计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2023,32(12):180−188 [doi: 10.15888/j.cnki.csa.009337]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

车辆轨迹中的隐性位置异常数据检测①

康　军, 吴子豪, 崔晟靖, 任海冰

(长安大学 信息工程学院, 西安 710064)
通信作者: 康　军, E-mail: junkang@chd.edu.cn

摘　要: 随着智能交通的发展, 大量的车辆轨迹数据被收集和存储, 但这些轨迹数据总是会存在异常轨迹点数据, 严重影响后续轨迹数据分析的准确性和有效性. 本文发现了一类隐性的位置异常轨迹数据, 此类异常数据用传统的基于移动特征阈值的检测方法难于发现, 但对轨迹数据分析过程同样有着重要的影响. 针对此类异常轨迹数据, 本文以部分西安市出租车轨迹数据为例, 提出了一种基于浮动网格和聚类方法的隐性异常轨迹数据检测方法, 并实现了数据的并行化方式. 实验结果展示所提方法检测隐性位置异常的数据召回率、精确率能够达到 0.90, 并且 *F*1-*score* 在 0.88–0.91 范围. 检测出这种隐性异常轨迹数据, 有利于后续的时空轨迹数据分析与应用.

关键词: 智能交通; 轨迹数据; 异常检测; 聚类; K-means; 检测方法

引用格式: 康军,吴子豪,崔晟靖,任海冰.车辆轨迹中的隐性位置异常数据检测.计算机系统应用,2023,32(12):180–188. http://www.c-s-a.org.cn/1003-3254/9337.html

Implicit Positional Anomaly Data Detection in Vehicle Trajectories

KANG Jun, WU Zi-Hao, CUI Sheng-Jing, REN Hai-Bing

(School of Information Engineering, Chang'an University, Xi'an 710064, China)

Abstract: With the development of intelligent transportation, a large amount of vehicle trajectory data is collected and stored. However, the trajectory data always has anomalous trajectory point data, seriously affecting the accuracy and effectiveness of subsequent trajectory data analysis. This study finds a class of implicit positional anomaly trajectory data that is difficult to be detected by traditional detection methods based on movement feature thresholds but plays a vital role in trajectory data analysis. To this end, this study proposes a method to detect the implicit anomalous trajectory data based on floating grid and clustering method. The parallelization method of data is realized by taking the trajectory data of some cabs in Xi'an as an example. The experimental results show that the data recall and accuracy of the proposed method to detect the hidden location anomaly could reach 0.90, and the *F*1-*score* is in the range of 0.88–0.91. The detection of such implicit anomalous trajectory data is beneficial to subsequent analysis and application of spatio-temporal trajectory data.

Key words: intelligent transportation; trajectory data; anomaly detection; clustering; K-means; detection method

随着基于位置服务 (LBS) 技术的发展, 越来越多的车辆位置移动数据产生并被存储下来, 这些海量数据中隐藏着大量的信息, 挖掘这些隐藏的信息即轨迹数据挖掘有助于城市交通管理决策, 有利于智慧交通城市的实现[1]. 受到位置采样环境、采样设备精度及采样频率等因素的影响, 原始车辆轨迹数据中会普遍存在异常数据, 如轨迹数据定位信息出现较大偏差或车辆速度明显异常, 这对基于车辆轨迹数据的后续研究工作将产生重要的消极影响. 因此, 为了保证车辆轨迹数据质量, 检测、识别和过滤这些轨迹异常数据则成为轨迹数据挖掘预处理阶段的一个重要步骤.

车辆轨迹数据是指按固定周期从车辆的位置传感

① 基金项目: 陕西省重点研发计划 (2020ZDLGY09-02)
收稿时间: 2023-06-07; 修改时间: 2023-07-12; 采用时间: 2023-07-27; csa 在线出版时间: 2023-09-22
CNKI 网络首发时间: 2023-09-26

器中读取到的一系列轨迹点时间序列数据, 其中每个轨迹点数据包括采样时刻、当前经纬度坐标等内容. 在采集车辆轨迹数据的过程中, 由于环境和设备等因素的影响, 原始车辆轨迹数据中总是存在异常轨迹点数据. 其中, 常见的异常轨迹点是移动特征异常轨迹点, 表现为在车辆移动过程中, 连续多个采样点中所包含的速度、加速度、方向角等车辆移动特征数据不符合常理即超出合理数值阈值范围. 对于此类异常轨迹点, 常用的检测方法是直接对相邻轨迹点之间提取的速度、加速度或方向角等移动特征数据进行取值合理性检测. 例如, 康军等人[2]计算相邻轨迹点之间速度, 通过最大速度限制, 将速度异常的轨迹点从数据中删除; 周洋等人[3]根据轨迹点经纬度计算速度、加速度变量, 并依据变量值剔除异常轨迹点; Hsueh 等人[4,5]在对轨迹数据的预处理中, 速度超过阈值的轨迹点被视为异常值被剔除; Ebel 等人[6]将相邻轨迹点间的平均速度与预设速度阈值进行比较, 若轨迹点的速度超出阈值范围则被视为异常轨迹点; Mohamed 等人[7]使用轨迹点的速度和方向角等移动特征, 通过应用 3 个过滤器, 对异常轨迹点的检测和剔除; Patil 等人[8]使用更全面的轨迹点移动特征 (距离、速度和加速度) 的组合进行异常轨迹点的检测; Ma 等人 [9]使用移动特征阈值对轨迹数据进行预处理, 将超出阈值的轨迹点视为异常轨迹点. 上述异常轨迹点的检测方法能够高效的将移动特征异常的轨迹点检测出来, 在大量的轨迹计算文献中都有成功运用.

