自适应参数的轨迹压缩算法*

龙 浩[1,2], 张书奎[1,3†], 孙鹏辉[4]

(1. 徐州工业职业技术学院 信息与电气工程学院, 江苏 徐州 221002; 2. 苏州大学 计算机科学与技术学院, 江苏 苏州 215006; 3. 江苏省现代企业信息化应用支撑软件工程技术研发中心, 江苏 苏州 215104; 4. 中国矿业大学 计算机科学与技术学院, 江苏 徐州 221002)

摘 要: 针对现有轨迹数据压缩算法难以确定压缩阈值的缺点,提出了自适应参数的轨迹压缩算法。该算法根据用户期望达到的压缩效果,按照优先保证压缩比的策略,在保证压缩效率和压缩效果的情况下,帮助用户自动确定压缩阈值,从而避免了用户需要根据自己的经验,进行反复实验来得到理想压缩阈值的过程。实验结果表明,自适应参数与非迭代的压缩算法相结合,在保证原压缩算法压缩效率的情况下,解决了压缩阈值难以确定的问题,同时还提高了原压缩算法的压缩效果;自适应参数与迭代的压缩算法相结合,会降低原压缩算法的压缩效率,但解决了压缩阈值难以确定的问题,同时还提高了原压缩算法的压缩效果。

关键词: 轨迹压缩; 自适应参数; 压缩阈值; 压缩比

中图分类号: TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2018)03-0685-04

doi:10.3969/j.issn.1001-3695.2018.03.010

Trajectory compression algorithm with adaptive parameter

Long Hao[1,2], Zhang Shukui[1,3†], Sun Penghui[4]

(1. *School of Information & Electrical Engineering, Xuzhou College of Industrial Technology, Xuzhou Jiangsu* 221002, *China*; 2. *School of Computer Science & Technology, Soochow University, Suzhou Jiangsu* 215006, *China*; 3. *Jiangsu Province Support Software Engineering R&D Center for Modern Information Technology Application in Enterprise, Suzhou Jiangsu* 215104, *China*; 4. *School of Computer Science & Technology, China University of Mining & Technology, Xuzhou Jiangsu* 221002, *China*)

Abstract: In order to overcome the disadvantage that it was hard to confirm the perfect compression threshold while the algorithm running, this paper presented trajectory compression algorithms with adaptive parameters. This method, based on the compression effort that users expected and the strategy ensuring the compression ratio priority, could help users to automatically determine the compression threshold while guaranteeing the compression efficiency and effect. It would avoid users found the perfect compression threshold by their experience and repeated experiments. The experimental results show: adaptive parameters combined with the non-iterative compression algorithms guarantee the compression efficiency of original compression algorithms and solve the problem that it is hard to confirm the perfect compression threshold, at the same time also improve the compression effect of the original compression algorithms; adaptive parameters combined with the iterative compression algorithms reduce the compression efficiency of original compression algorithms, but solve the problem that it is hard to confirm the perfect compression threshold, at the same time also improve the compression effect of the original compression algorithms.

Key words: trajectory compression; adaptive parameter; compression threshold; compression ratio

0 引言

近年来,随着 RFID、GPS、无线传感器等技术的不断发展和广泛应用,全球范围内的各种移动对象都可以得到有效的定位和跟踪。由此产生了海量的轨迹数据,这些数据蕴涵着丰富的信息,迫切需要研究人员对其进行扩展和分析。文献[1,2]中提出,移动对象的运动轨迹在本质上是连续的;但是,现有的采集、存储和处理技术迫使本文用离散的结构来描述移动对象的运动轨迹。轨迹的最简单描述是指用一系列带时间戳的坐标点来表示轨迹。随着时间的推移,这些轨迹数据量急剧增长,数据规模变得越来越庞大、复杂,给数据的存储、传输和分析带来了一系列的难题。因此,需要对移动对象的轨迹数据进行压缩,一方面,可以节约存储空间,便于轨迹的存储;另一方面,可以减少轨迹数据数量,便于轨迹的传输;更重要的是,通过对轨迹数据的压缩,可以保留轨迹中有用的信息,去除轨迹中冗余的信息,便于对移动对象的轨迹数据进行深入分析。移动对象轨迹数据压缩[3]的目的是在保留数据所包含信息的前提下,尽可能地减少数据量,缩小数据所占用的存储空间,即在保证移动对象轨迹准确性的前提下,去除冗余定位点,从而减少数据量。近年来,各个学科的快速发展以及互联网的广泛应用,为轨迹数据压缩技术提供了大量的技术支持和强大的动力。均匀采样算法是最简单、最原始的轨迹数据压缩算法;

收稿日期: 2016-10-31; **修回日期**: 2017-01-03 **基金项目**: 国家自然科学基金资助项目(61201212);江苏省自然科学基金资助项目(BK2011376);江苏省"六大人才高峰"项目(2014-WLW-010);苏州市融合通信重点实验室(SKLCC2013XX);江苏省产学研前瞻性项目(BY2012114);徐州市科技局应用基础研究计划资助项目

作者简介:龙浩(1984-),男,讲师,研究生,主要研究方向为数据挖掘、分布式计算(longhhao@163.com);张书奎(1966-),男(通信作者),教授,博士,主要研究方向为信息安全、分布式计算;孙鹏辉(1991-),男,研究生,主要研究方向为数据挖掘、并行计算.

