软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(3):734−755 [doi: 10.13328/j.cnki.jos.005438]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

路网感知的在线轨迹压缩方法*

左一萌[1,2], 林学练[1,2], 马 帅[1,2], 姜家豪[1,2]

[1](软件开发环境国家重点实验室(北京航空航天大学),北京 100191)

[2](大数据与脑机智能高精尖创新中心(北京航空航天大学),北京 100191)

通讯作者: 林学练, E-mail: linxl@buaa.edu.cn

摘 要: 随着定位技术的高速发展,定位传感器被广泛应用于智能手机、车载导航等移动设备中,用于采集移动对象位置数据并将数据上传至服务器.该技术的应用方便了位置跟踪、预测和分析,同时也带来了轨迹数据量大、数据冗余、传输和存储代价高等问题.轨迹压缩技术即是针对该问题而提出的,它通过保留关键轨迹点和去除冗余轨迹点信息,降低了轨迹数据的传输和存储开销.分析了近年来轨迹压缩领域的研究进展,针对现有研究工作的不足,提出了一种路网感知的在线轨迹压缩方法,包括针对轨迹压缩的距离有界的隐马尔可夫地图匹配算法和误差有界的高效轨迹压缩算法等,实现了该方法的原型系统 ROADER(road-network aware and error-bounded trajectory compression).基于真实数据集的实验结果表明,该系统在压缩率、误差和执行时间等方面均显著优于同类算法.

关键词: 时空数据;时序数据压缩;轨迹压缩;地图匹配;在线压缩

中图法分类号: TP311

中文引用格式: 左一萌,林学练,马帅,姜家豪.路网感知的在线轨迹压缩方法.软件学报,2018,29(3):734−755. http://www.jos.org.cn/1000-9825/5438.htm

英文引用格式: Zuo YM, Lin XL, Ma S, Jiang JH. Road network aware online trajectory compression. Ruan Jian Xue Bao/ Journal of Software, 2018,29(3):734−755 (in Chinese). http://www.jos.org.cn/1000-9825/5438.htm

Road Network Aware Online Trajectory Compression

ZUO Yi-Meng[1,2], LIN Xue-Lian[1,2], MA Shuai[1,2], JIANG Jia-Hao[1,2]

[1](State Key Laboratory of Software Development Environment (Beihang University), Beijing 100191, China)

[2](Beijing Advanced Innovation Center for Big Data and Brain Computing (Beihang University), Beijing 100191, China)

Abstract: With the rapid development of positioning technologies, positioning sensors are widely used in smart phones, car navigation system and other mobile devices. These positioning systems collect data points at certain sampling rates and produce massive trajectories, which further bring the challenges of storage and transmission of the trajectory data. The trajectory compression technique reduces the waste of the network bandwidth and the storage space by removing the redundant trajectory points and preserving the key trajectory points. This paper summarizes the progresses of trajectory compression researches and proposes a road-network aware and error bounded online trajectory compression system, named ROADER. The system includes a distance-bounded Hidden Markov map matching algorithm and error-bounded efficient trajectory compression algorithm. Experiments based on real data sets show that the system is superior to similar systems in terms of compression ratio, error occurrence and running time.

Key words: spatio-temporal data; time series compression; trajectory compression; map matching; online compression

* 基金项目: 国家自然科学基金(U1636210, 61421003); 国家重点基础研究发展计划(973)(2014CB340300)

Foundation item: National Natural Science Foundation of China (U1636210, 61421003); National Program on Key Basic Research Project (973) (2014CB340300)

本文由基于图结构的大数据分析与管理技术专刊特约编辑林学民教授、杜小勇教授、李翠平教授推荐.

收稿时间: 2017-07-30; 修改时间: 2017-09-05; 采用时间: 2017-11-07; jos 在线出版时间: 2017-12-05

CNKI 网络优先出版: 2017-12-06 15:23:11, http://kns.cnki.net/kcms/detail/11.2560.TP.20171206.1522.003.html

当前,GPS 定位系统在智能设备中被广泛应用,例如:智能手机中的定位系统可以采集行人行驶的位置信息;车辆可以通过车载定位系统采集车辆行驶过程中的位置信息.这些持续采集的 GPS 定位序列构成了行人或车辆的运动轨迹,其中包含了丰富的语义信息,可直接或间接地应用于个人行为分析[1−3]、路径推荐[4−6]以及城市道路的规划和设计等.随着轨迹数据重要性的增加,定位设备采集轨迹点的频率也在提高.它们以秒级的速率采集位置信息,这就导致每个移动设备产生更多的定位数据.例如:一个有数万辆车的租车公司,每天产生数亿的轨迹点数据.这些原始定位数据通过移动设备传输到服务器并进行存储,这不但占用了移动设备宝贵的网络带宽资源,也浪费了存储空间.轨迹压缩技术[7−18]即是针对该问题而提出的,该技术通过去除轨迹中的冗余数据点,以减少数据规模,从而节省传输流量和存储空间.

现实生活中,移动对象的运动通常受到道路结构的限制,因此在轨迹压缩技术中发展出了基于路网的轨迹压缩技术[7,8,19].基于路网的轨迹压缩通常首先对轨迹进行地图匹配,将原始轨迹表示为路网中的路径及路径上的匹配轨迹点,然后分别对路径和轨迹点进行重编码和压缩.文献[7]通过稀释(dilution)-地图匹配(map-matching)-编码(coding)这 3 个步骤实现了基于路网的轨迹压缩和编码;PRESS 系统[8]采用先地图匹配后压缩的策略达到相似的效果.这些技术不仅减少了数据量,还保留了移动对象所在的道路信息,因此逐渐被学者所关注.然而,从压缩效果和执行效率的角度看,这些方法还存在一些明显的不足.

- 首先,虽然基于路网的轨迹压缩技术都采用了地图匹配技术,但是现有的地图匹配技术都不是针对轨迹压缩而设计的,因此直接使用这些地图匹配技术将导致不合理的匹配,进而影响了压缩效果.比如车辆和行人的运动具有一定的随意性,即,人在行走或驾驶过程中可能会穿过一片草坪中的小径或者在远离道路的平地运动,这种情况下,若将这类轨迹匹配到道路上是不合理的.又比如,移动对象在真实的道路上运动,但是该道路尚未被收录到路网数据中,导致轨迹点被匹配到其他道路上;或者实际上相连的两条道路,在地图中并未表现为连通的.图 1 是不在地图道路上行驶的移动对象及其轨迹的真实案例.若不加针对性地处理,这些情况都将使地图匹配过程中断或者匹配结果不合理,进一步导致不良的压缩效果;
- 其次,现有的基于路网的轨迹压缩方法的压缩率不够好.比如 PRESS 系统,其采用的匹配轨迹压缩算法是时序数据压缩中的 BOPW 方法[20],压缩率存在很大的提升空间;
- 再次,现有的基于路网的轨迹压缩方法都是离线的算法,适合在后台执行.然而移动对象的轨迹通常在移动端产生,若将原始轨迹上传至服务端再进行压缩,则会在移动端产生大量的流量消耗和存储开销.因此,在移动端对轨迹进行压缩是更好的方法.

(a) 某专车穿过小区 (b) 某行人不在道路上移动 (c) 某专车有段时间沿着一条不在地图上的小路行驶

Fig.1 Trajectories of moving objects

图 1 车辆或行人的行驶轨迹

针对以上问题,本文提出了一种路网感知的在线轨迹压缩方法.首先,本文在综合考虑移动轨迹的特点和地图质量的基础上,针对轨迹压缩的需要,设计了一种距离有界的地图匹配算法;其次,基于 O'Rourke[21]算法,本文

设计了更高效的匹配轨迹压缩算法;再次,本文设计和实现了 ROADER(road-network aware and error-bounded trajectory compression)系统,它通过滑动窗口机制和在线地图加载技术,集成路网匹配、匹配轨迹压缩和未匹配轨迹压缩等算法,实现了在移动端的轨迹压缩;最后,我们通过实验验证了 ROADER 系统的执行效果和效率.实验说明,该系统在压缩率、误差和执行时间等方面均大幅优于同类算法或系统.

本文第 1 节介绍轨迹压缩领域的相关研究.第 2 节定义本文出现的相关概念,同时介绍本文的工作基础.第 3 节提出一种距离有界的地图匹配算法.第 4 节提出基于路网的轨迹压缩算法.基于上述算法,第 5 节设计并实现路网感知的在线轨迹压缩系统.第 6 节在真实数据集下,通过实验验证本文系统.第 7 节总结全文,并对后续研究进行初步探讨.

1 轨迹压缩相关研究

定位设备采集的轨迹通常由一系列轨迹点组成,每个轨迹点至少包括相应的位置信息和时间信息.轨迹压缩技术去除轨迹中冗余的轨迹点,保留轨迹中包含重要信息的点.为便于讨论,本节将这些轨迹压缩技术分为 3 类:(1) 线段简化算法,(2) 基于路网的轨迹压缩算法,(3) 其他轨迹压缩算法.

1.1 线段简化算法

线段简化的思想来自于计算几何,其目的是用粗粒度的折线代替细粒度的折线,且使后者到前者的最大距离小于用户给定的阈值.最优线段简化算法的时间复杂度为 $O(n^2)$[22],其中,n 为输入的轨迹点数.最优算法的时间复杂度较高,不适用于大规模数据的压缩.因此,许多工作致力于时间复杂度相对较低的次优算法,这些算法按其技术特点又可分为批处理算法、在线算法和单遍扫描算法.

1.1.1 批处理算法

批处理算法采用全局扫描的策略,需要在压缩开始之前加载所有的数据点.批处理算法可以是自顶向下或是自底向上的.在自顶向下的算法中,轨迹点序列被递归地划分为子序列,直到满足一定的停止条件.Douglas-Peucker 算法[10]是经典的自顶向下的算法,该算法将原始轨迹的起始点和终点连接起来,然后计算每个点到这条连线的距离,若某一轨迹点到连线的距离最大且超过设定的阈值,则以该点作为分割点将轨迹分为两个子序列,之后递归地进行上述过程,直到轨迹无需划分.Douglas-Peucker 算法的时间复杂度是 $O(n^2)$.Meratnia 提出的 TD- TR[9]算法是对 Douglas-Peucker 的改进,主要变化是用同步距离(SED)[23]代替了原算法采用的垂直距离(PED),其他计算过程都类似.自底向上的算法是对自顶向下算法的补充,典型的算法是 Theo Pavlidis[24],该算法开始时用 $n/2$ 个子轨迹段近似表示长度为 n 的原始轨迹,之后,该算法将误差最小的相邻子轨迹递归合并,直到满足停止条件.批处理算法具有很高的时间复杂度和空间复杂度,不适合于在线场景和资源有限的计算环境[11].

1.1.2 在线压缩算法

在线算法使用了窗口机制,只在窗口内进行全局扫描,因此不需要获得整个轨迹信息就可以进行轨迹压缩.文献[25]提出了滑动窗口(sliding-window)算法和开放窗口(opening-window)两种在线算法.这类算法首先建立一个窗口,然后对窗口内的轨迹进行压缩,之后对后续的轨迹数据重复该过程.这两种算法的时间复杂度为 $O(n^2)$.BQS[12]和 SQUISH-E[13]进一步优化开放窗算法:BQS[12]利用凸包优化技术,从窗口的轨迹中挑选出最多 8 个特殊点,以此加快算法执行速度;SQUISH-E[13]算法是开放窗口和自底向上算法的结合,该算法使用双向链表提高算法的效率.文献[26]提出将滑动窗口中偏移距离最大的轨迹点作为当前轨迹点能否被压缩的判据,以此提高轨迹的压缩时间和压缩效率.在线算法仍具有较高的时间复杂度和空间复杂度.

1.1.3 单遍扫描算法

单遍扫描算法对每个轨迹点仅扫描和处理一次,因而不需要窗口提前缓冲轨迹点.显然,单遍扫描算法具有线性的时间复杂度和 $O(1)$的空间复杂度.文献[14]中提出的算法将固定数目的连续数据点划分为区段,保留每个区段中第n个点或一个随机点.虽然这个过程很快,但该算法的误差没有上界.Reumann-Witkam[15]建立了一个平行于前两个数据点连线的长方形条块,并用一条线段表示该条块中的所有数据点.Reumann-Witkam 执行速度很快,但由于条块的方向受限于前两个数据点,很难包含更多的点,因而压缩率不佳.在 20 世纪 70 年代后期出现

了扇区相交(sector intersection,简称 SI)算法[16,17],制图学科中的 Sleeve 算法[18]也采用了与 SI 算法相同的思想,它们都是单遍扫描算法.本文作者近期提出的 OPERB 算法[11]也实现了误差有界的单遍扫描算法.需要指出的是:目前的这些单遍扫描算法[11,16−18]都只支持垂直距离,不支持同步距离.

1.2 基于路网的轨迹压缩

移动对象通常在路网上行驶,因此其移动经常受路网的限制.基于路网的轨迹压缩[7,8,19]保留移动对象在路网中的行驶路径,更适用于基于位置的服务等应用.通常,基于路网的轨迹压缩首先对轨迹进行地图匹配,将原始轨迹表示为路网中的路径及匹配轨迹点,然后分别对路径和匹配轨迹点进行重编码和压缩[27,28].