但是, 在研究过程中我们发现了一种特殊的异常轨迹点, 如图 1 所示, 在距离周围路段相对较远的较小的空间范围内, 连续异常轨迹点 g_7–g_{14} 嵌入在正常的轨迹点序列中, 其移动特征数据并无异常, 但是在实际路网中却不存在和该轨迹点匹配的真实路段. 这些异常轨迹点主要是由于车辆在某些特定区域 (如科技园区) 的内部道路上行驶, 而这些内部道路在电子地图的路网数据中并不存在. 我们将上述异常轨迹点称作隐性位置异常轨迹点, 这些异常轨迹点用传统的基于移动特征的异常轨迹点检测方法难以检测, 若直接将其看作正常的轨迹点, 并进行后续的地图匹配处理则必定会产生匹配错误, 从而对后续一系列基于位置服务 (如车辆行为分析, 序列模式挖掘等) 的准确性造成不利影响.

隐性位置异常轨迹点呈现出一定的时空局部聚集性, 传统的异常点检测方法是根据轨迹点间的时空关系进行判断, 忽略了轨迹点和路段的空间关系, 导致难以检测出这种异常轨迹点. 综上, 本文提出一种针对隐性位置异常轨迹点的检测方法, 旨在通过利用轨迹数据的时空特性, 有效识别车辆轨迹中容易被忽略的隐性位置异常数据. 该方法通过 Geohash 编码[10]和浮动网格[2]筛选出初始核心异常轨迹点; 针对每个核心异常轨迹点进行简单聚类, 探索时空范围内的其他隐性位置异常轨迹点; 最终, 检测出隐性位置异常轨迹点集. 通过实验结果表明, 本文所提方法可以有效检测出隐性的位置异常轨迹点, 在提高车辆行为分析和交通管理等领域的准确性和效率方面具有重要意义.

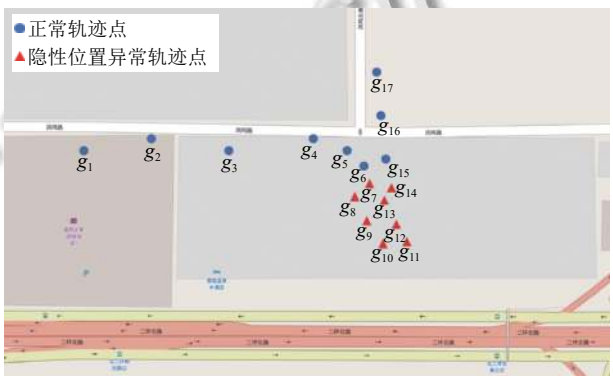

图 1 隐性位置异常轨迹点示例

1 位置异常轨迹点问题描述

本文所涉及的相关定义如下.

定义 1. 车辆轨迹 (T): 由同一车辆的一系列时间连续的轨迹点所组成的时间序列称为车辆轨迹 T, 其中 $T=(g_1,g_2,\cdots,g_n)$, $\forall g_i \in T, g_i=(id,time,lon,lat)$, 且 id 表示产生轨迹的车辆的车牌号, $time$ 表示采样时刻, lon 和 lat 分别该时刻轨迹点的经度和纬度坐标.

定义 2. 道路网络: 使用有向图 $G(V,E)$ 表示道路网络, 其中 V 表示道路网络中路段交点集合, E 表示道路网络中路段集合. 其中, $\forall e \in E$, $e=(id,ps)$, $e.id$ 表示路段的唯一标识, $e.ps$ 是电子地图中表示路段的一组二维经纬度点的序列, $\forall p \in e.ps, p=(lon,lat)$, 其中 $p.lon$、$p.lat$ 分别表示路段 e 的经纬度点的经度值和纬度值.

定义 3. Geohash 网格[10]: 本文采用定长 Geohash 编码方式将路网 $G(V,E)$ 划分成固定大小的矩形网格, 对于每个 Geohash 网格均存在一个全局唯一且定长的 Geohash 编码 ID_{Geo}, 则对于 $\forall p \in e.ps \wedge \forall e \in E$ 总是包含在一个唯一的 Geohash 网格中. 设 Geohash 网格的长和宽分别表示为 L_g 和 W_g, 且本文中都定为常数.

定义 4. Geohash 网格索引 (M): 是路网的各个 Geohash 网格编号 ID_{Geo} 到该网格所包含的路段集合

C_e的索引. 索引中每一项均表示为一个键值对, $\forall r \in M, r = \{(ID_{\text{Geo}} : C_e)\}$, 其中, ID_{Geo}为 Geohash 网格编号, C_e为编号为ID_{Geo}的 Geohash 网格所包含的所有的路段集合, 可定义为$C_e = \{e | \forall e \in E \wedge \exists p \in e.ps \rightarrow geohash(p) = ID_{\text{Geo}}\}$, 其中$geohash()$表示根据经纬度点计算其所在的 Geohash 网格的网格编号的标准方法.

