一种基于SDT算法的新的过程数据压缩算法

宁海楠
(南京大学 工程管理学院,江苏 南京 210093)

摘 要:针对流程工业实测过程数据压缩存储问题,在深入分析旋转门(SDT,Swing Door Trending)算法的基础上,提出了一种基于SDT算法新的过程数据压缩算法(NSDT, New Swing Door Trending)。NSDT算法采用曲线对过程数据进行拟和以实现数据压缩,与SDT算法相比能取得更好的压缩效果。根据理论分析和实验数据结果分析,证明了NSDT算法确实可以在不增加压缩误差的前提下,有效地提高压缩比。

关键词:旋转门算法;NSDT算法;实时数据压缩
中图分类号:TP274 **文献标识码**:A **文章编号**:1673-629X(2010)01-0025-04

A New Process Data Compression Algorithm Based on SDT Algorithm

NING Hai-nan
(School of Management and Engineering, Nanjing University, Nanjing 210093, China)

Abstract: To address the issue of process industry real-time process data compression, after in-depth analysis of SDT (Swing Door Trending) algorithm, a new process data compression algorithm (NSDT, New Swing Door Trending) based on the SDT algorithm is presented. NSDT algorithm is an algorithm which fits process data using curve, and compared with the SDT algorithm it can achieve better compression results. According to theoretical analysis and experimental data analysis, it is proved that NSDT algorithm can actually greatly improve the compression ratio while the compression error is not increased.

Key words: swing door trending algorithm; NSDT algorithm; real-time data compression

0 引 言

随着流程工业自动化水平的提高,大量的实时测量数据需要存储和管理。实时采集的过程数据量巨大,这就给网络系统和信息处理系统造成了极大的负担,数据压缩是解决这个问题的有效途径[1,2]。

对SDT(Swing Door Trending)算法作了深入的分析,然后在此基础上提出了一种新的过程数据压缩算法(NSDT, New Swing Door Trending)。根据理论分析和实验数据结果分析,证明了与SDT算法相比NSDT算法确实可以有效地提高压缩效果。

1 SDT算法

如图1所示,以数据点1为起始点,在垂直方向上距离数据点1为E的地方有两个支点。两个支点和过程数据之间的连线U,L构成了两扇虚拟的以支点为轴的"门"。算法开始时两扇门都是关闭的。在算法执行过程中两扇门只能开启不能关闭。对采集到的每个数据点,作以下处理[3]:

· 如果上支点和当前数据点连线的斜率大于U的斜率,逆时针旋转U使U穿过新的数据点;

· 如果下支点和当前数据点连线的斜率小于L的斜率,顺时针旋转L使L穿过新的数据点;

· 否则维持原状。

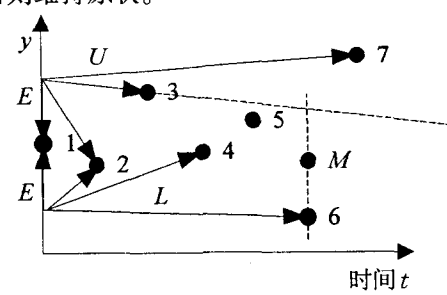

图1 SDT算法操作示例

当U的斜率大于L的斜率时,结束当前轮压缩,以当前数据点为起始点开始新一轮的压缩。图1中,在处理完数据点7后,U的斜率大于L的斜率,结束当前轮压缩,以数据点7为起点开始新一轮的压缩。一轮压缩结束后,算法需要记录压缩区间的起点和终点,数据用来在解压缩时进行数据重建。为了减小压缩误差,记录压缩区间终点时一般不直接记录终点处的原始数