文献[7]通过稀释(dilution)-地图匹配(map-matching)-编码(coding)这 3 个步骤实现了基于路网的轨迹压缩和编码:首先,利用 DP 算法[10]对移动端的原始轨迹数据进行压缩,这一步称为稀释(dilution).压缩后的轨迹数据通过移动端传输到服务器端,减少了移动端的带宽消耗;第 2 步为地图匹配(map-matching),将 GPS 轨迹点映射到路网上,将原始轨迹用路径信息表示;最后一步是对路径进行编码(conding),文献提出了贪婪路径序列和最短路径序列两种路径编码方法.需要指出的是,该算法的匹配和重编码都不适合在移动端完成.此外,先压缩后匹配的做法,也使得其匹配距离没有上界并且压缩效果较差.

PRESS 系统[8]采用先匹配后压缩的策略,其压缩分为无损的路径信息重编码和有损的匹配轨迹点压缩.对于轨迹的路径部分,压缩系统首先对整个路网进行预处理,计算最短路径,使得任意两点之间都可用最短路径表示;同时,由于不同的移动轨迹含有许多重复的轨迹(路径)片段,因此将轨迹中出现频率高的路段或路径用轨迹模式表示,进而减少数据规模.对路径上的轨迹点数据,该系统进行了有损压缩.PRESS 输出的压缩点的时间信息都是原始轨迹点的采样时间.PRESS 系统对道路匹配距离没有限制,因此匹配结果中匹配距离是无界的.此外,其压缩率不够好,且不支持在线压缩,其压缩的效果和性能都有很大的提升空间.

COMPRESS 系统[19]是 PRESS 系统[8]的后续发展,主要变化是改变了匹配点的压缩算法.COMPRESS 系统采用了基于计算几何的有损压缩方法,在宽度为ε的管道内找到一条包含顶点数最少的折线.COMPRESS 系统继承了 PRESS 的地图匹配方法,因此其匹配结果中的匹配距离是无界的.此外,COMPRESS 输出的压缩点的时间信息不是原始轨迹点的采样时间,这点与本文方法以及 PRESS 系统都不同.

1.3 其他轨迹压缩

轨迹信息还可以表示为便于人们理解的信息,例如兴趣点(POI)、标志建筑或者路口等.例如:文献[29]利用兴趣点形成的最小包围盒(MBB)表示压缩后的轨迹,对压缩后的轨迹相似性进行度量,并对轨迹进行聚类.这类工作与本文的相关性不高,故不展开赘述.

2 基本概念及方法描述

本节先说明相关概念,然后介绍基于 HMM 的地图匹配方法和匹配轨迹压缩的基础算法(O'Rourke 算法).

2.1 术语定义

基于路网的轨迹压缩方法首先进行道路匹配,将轨迹点序列匹配到路网中,得到原始轨迹的匹配路径信息和映射在匹配路径上的匹配轨迹点序列.下面首先说明道路匹配的相关概念.

定义 1(轨迹点). 轨迹点 p 定义为 $p=(x,y,t)$,其中,x,y,t 分别表示轨迹点的经度、纬度及时间.

定义 2(轨迹). 轨迹 $\vec{T}=\{p_1,...,p_n\}$ 为按时间递增排序的轨迹点序列,对于任意 $1\leqslant i\leqslant j\leqslant n$,有 $p_i.t<p_j.t$.

定义 3 (路网). 路网 $G=(V,E)$是有向图,其中,V 是顶点 $v=(x,y)$的集合,E 是路段 $r=(v_s,v_e)$的集合.

定义 4 (路径). 路径 $R=\{r_1,\ldots,r_m\}$是连续的路段序列,对于任意 $1\leqslant i<m$,有 $r_i.v_e=r_{i+1}.v_s$.

定义 5 (匹配点). 匹配点 $\overline{\overline{p}}$ 是轨迹点 p 在路段 r 上的投影点,定义为 $\overline{\overline{p}}=(r,d,t)$,其中,r 为路段,d 为匹配点 $\overline{\overline{p}}$ 相对 $r.v_s$ 的距离,t 是轨迹点 p 的时间.

定义 6(匹配距离). 匹配距离是轨迹点 p 到路段 r 的距离,表示为 $ped(p,r)$.匹配距离也是 p 到匹配点 $\overline{\overline{p}}$ 的距

离,即, $ped(p,r)=|p\bar{\bar{p}}|$.

定义 7(匹配轨迹). 给定子轨迹 $\ddot{T}=\{p_s,...,p_k\}(s\leqslant k)$,其匹配轨迹是匹配点序列,表示为 $\bar{\bar{T}}=\{\bar{\bar{p}}_s,...,\bar{\bar{p}}_k\}$.匹配轨迹也可以表示为 $\bar{\bar{T}}=(R,\bar{\bar{S}})$,其中,$R$ 为匹配轨迹的路径, $\bar{\bar{S}}=\{(d_s,t_s),...,(d_k,t_k)\}$ 为路径 R 上的轨迹点序列.此处, d_i 为 $\bar{\bar{p}}_i$ 相对路径 R 起始点的道路距离($s\leqslant i\leqslant k$).

其次,基于路网的轨迹压缩方法进一步对匹配轨迹进行压缩.压缩算法的核心是寻找穿越轨迹点序列的穿越直线,其含义如下.

定义 8(穿越直线). 考虑(d,t)二维空间中的时序数据$\{(t_s,d_s),\ldots,(t_k,d_k)\}(s<k)$,其中,$t$ 是时间,d 是变量.若对于每个时刻 $t_i(s\leqslant i\leqslant k)$,变量 d 都有一个取值范围$[a_i,w_i]$,则该时序序列可表示为三元组序列$\{(t_s,a_s,w_s),\ldots,(t_k,a_k,w_k)\}$.一条穿越范围序列$\{(t_s,a_s,w_s),\ldots,(t_k,a_k,w_k)\}$的直线称为穿越直线,表示为 L_{sk}.

2.2 基于HMM的地图匹配方法

目前的基于路网的轨迹压缩算法[7,8]都先将原始轨迹匹配到道路上,并普遍采用了基于隐式马尔可夫模型(HMM)的地图匹配算法[30−32].给定轨迹 $\ddot{T}=\{p_s,p_{s+1},...,p_k\}$,基于 HMM 的地图匹配算法将路段作为隐状态,将轨迹点作为观测状态,最终通过观测状态序列推测隐状态序列.对于每个轨迹点,HMM 方法计算其发射概率和转移概率,然后在路网 G 中找到一条联合概率最大的路径 R.

HMM 发射概率 $P(p|r)$表示了在路段 r 上采集到移动对象轨迹点 p 的概率,即轨迹点 p 匹配到路段 r 的概率.文献[25]证明发射概率相对于轨迹点 p 到路段 r 的匹配距离 $ped(p,r)$服从正态分布.令σ_{ped}为距离 ped 的标准差,轨迹点 p 在道路 r 上的发射概率可定义为

$$P(p\mid r)=\frac{1}{\sqrt{2\pi}\sigma_{ped}}\mathrm{e}^{-0.5\times\left(\frac{\|ped\|}{\sigma_{ped}}\right)^2} \tag{1}$$

相应地,转移概率 $P(\bar{\bar{p}}_i.r,\bar{\bar{p}}_{i+1}.r)$ 表示移动对象从匹配路段 $\bar{\bar{p}}_i.r$ 移动到另一条匹配路段 $\bar{\bar{p}}_{i+1}.r$ 的概率.其中,从 $\bar{\bar{p}}_i$ 沿道路至 $\bar{\bar{p}}_{i+1}$ 的距离又称为道路转移距离(如图 2 所示).令 $\bar{\bar{p}}_i$ 和 $\bar{\bar{p}}_{i+1}$ 的道路转移距离与轨迹点 p_i 和 p_{i+1} 的欧氏距离之差为Δd,文献[31]证明,Δd 服从指数分布.由此,移动对象从道路 $\bar{\bar{p}}_i.r$ 至道路 $\bar{\bar{p}}_{i+1}.r$ 的转移概率定义为

$$P(\bar{\bar{p}}_i.r,\bar{\bar{p}}_{i+1}.r)=\frac{1}{\beta}\times\mathrm{e}^{-\frac{\Delta d}{\beta}} \tag{2}$$

其中,β是指数分布中的参数.

此外,每个轨迹点 p 都有多条候选匹配路段,因此,一条轨迹就存在多条候选匹配路径.对于子轨迹$\{p_s,\ldots,p_i\}$及其某条候选匹配路径 R_i,对应了一条马尔可夫链,且存在一个联合概率 $P(\{p_s,\ldots,p_i\}|R_i)$,定义为

$$P(\{p_s,...,p_i\}\mid R_i)=\begin{cases}P(\{p_s,...,p_{i-1}\}\mid R_{i-1})\times P(\bar{\bar{p}}_{i-1}.r,\bar{\bar{p}}_i.r)\times P(p_i\mid\bar{\bar{p}}_i.r), & i>s\\ P(p_i\mid\bar{\bar{p}}_i.r), & i=s\end{cases} \tag{3}$$

给定输入轨迹点序列 $\ddot{T}=\{p_s,p_{s+1},...,p_k\}$,HMM 地图匹配算法输出匹配轨迹 $\bar{\bar{T}}=(\bar{\bar{p}}_s,\bar{\bar{p}}_{s+1},...,\bar{\bar{p}}_k)$.算法得到每个轨迹点的所有可能的匹配路段,计算其概率,并维持一个集合 *RSet* 保存当前所有可能的匹配路径 R.当算法执行到轨迹点 $\bar{\bar{p}}_i$ 时,选择 $P(\{p_s,\ldots,p_{i-1}\}|R_{i-1})$与 $P(\bar{\bar{p}}_{i-1}.r,\bar{\bar{p}}_i.r)$ 乘积最大的路径加入其中,并更新联合概率 $P(\{p_s,\ldots,p_i\}|R_i)$和当前路径集 *RSET*.轨迹结束时,算法选择联合概率 $P(p_k|R_k)$最大的路径作为匹配路径.

例 1:图 3 为 HMM 地图匹配算法的示意图,图中 p_1~p_4 为待匹配的轨迹点,r_1~r_5 为路段.每个轨迹点对应的实心点是轨迹点的候选路段,且每个实心点都有对应的发射概率,即轨迹点与候选路段的匹配概率.每两个实心点之间有对应的转移概率,以描述移动对象从一条路段移动到另一条路段的可能性.

(1) 轨迹点 p_1 的候选路段为$\{r_1,r_3,r_5\}$,此时,算法将 p_1 的所有候选路段作为候选路径;

(2) 当轨迹点 p_2 的候选路段$\{r_3,r_5\}$加入候选路径后,此时,候选路径为$\{(r_1,r_3),(r_3),(r_5)\}$,且每条候选路径都可以计算其联合概率;

(3) 当最后一个轨迹点 p_4 的候选路段加入后,候选路径更新为$\{(r_1,r_3,r_4),(r_3,r_4),(r_5,r_4)\}$,算法最终选择其中

联合概率最大的路径作为轨迹(p_1,p_2,p_3,p_4)的匹配路径.

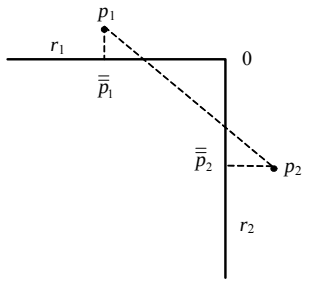


Fig.2 Transition distance
图 2 转移距离示意图

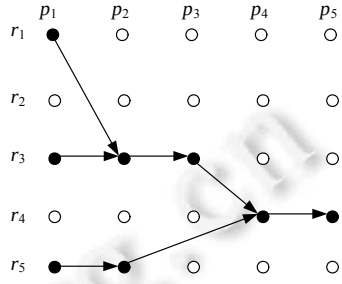


Fig.3 HMM map-matching
图 3 HMM 地图匹配示意图

2.3 O'Rourke算法

本文还采用了经典的 O'Rourke[21]算法,用于判断是否存在一条直线同时穿过一组连续的取值范围.

如图 4 所示,考虑由(d,t)构成的二维空间,O'Rourke 算法的目标是在(d,t)二维平面中找到一条穿越取值范围$\{(t_s,a_s,w_s),\ldots,(t_k,a_k,w_k)\}(s<k)$的穿越直线 L_{sk}.O'Rourke 算法通过半平面相交的方法寻找穿越直线.首先,对于第 i 个范围(t_i,a_i,w_i),穿过它的直线必定满足 $a_i \leqslant k\times t_i+b \leqslant w_i$.通过变形进一步得到由变量 k 和 b 构成的方程组:

$$\begin{cases} b \geqslant (-t_i)\times k + a_i \\ b \leqslant (-t_i)\times k + w_i \end{cases}.$$

该方程组在 k-b 空间下表示两个边界平行的半平面相交区域(如图 5 所示).

因此,若存在穿过范围序列$\{(t_s,a_s,w_s),\ldots,(t_k,a_k,w_k)\}$的直线,则这些半平面在 k-b 空间下存在共同的相交区域,且该区域为凸多边形;否则,这些半平面没有共同的相交区域.