定义 5. 浮动网格: 给定一个轨迹点g_i, 其浮动网格c_{g_i}为以g_i的经纬度坐标为中心的一个边长为 L 的正方形区域. 对于一个车辆轨迹 T 中所包含的每个轨迹点均存在一个与其对应的浮动网格. 对于任一个浮动网格c_{g_i}, 设其对应的 4 个顶点分别表示为$v_{c_{g_i}}^{lu}, v_{c_{g_i}}^{ru}, v_{c_{g_i}}^{rd}, v_{c_{g_i}}^{ld}$.

定义 6. 轨迹点候选路段集合(C_{condi}): 给定一个轨迹点g_i, 设其对应的浮动网格为c_{g_i}且该浮动网格对应的 4 个顶点分别为$v_{c_{g_i}}^{lu}, v_{c_{g_i}}^{ru}, v_{c_{g_i}}^{rd}, v_{c_{g_i}}^{ld}$, 则由 $Geohash()$ 方法可确定c_{g_i}的 4 个顶点分别所对应的 Geohash 网格及其编号表示为$ID_{c_{g_i}}^{lu}$, $ID_{c_{g_i}}^{ru}$, $ID_{c_{g_i}}^{rd}$, $ID_{c_{g_i}}^{ld}$. 根据 Geohash 网格索引 M 可获得上述 Geohash 网格所包含的路段集合, 表示为$C_e^{(i0)}$, $C_e^{(i1)}$, $C_e^{(i2)}$, $C_e^{(i3)}$, 则轨迹点g_i的候选路段集合$C_{\text{condi}}^{(i)}$可定义为$C_{\text{condi}}^{(i)} = C_e^{(i0)} \cup C_e^{(i1)} \cup C_e^{(i2)} \cup C_e^{(i3)}$, 其中$C_e^{(ij)}$为浮动网格第$j$个顶点所在 Geohash 网格对应的路段集合, $0 \leqslant j \leqslant 3$.

定义 7. 隐性位置异常轨迹点集合(C_o):在车辆轨迹序列中, 移动特征数据总是满足阈值约束, 但在实际路网中却不存在与之匹配的真实路段的轨迹点, 将其称之为隐性位置异常轨迹点. 根据隐性位置异常轨迹点的判断条件的不同, 可将其分为两类即核心位置异常轨迹点和位置异常轨迹点, 其中对于一个轨迹点 O, 如果其候选路段集合$C_{\text{condi}}^{O} = \varnothing$, 则称 O 为核心位置异常轨迹点, 而位置异常轨迹点需要通过本文所提的隐性位置异常轨迹点聚类算法进行判断. 隐性位置异常轨迹点集合C_o则定义为当前轨迹序列中所有的核心位置异常轨迹点和位置异常轨迹点的集合.

本文所提的隐性位置异常数据的检测方法是指: 给定车辆行驶轨迹T和道路网络$G(V,E)$, 根据 Geohash 网格索引M及浮动网格候选路段选择方法和隐性位置异常轨迹点聚类方法, 在轨迹T中检测隐性位置异常轨迹点集合C_o.

2 基于聚类的隐性位置异常数据检测方法

2.1 隐性位置异常点的特征与检测方法的关联分析

隐性位置异常轨迹点的特征为点远离路段并且具有一定时空聚集性, 因此, 在检测这些点时, 需要使用一种方法来定位这些点, 并将其周围可能存在的异常点识别出来. 而浮动网格与聚类算法结合的方法可以很好地满足这一需求.

浮动网格的特性是可以根据轨迹点来动态调整网格的大小和位置, 从而有效地检索出距离该轨迹点一定范围内的路段, 具体见第 2.2 节; 因此, 对于远离路段的隐性位置异常点, 可以通过浮动网格来检测它们. 聚类算法是一种无监督学习方法, 可以将具有相同特征的数据分为一组; 在这种情况下, 将浮动网格检测出的隐性位置异常点为中心, 使用聚类算法, 将该轨迹点周围具有相似时空属性的轨迹点分为同一异常轨迹点类簇, 作为核心位置异常轨迹点的补充. 因此, 浮动网格与聚类算法结合的检测方法可以有效地定位远离路段且具有一定时空聚集性的隐性位置异常点.

本文所提出的隐性位置异常数据检测方法主要包含两个步骤: 第 1 步是确定核心位置异常轨迹点数据, 即当轨迹序列中的一个轨迹点通过浮动网格确定的候选路段集合为空时, 即可以确定其为核心位置异常轨迹点; 第 2 步是面向核心位置异常轨迹点, 根据隐性位置异常数据的局部聚集性假设, 采用聚类算法检测聚集在核心位置异常轨迹点附近的在其他位置异常轨迹点. 最终检测出正常轨迹中的隐性位置异常轨迹点.

2.2 核心位置异常轨迹点的选取方法

(1) 路网网格划分

本文首先使用 Geohash 网格划分方法[2,11], 将城市道路网络$G(V,E)$按照相同大小的 Geohash 网格划分为多个网格区域. 其中, 每个网格均有一个全球唯一的字符串类型的网格编号ID_{Geo}; 同时, 根据 Geohash 网格编号长度的不同, 相对应的每个网格的大小也会不同, 即网格编号长度越长, 网格的个数越多, 单个网格覆盖范围越小; 网格编号长度越短, 网格的个数越少, 单个网格覆盖范围越大.