收稿日期:2009-04-13;**修回日期**:2009-08-17
作者简介:宁海楠(1983-),男,江苏盐城人,硕士,研究方向为流程工业实时数据库系统。

据，而是记录在处理完压缩区间终点时直线 U, L 在终点处函数值的均值(例如图 1 中 M 处的 y 值)[4,5]。

SDT 算法在压缩时，一般是对每个测点产生的数据分别压缩。这样，如果某测点数据变化很慢可能会出现一轮压缩持续时间过长的现象，这就无法保证在线趋势提取的实时性。为了避免此类情况发生，预先设定强制记录限 FSRL(Forced Storage - Recording Limit)。一旦当前压缩区间的长度达到 FSRL，就强行结束当前轮压缩并重新开始一轮压缩[6]。

如果所有数据都需要相同的存储空间，则 SDT 算法的压缩比 CR(compression ration)为[7]：

$$CR = \frac{\text{原始数据数量}}{\text{压缩区间数量} \times 2}$$

SDT 算法本质上是一种利用直线对过程数据进行拟合以实现数据压缩的算法。容易证明，SDT 算法中记录限 E 是用户能容忍的最大误差。SDT 算法可以在最大误差不超过 E 的条件下，找到尽可能最长的直线趋势。

由于 SDT 算法中的拟合直线经过压缩区间起始数据点，拟合直线的斜率是唯一的未知量。SDT 算法的执行过程就是确定拟合直线的斜率的过程。事实上，SDT 算法可以看成求解在压缩误差不超过记录限 E 的条件下令每轮压缩生成的压缩区间最长的拟合直线的斜率问题的一种解法。

2 NSDT 算法

在很多情况下过程数据的变化趋势并不是线性的，SDT 算法不是过程数据的最优压缩算法。在 SDT 算法的基础上发展出了一种新的过程数据压缩算法，称这种新的压缩算法为 NSDT 算法。

2.1 算法原理

SDT 算法是一种用直线对过程数据进行拟合的算法[8]，这就使笔者联想到如果用曲线比如抛物线对过程数据进行拟合会产生什么样的结果？抛物线函数表达式中有 3 个参数，令拟合抛物线经过压缩区间起始数据点可消去一个参数。但是，在压缩算法执行中确定两个未定参数同样很困难。为简化问题，令拟合抛物线函数表达式为如下形式：

$$y = k \cdot t^2 + b, \quad t = 0,1,2,\cdots,n-1 \qquad (1)$$

其中，t 为时刻，n 为压缩区间长度。由于拟合抛物线经过压缩区间起始数据点，b 为压缩区间起始数据点值。这样，拟合抛物线函数表达式中只有一个未定参数 k。仿照 SDT 算法的实现，容易实现用该表达式拟合过程数据的压缩算法。当过程数据的变化趋势近似于抛物线时，该算法的压缩效率得以提高。但是，当过程数据的变化趋势近似于直线时，与 SDT 算法相比该算法的压缩效率很低。所以，要想取得比 SDT 算法更好的压缩效果，新的算法必须能根据实际数据的变化趋势在运行时动态调整拟合函数表达式的形式。

因此，NSDT 算法的拟合函数形式调整为：

$$y = k \cdot t^a + b, \quad t = 0,1,2,\cdots,n-1 \qquad (2)$$

该式中有两个未定参数 k 和 a。为使问题简化，预先给定参数 a 可能的几个取值。这样，NSDT 算法只需在每轮压缩中，计算取参数 a 的每个可能取值时产生的压缩区间长度。显然，参数 a 所有可能取值中产生的压缩区间最长的那一个就是所求的解。

如何确定参数 a 的可能取值集合呢？可以粗略地把过程数据的变化趋势分为三种类型：第一种是平稳变化，即相邻时刻的两个过程数据差值始终保持不变；第二种是变化越来越快，即相邻时刻的两个过程数据差值绝对值越来越大；第三种是变化越来越慢，即相邻时刻的两个过程数据差值绝对值越来越小。对第一种数据，参数 a 取 1 时应该能取得好的压缩效果，对第二种数据参数 a 应该取一个大于 1 的数，对第三种数据参数 a 则最好取一个小于 1 的数。所以，参数 a 可能在取值集合中应该至少有 3 个元素。最终，经过理论分析和实验测定，给定参数 a 可能的取值集合 $A = \{1, 2, 0.5, 0.75\}$。