1973 年,Douglas 等人[4]提出的道格拉斯—普克算法是最经典的轨迹压缩算法之一;2004 年,Meratnia 等人[2]提出了基于速度的自顶向下算法和基于时间比的自顶向下算法,基于时间比的自顶向下算法是道格拉斯—普克算法的一种变形,后来,Meratnia 又提出了支持在线轨迹数据实时压缩的开窗算法;2006 年,Potamias 等人[5]提出了面向小内存设备轨迹数据压缩的 STTrace 算法;2009 年,Schmid 等人[6]提出使用路网语义信息的形式代替轨迹点存储压缩轨迹,之后,许多学者基于路网语义信息进行了大量的研究[7,8];与 STTrace 算法类似,2014 年,Song 等人[9]提出了启发式空间质量简化算法;2014 年,Muckell 等人[10]将压缩比引入到启发式空间质量简化算法中,提出了启发式空间质量简化扩展算法,该算法能够在压缩比和压缩误差方面灵活地调整压缩。

尽管这些算法均具有较高的计算性能,但现有的压缩算法都是基于固定压缩阈值,如垂直距离阈值和速度差值阈值,来判定轨迹点是否被保留。但是压缩阈值的设置需要依靠用户的大量经验,而且这个过程是一个不确定的、盲目的、赌博式的过程,这导致了每次设置压缩阈值时,用户都需要通过反复地尝试、大量地实验,才能确定符合用户要求的压缩阈值。

本文针对现有轨迹数据压缩算法压缩阈值难以确定的缺点,提出了自适应参数的轨迹压缩算法,在保证压缩效率和压缩效果的情况下,帮助用户自动确定压缩阈值,从而避免用户根据自己的经验、反复进行实验来确定理想的压缩阈值的过程。在本文中,首先,结合自适应参数的轨迹压缩算法和同步欧氏距离,引入压缩比 ,充分考虑轨迹的时空特性,提出了自适应阈值压缩算法(adaptive threshold compression algorithm, ATC);然后,结合自适应参数的轨迹压缩算法和移动对象速度差值压缩思想,引入压缩比 ,充分考虑了轨迹的时空特性和运动趋势,提出了自适应速度差值压缩算法(adaptive speed difference compression algorithm, ASDC)。

1　相关工作

移动对象时空轨迹是记录移动对象位置、属性和时间的序列[11]。也可以说,移动对象时空轨迹就是时间到空间的映射 $O:R^+\rightarrow R^d$。其中,O 是一个以时间作为自变量的连续函数,通过此函数可以得到在某一时刻 $t(t\in R^+)$ 时,该移动对象在 d 维空间 R^d(通常指二维空间或者三维空间)中所处的位置坐标。为了更好地描述本文中的方法,需要对轨迹及其相关属性进行形式化描述。文献[12]对轨迹进行了相关的形式化定义,本文同样适用。移动对象的轨迹是由一系列带时间戳的坐标点组成的序列 $TR_i=p_1p_2p_3\cdots p_j\cdots p_{\text{len}_i}$。其中,$p_j(1\leqslant j\leqslant \text{len}_i)$ 是一个 d 维的记录点。轨迹 TR_i 的长度 len_i 可以不同于其他轨迹。轨迹 $p_{c1}p_{c2}\cdots p_{ck}$ 是 TR_i 的一条子轨迹。

定义 1　同步欧氏距离(sed)[5]。对于轨迹中的每一个点,该点的同步欧氏距离是通过在该点的前驱点和后继点之间插值得到的同步位置点与该点的真实位置之间的距离。如图 1 所示,轨迹点的同步欧氏距离表示为 sed,垂直距离表示为 $d_\perp$。

由图 1 可以看出,p_ip_{i+2} 为当前线段,p_{i+1} 为当前正在检测的点,p'_{i+1} 是 p_{i+1} 在线段 p_ip_{i+2} 上的同步点,则 p'_{i+1} 坐标为

$$\begin{aligned} t'_{i+1}&=t_{i+1}\\ x'_{i+1}&=x_{i+1}+\frac{t_{i+1}-t_i}{t_{i+2}-t_i}(x_{i+2}-x_i)\\ y'_{i+1}&=y_{i+1}+\frac{t_{i+1}-t_i}{t_{i+2}-t_i}(y_{i+2}-y_i) \end{aligned}\tag{1}$$