如图 2 中给出了以 7 字符长度划分出的 4 个 Geohash 网格的示意图, 其中 4 个网格的ID_{Geo}分别为 wqj76q、wqj76w、wqj76m 和 wqj76t. 采用 Geohash 网格系统进行路网网格化划分的优势在于当判断路网中各位置点所在网格编号时, 仅通过标准化的计算即可获得其所在的网格编号, 用于划分路网的网格信息不需要提前生成和保存.

路网$G(V,E)$经过 Geohash 网格划分后, 对于$\forall p \in e.ps, \forall e \in E$, 通过$geohash(p)$即可获得的其对应的 Geo-

hash 网格编号, 因此可容易计算每条路段所对应的 1 个或多个 Geohash 网格, 通常一个 Geohash 网格会覆盖多条路段. 为了快速找到 Geohash 网格中包含的路段, 本文进一步构建了 Geohash 网格索引 M, 该索引以 Geohash 网格编号为 Key 以查询该网格所覆盖的所有路段. 所建立索引 M 部分数据如表 1 所示, 通过该索引, 能够根据给定的 Geohash 网格编号 ID_{Geo} 快速查询到其所覆盖的路段集合 C_e.

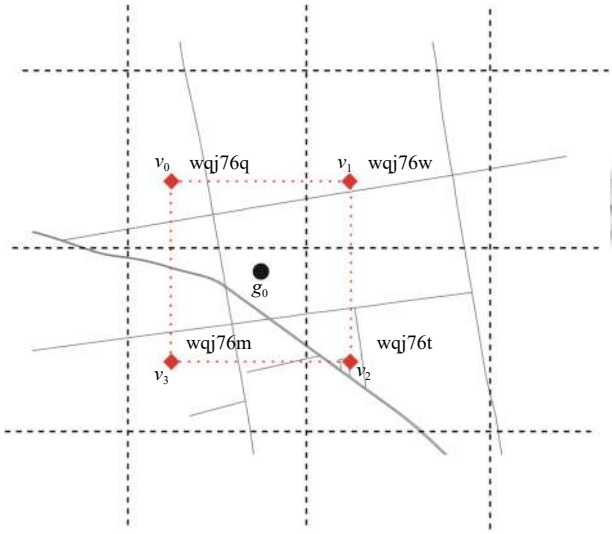

图 2 区域 Geohash 网格划分示意图

表 1 Geohash 网格索引示例

网格编号	路段属性		
	路段编号	路段坐标点序列	
wqj6z86	51485700348	(108.96546, 34.23522) (108.96580, 34.23520)	…
	51482706447	(108.96625, 34.23557) (108.96581, 34.23557)	…
	51482706471	(108.96539, 34.23478) (108.96485, 34.23481)	…
wqj6wfb	51482710593	(108.93050, 34.20519) (108.93006, 34.20519)	…
	51482707522	(108.92962, 34.20510) (108.93006, 34.20510)	…
…	…	…	…

(2) 浮动网格以及候选路段集合的设定

对于移动的车辆的轨迹 T 中的轨迹点 g_i, 本文采用定义 5 中所定义的浮动网格方法获得定义 6 所定义的候选路段集合 $C_{\text{condi}}^{(i)}$. 例如在图 2 中, 轨迹点 $g_0 \in T$ 的浮动网格对应的矩形的 4 个顶点分别为 v_0, v_1, v_2, v_3, 经过 *geohash* 方法计算可知其对应的 Geohash 网格编号分别为 wqj76q、wqj76w、wqj76m 和 wqj76t. 利用 Geohash 网格索引 M, 即可获得上述 4 个网格所覆盖的路段集合即为轨迹点 g_0 的候选路段集合.

(3) 核心位置异常轨迹点的确定

对于 $\forall g_i \in T$, 如果其通过浮动网格获得的候选路段集合 $C_{\text{condi}}^{(i)} = \varnothing$, 即 4 个网格中都不存在路段, 则 g_i 被确认为核心位置异常轨迹点. 核心位置异常轨迹点的候选路段集合为空集, 在轨迹数据预处理环节的地图匹配过程中就无法找到其匹配路段, 这就对轨迹数据的后续分析产生的一定的影响, 因此有必要把这类轨迹点在轨迹数据预处理之前检测出来.

当确定核心位置异常轨迹点, 根据隐性位置异常数据的局部聚集性假设, 则必然在核心位置异常轨迹点周围存在其他隐性位置异常点, 此类轨迹点的候选路段集合并不为空且其移动特征参数往往也在正常的阈值范围内, 在传统的轨迹异常检测环节中是无法检测出来的, 而且通过地图匹配也能确定其各自的匹配路段, 但此类轨迹点实际上并不是在路网中的路段上移动, 因此地图匹配结果往往也是错误的, 因此此类位置异常轨迹点相对其他类型的异常轨迹点更为隐蔽且难以检测, 这也是本文将其和核心位置异常轨迹点统称为隐性位置异常点的原因. 为了检测核心位置异常轨迹点周围的其他隐性位置异常点, 本文提出了一种面向隐性位置异常点检测的聚类方法.