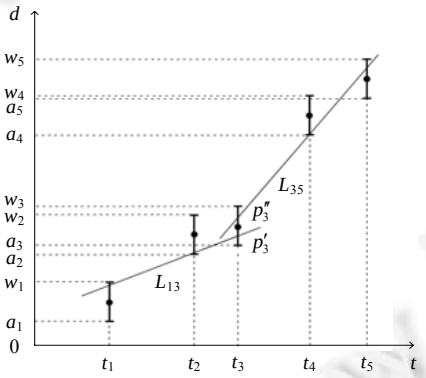


Fig.4 Passing lines in t-d space
图 4 t-d 空间下穿越直线求解示意图

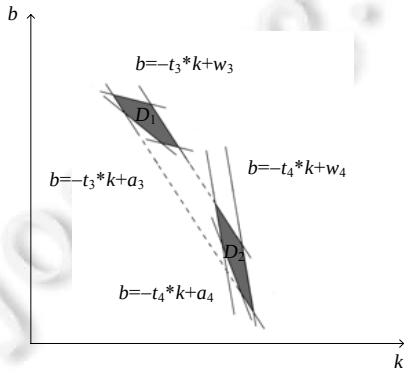


Fig.5 Half-Plane intersection in k-b space
图 5 k-b 空间下半平面相交示意图

给定范围序列$\{(t_1,a_1,w_1),\ldots,(t_n,a_n,w_n)\}$,O'Rourke 算法以$(t_1,a_1,w_1)$作为初始范围$(t_s,a_s,w_s)$,依次处理每个数据范围$(t_i,a_i,w_i)$,$1\leqslant i\leqslant n$,计算 k-b 空间下的相交区域 D,若 D 不为空,则继续下一个数据范围;否则,输出序列$\{(t_s,a_s,w_s),\ldots,(t_{i-1},a_{i-1},w_{i-1})\}$的穿越直线,并且以$(t_{i-1},a_{i-1},w_{i-1})$作为新的初始范围$(t_s,a_s,w_s)$,重复上述过程,直到所有的数据范围都处理完成.O'Rourke 算法的时间复杂度为 $O(n)$.

例 2:给定范围序列$\{(t_1,a_1,w_1),\ldots,(t_5,a_5,w_5)\}$,图 4 的 L_{13} 和 L_{35} 为该范围序列的穿越直线,图 5 为半平面交集求解过程.图 5 中,范围序列$\{(t_1,a_1,w_1),(t_2,a_2,w_2),(t_3,a_3,w_3)\}$的半平面在 k-b 空间下相交于 D_1,范围(t_4,a_4,w_4)在 k-b

空间下的半平面相交区域与 D_1 不重叠.因此,算法输出{$(t_1,a_1,w_1),(t_2,a_2,w_2),(t_3,a_3,w_3)$}的穿越直线 L_{13}.然后从(t_3,a_3,w_3)开始到序列结束,范围序列{$(t_3,a_3,w_3),(t_4,a_4,w_4),(t_5,a_5,w_5)$}在 k-b 空间下相交于 D_2.因此,算法输出{$(t_3,a_3,w_3),(t_4,a_4,w_4),(t_5,a_5,w_5)$}的穿越直线 L_{35}.注意:直线 L_{13} 和 L_{35} 在 t_3 时刻是不连续的.

3 距离有界的地图匹配方法

基于路网的轨迹压缩算法[7,8,19]首先将原始轨迹匹配到路网上,并且普遍采用了基于 HMM 的路网匹配算法[30−32].但这些 HMM 地图匹配算法都不是针对轨迹压缩而设计的,因而存在明显的缺陷.

(1) 没有考虑路网质量对匹配效果的影响,进而影响了压缩率;

(2) 对于最大匹配距离和移动对象的运动方向没有限制,导致了不合理的匹配,同样影响了压缩的效果和性能.

针对上述问题,本节首先分析了路网数据质量问题,然后提出了一种距离有界的在线地图匹配方法,通过限制最大匹配距离、将移动对象运动方向加入计算等方法,提高了匹配的效果,使其更适合于轨迹压缩.

3.1 路网质量的考虑

实际应用中,由于地图数据不精确或地图更新不及时等原因,通常会出现路网中相邻的道路不相连的情况.这种情况会影响到转移概率的计算,从而导致匹配结果不准确.如图 6 所示,在实际中,路段 r_2 和路段 r_3 是相连的,即,移动对象可以由路段 r_2 移动到路段 r_3,因此,轨迹点 p_i 的匹配路段为 r_2,轨迹点 p_{i+1} 的匹配路段为 r_3.然而,由于路网数据质量问题导致路段 r_2 和路段 r_3 不相连,使其不符合地理信息数据质量的逻辑一致性原则[33,34](在著名的 Openstreetmap 中,这种现象十分普遍.例如,文献[34]对法国两个不同区域的 Openstreetmap 进行评估,发现 68%被测试区域的路网结构不符合逻辑一致性.本文对 Openstreetmap 北京地图进行统计,由上述不连接造成的违背逻辑一致性的情况约占所有路段的 34%),造成不合理的转移概率和匹配结果.

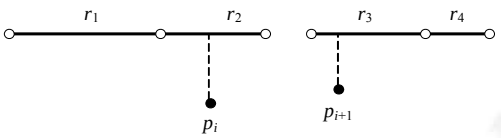


Fig.6 Connected road segments in reality

图 6 路段连通性示意图

对此,本文统计了某专车的北京轨迹,通过大量的轨迹来判断两个相近路段之间的连接性,以此为基础完善路网信息.统计结果表明:在本文使用的路网数据 OpenStreetMap 中,两条距离相近但不直接连接路段的平均距离为 48m,而其中的大部分实际上是存在通路的.

3.2 距离有界的路段选择

路网匹配时,需要从路网中挑选候选路段.传统的 HMM 地图匹配方法考虑路网中的所有路段,这会导致轨迹点到匹配路段的匹配距离没有上界.这样的匹配是极不合理的(如图 7 所示).针对这种情况,本文引入距离限制.给定距离阈值δ和轨迹点 p,本文匹配方法只考虑与 p 的距离小于δ的路段.若轨迹点的匹配范围内不存在匹配路段,此时我们可以认为地图中没有包含移动对象所在的路段数据,或者移动对象在一块没有路段的平地上行驶.对于没有匹配路段的轨迹点,本文采用保存原始轨迹点的策略.这种方法保证了原始轨迹点和匹配轨迹点的距离是有界的,并且提高了匹配轨迹与原始轨迹的相似性.

图 8 为由 9 个轨迹点组成的轨迹在路网上的示例图.圆圈半径表示距离阈值δ,轨迹点 p_1,p_2,p_3,p_7,p_8,p_9 在阈值范围内存在候选路段,轨迹点 p_1,p_2 的候选路段为{r_1},轨迹点 p_3 的候选路段为{r_2,r_3},轨迹点 p_7,p_8,p_9 的候选路段为{r_5}.而轨迹点 p_4,p_5,p_6 在阈值范围内没有候选路段,因此,算法保留原始轨迹点{p_4,p_5,p_6}.

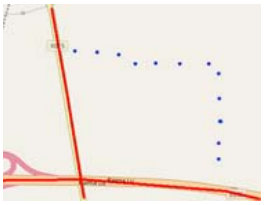


Fig.7 Unbounded map-matching

图 7 匹配距离无界示意图

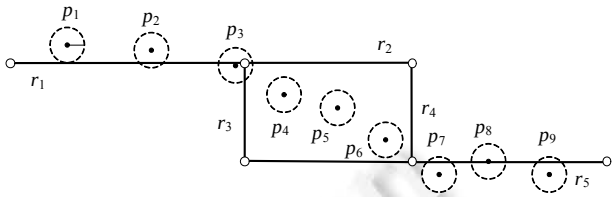


Fig.8 Bounded map-matching

图 8 距离有界的候选路段示意图

3.3 发射概率的计算

由于候选路段受到距离和行驶方向的影响,本文在公式(1)的基础上,将发射概率定义为

$$P = f(\alpha)\times\frac{1}{\sqrt{2\pi}\sigma_{ped}}\mathrm{e}^{-0.5\times\left(\frac{\|ped\|}{\sigma_{ped}}\right)^2} \tag{4}$$

其中,

$$f(\alpha)=\begin{cases}0, & |\alpha|-\theta>0\\ 1-\dfrac{|\alpha|}{\theta}, & |\alpha|-\theta\leqslant 0\end{cases}.$$

ped 为轨迹点到匹配道路的匹配距离,σ_{ped} 为 ped 的标准差;θ为角度常数,缺省值是π/4;α为轨迹点移动方向偏离路段方向的夹角(如图 9 所示),且$-\pi<\alpha<\pi$.本文只选取夹角绝对值小于θ的路段作为轨迹点的候选路段.

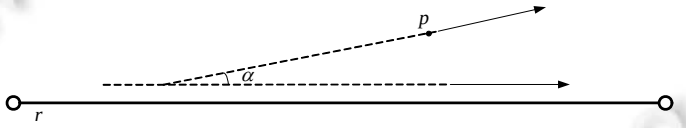


Fig.9 Direction of the moving object deviates from the direction of the road segment

图 9 轨迹点方向偏离路段规划方向的角度α

3.4 路径匹配算法

在上述方法的基础上,本文设计实现了在线地图匹配.在地图匹配过程中,当输入的轨迹点无法匹配到路网,从而导致匹配算法中断时,算法选择联合概率最大的匹配路径作为轨迹的最佳匹配路径.若轨迹点在匹配距离阈值内没有候选路段,则算法保存原始轨迹点.

算法 1 为本文地图匹配算法伪代码 *DistBoundedMapMatching*,算法输入为轨迹 $\ddot{T}=\{p_1,p_2,...,p_n\}$,输出为匹配子轨迹集 $\overline{\overline{TSet}}$ 和没有匹配路段的未匹配子轨迹集 $\overline{TSet}$.

算法 1. 地图匹配算法.

输入:轨迹 $\ddot{T}=\{p_1,p_2,...,p_n\}$ 及最大匹配距离δ;

输出:匹配子轨迹集 $\overline{\overline{TSet}}$ 和未匹配子轨迹集 $\overline{TSet}$.

1. $\overline{\overline{TSet}}=\varnothing;\overline{TSet}=\varnothing$;*Rset*=∅; //*RSet* 保存当前所有候选路径
2. **for** *i*=1 **to** *N* **do**
3. **if** *candRoads*(p_i)=∅ **then** //p_i 没有候选道路
4. **if** $\bar{T}=\varnothing$ **then**
5. $\overline{\overline{TSet}}\leftarrow\bar{\bar{T}}$;
6. $\bar{T}\leftarrow p_i$;

7. **if** $candRoads(p_i)\neq\varnothing$ **then**
8. **if** $RSet=\varnothing$ **then**
9. $RSet\leftarrow candRoads(p_i)$; $\overline{TSet}\leftarrow\overline{T}$;
10. **else for** $\overline{\overline{p}}_i$ **in** $candRoads(p_i)$
11. $\overline{\overline{T}}=RSet[\max(P(\{p_s,...,p_{i-1}\}\mid R_{i-1})\times P(\overline{\overline{p}}_{i-1}.r,\overline{\overline{p}}_i.r))]$;
12. $\overline{\overline{T}}\leftarrow\overline{\overline{p}}_i$; update $P(\{p_s,\ldots,p_i\}|R_i)$ and $RSet$; //更新 $\overline{\overline{TSet}}$ 的联合概率及 $RSet$
13. return $\overline{\overline{TSet}}$ & $\overline{TSet}$;

算法步骤解释如下:

- 第 1 行中,$RSet$ 保存当前所有的匹配路径;
- 第 2 行遍历轨迹点;
- 第 3 行中函数 $candRoads(p_i)$求轨迹点 p_i 的候选路段;
- 第 3 行~第 5 行表示若轨迹点 p_i 没有匹配路段,且当前未匹配子轨迹 $\overline{T}$ 为空时,则将当前匹配子轨迹 $\overline{\overline{T}}$ 放入匹配子轨迹集 $\overline{\overline{TSet}}$ 中;
- 第 6 行为将没有匹配路段的轨迹点 p_i 插入当前未匹配子轨迹 $\overline{T}$;
- 第 7 行,若轨迹点 p_i 有匹配路段,则继续往下执行;
- 在第 8 行、第 9 行,当 p_i 存在匹配路段时,若 $RSet$ 为空,则 p_i 为当前轨迹段的起点,将 p_i 的所有匹配路段作为当前的匹配路径集;
- 若 $RSet$ 不为空,算法 10 行遍历 p_i 所有的候选路段,并得到路段上的匹配轨迹点 $\overline{\overline{p}}_i$;
- 第 11 行,选择 $RSet$ 中 $P(\{p_s,...,p_{i-1}\}\mid R_{i-1})\times P(\overline{\overline{p}}_{i-1}.r,\overline{\overline{p}}_i.r)$ 值最大的路径作为当前匹配轨迹 $\overline{\overline{T}}$ 的路径;
- 第 12 行将 p_i 对应的匹配轨迹点 $\overline{\overline{p}}_i$ 插入 $\overline{\overline{T}}$,更新 $\overline{\overline{T}}$ 对应路径的联合概率 $P(\{p_s,\ldots,p_i\}|R_i)$及当前的 $Rset$;
- 第 13 行,算法最终返回 $\overline{\overline{TSet}}$ 和 $\overline{TSet}$.