根据式(1),p_{i+1} 到线段 p_ip_{i+2} 的同步欧氏距离的计算方法为

$$\text{sed}=\sqrt{(x_{i+1}-x'_{i+1})^2+(y_{i+1}-y'_{i+1})^2}\tag{2}$$

定义 2　速度差值。移动对象速度的差值,反映了移动对象的运动趋势。如图 2 所示,移动对象从前一点移入采样点的速度(到达速度)表示为 v_{in};移动对象从采样点移向后一点的速度(离开速度)表示为 v_{out};速度差值表示为 sd。

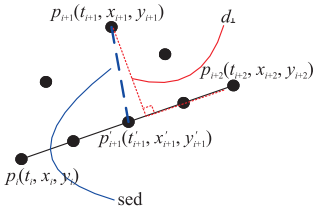

图1　同步欧氏距离

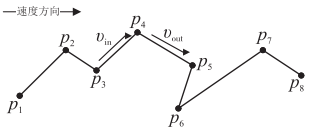

图2　速度差值示意图

由图 2 可以看出,v_{in} 和 v_{out} 分别为采样点的到达速度和离开速度,则速度差值的计算公式定义为

$$sd=|v_{\text{in}}-v_{\text{out}}|\tag{3}$$

根据压缩方式的不同[3],一般可以将轨迹数据压缩算法分为分段线性逼近压缩算法和非线性逼近压缩算法。本文主要研究和分析了分段线性逼近压缩算法。一般情况下,分段线性逼近压缩算法较为简单,计算量较小,能够有效地压缩轨迹数据。本文详细分析了五种常见的分段线性逼近压缩算法,包括均匀采样算法、道格拉斯—普克算法、自顶向下的时间比算法、基于速度的自顶向下算法和开窗算法。

均匀采样算法是有损压缩算法中最原始的算法。均匀采样算法的主要优点是以线性时间运行,且实现简单,算法复杂度为 $O(n)$;它的主要缺点是采样点之间在速度和方向上急剧变化时也不会被存储,这样会与原始轨迹之间存在较大的误差。道格拉斯—普克算法[4]是最经典的压缩算法之一,算法复杂度为 $O(n\log n)$,它具有平移和旋转不变性,给定曲线与阈值后,压缩结果是一定的。但道格拉斯—普克算法是以垂直距离作为关键点是否保留的判断依据,这样就忽略了轨迹数据中时间维度的信息;此外它还是一个批处理算法,也就是说,整个轨迹必须在压缩之前被存储,只能离线地进行轨迹数据的压缩。因此,它通常不适合于移动和实时应用程序。自顶向下的时间比算法[1,2]是道格拉斯—普克算法的扩展,它的算法复杂度同样为 $O(n\log n)$,自顶向下的时间比算法在继承了道格拉斯普克算法的平移和旋转不变性等优势的基础上,将同步欧氏距离作为关键点是否保留的判断依据,同时考虑了移动对象运动轨迹的空间特性和时间特性,使得最终得到的轨迹具有更好的拟合效果。但它仍是一个批处理算法,也就是说,整个轨迹必须在压缩之前被存储,只能离线地进行轨迹数据的压缩。因此,它通常不适合于移动和实时应用程序。基于速度的自顶向下算法[1]利用时间序列中隐藏的时空信息改进了现有的压缩算法,它的算法复杂度为 $O(n^2)$,尽管它的计算复杂度较高,但其同时考虑了轨迹的空间特性和时间特性,最终可以得到具有较好拟合效果的压缩结果。开窗算法的算法[11]复杂度为 $O(n^2)$,尽管开窗算法需要较高的计算代价,但它依然很流行,因为它是一种在线的压缩算法,并且具有较好的抗噪性。

如表 1 所示,道格拉斯—普克算法和自顶向下的时间比算法都是离线的压缩算法,它们只有在得到了全部的轨迹点之后才能开始进行压缩;而开窗算法是在线的压缩算法,它们可以在获得轨迹点的同时进行压缩,在线算法的优势在于支持实时应用程序和有效地降低内存的使用量;均匀采样算法和基于速度的自顶向下算法既可以实现在线压缩,又可以实现离线压缩。虽然,上述的经典压缩算法在压缩轨迹数据时,都具有较

好的压缩效果;但是,它们也存在着许多的不足。例如,上述压缩算法都需要用户提前设定压缩阈值,但压缩阈值的设定又需要依靠用户的经验,而这个过程往往是不确定的、盲目的、赌博式的。因此,本文提出了自适应参数的轨迹压缩算法。