2.3 面向隐性位置异常点检测的聚类方法

在已知核心位置异常轨迹点的条件下, 对于其周围的其他隐性位置异常轨迹点的检测, 通过本文所提出的一种简化的聚类算法对其进行聚类检测. 由于本文仅关注隐性位置异常轨迹点所构成的类簇, 因此所检测的轨迹点数据可分为两类, 一类是属于隐性位置异常类簇的轨迹点, 一类则是不属于上述类簇的轨迹点. 面向隐性位置异常点检测的聚类方法通过迭代的方式仅不断更新隐性位置异常类簇集合.

假设在进行隐性位置异常点检测之前能够获得当前车辆在设定的时段内的完整轨迹点序列. 首先对轨迹序列逐点判断其是否为核心位置异常轨迹点, 设第 j 个轨迹点 $g_j \in T$ 被确定为核心位置异常轨迹点, 则以 g_j 为第 j 个隐性位置异常类簇的初始的类簇中心 g_{center}, 即作为聚类算法的输入. 隐性位置异常点的聚类方法的伪代码算法 1 所示.

算法 1. 聚类算法伪代码

输入: 车辆轨迹 T, Geohash 网格索引 M, 类簇中心 g_{center}.
输出: 核心点聚类结果 C_o.

1. 输入车辆行驶轨迹T, C_o←g_{center}
2. $step$←1 // 初始化步数
3. j←$indexOf(T,g_{\text{center}})$
4. **While** $j-step>0$ **do**
5. **If** g_{j-step} is not O **then**
6. dis←$\|g_{\text{center}}-g_{j-step}\|_{\text{greatCircle}}$
7. C_{condi}←$getCondidates(g_{j-step},M)$
8. $pdis$←$d(g_{j-step},C_{\text{condi}})$//计算最小投影距离
9. **If** $pdis>dis$ **then**
10. $C_o.add(g_{j-step})$ //将g_{j-step}加入异常类簇
11. g_{center}←$updateCenter(g_{j-step},g_{\text{center}})$
12. **Else**
13. **break**
14. **End If**
15. **End If**
16. $step$←$step$+1 //向前进一步迭代
17. **End While**
18. return C_o

开始以g_j为中心按照时间顺序开始向前和向后逐跳遍历其他非核心位置异常轨迹点, 其中伪代码只给出向前遍历, 而向后遍历与向前同理.

首先向前遍历g_{j-1}, 如果该轨迹点已经被判定为隐性位置异常轨迹点, 那么直接进入下一次循环, 否则得到g_{j-1}候选路段集合为$C_{\text{condi}}^{(j-1)}$, 设$d(g_{j-1},C_{\text{condi}}^{(j-1)})$是轨迹点$g_{j-1}$到其候选路段集合中路段的最小投影距离, $d(g_{j-1},g_{\text{center}})$是轨迹点$g_{j-1}$与$g_{\text{center}}$的距离. 如果$d(g_{j-1},g_{\text{center}})<d(g_{j-1},C_{\text{condi}}^{(j-1)})$, 则将$g_{j-1}$加入隐性位置异常类簇并更新隐性位置异常类簇中心 (更新算法如式 (1) 所示), 然后对$step$加一; 否则视轨迹点g_{j-1}为正常轨迹点, 并结束这一方向的判断.

然后, 用上述方法向后遍历并检测, 直到停止这一方向的检测. 至此聚类算法的向前和向后的迭代过程完成, 最后得到隐性位置异常类簇C_o.

上述伪代码中, $getCondidates$表示第 2.2 节中基于浮动网格提取轨迹点候选路段方法, $d(g_{j-step},C_{\text{condi}})$表示根据轨迹点和候选路段集合得到轨迹点到路段的最小投影距离, $updateCenter$利用已有的隐性异常轨迹点类簇中心g_{center}和新增轨迹点g_{j-step}来更新g_{center}. 更新方法如式 (1) 中, $g_{\text{center}}^{\text{old}}$表示当前迭代时隐性位置异常类簇的中心位置, n 表示当前迭代时隐性位置异常类簇中点个数. g_{j-1}表示当前需加入隐性位置异常类簇的点.

$$g_{\text{center}}^{\text{new}}=\frac{g_{\text{center}}^{\text{old}}\times n+g_{j-1}}{n+1} \tag{1}$$

需要说明的是, 每次隐性位置异常点的聚类过程的触发条件是发现新的核心位置异常轨迹点. 在聚类过程中, 如出现时间连续的核心位置异常轨迹点, 则在同一次聚类过程中这些核心位置异常轨迹点不用判断, 自动进入隐性位置异常类簇; 如果隐性位置异常类簇持续不再更新, 则当前聚类过程结束. 然后, 继续从上次向后遍历检测的最后一个轨迹点继续开始搜索新的核心位置异常轨迹点, 一旦确定有新的核心位置异常轨迹点, 则触发新的隐性位置异常点的聚类过程.

2.4 并行化隐性位置异常轨迹点检测方法

面对持续增长的海量轨迹数据, 为了更高效地实现隐性位置异常轨迹点的检测, 本文提出了一种并行化隐性位置异常轨迹点检测方法.