记轨迹 $\ddot{T}$ 的长度为 n,每个轨迹点平均拥有 m 条候选路段,最长匹配子轨迹和最长未匹配子轨迹的长度分别为 k 和 w.在算法执行前,本文对路网进行划分网格预处理工作,其目的是在计算每个轨迹点的候选路段时,将轨迹点的坐标值转换为网格的索引 ID,并得到对应网格内的所有路段,即,获取每个轨迹点的候选路段时间的复杂度为 $O(1)$.当轨迹点的候选路段集为空时,轨迹点加入未匹配子轨迹,该操作的时间复杂度为 $O(1)$;当轨迹点的候选路段集不为空时,遍历所有候选路段并将其加入当前匹配路径,该过程的时间复杂度为 $O(m)$.因此,匹配算法总体的时间复杂度为 $O(n\times m)$.算法当前候选路径集 $RSet$ 需要 $O(km^2)$空间;函数 $candRoads(p_i)$的空间复杂度为 $O(m)$;当前匹配子轨迹和未匹配子轨迹的分别需要 $O(k)$和 $O(w)$空间,故算法总的空间复杂度为 $O(km^2)$.

4 误差有界的匹配轨迹压缩

本节在 O'Rourke[21]算法的基础上给出了匹配轨迹的一种误差有界的高效压缩方法,该方法通过回溯的方式解决了直接采用 O'Rourke 算法压缩轨迹所导致的线段不连续的问题.

4.1 问题描述

匹配轨迹 $\overline{\overline{T}}=\{\overline{\overline{p}}_s,...,\overline{\overline{p}}_k\}$ 可以等价地表示为 $\overline{\overline{T}}=(R,\overline{\overline{S}})$, 其中,$R$ 为匹配轨迹的路径, $\overline{\overline{S}}=\{(d_s,t_s),...,(d_k,t_k)\}$ 为路径 R 上的轨迹点序列,并且 d_i 为 $\overline{\overline{p}}_i$ 相对路径 R 起点的道路距离.由此,R 可以采用更紧凑的编码方式,从而达到无损压缩的目的(关于 R 的编码见文献[7,28],不在本文的考虑范围).而 $\overline{\overline{S}}$ 是时序数据[20,35,36],可以被进一步压缩.本文在算法 O'Rourke[21]的基础上,设计了一种误差有界的压缩算法.相比 PRESS 所采用的压缩算法[8],O'Rourke 算法可以用一条线段表示更多的轨迹点,因而具有潜在的更好的压缩率.然而,直接采用 O'Rourke 算法又将导致线段不连续的问题,如图 4 所示:穿越直线 L_{13} 与穿越直线 L_{35} 在 t_3 时刻与范围(t_3,a_3,w_3)相交于不同的两点,导致在 t_3 时刻 L_{13} 与 L_{35} 不连续.这种情况一方面影响压缩效果,另一方面,这种不连续意味着需要保存更

多的信息,比如对于图 4 的 t_3 时刻,需要保存 p_3' 和 p_3'' 两个点的信息,这不利于提高压缩率.针对这个问题,本文提出一种保证压缩结果连续的压缩方法.

4.2 算法原理

给定轨迹点序列 $\overline{\overline{S}}=\{(d_s,t_s),...,(d_k,t_k)\}$ 及误差阈值 ε, $\overline{\overline{S}}$ 可以表示为范围序列 $\{(t_s,d_s-\varepsilon,d_s+\varepsilon),\ldots,(t_k,d_k-\varepsilon,d_k+\varepsilon)\}$.算法分两个步骤:首先,采用 O'Rourke 算法计算每个最长的子范围序列以及该序列的穿越直线集;其次,逆向确定唯一的穿越直线序列.

(1) 计算每个最长子范围序列,并得到每个最长子范围序列的穿越直线集.

算法以 $(t_s,d_s-\varepsilon,d_s+\varepsilon)$ 作为初始范围,计算过程如下.

a) 利用 O'Rourke 算法计算能同时被穿越直线穿过的最长子范围序列,$\{(t_i,d_i-\varepsilon,d_i+\varepsilon),\ldots,(t_j,\ d_j-\varepsilon,d_j+\varepsilon)\}$($s\leqslant i\leqslant j\leqslant k$),得到该范围序列内的穿越直线集 L_{ij},表现为 O'Rourke 算法中 *k-d* 空间下的一个凸多边形.此时,(d_j,t_j) 为子范围序列的端点,称为断点;

b) 计算穿越直线集 L_{ij} 中斜率最大与最小的直线,分别得到其在范围 $(t_j,d_j-\varepsilon,d_j+\varepsilon)$ 内的交点 (a_j,t_j) 和 (w_j,t_j).将 (t_j,a_j,w_j) 作为 t_j 时刻新的范围,并将它作为后续计算的初始范围;

c) 重复上述过程 a)和过程 b),直到处理完所有轨迹点.此时得到 t_n 时刻新的范围 (t_n,a_n,w_n).

例 3:图 10 是在 *t-d* 空间和 *k-b* 空间下,匹配轨迹 $\overline{\overline{T}}=\{\overline{\overline{p}}_1,\overline{\overline{p}}_2,\overline{\overline{p}}_3,\overline{\overline{p}}_4,\overline{\overline{p}}_5\}$ ($\overline{\overline{T}}$ 还可表示为 $\overline{\overline{T}}=(R,\overline{\overline{S}})$)在压缩误差 ε 下的压缩过程示意图.图 10(a1)中,$\{\overline{\overline{p}}_1,\overline{\overline{p}}_2,\overline{\overline{p}}_3\}$ 在取值范围内存在穿越直线,$\{\overline{\overline{p}}_1,\overline{\overline{p}}_2,\overline{\overline{p}}_3,\overline{\overline{p}}_4\}$ 在误差范围内不存在穿越直线.在 $\{\overline{\overline{p}}_1,\overline{\overline{p}}_2,\overline{\overline{p}}_3\}$ 所有的穿越直线中,斜率最大和最小的穿越直线与 $\overline{\overline{p}}_3$ 点处的误差范围 $(t_3,d_3-\varepsilon,d_3+\varepsilon)$ 交于 (a_3,t_3) 和 (w_3,t_3) 两点.该过程在 *k-b* 空间下对应于图 10(b1),$\{\overline{\overline{p}}_1,\overline{\overline{p}}_2,\overline{\overline{p}}_3\}$ 对应的平行直线形成凸多边形 D_1,$\overline{\overline{p}}_4$ 点对应的两条平行直线 $b=-t_4\times k+d_4-\varepsilon$ 和 $b=-t_4\times k+d_4+\varepsilon$ 与 D_1 不相交.因此,算法保存当前的凸多边形 D_1,并由凸多边形 D_1 可得到 $\overline{\overline{p}}_3$ 点的新的取值范围 (t_3,a_3,w_3).

图 10(a2)中,从 $\overline{\overline{p}}_3$ 点以新的误差范围 (t_3,a_3,w_3) 作为 t_3 时刻的范围,重新开始计算 $\overline{\overline{p}}_3,\overline{\overline{p}}_4$ 及其后面的点的误差范围内是否存在共同的穿越直线,直到该轨迹在点 $\overline{\overline{p}}_5$ 结束时,$\{\overline{\overline{p}}_3,\overline{\overline{p}}_4,\overline{\overline{p}}_5\}$ 的穿越直线中,斜率最大和最小的穿越直线在 $\overline{\overline{p}}_5$ 点处的误差范围 $(t_5,d_5-\varepsilon,d_5+\varepsilon)$ 交于 (a_5,t_5) 和 (w_5,t_5) 两点,此时,t_5 时刻的范围更新为 (t_5,a_5,w_5).该过程在 *k-b* 空间下对应于图 10(b2),$\{\overline{\overline{p}}_3,\overline{\overline{p}}_4,\overline{\overline{p}}_5\}$ 误差范围内的穿越直线形成凸多边形 D_2,沿 D_2 以斜率 $-t_5$ 的 2 条直线与 *b* 轴相交于 a_5 和 w_5.

(2) 逆向二次更新断点处的范围,确定唯一的穿越直线序列.

令压缩后的匹配轨迹为 $\overline{\overline{S}}'=\{(d_s',t_s'),...,(d_m',t_m')\}$,计算过程如下:

a) 二次更新 t_m' 时刻的范围 $(t_m',a_m',w_m')=(t_m',a_m,w_m)$,确定 $\overline{\overline{S}}'$ 中最后一个压缩轨迹点:

$$(d_m',t_m'),d_m'=(a_m'+w_m')/2,t_m'=t_k;$$

b) 设 $t_{m-1}'=t_f$,且过 (a_m',t_m') 和 (w_m',t_m') 的穿越直线在 t_f 范围内交于 (a_f',t_f) 和 (w_f',t_f) 两点.二次更新 t_{m-1}' 时刻的新范围为 (t_{m-1}',a_f',w_f'),确定压缩轨迹点 (d_{m-1}',t_{m-1}'),其中,

$$d_{m-1}'=(a_f'+w_f')/2,t_{m-1}'=t_f;$$

c) 重复 b)过程,依次确定 $\overline{\overline{S}}'$ 中后续的值,直到确定 $\overline{\overline{S}}'$ 的起点 (d_s',t_s').

例 4:图 11 为在图 10 的基础上,确定唯一穿越直线序列并得到压缩匹配轨迹 $\overline{\overline{S}}'$ 的过程.

图 11(a1)中,二次更新 t_5 时刻的范围 $(t_5,a_5',w_5')=(t_5,a_5,w_5)$,确定 $\overline{\overline{S}}'$ 中最后一个轨迹点 $\overline{\overline{p}}_5'=(d_5',t_5)$,其中,$d_5'=(a_5'+w_5')/2$.过 (a_5',t_5) 和 (w_5',t_5) 的穿越直线在断点 $\overline{\overline{p}}_3$ 的范围内交于 (w_3',t_3) 和 (a_3',t_3) 两点.二次更新 t_3 时刻的新范围为 (t_3,a_3',w_3'),确定压缩轨迹点 $\overline{\overline{p}}_3'=(d_3',t_3)$,其中,$d_3'=(a_3'+w_3')/2$.该过程在 *k-b* 空间下对应于图 11(b1),沿 D_2 以斜率 $-t_5$ 的 2 条直线与 *b* 轴相交于 a_5' 和 w_5';沿 D_2 以斜率 $-t_3$ 的 2 条直线与 *b* 轴相交于 a_3' 和 w_3'.d_5' 和 d_3' 分别取 a_5' 和 w_5' 的中点和 a_3' 与 w_3' 的中点.

在图 11(a2)中,过 (w_3',t_3) 和 (a_3',t_3) 的穿越直线在断点 $\overline{\overline{p}}_1$ 的范围内交于 (a_1',t_1) 和 (w_1',t_1) 两点.二次更新 t_1 时刻的新范围为 (t_1,a_1',w_1'), 确定压缩轨迹点 $\overline{\overline{p}}_1'=(d_1',t_1)$,其中, $d_1'=(a_1'+w_1')/2$.该过程在 *k*-*b* 空间下对应于图 11(b2),沿 D_2 以斜率 $-t_1$ 的 2 条直线与 *b* 轴相交于 a_1' 和 w_1',d_1' 取 a_1' 与 w_1' 的中点.

最终,压缩后的匹配轨迹为 $\overline{\overline{T}}'=\{R,\overline{\overline{S}}'\}$, 其中, $\overline{\overline{S}}'=\{\overline{\overline{p}}_1',\overline{\overline{p}}_3',\overline{\overline{p}}_5'\}$.