NSDT 算法是一种用式(2)描述的函数对过程数据进行拟合以实现数据压缩的算法。

2.2 算法描述

如图 2 所示，以数据点 1 为起始点，在垂直方向上距离数据点 1 为 E 的地方有两个支点。A 中的每个元素对应两个连接支点和过程数据的直线或曲线，把这两个直线或曲线看成两扇虚拟的以支点为轴的“门”。A 中有四个元素，共有八扇门。算法开始时这八扇门都是关闭的。在算法执行过程中八扇门只能开启不能关闭。以 a_i 表示集合 A 中的第 $i(i = 0,1,2,3)$ 个元素，以 U_i 和 L_i 表示 a_i 对应的两个直线或曲线的描述函数，则 U_i 和 L_i 的函数表达式为：

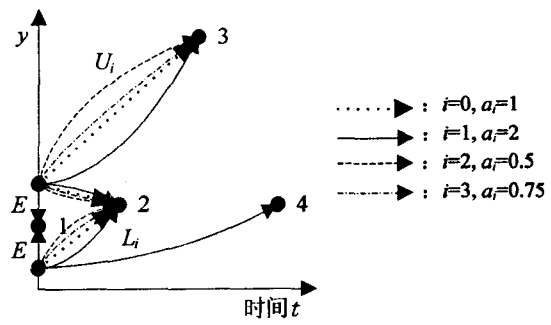

图 2 NSDT 算法操作示例

$$U_i = KU_i \cdot t^{a_i} + bU \qquad (3)$$

$$L_i = KL_i \cdot t^{a_i} + bL \tag{4}$$

其中 t 为时刻。

以 $y_1, y_2, \cdots, y_{k-1}, y_k, y_{k+1}, \cdots, y_n$ 表示实际过程数据,假设当前数据点为 y_k,NSDT算法在接收到 y_k 后作以下处理:

(1) 如果当前数据点为一轮压缩的起始点,置 $bU = y_k + E$ 和 $bL = y_k - E$,结束对 y_k 的处理;

(2) 对 i 的任一可能值,如果 $KU_i \leqslant KL_i$,判断若让 U_i 经过 y_k 是否会增大 KU_i,如果会使 KU_i 增大则增大 KU_i 使 U_i 经过 y_k;

(3) 对 i 的任一可能值,如果 $KU_i \leqslant KL_i$,判断若让 L_i 经过 y_k 是否会减小 KL_i,如果会使 KL_i 减小则减小 KL_i 使 L_i 经过 y_k;

(4) 如果对 i 的所有可能值都有 $KU_i > KL_i$,结束当前轮压缩,以 y_k 为起点开始新一轮的压缩。NSDT算法同样需要在一轮压缩结束后,记录压缩区间的起点和终点数据,同时 NSDT 算法还需要记录最后一个令 $KU_i \leqslant KL_i$ 的 i 值。NSDT算法记录的终点值也不是终点处的原始数据,而是处理完终点数据后 U_i 和 L_i 在终点处函数值的均值 。

NSDT算法在压缩时,同样可能因为某测点数据变化很慢出现一轮压缩持续时间过长的现象,所以NSDT算法同样必须预先设定强制记录限 FSRL (Forced Storage - Recording Limit)。

3　性能分析

为验证 NSDT 算法的性能,对合成数据和实际工业过程数据进行了仿真计算。文中用 CE 表示重建数据和原始数据差值的平方和。

3.1　合成数据计算

实际工业过程数据有其固有的特性,一般认为用正弦曲线来模拟实际数据是比较合适的。生成合成数据时,需要考虑正弦波的波幅和采样精度,这两点都会对最终的压缩效果产生影响。实际上,波幅对压缩效果的影响可以通过调整记录限 E 的值来抵消,仅通过调整采样精度来生成不同的合成数据。表 1 列出了数据产生函数。后三组数据在原数据上叠加了噪声 $N(t)$。$N(t)$ 为取值范围在[-0.75, 0.75]的随机数。表 2 列出了对合成数据仿真计算的结果。对每组数据都给出了两组计算结果,以此来反映不同的记录限 E 对压缩效果的影响。