表 1　轨迹数据压缩算法总结(其中 n 为轨迹大小)

算法名称	英文名称	复杂度	复杂度	误差指标
均匀采样算法	Uniform Sampling Algorithm	$O(n)$	在线/离线	目标压缩比(λ)
道格拉斯—普克算法	Douglas-Peucker Algorithm	$O(n \log n)$	离线	空间距离(ε)
自顶向下的时间比算法	Top-Down Time Ratio Algorithm	$O(n \log n)$	离线	同步欧氏距离(ε)
基于速度的自顶向下算法	Top-Down Speed-Based Algorithm	$O(n^2)$	在线/离线	速度差值(v)
开窗算法	Opening Window Algorithm	$O(n^2)$	在线	空间距离(ε)

2　自适应参数的轨迹压缩算法

当前绝大部分轨迹数据压缩算法,都需要用户预先设置压缩阈值(垂直距离或速度差值)之后,才能够对轨迹数据进行压缩;然而,压缩阈值的设置需要依靠用户的大量经验,这是一个盲目的、赌博式的过程,为了找到符合用户要求的压缩阈值,往往需要进行大量的实验。Muckell 等人[10]将压缩比引入到启发式空间质量简化算法中,提出了启发式空间质量简化扩展算法,该算法通过预先设置压缩比和压缩误差,实现了算法压缩过程的灵活调整,以便得到具有更好压缩效果的新轨迹。本文借鉴文献[10]中,利用压缩比控制压缩过程的思想,充分考虑移动对象轨迹数据的时空特性,提出了自适应参数的轨迹压缩算法。该方法将用户从预设压缩阈值的繁复过程中解脱出来,将这一繁琐过程交给压缩算法自动处理,提高了压缩效率;同时,能够得到具有更好压缩效果的新轨迹。该思想如下:首先,用户根据自己希望达到的压缩效果,预先设置希望达到的压缩比 λ;然后,自适应参数的轨迹压缩算法自动确定压缩阈值,对原始轨迹数据进行压缩,压缩后,判断压缩比是否达到用户要求,若未达到,则改变压缩阈值后,再次进行压缩,直至达到用户预先设定的压缩比。

2.1　自适应阈值压缩算法

本文结合自适应参数的轨迹压缩算法和同步欧氏距离 sed,引入压缩比 λ,充分考虑了轨迹的时空特性,提出了自适应阈值压缩算法(adaptive threshold compression algorithm, ATC)。该算法是道格拉斯—普克算法的一个变种,用同步欧氏距离 sed 替换垂直距离 ε,放弃原算法中预设的压缩阈值,而改用预设压缩比 λ 的方法来自动确定压缩阈值。该算法不仅能够得到具有更高拟合度的新轨迹,还避免了通过反复实验确定压缩阈值的过程。ATC 算法主要思想如下:a)计算最大同步欧氏距离 maxSedDistance;b)根据压缩比 λ,自动确定 ATC 算法的阈值 ε;c)对原始轨迹进行压缩。在第一阶段中,首先,计算每个轨迹数据点的同步欧氏距离 sedDistance(第 1、2 行);然后,选取其中最大的同步欧氏距离(第 3、4 行)。在第二阶段中,根据压缩比 λ,自动确定 ATC 算法的阈值 (第 6 ~ 9 行)。在第三阶段中,对原始轨迹进行压缩。首先,保存最大同步欧氏距离 maxSedDistance 大于压缩阈值 ε 的点(第 10、11 行);然后,将原始轨迹在保留点处划分为两条新轨迹,保留点前的所有点作为新轨迹 trajectory1,保留点及保留点后的所有点作为新轨迹 trajectory2(第 12、13 行);最后,迭代执行 ATC 算法(第 14、15 行),直到实际压缩比 radio 小于预设压缩比 λ 结束。

算法 1　ATC(traj ,λ)
输入:原始轨迹 traj,压缩比 λ。
输出:压缩后的轨迹 ReturnPoints。
/ * 第一阶段,最大同步欧氏距离计算 * /
1　for each $point_i \in$ traj do
2　　sedDistance←计算 $point_i$ 的同步欧氏距离;
3　　if sedDistance > maxSedDistance then
4　　maxSedDistance←sedDistance;
5　end for
/ * 第二阶段,压缩阈值确定 * /
6　sed←maxSedDistance;/ * * /
7　radio←1 - len(ReturnPoints)/len(traj);/ * 计算轨迹数据的压缩比 * /
8　if radio > λ
9　　sed←0.9 * sed;/ * 缩小压缩阈值 sed * /
/ * 第三阶段,轨迹进行压缩 * /
10　if maxSedDistance > sed
11　　ReturnPoints←该点坐标信息;
/ * 将当前轨迹在该点处划分为两段新轨迹 * /
12　　trajectory1←该点之前的点组成的轨迹; / * 将前一段轨迹存入 trajectory1 中 * /
13　　trajectory2←该点之后的点组成的轨迹; / * 将后一段轨迹存入 trajectory2 中 * /
14　ATC(trajectory1,λ);/ * 对前一段轨迹进行迭代压缩 * /
15　ATC(trajectory2,λ);/ * 对后一段轨迹进行迭代压缩 * /
end