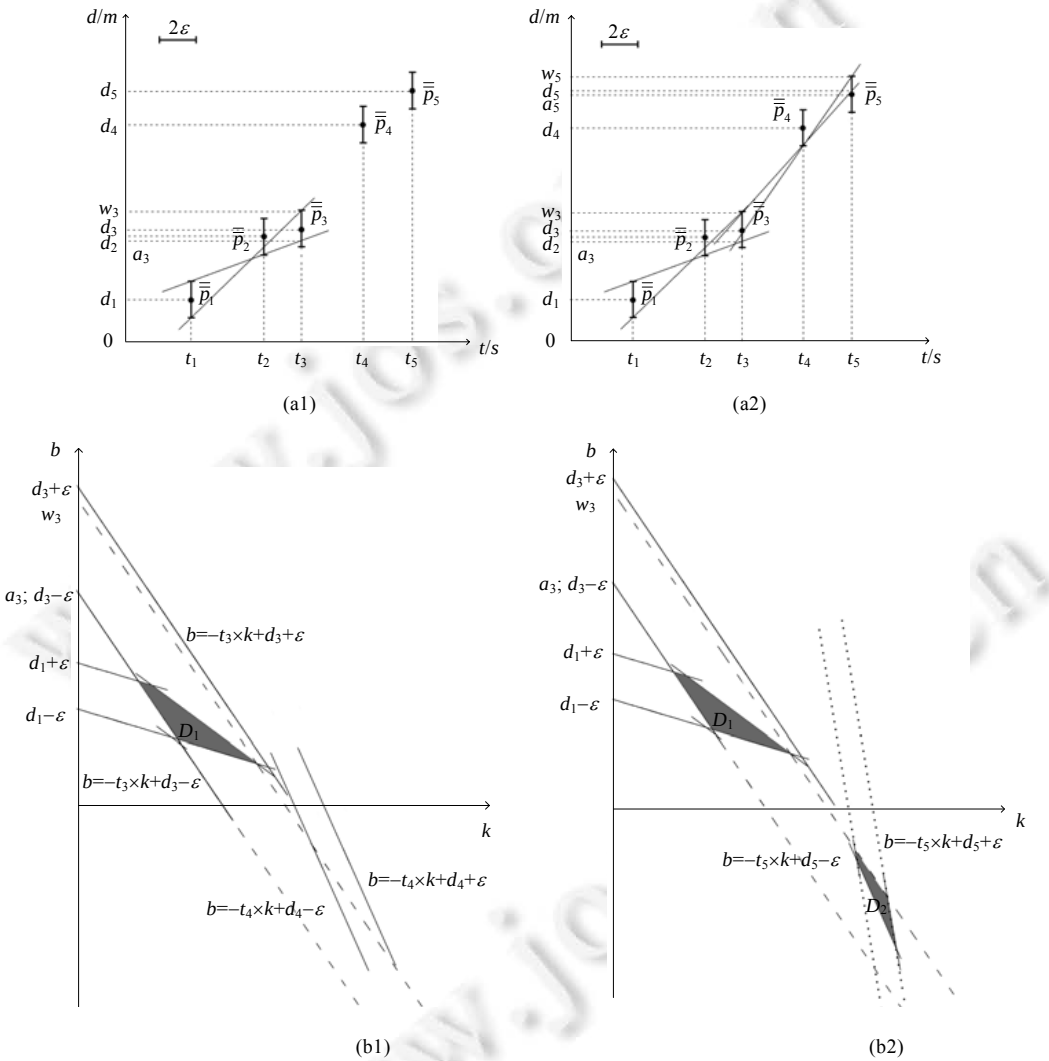


Fig.10 Find the potential passing lines by algorithmO'Rourke

图 10 利用 O'Rourke 算法压缩匹配轨迹

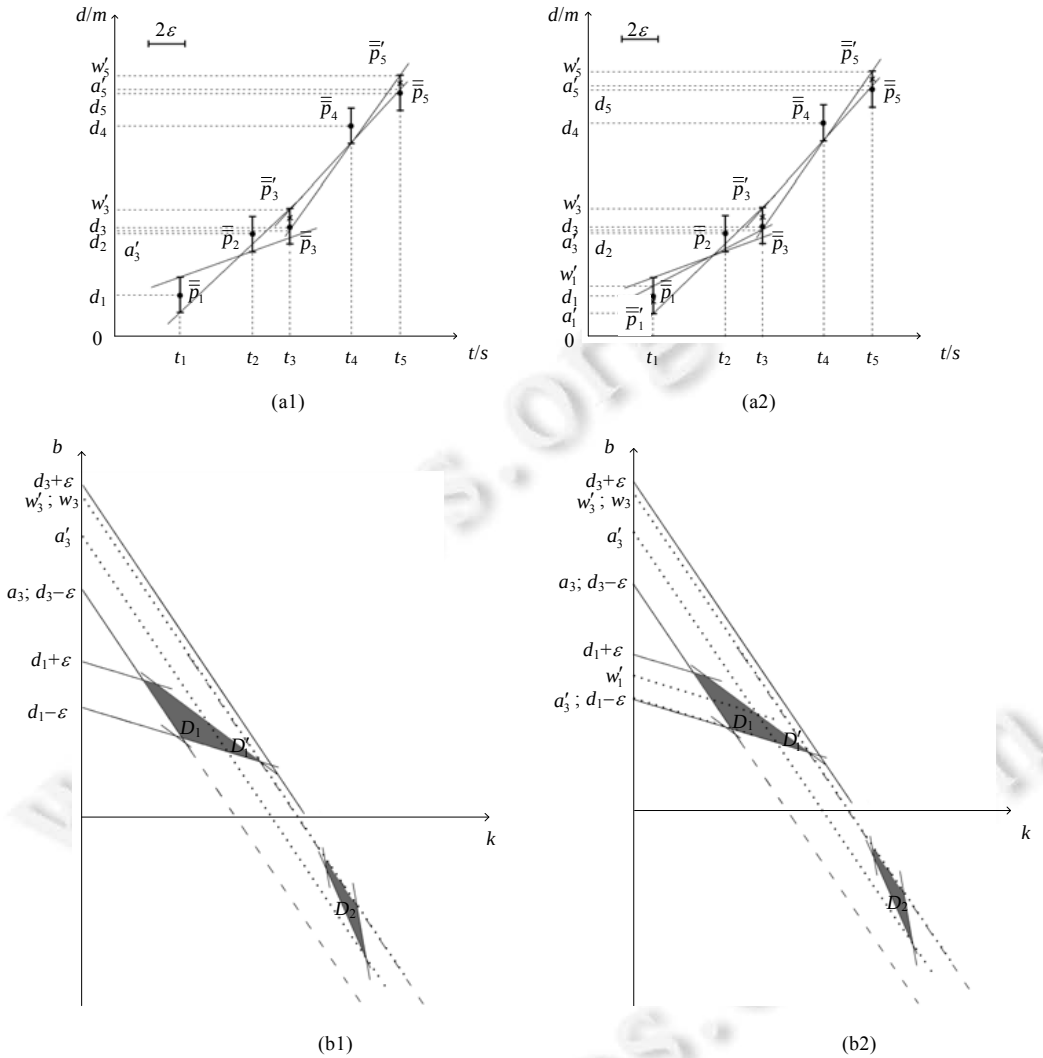


Fig.11 Determine the passing lines

图 11 逆向确定穿越直线

4.3 算法实现

算法 2 为匹配轨迹压缩算法 *RoadBasedCompression*.算法的输入为匹配轨迹 $\overline{\overline{T}}$ 及误差阈值ε,输出为压缩后的匹配轨迹 $\overline{\overline{T}}'$.

算法 2. 匹配轨迹压缩算法.

输入:匹配轨迹 $\overline{\overline{T}}=\{\overline{\overline{p}}_s,...,\overline{\overline{p}}_k\}$ 及误差阈值ε;

输出:压缩后的匹配轨迹 $\overline{\overline{T}}'=\{\overline{\overline{p}}'_s,...,\overline{\overline{p}}'_m\}$.

1. *polygonList*=∅; *index*=*s*; $\overline{\overline{S}}=\{(d_s,t_s),...,(d_k,t_k)\}$ //*index* 指向当前断点的位置
2. **for** *i*=*s* **to** *k* **do**
3. **if** *intersection*(*polygon*, $\overline{\overline{p}}_i$)=∅||*i*=*k* **then** //不存在穿越直线或者轨迹结束
4. *polygonList*←*polygon*; //将当前凸多边形加入 *polygonList*

5. (a_{i-1},w_{i-1})=*updateRange*(t_{i-1});
6. **if** $i<n$ **then** *index*←(−−*i*) //更新 *index*
7. **if** *intersection*(*polygon*, $\overline{\overline{p}}_i$)≠∅ **then** //存在穿越直线
8. *polygon*($\overline{\overline{p}}_i$); //更新加入 $\overline{\overline{p}}_i$ 后的凸多边形
9. **for** *j*=*polygonList.size* **to** 1 **do**
10. (a'_j,w'_j) =*updateRange*(*polygonList*[*j*]); $d'_j=(a'_j+w'_j)/2$; //更新误差范围
11. $\overline{\overline{S}}' \leftarrow (d'_j,t_j)$;
12. return $\overline{\overline{T}}'=\overline{\overline{S}}'$;

算法步骤解释如下:

- 第 1 行中,*polygonList* 为凸多边形的列表,保存当前所有的凸多边形,*index* 指向当前断点的位置,初始时,使其指向第 1 个轨迹点.$\overline{\overline{S}}$ 为 $\overline{\overline{T}}$ 对应的匹配路径轨迹;
- 第 2 行遍历轨迹点序列;
- 第 3 行~第 6 行表示若第 *i* 个轨迹点与当前的凸多边形无交点,即不存在穿越直线,则将当前凸多边形加入 *polygonList* 中,并更新断点时刻的范围;
- 否则,在第 8 行更新当前的凸多边形 *polygon*;
- 第 9 行倒序遍历凸多边形列表 *polygonList*;
- 第 10 行和第 11 行重新更新每个断点的凸多边形,二次更新断点的范围 (t_j,a'_j,w'_j),则得到压缩轨迹点 (d'_j,t_j), 其中, d'_j 取 a'_j 和 w'_j 的中值,并将压缩轨迹点 (d'_j,t_j) 加入压缩轨迹 $\overline{\overline{S}}'$ 中;
- 第 12 行将 $\overline{\overline{S}}'$ 构造为 $\overline{\overline{T}}'$,返回 $\overline{\overline{T}}'$.

记匹配轨迹 $\overline{\overline{T}}$ 的长度为 *n*,压缩后匹配轨迹 $\overline{\overline{T}}'$ 的长度为 *m*,算法第 1 步采用 O'Rourke 算法得到每个最长子范围序列的穿越直线集,这个过程的时间复杂度为 $O(n)$;第 2 步逆向确定唯一穿越直线序列的时间复杂度为 $O(m)$,故匹配轨迹压缩算法的时间复杂度为 $O(n+m)$.在第 1 步中需要保存 *k*-*b* 空间下每个断点对应的凸多边形,该算法的空间复杂度为 $O(m)$.

5 路网感知的轨迹压缩系统

基于上文提出的地图匹配算法和轨迹压缩算法,本节设计并实现了路网感知的在线轨迹压缩系统.该系统可在移动端对轨迹进行在线压缩,不仅减少了移动端上传轨迹的流量消耗,也节省了服务端的存储空间.

5.1 系统结构与工作原理

路网感知的轨迹压缩系统的组成结构如图 12 所示.该系统由地图加载(map loading)、地图匹配(map matching)、匹配轨迹压缩(road-based simplification)和未匹配轨迹压缩(piece-wise line simplification)这 4 个部分组成.

- 地图加载:由于路网感知的轨迹压缩系统运行在客户端,因此客户端需存在相应地图. ROADER 系统允许客户端离线下载指定地图,也可以在压缩的过程中在线下载移动轨迹区域的地图. ROADER 对地图进行网格划分,以每个网格作为地图加载的最小单位.离线下载地图时,地图加载模块按照指定的区域范围,将区域中所有网格内的地图加载到客户端.当客户端存储量较小时,可以选择在线加载移动轨迹所在的区域,若客户端没有轨迹所在区域的地图,则将轨迹所在的网格以及周围的 8 个网格内的地图加载到客户端.在线加载地图按需下载部分地图数据,节省了客户端的流量和存储量;
- 地图匹配:该模块实现了距离有界的地图匹配.为使匹配结果更合理,在匹配过程中,轨迹点选择匹配距离阈值δ内的路段作为匹配路段.若轨迹点在阈值δ内无候选路段,则保留原始的轨迹点.地图匹配的输出是匹配轨迹和未匹配的轨迹;
- 匹配轨迹压缩:该模块对地图匹配结果中的匹配轨迹进行压缩,且保证压缩误差小于给定的阈值ε.匹

配轨迹压缩算法在提高压缩率的同时,也保证压缩结果折线是连续的;

• 未匹配轨迹压缩:该模块利用基于同步距离的线段简化算法 TD-TR[9]对地图匹配结果中未匹配的轨迹进行压缩,压缩阈值大小为 $\sqrt{\delta^2+\varepsilon^2}$.线段简化算法实现原理简单、压缩率高.

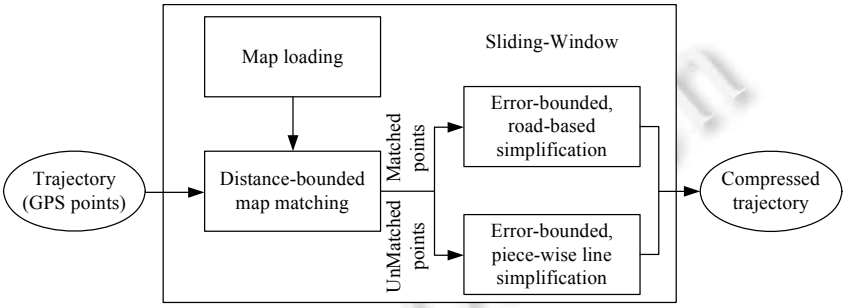


Fig.12 Framework of ROADER

图 12 ROADER 系统结构图

5.2 算法实现

算法 3 *RoadAwareCompression* 是 ROADER 主算法.

算法 3. 路网感知的轨迹压缩算法.

输入:轨迹 $\ddot{T}=\{p_0,p_1,...,p_n\}$,窗口大小 *WSIZE*,最大匹配距离 δ 和最大匹配轨迹误差 ε;

输出:压缩后的匹配子轨迹集 $\overline{\overline{TSet}}'$ 及压缩后的未匹配子轨迹集 $\overline{TSet}'$.

1. While (having points in $\ddot{T}$) do
2. T=*LoadSubTraj*($\ddot{T}$,*WSIZE*); G=*LoadMap*(T); //加载轨迹至窗口及窗口内轨迹的地图
3. ($\overline{\overline{TSet}}'$,$\overline{TSet}'$)=*DistBoundedMapMatching*(T,G,δ); //得到匹配轨迹 $\overline{\overline{TSet}}$ 和未匹配轨迹 $\overline{TSet}$
4. $\overline{\overline{TSet}}'$ =*RoadBasedCompression*($\overline{\overline{TSet}}$,ε); //对 $\overline{\overline{TSet}}$ 执行基于路网的压缩
5. $\overline{TSet}'$ =*LineSimplification*($\overline{TSet}$, $\sqrt{\delta^2+\varepsilon^2}$); //对 $\overline{TSet}$ 执行基于线段的压缩
6. output $\overline{\overline{TSet}}'$ and $\overline{TSet}'$ //输出数据

算法过程如下.