3.2　实际工业过程数据计算

选取实际工业过程数据是扬子石化的丙烯 - 水换热器中的温度和压力数据。这些数据中每一变量的采样周期为 1min,总共包括 2030 个原始数据。由于实际数据的波动情况不同,为了便于对比不同组数据,统一调整每组数据记录限 E 的值,使对不同组数据用 NSDT 算法压缩得到相同的压缩比。表 3 是利用 SDT 算法和 NSDT 算法对丙烯 - 水换热器中的温度和压力数据的计算结果。

表 1　合成数据的产生函数

数据	产生函数	备注
1	y=10sin(t)	t 为均匀分布于[0, 200π]的 4000 个数据
2	y=10sin(t)	t 为均匀分布于[0, 100π]的 4000 个数据
3	y=10sin(t)	t 为均匀分布于[0, 50π]的 4000 个数据
4	y=10sin(t) + N(t)	t 为均匀分布于[0, 200π]的 4000 个数据
5	y=10sin(t) + N(t)	t 为均匀分布于[0, 100π]的 4000 个数据
6	y=10sin(t) + N(t)	t 为均匀分布于[0, 50π]的 4000 个数据

表 2　合成数据仿真计算结果

数据	压缩算法	压缩比(CR)	压缩误差(CE)	FSRL	记录限 E
1	SDT	4.98753	1675.41	100	1.0
	NSDT	6.64452	1196.33		
1	SDT	3.32779	313.786	100	0.5
	NSDT	5	286.821		
2	SDT	9.95025	1566.12	100	1.0
	NSDT	13.245	1336.75		
2	SDT	6.64452	332.127	100	0.5
	NSDT	10	325.517		
3	SDT	19.802	1657.54	100	1.0
	NSDT	26.3158	1376.87		
3	SDT	13.245	394.445	100	0.5
	NSDT	20	325.407		
4	SDT	4.0404	1209.34	100	1.0
	NSDT	4.83092	1182.85		
4	SDT	2.13789	280.619	100	0.5
	NSDT	2.63505	318.265		
5	SDT	6.47249	1170.28	100	1.0
	NSDT	8.26446	1118.87		
5	SDT	2.39234	290.557	100	0.5
	NSDT	3.01205	328.05		
6	SDT	10.1781	1074.53	100	1.0
	NSDT	13.3333	1065.49		
6	SDT	2.57732	299.681	100	0.5
	NSDT	3.08642	338.683		

表 3　实际工业过程数据计算结果

数据	压缩算法	压缩比(CR)	压缩误差(CE)	FSRL	记录限 E
7	SDT	14.9265	153.035	100	0.51
	NSDT	17.807	145.738		
8	SDT	15.9843	30.6388	100	0.25
	NSDT	17.807	30.3509		
9	SDT	14.7101	161.119	100	0.58
	NSDT	17.807	159.599		
10	SDT	14.5	99.7887	100	0.45
	NSDT	17.807	95.1839		
11	SDT	14.0972	0.104977	100	0.0147
	NSDT	17.807	0.106721		
12	SDT	15.3788	0.068507	100	0.0105
	NSDT	17.807	0.063997		

3.3　计算结果分析

表 2 的计算结果表明,NSDT 算法确实是正确的

和有效的。在对数据集1,2,3的计算中,NSDT算法的压缩比CR比SDT算法平均增加了约40%,压缩误差CE平均减小了约15%。这就表明,NSDT算法确实可以有效地提高压缩效果。另外,注意到,当记录限E为1.0时NSDT算法的压缩比CR比SDT算法平均增加了约30%,而当E为0.5时平均增加了约50%,这说明,NSDT算法比SDT算法更适合压缩要求比较高的情况。在对数据集4,5,6的计算中,NSDT算法的压缩比CR比SDT算法平均增加了约30%,压缩误差CE几乎相等。这就说明,在有噪声的情况下,NSDT算法与SDT算法相比仍然能显著地提高压缩效果。