要实现并行化隐性位置异常轨迹点检测, 首先需要对数据进行合理的分区, 车辆轨迹数据按照分区策略进行分布式存储. 本文以产生轨迹数据的车辆的车牌号的 $hash$ 值为 key 对数据进行分区, 同一辆车的轨迹数据保存在同一数据分区. 由于分布式存储系统所能够支持的最大分区数是一个预设参数, 实际中车辆数量会超过系统的最大分区数, 在这种情况下, 采用式 (2) 计算轨迹数据分区的 key:

$$key=Hashcode(id)\bmod m \tag{2}$$

其中, id表示车辆的车牌号; $Hashcode()$ 表示计算 $hash$ 值方法; m 为系统最大分区数. 数据分区后, 一个分区中可能存在不同的车辆轨迹数据, 可通过使用分布式计算环境中的算子$groupBykey$根据轨迹数据的 id, 将同一个分区内的车辆的轨迹数据聚合成不同的组后再进行分析. 对聚合后的每组轨迹数据执行算法 2 中方法进行隐性位置异常轨迹点检测. 具体如下.

算法 2. 隐性位置异常轨迹点检测伪代码

输入: 轨迹数据T, Geohash 网格索引M.
输出: 检测的异常点集合C_o.

1. 输入车辆行驶轨迹T, $clusterps$←∅, C_o←∅
2. ts←$T.groupByKey(cid)$
3. **For** each t in ts **do**
4. **For** i=0 to $t.len$ **do**
5. C_{condi}←$getCondidates(g_i,M)$
6. **If** C_{condi}=∅ **then**
7. $C_o.add(g_i)$ //将轨迹点加入异常类簇
8. $clusterps.add(g_i)$ //将轨迹点加入核心位置异常轨迹点集合
9. **End If**
10. **End For**
11. **End For**
12. **For** g_{center} in $clusterps$ **do**
13. C_o←$clustering(T,g_{\text{center}})$ //进行聚类检测
14. **End For**
15. return C_o

(1) 针对轨迹数据T, 通过分组函数将每辆车的轨迹分到一个组, 针对每个车辆轨迹的轨迹点, 求取其候选路段集合, 如果候选路段集合为空, 那么将该轨迹点识别为隐性异常轨迹点, 并将其加入到核心位置异常点集合中;

(2) 遍历核心位置异常轨迹点集合, 针对每一个核心位置异常轨迹点, 进行算法 1 的隐性位置异常点的聚类方法并返回异常类簇, 最终返回所有的隐性位置异常轨迹点.

算法 2 中方法*clustering*()是运用第 2.3 节中所提的聚类方法, 在获得一个新的核心位置异常轨迹点后, 检测该轨迹点附近是否存在其他隐性位置异常轨迹点并构建对应的隐性位置异常类簇.

3 测试及结果分析

本实验章节旨在评估所提出的隐性位置异常轨迹点检测方法的性能. 通过合适的评价指标来检测本文提出方法的准确性; 通过对比单机计算和并行计算下的计算延迟, 我们将探讨并行化对算法性能的影响. 通过这些实验, 我们能够验证本文方法在隐性位置异常轨迹点检测方面的准确性以及效率.

3.1 实验设置

(1) 测试数据

实验数据采用西安市市区路网数据以及西安市出租车真实 GPS 移动数据. 西安市市区路网包含 24 050 条路段, 每条路段由路段编号, 路段经纬度坐标序列、路段长度以及路段类别组成. 轨迹数据包含了 5 辆车的连续轨迹序列, 总计大约 12 000 个轨迹点, 轨迹数据的采样间隔为 30 s, 每个轨迹点包含 5 个属性, 车辆车牌号、采样时刻、经度、纬度和速度. 对于上述轨迹数据, 采用人工标定的方法, 对真实的隐性位置异常轨迹点进行了标定, 其中将 397 个轨迹点标记为隐性位置异常轨迹点.

(2) 路网网格化和 Geohash 网格索引建立

实验使用 Geohash 网格法对西安市市区进行了网格划分, Geohash 网格的编码长度设置为 7 字符长度, 从而将西安市区大约划分为 97 152 个 7 字符 Geohash 网格, 每个的网格约为 152 m×152 m的矩形区域.

基于上述 Geohash 网格设定参数, 对于每条路段的经纬度坐标序列中的每个坐标点, 离线计算其对应的 Geohash 网格编码. 然后, 根据计算结果构建 Geohash 网格索引. 由于路网路段在不同区域分布密度不同, 有些网格中不存在路段. 对于所覆盖的路段集合为空的网格, 在索引中不进行记录, 最终得到 Geohash 网格索引共包括 27 956 个有效 Geohash 网格.