(1) 加载子轨迹至窗口(*LoadSubTraj*),并读取窗口内轨迹的路段信息(*LoadMap*);

(2) 执行距离有界地图匹配(*DistBoundedMapMatching*),匹配结果为匹配子轨迹集 $\overline{\overline{TSet}}$ 和未匹配子轨迹集 $\overline{TSet}$;

(3) 对匹配结果中的每个子轨迹分别进行误差有界的轨迹压缩并输出压缩结果:对 $\overline{\overline{TSet}}$ 中的每条匹配子轨迹 $\overline{\overline{T}}$ 执行基于路网的轨迹压缩(*RoadBasedCompression*);对 $\overline{TSet}$ 中的每条未匹配子轨迹 $\overline{T}$ 执行基于线段的轨迹压缩(*LineSimplification*);

(4) 算法重复以上过程,直到所有的轨迹点都处理完成.

令轨迹 $\ddot{T}$ 的长度为 n,每个轨迹点平均拥有 m 条候选路段,轨迹 $\ddot{T}$ 中匹配轨迹 $\overline{\overline{T}}$ 所占比例为 γ,其中,最长匹配子轨迹的长度为 k,则 ROADER 系统加载轨迹(*LoadSubTraj*)过程的时间复杂度为 $O(n)$;地图加载(*LoadMap*)的时间复杂度为 $O(|G|)$,$|G|$为地图大小;距离有界的地图匹配(*DistBoundedMapMatching*)时间复杂度为 $O(mn)$;匹配轨迹基于路网的轨迹压缩过程的时间复杂度为 $O(\gamma n)$;未匹配轨迹基于线段的压缩算法 TD-TR[9]的时间复杂度为 $O((1-\gamma)^2n^2)$.故 ROADER 系统总时间复杂度为 $O((1-\gamma)^2n^2+mn+|G|)$.对于算法的空间复杂度,算法的空间主要用于地图加载和地图匹配,复杂度为 $O(|G|+km^2)$.

例 5:图 13 是长度为 10 的轨迹 $\ddot{T}=\{p_1,...,p_{10}\}$ 的压缩示例,虚线表示路网中没被收录的小径,路段 r_1~

r_4 为路网中的路段.移动对象的轨迹是沿着路段 r_1 运动到图中虚线表示的小径,再运动到路段 r_3 和路段 r_4.

(1) ROADER 首先加载轨迹 $\ddot{T}=\{p_1,...,p_{10}\}$,然后加载轨迹所在的区域的地图;

(2) ROADER 对窗口内的轨迹进行地图匹配.给定最大匹配距离 δ,轨迹点选择最大匹配范围内的路段作为候选路段,若在最大范围内,轨迹点没有匹配路段,匹配算法保留其原始轨迹点.图 13 中的虚线圆代表最大匹配距离的范围,这种情况下,地图匹配的输出为匹配子轨迹序列 $\overline{\overline{TSet}}=\{\overline{\overline{T}}_1,\overline{\overline{T}}_2\}$ (其中,$\overline{\overline{T}}_1=\{\overline{\overline{p}}_1\overline{\overline{p}}_2\},\overline{\overline{T}}_2=\{\overline{\overline{p}}_7,\overline{\overline{p}}_8,\overline{\overline{p}}_9,\overline{\overline{p}}_{10}\}$)以及未匹配子轨迹序列 $\overline{TSet}=\{\overline{T}\}$ (其中,$\overline{T}=\{\overline{p}_3,\overline{p}_4,\overline{p}_5,\overline{p}_6\}$).注意:匹配子轨迹 $\overline{\overline{T}}_1$ 和 $\overline{\overline{T}}_2$ 还可以表示为

$$\overline{\overline{T}}_1=\{\{r_1\},\{(d_1,t_1),(d_2,t_2)\}\},\overline{\overline{T}}_2=\{\{r_3,r_4\},\{(d_7,t_7),(d_8,t_8),(d_9,t_9),(d_{10},t_{10})\}\};$$

(3) 分别对匹配轨迹和未匹配轨迹进行轨迹压缩.由于 $\overline{\overline{T}}_1$ 中只包含 2 个点,因此压缩后的轨迹仍保留起点和终点两个点,故压缩后 $\overline{\overline{T}}'_1=\{\{r_1\},\{(d'_1,t_1),(d'_2,t_2)\}\}$,$d'_1$ 和 d'_2 分别表示压缩后的轨迹点距离路径 $\{r_1\}$ 起点的距离.$\overline{\overline{T}}_2$ 被压缩为 (d'_7,t_7) 和 (d'_{10},t_{10}) 两个点,即 $\overline{\overline{T}}'_2=\{\{r_3,r_4\},\{(d'_7,t_7),(d'_{10},t_{10})\}\}$,$d'_7$ 和 d'_{10} 分别表示对应压缩后的轨迹点距离路径 $\{r_3,r_4\}$ 起点的距离.未匹配子轨迹 $\overline{T}$ 由基于同步距离的 DP 算法压缩为 $\overline{T}'=\{\overline{p}_3,\overline{p}_5,\overline{p}_6\}$;

(4) 最终的输出结果为 $\overline{\overline{TSet}}=\{\overline{\overline{T}}'_1,\overline{\overline{T}}'_2\}$ 及 $\overline{TSet}=\{\overline{T}'\}$.

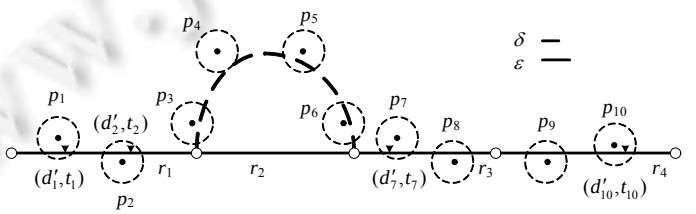


Fig.13 RoadAware compression of trajectory $\ddot{T}$

图 13 轨迹 $\ddot{T}$ 及其压缩结果示意图

6 实 验

6.1 实验设置

本文使用了某专车的北京市数据集和 Geolife 数据集.其中,从 Geolife 数据集中拆出了 Geolife 北京市数据子集和 Geolife 华盛顿州数据子集.

- 专车北京市数据:选取某专车公司在北京市 100 多辆车 2 天的行驶轨迹,共包含 719 816 个轨迹点.该数据集采集频率为:62.4%为 6s;15%为 5s;10.9%为 7s;3.46%为 8s;8.24%为其他.其中,最小的频率为 1s,占 0.0000347%;
- Geolife 数据:Geolife 北京市数据子集选取自 Geolife 数据集的 500 多条北京市的轨迹,共 876 198 个轨迹点;Geolife 华盛顿州数据子集选取自 Geolife 数据集的 79 条美国华盛顿地区的轨迹,共包含 245 168 个轨迹点.Geolife 数据集采样频率(按比例从高到低)为:74.23 为 5s;7.71%为 2s;7.44%为 1s;7.41%为 3s;2.17%为 4s;1.04%为其他;其中最小的频率为 1s,占 7.44%.

本文对比了两个路网感知的轨迹压缩系统,即本文系统 ROADER 和 PRESS 系统[8].PRESS 的地图匹配部分没有公开算法和源代码,只提供了".exe"可执行文件,故在实验中,ROADER 系统和 PRESS 系统的执行时间实验部分运行在 Windows 系统下.除此之外,PRESS 系统的压缩算法和 ROADER 系统的全部都运行在 linux 中.实验计算机的硬件配置为 32 Intel(R) Xeon(R) CPU E5-2650 0@2.00GHz,内存大小为 256G.

6.2 实验结果

本文进行了 5 组实验,前 4 组实验分别对比了 ROADER 和 PRESS 两个系统:(1) 压缩率;(2) 匹配距离;(3) 压缩误差;(4) 执行效率;第 5 组实验分析了移动端路网加载的效果.

6.2.1 压缩率

轨迹的压缩率(compression ratio,简称 CR)是衡量轨迹压缩效果的重要指标,本文将压缩率定义为

$$CR=\frac{M_{T'}}{M_{\bar{T}}}\times 100\%,$$

其中,$M_{T'}$ 为压缩后轨迹的存储量,$M_{\bar{T}}$ 为原始轨迹存储量.

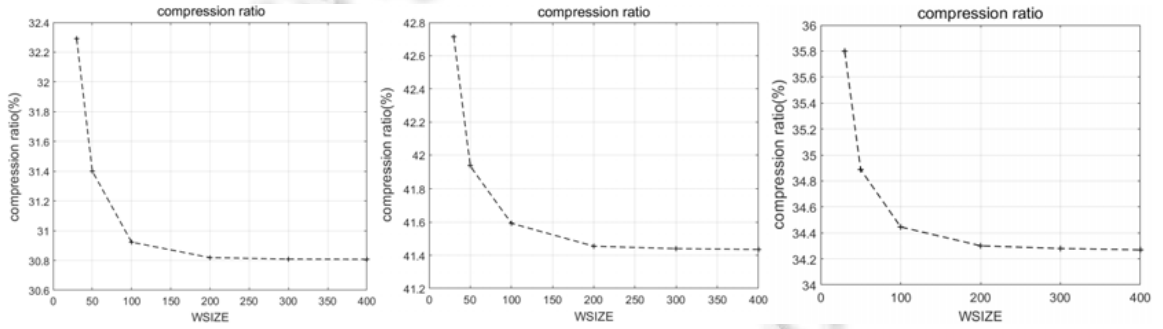
- 实验 A.1:窗口大小对 ROADER 压缩率的影响

为验证滑动窗口大小 *WSIZE* 对 ROADER 压缩率 *CR* 的影响,本文固定匹配距离阈值δ=50m,压缩误差ε=50m,将 *WSIZE* 从 30 增加至 400.图 14 为 3 个数据集的实验结果,从中可以发现:

(1) 轨迹压缩率 *CR* 随窗口 *WSIZE* 的增大而降低.由于随着窗口的增大,地图匹配会对匹配路径不断修正,而匹配准确性的提高有利于基于路网的压缩,因此压缩率会降低;

(2) 当窗口达到 200 之后,窗口的增大对压缩效果的提升很不明显;

(3) *WSIZE*>30 时,窗口大小对压缩率的影响并不大.专车北京市数据集的轨迹压缩率基本维持在 31%~32%之间;Geolife 北京市数据集的压缩率基本维持在 41.5%~42.7%之间;Geolife 华盛顿数据集的压缩率在 34%~35%之间.



(a) 专车北京数据集 (b) Geolife 北京数据集 (c) Geolife 华盛顿数据集

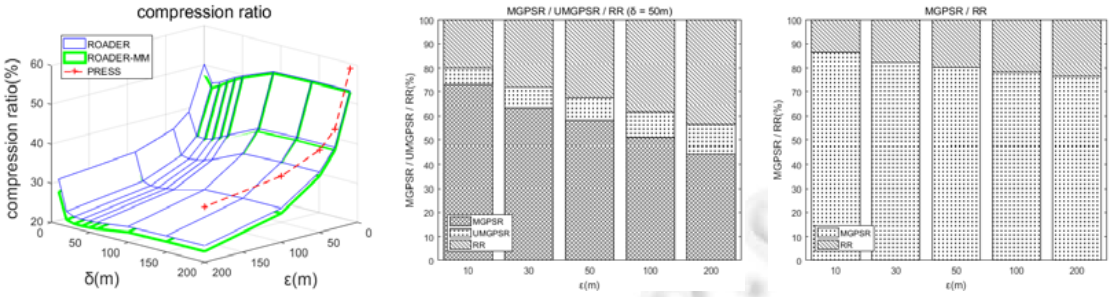
Fig.14 Impact of window size (*WSIZE*) on compression ratio (*CR*)

图 14 窗口大小(*WSIZE*)对压缩率(*CR*)的影响

- 实验 A.2:ROADER 与 PRESS 的压缩率对比

本组实验比较了 ROADER 系统和 PRESS 系统在不同数据集下的压缩率,以及在压缩轨迹结果中路径信息、轨迹点信息等各部分所占的比例.我们分别从 10m~200m 变换了匹配距离阈值δ和压缩误差阈值ε.实验结果如图 15~图 17 所示.其中,图 15(a)、图 16(a)和图 17(a)为这 3 个数据集的压缩率,蓝色细网格表示 ROADER 的总压缩率,绿色粗网格 ROADER-MM 表示 ROADER 中匹配轨迹(road-aware trajectories)的压缩率,红色虚线代表 PRESS 的压缩率.在 PRESS 中,由于它在地图匹配中没有距离上界,因此其压缩率只受压缩误差阈值ε的影响,为便于比较,我们将 PRESS 的结果画在图中δ=200m 处.图 15~图 17 的(b)图和(c)图分别为 3 个数据集下,ROADER 和 PRESS 的压缩结果数据中路径信息、轨迹点信息等各部分所占的比例.此外,在 ROADER 系统中,压缩后的轨迹点又分为匹配轨迹点和未匹配轨迹点,对此,本文用 $RR=\frac{M_R}{M_{T'}}\times 100\%$ 表示压缩结果中路径数据占存储量的百分比,用 $MR=\frac{M_{\overline{\overline{S}}}}{M_{T'}}\times 100\%$ 表示匹配轨迹点所占存储量的百分比,用 $NMR=\frac{M_{\overline{T'}}}{M_{T'}}\times 100\%$ 表示未

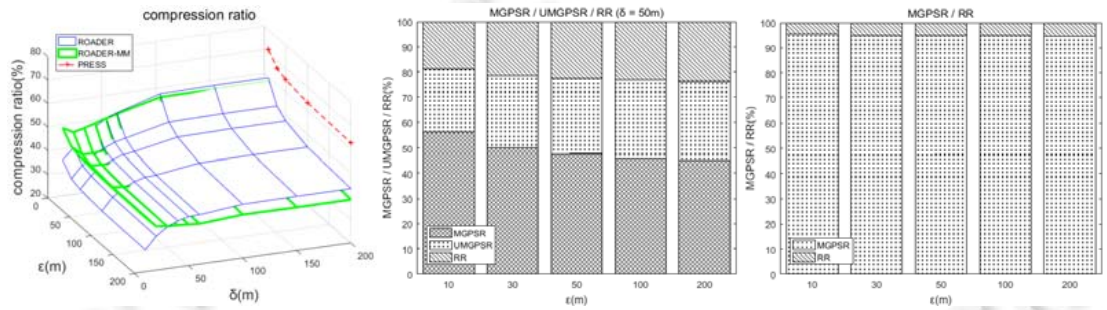
匹配轨迹点所占存储量的百分比.