2.2　自适应速度差值压缩算法

本文结合自适应参数的轨迹压缩算法和移动对象速度差值压缩思想,引入压缩比 λ,充分考虑了轨迹的时空特性和运动趋势,提出了自适应速度差值压缩算法(adaptive speed difference compression algorithm,ASDC)。该算法不仅具有更高的效率和能够得到具有更高拟合度的新轨迹,还避免了通过反复实验确定压缩阈值的过程。ASDC 算法的主要思想如下:a)根据压缩比 λ,自动确定 ASDC 算法的阈值 ε;b)对原始轨迹进行压缩。在第一阶段中,首先,根据压缩比 λ,确定阈值数组 maxSpeedDistance 的大小 index 并初始化阈值数组(第 1、2 行);然后,计算每个轨迹数据点的速度差值 speedDistance,将最大的 index 个速度差值 speedDistance 保存到阈值数组 maxSpeedDistance 中并按升序排序(第 3 ~ 8 行);最后,将阈值数组 maxSpeedDistance 中的第一个元素作为压缩阈值 ε(第 9 行)。在第二阶段中,对原始轨迹进行压缩,计算每个轨迹数据点的速度差值 speedDistance,速度差值 speedDistance 是否大于压缩阈值 ε,若大于压缩阈值 ε,则保存该点,直到所有的速度差值 speedDistance 都小于压缩阈值 ε 结束(第 10 ~ 15 行)。

算法 2　ASDC(traj,λ)
输入:原始轨迹 traj,压缩率 λ。
输出:压缩后的轨迹 ReturnPoints。
/ * 第一阶段,压缩阈值确定 * /
1　index←len(traj) * (1 - λ);/ * 计算阈值数组 maxSpeedDistance 的大小,index 向上取整 * /
2　maxSpeedDistance[index]←0;/ * 将大小为 index 的阈值数组 maxSpeedDistance 初始化为 0 * /
3　for each $point_i \in$ traj do
4　　speedDifference←计算 $point_i$ 的速度差值;
5　　if speedDifference > maxSpeedDistance[0] then/ * 如果 speedDifference 大于阈值数组 maxSpeedDistance 中的第一个元素,则将 speedDifference 赋值给 maxSpeedDistance[0] * /

```
6            maxSpeedDistance[0]←speedDifference;
7            maxSpeedDistance. sort( ); /* 将阈值数组 maxSpeed-
Distance 按升序排序 */
8    end for
9    sd←maxSpeedDistance[0]; /* 将阈值数组 maxSpeedDistance
中的第一个元素作为压缩阈值 sd */
/* 第二阶段,轨迹进行压缩 */
10   for each point_i ∈ traj do
11       speedDifference←计算 point_i 的速度差值;
12       if speedDifference > sd then
13           ReturnPoints←该点坐标信息;
14   end for
15   end
```

3　实验分析

为了验证本文提出的算法的应用性和性能,开发了轨迹数据分析系统 TrDaMiner。该系统由 Microsoft Visual Studio . Net 2015 开发,移动对象轨迹数据存储在 Microsoft SQL Server 2008 R2 中。实验进行的软/硬件环境为 Windows 7, Intel Core i5-3470 3.20 GHz 的 CPU,4 GB 内存。本文采用 GeoLife 数据集。

本文对自适应阈值压缩算法和自适应速度差值压缩算法进行了实验验证和性能评估。在确保压缩率相同(或相近)的情况下,在拟合效果和算法执行时间两方面,对 ATC 算法和 DP 算法、ASDC 算法和 TD_SP 算法进行了实验对比。

图 3、5 和图 4、6 分别展示了 DP 算法和 ATC 算法对 GeoLife数据集中的两条轨迹进行压缩,达到 99% 的压缩比时,得到的新轨迹的拟合效果。由图 3 和 4 对比、图 5 和 6 对比可以看出,两种压缩算法对相同的轨迹进行压缩,得到的新轨迹均具有较高的拟合度;而且,两种压缩算法在压缩率相同(或相近)的情况下,得到的新轨迹也具有相近的拟合效果。但是,DP 算法需要结合用户的大量经验和反复的压缩测试,才能够得到既具有较高压缩率,又具有较好拟合效果的压缩阈值(即用户预先设定的、作为阈值的垂直距离);而 ATC 算法只需用户根据自己的需要调整压缩率,就能得到满足用户要求、且具有较好拟合效果的新轨迹。