(3) 测量指标

为了验证本文所提的隐性位置异常轨迹检测方法的效果, 本文采用了精确率 (*precision*), 召回率 (*recall*), *F*1-*score*[12]来评价实验效果, 其中 *F*1-*score* 是对精确率和召回率的综合考量. 将正确识别的隐性位置异常轨迹点的情况定义为真正例 (*TP*), 将检测是隐性位置异常轨迹点但实际却不是的情况定义为假正例 (*FP*), 将检测为不是隐性位置异常轨迹点且实际也不是的情况定义为真反例 (*TN*), 将检测出不是隐性位置异常轨迹点但实际是的情况定义为假反例 (*FN*). 各指标的计算公式如下:

$$precision = \frac{TP}{TP+FP} \tag{3}$$

$$recall = \frac{TP}{TP+FN} \tag{4}$$

$$F1\text{-}score = \frac{2\times precision\times recall}{precision+recall} \tag{5}$$

3.2 实验结果

(1) 单机测试结果

1) 隐性位置异常轨迹点检测示例

如图 3 所示, 举例部分轨迹$g_{455}-g_{469}$的检测结果, 其中方格为 Geohash 网格, 其中人工标定的异常轨迹点为$g_{457}-g_{465}$, 本文检测方法的检测结果为 $g_{457}-g_{467}$. 从轨迹点g_{455}开始, 由于轨迹点$g_{455}-g_{457}$离路段较近, 候选路段集合不为空, 所以被视为正常轨迹点, 而$g_{458}-g_{465}$通过浮动网格筛选的候选路段集合为空, 被视为核心位置异常轨迹点, 但是在以g_{458}为核心的聚类前向遍历的过程中, 因为g_{457}距离聚类中心g_{458}更近, 所以也会将g_{457}识别为一个异常轨迹点. 以此类推, 轨迹点g_{466}、g_{467}虽然不是核心异常轨迹点, 但是其会被g_{465}向后遍历识别为隐性异常轨迹点; 所以示例中的轨迹被识别为隐性异常轨迹点的为$g_{457}-g_{467}$. 虽然该实例对轨迹点g_{466}、g_{467}识别错误, 但从结果来看, 这对于后续的地图匹配等预处理影响不大, 但是如果使用移动特征异常轨迹点的判断方法来识别此轨迹段, 则难以检测其中的异常轨迹点, 因为这些轨迹点的速度, 加速度等车辆移动特征并没有超出正常范围.

2) 准确率分析

为测试不同边长浮动网格, 对检测效果的影响, 本文在测试时将浮动网格边长 L 分别设置为 Geohash 网格的相同边长、3/4 边长和 1/2 边长, 即 152 m, 114 m

和 76 m, 测试结果如图 4.

在图 4 中, 纵轴表示评价指标得分, 横轴表示不同的评价指标, 不同图例对应于浮动网格边长 L 不同取值情况下的测试结果. 测试结果表明, 随着 L 的增加, 精确率逐渐增加, 召回率逐渐下降, $F1$-*score* 在 L=114 m 时取得最好分数. 当浮动网格较小时, 在对轨迹点进行候选路段筛选时, 所确定的筛选区域即浮动网格 4 个顶点对应的 Geohash 网格范围相对较小, 有利于核心位置异常点的判断, 造成漏检的情况较少; 但同时易将一些偏离路段较远的正常轨迹点错误的标定位隐性位置异常轨迹点, 这样就存在纳伪现象, 造成精确率偏低. 当浮动网格较大时, 纳伪现象减少, 精确率呈现上升趋势, 但同时轨迹点候选路段筛选区域变大, 容易造成漏检, 因此召回率则呈现下降趋势. 在测试中, 当令浮动网格边长为 L=114 m 时, $F1$-*score* 可达到最高值即 0.91, 此时在准确率和召回率实现了折中均衡.