(a) ROADER 和 PRESS 压缩率对比 (b) ROADER 压缩结果各部分占比 (c) PRESS 压缩结果各部分占比

Fig.15 Compression ratio and the proportion of each part of service car data set

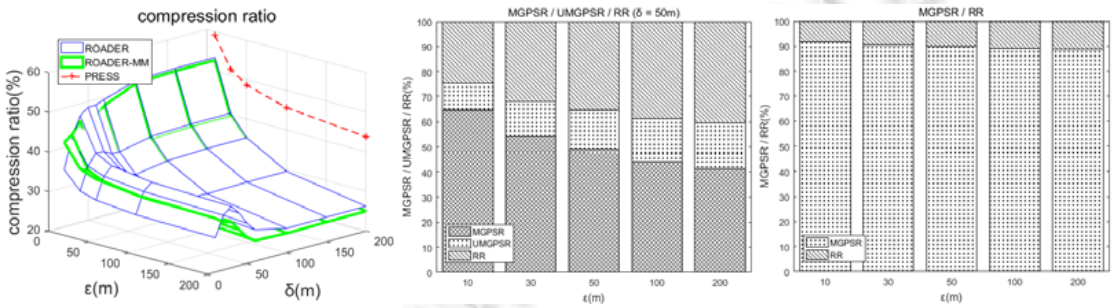
图 15 专车北京市数据集的压缩率及各部分占比



(a) ROADER 和 PRESS 压缩率对比 (b) ROADER 压缩结果各部分占比 (c) PRESS 压缩结果各部分占比

Fig.16 Compression ratio and the proportion of each part of Geolife-Beijing data set

图 16 Geolife 北京市数据集的压缩率及各部分占比



(a) ROADER 和 PRESS 压缩率对比 (b) ROADER 压缩结果各部分占比 (c) PRESS 压缩结果各部分占比

Fig.17 Compression ratio and the proportion of each part of Geolife-Washington data set

图 17 Geolife 华盛顿数据集的压缩率

图 15~图 17 说明:

(1) 在 PRESS 中,压缩率表现为随ε的增加而减小.在 ROADER 中,压缩率同时受δ和ε大小的影响.在同一阈值δ下,压缩率随ε的增加而变小,出现该结果的原因主要是匹配轨迹$\overline{\overline{T}}$和未匹配轨迹$\overline{T}$的压缩都受ε变化的影响.同一ε下,ROADER-MM 的压缩率总体上表现为随着δ的增加先变小后变大,在δ=20m~60m 之间,压缩率达到最佳,且ε越大,使压缩率达到最小的最佳δ也越大;

(2) ROADER 的压缩率整体上优于 PRESS.在专车北京市数据集、Geolife 北京市数据集和 Geolife 华盛顿数据集下,ROADER 的压缩率分别优于 PRESS 的压缩率 5%~13%,12%~35%和 6%~26%;

(3) ROADER 中,固定匹配距离δ=50m,变化压缩误差ε,则其压缩结果数据中路径信息的比例(*RR*)相比 PRESS 的 *RR* 大 7%~20%,这从侧面说明 ROADER 对匹配轨迹点的压缩效果明显优于 PRESS.而且,ROADER 还可进一步通过路径编码提高压缩率;

(4) 在 Geolife 数据集下,由于该数据集中包含许多不在路网上的行人轨迹,所以在该数据集下,*UMGPSR* 的值相对其他数据集较大.对于这部分轨迹,ROADER 采用基于同步距离的 DP 压缩算法[9],因此从图 16(a)和图 17(a)中可以看出:在δ取值为 10m~30m 之间时,ROADER 受 DP 压缩算法的影响较大.

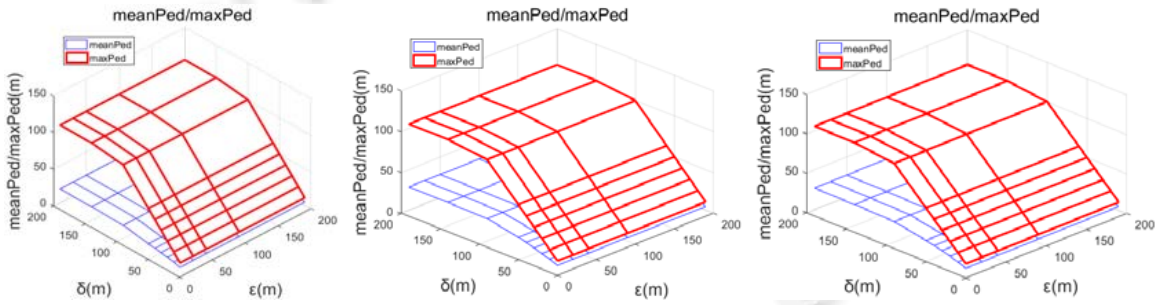
6.2.2 匹配距离

匹配距离指轨迹点到匹配路段的垂直距离,是衡量匹配正确性的主要指标之一.该组实验分别在匹配距离阈值δ和压缩阈值ε下,对 ROADER 和 PRESS 的匹配距离 ped 进行分析.表 1 为 3 个数据集下,PRESS 的平均匹配距离 meanPed 和最大匹配距离 maxPed,以及在所有匹配结果中匹配路段的方向与轨迹点运动方向相反的比例.图 18 为 3 个数据集下,ROADER 的平均匹配距离 meanPed 和最大匹配距离 maxPed.

Table 1 Experimental results of PRESS system in each data set

表 1 PRESS 系统在各数据集下的实验结果

数据集	meanPed(m)	maxPed(m)	匹配方向相反(%)
专车(北京)	525.213	40672.470	13.65
Geolife(北京)	382.613	28657.781	40.25
Geolife(华盛顿州)	38.747	17171.455	27.86



(a) 专车北京数据集 (b) Geolife 北京数据集 (c) Geolife 华盛顿数据集

Fig.18 MeanPed and maxPed of ROADER system

图 18 ROADER 系统的平均匹配距离(meanPed)和最大匹配距离(maxPed)

实验结果说明:

(1) PRESS 的匹配距离最大可能达到 40km 以上,即没有边界;

(2) ROADER 的 meanPed 和 maxPed 都显著小于 PRESS,且 ROADER 的匹配距离是有界的;

(3) ROADER 中,meanPed 随δ的增加而增大,且不受压缩阈值ε的影响.专车北京市数据集下的 meanPed 低于 Geolife 数据集下的 meanPed 约 10%左右.原因是 Geolife 数据集的行人轨迹多,导致未匹配轨迹占比大,从而 meanPed 也较大;

(4) ROADER 中,maxPed 随δ的增加而增大,且不受压缩阈值ε的影响.

6.2.3 匹配轨迹的压缩误差

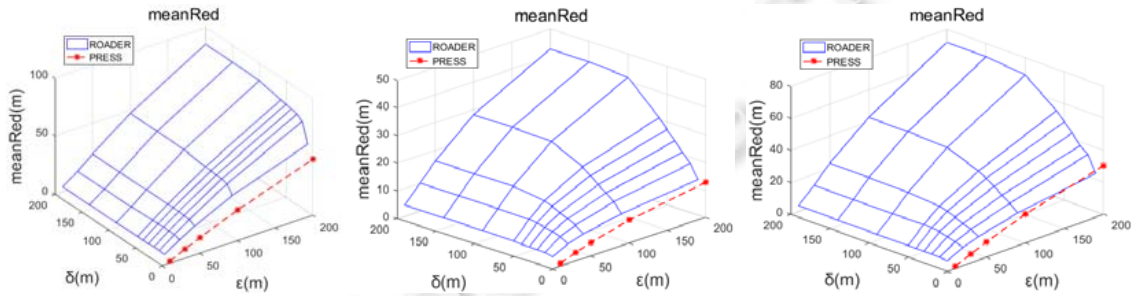
匹配轨迹的压缩误差是指轨迹点的匹配点相对压缩轨迹的对应点的距离,是衡量轨迹压缩的重要指标之一.本组实验分别在匹配距离阈值δ和压缩阈值ε下,对 ROADER 和 PRESS 的压缩误差 red 进行实验.图 19 和图 20 分别是 ROADER 和 PRESS 在 3 个数据集下的平均误差 meanRed 和最大误差 maxRed.实验结果说明:

(1) ROADER 压缩误差受ε和δ的共同影响;

(2) ROADER 和 PRESS 的 meanRed 都随ε的增加而增大.ROADER 的 meanRed 开始随δ的增加而缓慢增大,当δ大于一定值后,meanRed 的增加变得缓慢;

(3) ROADER 最好的 meanRed 较 PRESS 的 meanRed 值大 2%~9%;ROADER 最差的 meanRed 较 PRESS 的 meanRed 值大 2%~35%;

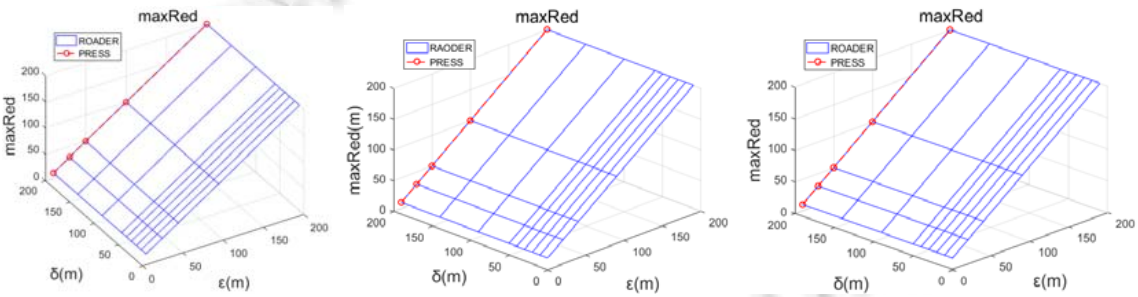
(4) ROADER 和 PRESS 的最大压缩误差 maxRed 均接近于ε.



(a) 专车北京数据集 (b) Geolife 北京数据集 (c) Geolife 华盛顿数据集

Fig.19 MeanPed of compression algorithm of ROADER and PRESS

图 19 ROADER 和 PRESS 压缩算法的平均误差(meanRed)



(a) 专车北京数据集 (b) Geolife 北京数据集 (c) Geolife 华盛顿数据集

Fig.20 MaxPed of compression algorithm of ROADER and PRESS

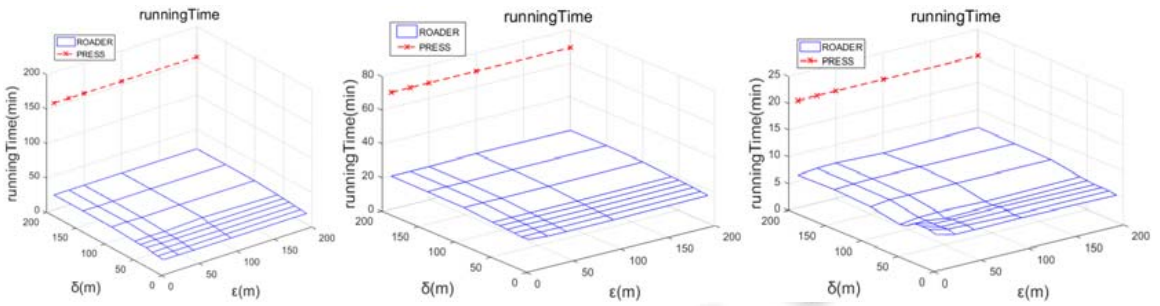
图 20 ROADER 和 PRESS 压缩算法的最大误差(maxRed)

6.2.4 执行时间

为比较 ROADER 和 PRESS 的执行时间 runningTime,本组实验分别调节匹配距离阈值δ和压缩阈值ε,实验结果如图 21 所示.实验结果说明:

(1) ROADER 的执行时间随着ε的增大而减小;随δ的增大先增大后减小,再增大.PRESS 压缩算法的执行时间随ε的增大而减小;

(2) ROADER 整体快于 PRESS.实验中,压缩阈值ε取不同值时,在专车北京市数据集下,PRESS 的执行时间约为 150 分钟,而 ROADER 的执行时间快于 PRESS 近 120 分钟;在 Geolife 北京数据集下,PRESS 的执行时间约为 70 分钟,ROADER 快于 PRESS 约 50 分钟;在 Geolife 华盛顿数据集下,PRESS 的执行时间约为 20 分钟,ROADER 快于 PRESS 约 14 分钟.