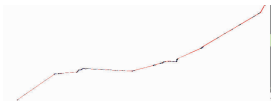

图3　压缩比为99%时,DP算法压缩后得到的新轨迹的拟合效果图

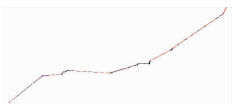

图4　压缩比为99%时，ATC算法压缩后得到的新轨迹的拟合效果图

图5　压缩比为99%时，DP算法压缩后得到的新轨迹的拟合效果图

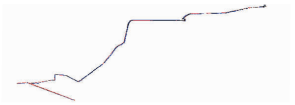

图6　压缩比为99%时，ATC算法压缩后得到的新轨迹的拟合效果图

图 7 ~ 9 分别展示了 TD_SP 算法、TD_SP_NEW 算法和 ASDC 算法对 GeoLife 数据集中的轨迹进行压缩,达到 98.5% 的压缩比时,得到的新轨迹的拟合效果。TD_SP 算法为 Nirvana Meratnia 和 Rolf A. de By 提出的基于速度的自顶向下算法,该算法是一个迭代的过程;TD_SP_NEW 算法是本文在 TD_SP 算法的基础上进行了改进,得到的一个非迭代执行的算法;而 ASDC 算法则是本文结合自适应参数的轨迹压缩算法和移动对象速度差值压缩思想,引入压缩比 ,充分考虑了轨迹的时空特性和运动趋势,得到的一种新的压缩算法。

由图 7 ~ 9 对比可以看出,三种压缩算法对相同的轨迹进行压缩,得到的新轨迹均具有较高的拟合度;而且,两种压缩算法在压缩率相同(或相近)的情况下,得到的新轨迹也具有相近的拟合效果。但是,TD_SP 算法和 TD_SP_NEW 算法要结合用户的大量经验和反复的压缩测试,才能得到既具有较高压缩率,又具有较好拟合效果的压缩阈值(即用户预先设定的、作为阈值的速度差值);而 ASDC 算法只需用户根据自己的需要调整压缩比,就能得到满足用户要求、且具有较好拟合效果的新轨迹。

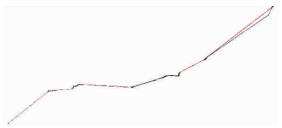

图7　压缩比为98.5%时，TD_SP算法压缩后得到的新轨迹的拟合效果图

图8　压缩比为98.5%时，TD_SP_NEW算法压缩后得到的新轨迹的拟合效果图

图 10 反映了 DP、ATC、TD_SP、TD_SP_NEW 和 ASDC 算法对 GeoLife 数据集中不同的轨迹压缩,达到相同(或相近)的压缩比所需要的执行时间。由图 10 可以看出,DP 算法和 TD_SP 算法在数据量较小时,它们具有相近的压缩效率;而当数据量较大时,DP 算法比 TD_SP 算法的效率更高,因此,DP 算法适合任意规模的轨迹数据压缩,TD_SP 算法则适合较小规模的轨迹数据压缩。ATC 算法的压缩效率最低,所以,它只适用于小规模的轨迹数据压缩。TD_SP_NEW 算法和 ASDC 算法始终具有较高的压缩效率,因此它们适合任意规模的轨迹数据压缩。

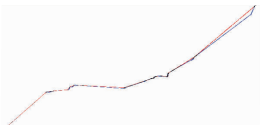

图9　压缩比为98.5%时，ASDC算法压缩后得到的新轨迹的拟合效果图

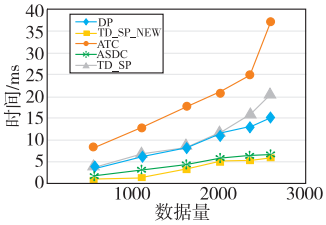

图10　DP、ATC、TD_SP、TD_SP_NEW和ASDC算法达到相同(或相近)压缩比时所需要的执行时间对比图

由图 3 ~ 9 可以看出,五种压缩算法均能得到较高拟合度的压缩轨迹;而由图 10 可以看出,ASDC 算法的压缩效率略低于 TD_SP_NEW 算法,而高于其他的三种算法,但 ATC 算法的压缩效率则远远低于其他四种压缩算法。因此,本文提出的自适应参数的轨迹压缩算法,更适合非迭代过程的压缩算法。将自适应参数的轨迹压缩算法引入非迭代过程的压缩算法,可以在保证压缩效率和压缩效果的情况下,帮助用户自动确定压缩阈值,从而,避免了用户根据自己的经验,反复进行实验来得到理想的压缩阈值的过程。

