	0.015	15.535
平均值方法	74 880	4 820	0.016	15.535

对压缩后的数据进行信息重建，数据曲线如图 3 所示，其中，采集日期的格式为“月-日”。

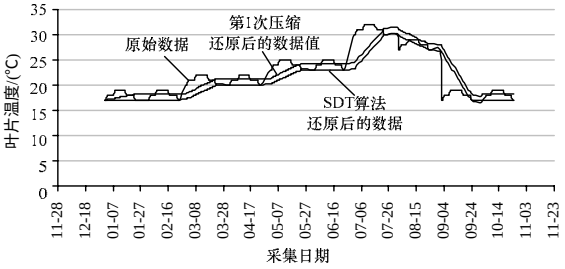


图 3 数据重建曲线

压缩误差采用均方解压缩误差指标来表示：

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-y_i')^2}$$

其中，y_i 为原文件数据点值；y_i'为压缩后重建数据曲线相对应的数据值。平均值方法与 SDT 算法误差比较如表 2 所示。

表 2 SDT 算法与平均值方法的误差比较

压缩类型	压缩误差
SDT 算法	2.687
平均值方法	2.411

可以看出，平均值方法的偏差较小，更接近原始数据。在压缩性能很接近的情况下，数据更接近原始数据更好。因此，第 1 次压缩本文决定采用平均值方法，不使用 SDT 算法。

4.2 改进的 LZW 算法与原 LZW 算法的仿真比较

分别利用改进的 LZW 算法和原 LZW 算法对中国农业科学院柑桔研究所提供的 2009 年反映柑橘叶片含水量的 74 880 点实时数据进行压缩，设概率阈值 PT＝1%，串长 str＝3，仿真压缩结果如表 3 所示。可以看出，改进 LZW 算法与原 LZW 算法相比，压缩比较高且压缩时间较短，性能更优。

表 3 LZW 算法与改进的 LZW 算法比较

压缩类型	原始文件/点	压缩后文件/点	压缩时间/s	压缩比
LZW 算法	74 880	16 564	2.078	4.521
改进的 LZW 算法	74 880	16 552	2.047	4.524

4.3 SDT、LZW 和二次压缩算法的仿真比较

分别利用 SDT 算法、LZW 算法和本文提出的二次压缩算法对中国农业科学院柑桔研究所提供的 2009 年反映柑橘叶片含水量的 74 880 点实时数据进行压缩，压缩性能结果如表 4 所示。可以看出，本文提出的二次压缩算法性能最好，SDT 算法次之，LZW 算法最差，由此证明了二次压缩算法的正确性和有效性。

表 4 3 种压缩算法的性能比较结果

压缩类型	原始文件/点	压缩后文件/点	压缩时间/s	压缩比	XN
SDT 算法	74 880	4 820	0.016	15.535	0.485 469
LZW 算法	74 880	16 564	1.968	4.521	0.001 149
二次压缩算法	74 880	1 279	0.031	58.546	0.944 290

5 结束语

本文针对 RWM and DEWS 的功能及数据特点，提出一种二次压缩算法，并且通过仿真比较分析，证明该二次压缩算法的压缩率有了显著提高，大大缩短了压缩时间，具有很强的应用价值。不过在使用改进的 LZW 算法进行第 2 次压缩时，概率阈值和串长 str 的取值都会影响到最终的压缩比和压缩速度。本文根据压缩数据的变化特征进行多次压缩实验来决定这些参数的取值，从而使压缩效果达到最好。这些参数如何取值目前没有明确的公式可以使用，不过通过反复实验发现，参数取值与数据采集的时间有很大关联。因此，下一步研究决定使用时间标签法来进行数据压缩，从而得到数据采集周期与参数取值之间的对应关系，然后用反复实验的方法来证明这种对应关系，如果误差不大，就可以提出一个较为实用的参数取值公式。

参考文献

[1] 周学文, 汤同奎, 邵惠鹤. SDT 算法及其在过程数据压缩中的应用[J]. 计算机应用与软件, 2003, 20(1): 47-49.

[2] 卓 越, 杨长生. 一种基于自适应字典的通用无损压缩算法[J]. 计算机工程, 2001, 27(2): 149-151.

[3] 许 霞, 马光思, 鱼 涛. LZW 无损压缩算法的研究与改进[J]. 计算机技术与发展, 2009, 19(4): 125-127.

[4] 张风林, 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.

[5] 王 泉, 齐 春, 罗新民. LZW 压缩算法的改进及其参数优化分析[J]. 重庆邮电学院学报: 自然科学版, 2005, 17(3): 351-355.

[6] 郑 莹, 李光耀, 孙燮华. 一种新的非线性分形压缩算法[J]. 计算机工程, 2008, 34(11): 21-22.

[7] 冯晓东, 邵惠鹤. 一种改进的过程数据压缩算法及其性能分析[J]. 测控技术, 2002(增刊 1): 136-139.

编辑 顾姣健

(上接第 39 页)

持资源消耗低、解析速度高等特点的同时，扩展支持了拉 DOM 的特性，具有对文档操作方便的特点，有更强的实用性。

参考文献

[1] Open Geospatial Consortium. OGC07-036-2007 Geography Markup Language(GML) Implementation Specification[S]. 2007.

[2] 兰小机, 闾国年, 张书亮, 等. 一种通用 GML3.0 解析引擎的设计与实现[J]. 地球信息科学, 2005, 7(1): 59-64.

[3] 张 迪, 朱 敏, 张凌立. 基于 SAX 的 XML 解析与应用[J]. 计算机与数字工程, 2008, 36(7): 103-106.

[4] 荣 江, 赵庆展, 李 伟. StAX 技术在 XML 文档解析中的应用[J]. 农业网络信息, 2008, (5): 165-167.

[5] 兰小机, 闾国年, 张书亮, 等. GML3.0 应用模式的开发及应用研究[J]. 计算机工程, 2005, 31(21): 34-36.

[6] Lu Chang-Tien, Santos R, Sripada L, et al. Advances in GML for Geospatial Applications[J]. GeoInformatica, 2007, 11(1): 131-157.

[7] 任 鑫, 曹冬磊, 金蓓弘. 支持 StAX 的高效 XML 解析器的设计与实现[J]. 计算机科学, 2006, 33(6): 128-131.

编辑 顾姣健