(a) 专车北京数据集 (b) Geolife 北京数据集 (c) Geolife 华盛顿数据集

Fig.21 Running time of ROADER and PRESS

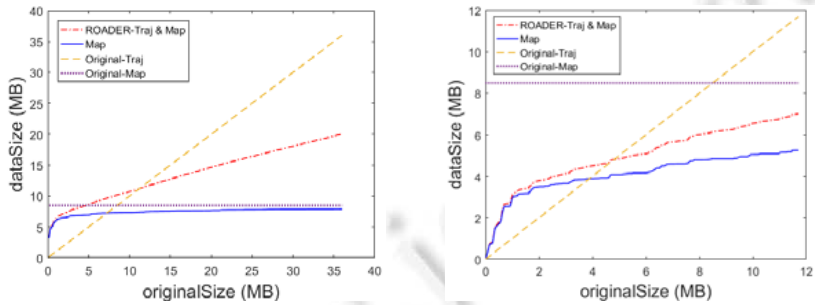
图 21 ROADER 和 PRESS 的执行时间

6.2.5 ROADER 地图加载的网络流量分析

为分析 ROADER 在线下载地图的客户端流量消耗情况,本文在专车北京数据集和 Geolife 北京数据集下进行了在线地图下载实验.实验结果如图 22 所示.其中,*Original-Traj* 是原始轨迹的大小,即客户端直接将原始轨迹上传至服务端所产生的流量;*Origi-Map* 是离线下载整个地图时客户端产生的流量;*Map* 是客户端在线下载地图时产生的流量;*ROADER-Traj & map* 是 ROADER 在客户端在线执行时,上传压缩轨迹和在线下载地图产生的流量之和.实验结果说明:

(1) 原始轨迹数据量较小时,客户端的流量消耗主要来自地图下载产生的流量.随着原始轨迹数据量的增加,下载地图消耗的流量趋于稳定;

(2) 专车数据和 Geolife 数据有明显的不同.专车轨迹覆盖区域比较广,因此在线下载的地图数据量较快接近于地图总数据量;而 Geolife 中的移动对象为行人,其通常在有限的且较固定的范围内活动,因此其客户端只需下载有限的地图就能满足其轨迹的匹配需求.



(a) 专车北京数据集 (b) GeoLife 单个移动对象

Fig.22 Cost of the client's transmission of a moving object

图 22 单个移动对象的客户端数据流量

7 结 论

本文提出一种路网感知的在线轨迹压缩系统,包括适合于轨迹压缩的距离有界的隐马尔可夫地图匹配算法和误差有界的高效轨迹压缩算法等.给定匹配距离阈值和压缩误差阈值,该系统可以保证最终的压缩结果与原始轨迹的距离在阈值之内,从而保证了压缩轨迹和原始轨迹的相似性.该系统不仅适用于移动对象不在路网道路中行驶的轨迹压缩,也适用于路网信息不完整情况下的压缩.基于真实数据集的实验证明,该系统在压缩率、误差和执行时间方面均显著优于同类算法.

未来还将从以下几方面继续改进该系统:(1) 在 HMM 地图匹配模型的计算中加入其他能够提高匹配准确率的因素,例如速度等;(2) 对匹配道路进行模式挖掘或对路径进行编码,进一步提高道路存储的压缩率.

References:

[1] Giannotti F, Nanni M, Pedreschi D, Pinelli F, Renso C, Rinzivillo S, Trasarti R. Unveiling the complexity of human mobility by querying and mining massive trajectory data. The VLDB Journal — The Int'l Journal on Very Large Data Bases, 2011,20(5):695−719. [doi: 10.1007/s00778-011-0244-8]

[2] Li Q, Zheng Y, Xie X, Chen Y, Liu W, Ma WY. Mining user similarity based on location history. In: Proc. of the 16th ACM SIGSPATIAL Int'l Conf. on Advances in Geographic Information Systems. ACM Press, 2008. 34. [doi: 10.1145/1463434.1463477]

[3] Zheng Y, Li Q, Chen Y, Xie X, Ma WY. Understanding mobility based on GPS data. In: Proc. of the 10th Int'l Conf. on Ubiquitous Computing. ACM Press, 2008. 312−321. [doi: 10.1145/1409635.1409677]

[4] Doytsher Y, Galon B, Kanza Y. Storing routes in socio-spatial networks and supporting social-based route recommendation. In: Proc. of the 3rd ACM SIGSPATIAL Int'l Workshop on Location-Based Social Networks. ACM Press, 2011. 49−56. [doi: 10.1145/2063212.2063219]

[5] Zheng Y, Zhang L, Ma Z, Xie X, Ma WY. Recommending friends and locations based on individual location history. ACM Trans. on the Web (TWEB), 2011,5(1):5. [doi: 10.1145/1921591.1921596]

[6] Zheng Y, Zhang L, Xie X, Ma WY. Mining interesting locations and travel sequences from GPS trajectories. In: Proc. of the 18th Int'l Conf. on World Wide Web. ACM Press, 2009. 791−800. [doi: 10.1145/1526709.1526816]

[7] Gotsman R, Kanza Y. A dilution-matching-encoding compaction of trajectories over road networks. GeoInformatica, 2015,19(2):331−364. [doi: 10.1007/s10707-014-0216-4]

[8] Song R, Sun W, Zheng B, Zheng Y. PRESS: A novel framework of trajectory compression in road networks. Proc. of the VLDB Endowment, 2014,7(9):661−672. [doi: 10.14778/2732939.2732940]

[9] Meratnia N, Rolf A. Spatiotemporal compression techniques for moving point objects. In: Proc. of the Int'l Conf. on Extending Database Technology. Berlin, Heidelberg: Springer-Verlag, 2004. 765−782. [doi: 10.1007/978-3-540-24741-8_44]

[10] Douglas DH, Peucker TK. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: The Int'l Journal for Geographic Information and Geovisualization, 1973,10(2):112−122. [doi: 10.3138/FM57-6770-U75U-7727]

[11] Lin X, Ma S, Zhang H, Wo T, Huai J. One-Pass error bounded trajectory simplification. Proc. of the VLDB Endowment, 2017,10(7):841−852. [doi: 10.14778/3067421.3067432]

[12] Liu J, Zhao K, Sommer P, Shang S, Kusy B, Jurdak R. Bounded quadrant system: Error-bounded trajectory compression on the go. In: Proc. of the 2015 IEEE 31st Int'l Conf. on Data Engineering (ICDE). IEEE, 2015. 987−998. [doi: 10.1109/ICDE.2015.7113350]

[13] Muckell J, Olsen PW, Hwang JH, Hwang JH, Lawson CT, Ravi SS. Compression of trajectory data: A comprehensive evaluation and new approach. GeoInformatica, 2014,18(3):435−460. [doi: 10.1007/s10707-013-0184-0]

[14] Shi W, Cheung CK. Performance evaluation of line simplification algorithms for vector generalization. The Cartographic Journal, 2006,43(1):27−44. [doi: 10.1179/000870406X93490]

[15] Reumann K, Witkam APM. Optimizing curve segmentation in computer graphics. In: Proc. of the Int'l Computing Symp. 1974. 467−472.

[16] Sklansky J, Gonzalez V. Fast polygonal approximation of digitized curves. Pattern Recognition, 1980,12(5):327−331. [doi: 10.1016/0031-3203(80)90031-X]

[17] Williams CM. An efficient algorithm for the piecewise linear approximation of planar curves. Computer Graphics and Image Processing, 1978,8(2):286−293. [doi: 10.1016/0146-664X(78)90055-2]

[18] Zhao Z, Saalfeld A. Linear-Time sleeve-fitting polyline simplification algorithms. Proc. of the AutoCarto, 1997,13:214−223.

[19] Han Y, Sun W, Zheng B. COMPRESS: A comprehensive framework of trajectory compression in road networks. ACM Trans. on Database Systems (TODS), 2017,42(2):11. [doi: 10.1145/3015457]

[20] Keogh E, Chu S, Hart D, Pazzani M. An online algorithm for segmenting time series. In: Proc. of the IEEE Int'l Conf. on Data Mining (ICDM 2001). IEEE, 2001. 289−296. [doi: 10.1109/ICDM.2001.989531]

[21] O'Rourke J. An on-line algorithm for fitting straight lines between data ranges. Communications of the ACM, 1981,24(9):574−578. [doi: 10.1145/358746.358758]

[22] Chan WS, Chin F. Approximation of polygonal curves with minimum number of line segments or minimum error. Int'l Journal of Computational Geometry & Applications, 1996,6(1):59−77. [doi: 10.1142/S0218195996000058]

[23] Potamias M, Patroumpas K, Sellis T. Sampling trajectory streams with spatiotemporal criteria. In: Proc. of the 2016 18th Int'l Conf. on Scientific and Statistical Database Management. IEEE, 2006. 275−284. [doi: 10.1109/SSDBM.2006.45]

[24] Pavlidis T, Horowitz SL. Segmentation of plane curves. IEEE Trans. on Computers, 1974,100(8):860−870. [doi: 10.1109/T-C.1974.224041]

[25] Diggelen FV. System design & test-gnss accuracy-lies, damn lies, and statistics—This update to a seminal article first published here in 1998 explains how statistical methods can create many different. GPS World, 2007,18(1):26−33.

[26] Wu JG, Liu M, Wei G, Liu LF. An inproved trajectory data compression algorithm of sliding window. Computer Technology and Development, 2015,35(5):1209−1212 (in Chinese with English abstract). [doi: 10.3969/j.issn.1673-629X.2015.12.011]

[27] Gao Q, Zhang FL, Wang RJ, Zhou F. Trajectory big data: A review of key technologies in data processing. Ruan Jian Xue Bao/Journal of Software, 2017,28(4):959−992 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5143.htm [doi: 10.13328/j.cnki.jos.005143]

[28] Wang H, Zhang M, Yang R, Lin X, Wo T, Ranjan R, Xu J. SMTP: An optimized storage method for vehicle trajectory data exploiting trajectory patterns. In: Proc. of the 2016 IEEE 18th Int'l Conf. on High Performance Computing and Communications, IEEE 14th Int'l Conf. on Smart City, IEEE 2nd Int'l Conf. on Data Science and Systems (HPCC/SmartCity/DSS). IEEE, 2016. 773−780. [doi: 10.1109/HPCC-SmartCity-DSS.2016.0112]

[29] Zhao XL, Xu WX. Spatio-Temporal trajectory clustering based on interesting location compression. Journal of Beijing Jiaotong University, 2011,35(3):53−57,61 (in Chinese with English abstract). [doi: 10.3969/j.issn.1673-0291.2011.03.009]

[30] Hummel B. Map matching for vehicle guidance. In: Proc. of the Dynamic and Mobile GIS: Investigating Space and Time. 2006. 437−438.

[31] Newson P, Krumm J. Hidden Markov map matching through noise and sparseness. In: Proc. of the 17th ACM SIGSPATIAL Int'l Conf. on Advances in Geographic Information Systems. ACM Press, 2009. 336−343. [doi: 10.1145/1653771.1653818]

[32] Goh CY, Dauwels J, Mitrovic N, Asif MT, Oran A, Jaillet P. Online map-matching based on hidden Markov model for real-time traffic sensing applications. In: Proc. of the 2012 15th Int'l IEEE Conf. on Intelligent Transportation Systems (ITSC). IEEE, 2012. 776−781. [doi: 10.1109/ITSC.2012.6338627]

[33] Zhou P, Huang W, Jiang J. Validation analysis of OpenStreetMap data in some areas of China. The Int'l Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 2014,40(4):383. [doi: 10.5194/isprsarchives-XL-4-383-2014]

[34] Girres JF, Touya G. Quality assessment of the French OpenStreetMap dataset. trans. in GIS, 2010,14(4):435−459. [doi: 10.1111/j.1467-9671.2010.01203.x]

[35] Palpanas T, Vlachos M, Keogh E, *et al*. Online amnesic approximation of streaming time series. In: Proc. of the 20th Int'l Conf. on Data Engineering. IEEE, 2004. 339−349. [doi: 10.1109/ICDE.2004.1320009]

[36] Lazaridis I, Mehrotra S. Capturing sensor-generated time series with quality guarantees. In: Proc. of the 19th Int'l Conf. on Data Engineering. IEEE, 2003. 429−440. [doi: 10.1109/ICDE.2003.1260811]

附中文参考文献:

[26] 吴家皋,刘敏,韦光,刘林峰.一种改进的滑动窗口轨迹数据压缩算法.计算机技术与发展,2015,35(5):1209−1212. [doi: 10.3969/j.issn.1673-629X.2015.12.011]

[27] 高强,张凤荔,王瑞锦,周帆.轨迹大数据:数据处理关键技术研究综述.软件学报,2017,28(4):959−992. http://www.jos.org.cn/1000-9825/5143.htm [doi: 10.13328/j.cnki.jos.005143]

[29] 赵秀丽,徐维祥.基于有趣地点压缩的时空轨迹聚类.北京交通大学学报,2011,35(3):53−57,61. [doi: 10.3969/j.issn.1673-0291.2011.03.009]

左一萌(1993－),女,山西长治人,硕士生,主要研究领域为大规模数据管理系统,移动计算,时序分析.

马帅(1975－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据库理论和系统,社交数据和图分析,密集型数据计算.

林学练(1978－),男,博士,讲师,主要研究领域为大规模数据管理系统,密集型数据计算,移动计算,时序分析.

姜家豪(1993－),男,硕士生,CCF 学生会员,主要研究领域为大规模数据管理系统,移动计算,时序分析.