4　结束语

绝大部分现有的轨迹数据压缩算法,都需要用户根据自己的经验,通过反复实验来确定算法的压缩阈值,本文提出的自适应轨迹数据压缩思想,能够有效地帮助用户自动确定算法的压缩阈值;但自适应参数的轨迹压缩算法,目前存在一个明显的不足,即当将自适应参数的轨迹压缩算法引入迭代过程的压缩算法中时,该思想会比较明显地降低原算法的压缩速度,增加原算法的压缩时间;但是,将自适应参数的轨迹压缩算法引入非迭代过程的压缩算法时,该思想则不会明显改变原算法的压缩效率,能够达到更好的压缩效果。希望本文提出的自适应参数的轨迹压缩算法,能够起到抛砖引玉的作用,吸引更多的研究人员来完善这类压缩算法,从而发现并提出更有价值、更高拟合度、更快压缩速度的轨迹数据压缩算法。

(下转第 716 页)

个优化的隐子空间内,LMTCF 桥接多个用户/项目源领域隐因子,并保留目标数据的局部几何结构,从而更好地解决协同过滤中存在的数据稀疏性问题,且还能有效克服现有方法存在的负迁移和迁移不充分的问题。所提算法采取目标和源领域数据间的距离来简单计算各源领域的加权权值,这可能带来无关源数据的硬性迁移,从而降低目标学习性能。因此,为了进一步避免源领域选择中可能存在的负迁移效应,提出一种有效选择源领域的相关度量准则是本文值得进一步探索的问题。

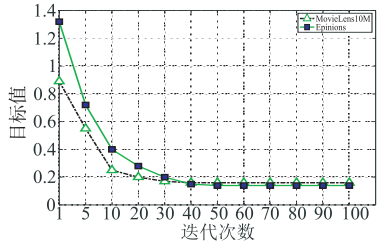

图4 算法收敛性

参考文献:

[1] Adomavicius G, Tuzhilin A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions [J]. IEEE Trans on Knowledge & Data Engineering, 2005,17(6):734-749.

[2] 郑孝遥, 鲍煜, 孙忠宝, 等. 一种基于信任的协同过滤推荐模型[J]. 计算机工程与应用, 2016,52(5):50-54.

[3] 郭彩云, 王会进. 改进的基于标签的协同过滤算法[J]. 计算机工程与应用, 2016,52(8):56-61.

[4] Gu Quanquan, Zhou Jie, Ding C. Collaborative filtering: weighted nonnegative matrix factorization incorporating user and item graphs [C]//Proc of SIAM International Conference on Data Mining. Bethesda:SIAM, 2010:199-210.

[5] 韩亚楠, 曹菡, 刘亮亮. 基于评分矩阵填充与用户兴趣的协同过滤推荐算法[J]. 计算机工程, 2016,42(1): 36-40.

[6] 秦国, 杜小勇. 基于用户层次信息的协同推荐算法[J]. 计算机科学, 2004,31(10):138-140.

[7] Pan S J, Yang Qiang. A survey on transfer learning[J]. IEEE Trans on Knowledge & Data Engineering, 2010,22(10):1345-1359.

[8] Pan Weike, Liu N N, Xiang E W, *et al*. Transfer learning to predict missing ratings via heterogeneous user feedbacks [C]//Proc of the 22nd International Joint Conference on Artificial Intelligence. [S. l.]:AAAI Press, 2011:2318-2323.

[9] Pan Weike, Xiang E W, Liu N N, *et al*. Transfer learning in collaborative filtering for sparsity reduction [C]//Proc of the 24th AAAI Conference on Artificial Intelligence. [S. l.]: AAAI Press, 2010: 230-235.

[10] Pan Weike, Xiang E, Yang Qiang. Transfer learning in collaborative filtering with uncertain ratings [C]//Proc of the 24th AAAI Conference on Artificial Intelligence. [S. l.]: AAAI Press, 2012: 662-668.

[11] Bell R M, Koren Y. Scalable collaborative filtering with jointly derived neighborhood interpolation weights[C]//Proc of IEEE International Conference on Data Mining. Washington DC: IEEE Computer Society, 2007: 43-52.

[12] Long Mingsheng, Wang Jianmin, Ding Guiguang, *et al*. Transfer learning with graph co-regularization [J]. IEEE Trans on Knowledge & Data Engineering, 2014,26(7):1805-1818.

[13] Li Bin, Yang Qiang, Xue Xiangyang. Can movies and books collaborate? Cross-domain collaborative filtering for sparsity reduction[C]//Proc of the 21st International Joint Conference on Artificial Intelligence. 2009:2052-2057.