表3显示了对实际过程数据进行压缩时,在相同的记录限E的条件下,NSDT算法的压缩比CR比SDT算法平均增加了约20%。表明在处理实际过程数据时,使用NSDT算法可以明显地减少对存储空间的需求,减小现场总线网络发生拥塞的可能性,提高控制系统性能。另外,对不同组的实际过程数据,在NSDT算法压缩比CR相同的条件下,SDT算法的压缩比CR也很接近。这就是说,NSDT算法对压缩效果的提高是比较稳定的。可以相信,在大多数工业生产条件下,NSDT算法与SDT算法相比压缩效果会有较为显著的提高。

4 结束语

该研究提出了一种基于SDT算法的新的过程数据压缩算法,编写了仿真软件并对合成数据和实际过程数据进行了仿真计算。与SDT算法相比,新的算法能在不增加压缩误差的条件下有效地提高压缩比。

参考文献:

[1] 徐 慧.实时数据库中数据压缩算法的研究[D].杭州:浙江大学,2006.

[2] 赵光煌,赵 平.实时数据库的现状及发展趋势[J].天津农学院学报,2002(16):58-63.

[3] Bristol E H. Swing Door Trending: Adaptive Trending Recording[R]. Institute Society of American: Research Triangle Park. NC, 1990:749-754.

[4] Mah R S H. Process Trending with Piecewise Linear Smoothing[J]. Computer Chemical Engineering, 1995, 19(2):129-137.

[5] Hales J C, Sellars H L. Historical data recording for process computers[J]. Chem. Eng. Prog., 1981(11):38-43.

[6] 王正洪.改进的SDT算法及其在过程数据压缩中的应用[J].煤矿自动化,2000(6):9-11.

[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,2002,21(1):136-139.

[8] 稆月强.工业历史数据库的研究[D].杭州:浙江大学,2007.

(上接第24页)

是具有动态效果的图,假设人的视线与XOY面成30°视角所看到的场景效果。

参考文献:

[1] Hartley R, Zisserman A. Multiple View Geometry in Computer Vision[M]. Cambridge: Cambridge University Press, 2004.

[2] Seitz S M, Kim J. Multiperspective Imaging[J]. Computer Graphics and Applications. IEEE, 2003, 23(6):16-19.

[3] 刘 钢,彭群生,鲍虎军.基于多幅图像的场景交互建模系统[J].计算机辅助设计与图形学学报,2004,16(10):1419-1424.

[4] Saxena A, Chung S H, Ng A Y. Learning depth from single monocular images[M]// Neural Information Processing Systems. [s.l.]:[s.n.],2005.

[5] Saxena A, Sun M, Ng A Y. Learning 3d scene structure from a single still image[C]// International Credit Card Validation Workshop on 3D Representation for Recognition (3dRR-07). [s.l.]:[s.n.],2007.

[6] Hoiem D, Efros A, Herbert M. Geometric context from a single image[C]// Tenth IEEE International Conference on Computer Vision. [s.l.]:[s.n.],2005:654-661.

[7] Delage E, Lee H, Ng A Y. A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image[C]// Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. [s.l.]:[s.n.],2006:2418-2428.

[8] Lin L, Zeng K, Wang Y, et al. 3D structure inference by integrating segmentation and reconstruction from a single image [J]. Computer Vision, IET, 2008, 2(1):15-22.

[9] Hoiem D, Efros A, Hebert M. Automatic photo pop-up[C]// Proceedings of ACM SIGGRAPH. [s.l.]:[s.n.],2005:577-584.

[10] Levin A, Lischinski D, Weiss Y. A closed form solution to natural image matting[C]//Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. NewYork:[s.n.],2006:61-68.

[11] Levin A, Lischinski D, Weiss Y. A closed-form solution to natural image matting[J]. IEEE transactions on pattern analysis and machine intelligence, 2008, 30(2):228-242.