[14] Li Bin, Yang Qiang, Xue Xiangyang. Transfer learning for collaborative filtering via a rating-matrix generative model [C]//Proc of the 26th Annual International Conference on Machine Learning. New York:ACM Press, 2009: 617-624.

[15] Shi Jiangfeng, Long Mingsheng, Liu Qiang, *et al*. Twin bridge transfer learning for sparse collaborative filtering[C]//Advances in Knowledge Discovery and Data Mining. Berlin:Springer, 2013: 496-507.

[16] Herlocker J, Konstan J A, Riedl J. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms [J]. Information Retrieval Journal, 2002,5(4):287-310.

[17] Tao Jianwen, Wen Shiting, Hu Wenjun. Multi-source adaptation learning with global and local regularization by exploiting joint kernel sparse representation [J]. Knowledge-Based Systems, 2016,98: 76-94.

[18] Belkin M, Niyogi P. Laplacian eigenmaps and spectral techniques for embedding and clustering [J]. Advances in Neural Information Processing Systems, 2002,14(6):585-591.

[19] Boyd S, Vandenberghe F. Convex optimization[M]. California:Stanford University Press, 2003:215-273.

[20] Salakhutdinov R, Mnih A. Probabilistic matrix factorization[J]. Advances in Neural Information Processing Systems, 2008,20(2):1257-1264.

[21] Zhang Sheng, Wang Weihong, Ford J, *et al*. Learning from incomplete ratings using non-negative matrix factorization[C]//Proc of SIAM International Conference on Data Mining. Bethesda: SIAM, 2006:549-553.

(上接第688页)

参考文献:

[1] 江俊文, 王晓玲. 轨迹数据压缩综述[J]. 华东师范大学学报: 自然科学版, 2015(5): 61-76.

[2] Meratnia N, Rolf A. Spatiotemporal compression techniques for moving point objects [M]//Advances in Database Technology-EDBT. Berlin Heidelberg:Springer, 2004:765-782.

[3] 张达夫, 张昕明. 基于时空特性的 GPS 轨迹数据压缩算法[J]. 交通信息与安全, 2013,31(3):6-9.

[4] Douglas D H, Peucker T K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature [J]. Cartographica: The International Journal for Geographic Information and Geovisualization, 1973,10(2):112-122.

[5] Potamias M, Patroumpas K, Sellis T. Sampling trajectory streams with spatiotemporal criteria[C]//Proc of the 18th International Conference on Scientific and Statistical Database Management. Vienna Austria: IEEE Press, 2006:275-284.

[6] Schmid F, Richter K F, Laube P. Semantic trajectory compression [M]//Advances in Spatial and Temporal Databases. Berlin Heidelberg:Springer, 2009: 411-416.

[7] Muckell J, Hwang J H, Patil V, *et al*. SQUISH: an online approach for GPS trajectory compression [C]//Proc of the 2nd International Conference on Computing for Geospatial Research & Applications. New York:ACM Press, 2011: 13.

[8] Richter K F, Schmid F, Laube P. Semantic trajectory compression: representing urban movement in a nutshell [J]. Journal of Spatial Information Science, 2012,2012(4):3-30.

[9] Song Renchu, Sun Weiwei, Zheng Baihua, *et al*. PRESS: a novel framework of trajectory compression in road networks [J]. Proceedings of the VLDB Endowment, 2014,7(9): 661-672.

[10] Muckell J, Jr Olsen P W, Hwang J H, *et al*. Compression of trajectory data: a comprehensive evaluation and new approach [J]. GeoInformatica, 2014,18(3): 435-460.

[11] Lee J G, Han Jiawei, Whang K Y. Trajectory clustering: a partition and group framework[C]//Proc of ACM SIGMOD International Conference on Management of Data. New York:ACM Press, 2007: 593-604.

[12] Lee J G, Han Jiawei, Li Xiaolei. Trajectory outlier detection: a partition-and-detect framework[C]//Proc of the 24th IEEE International Conference on Data Engineering. 2008: 140-149.

[13] Keogh E, Chu S, Hart D, *et al*. An online algorithm for segmenting time series[C]//Proc of the 1st IEEE International Conference on Data Mining. Washington DC:IEEE Computer Society, 2001:289-296.

[14] 李川, 张彪, 李艳梅, 等. 路网空间中 GPS 轨迹压缩的新方法[J]. 北京邮电大学学报, 2015,38(2): 94-97.

[15] 吴家皋, 钱科宇, 刘敏, 等. 基于综合时空特性的混合式轨迹压缩算法[J]. 计算机应用, 2015,35(5):1209-1212.

[16] 吴家皋, 刘敏, 韦光, 等. 一种改进的滑动窗口轨迹数据压缩算法[J]. 计算机技术与发展, 2015,25(12):47-51.