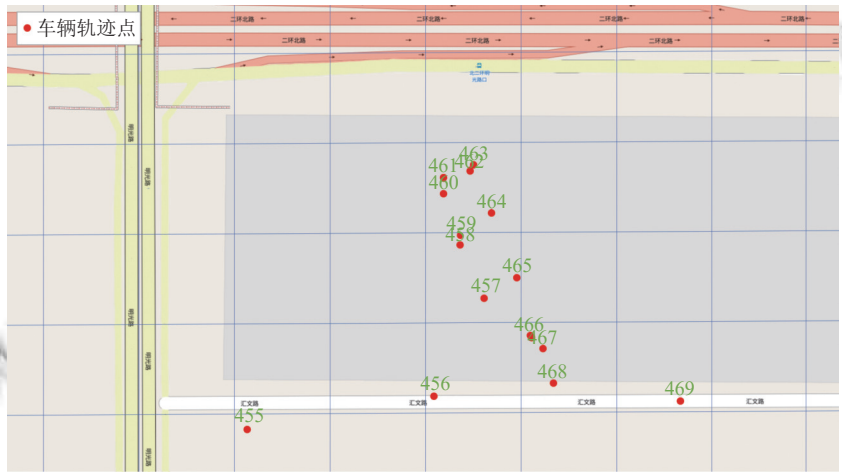

图 3 部分轨迹点的隐性位置异常点检测示意图

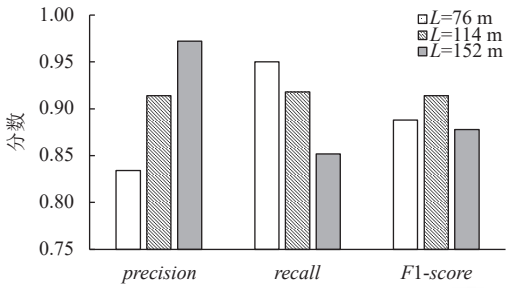

图 4 不同 L 各评价指标比较

3.3 检测方法并行化性能测试

(1) 测试环境

为了能够处理大规模轨迹数据的隐性位置异常点检测, 本文在第 2.4 节提出了一种并行化隐性位置异常轨迹数据检测方法, 该方法在目前流行的大数据计算框架 Spark 环境中进行实现. 为了模拟大数据集群环境, 本文在 Docker 容器中部署了 Spark+HDFS 集群环境, 包括 6 个 Spark Worker 节点和 3 个 HDFS DataNode 节点, 其中 Spark Worker 节点负责所提方法的分布式并行计算, HDFS DataNode 节点负责测试数据的分布式存储. 测试的硬件环境为 4 核 CPU, 16 GB 内存, 1 TB 硬盘.

(2) 测试数据及数据分区数的确定

为了检测本文所提的并行化隐性位置异常轨迹数据检测方法的性能, 本文采集了西安市 589 辆出租车共计 150 万条轨迹点数据.

在分布式环境进行计算时, HDFS 分区数对计算性能也会产生较大影响. 本文首先对 150 万条轨迹点数据, 在 HDFS 设置了 3, 4, 5, 6, 7 个分区数的情况下, 分别测试了本文所提方法的执行时间, 测试结果如图 5 所示.

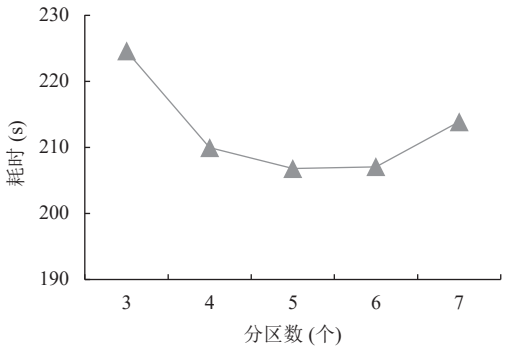

图 5 各个 HDFS 分区数条件下的时间性能测试

由测试结果显示当 CPU 核数固定的情况下, 当分区数设置较大时, 则对于各个分区的计算就需要付出

额外的调度时间损耗; 当分区数设置较小时, 则单个分区的数据量较大, 单节点的计算延迟就会增加, 从而导致总体计算延迟的增加. 在本次测试中, 当 HDFS 分区数为 5 时, 时间性能最好.

(3) 并行计算加速比测试及结果分析

本文将 HDFS 的分区数设置为 5, 分别采用 30 万, 60 万, 90 万, 120 万, 150 万条轨迹点数据进行加速比指标测试. 加速比指标的计算采用如下公式:

$$sp = \frac{t_{\text{single}}}{t_{\text{parallel}}} \tag{6}$$

其中, t_{single}, t_{parallel} 分别表示单机和并行化计算延迟.

不同数据规模条件下的单机和并行化计算延迟及加速比测试结果如表 2 所示, 并在图 6 中进行展示. 从中可以看出, 单机和并行方法的计算延迟都随数据量的增长而增长, 但并行方法增长幅度较单机慢; 并且在同等规模的隐性位置异常轨迹数据检测上, 并行化方法比单机方法要快接近一倍, 那是因为检测方法的并行化能够并行化数据, 同时利用多个计算资源进行任务处理, 提高了计算效率, 能够较好地适应大规模隐性位置异常轨迹数据的检测需要.

表 2 各种条件下的计算延迟及加速比测试结果

轨迹点总数 (万)	单机计算延迟 (s)	并行计算延迟 (s)	加速比
30	84.16	53.45	1.57
60	162.11	87.1	1.86
90	262.52	123.44	2.13
120	387.67	176.67	2.19
150	496.07	211.09	2.35

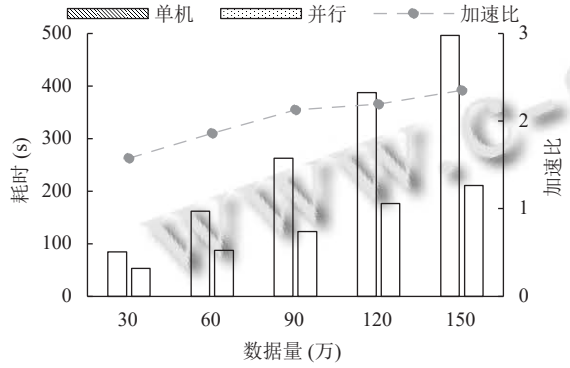

图 6 单机和并行化方法的时间消耗及加速比对比

4 结论

在对大量实际轨迹数据进行分析研究的过程中, 我们发现了一种特殊的位置异常轨迹数据并将其称之为隐性位置异常轨迹数据, 其移动特征参数往往在正常的阈值范围内, 但实际这些轨迹数据并不是车辆在路网路段上行驶产生的, 而是在某些特定区域的内部道路上行驶产生的, 这些内部道路在电子地图的路网数据中实际并不存在. 隐性位置异常轨迹数据通过常规的异常轨迹检测方式难以发现, 同时路网中根本不存在和这些隐性位置异常轨迹数据对应的真实路段, 如果直接对其进行地图匹配则显然又会出错, 从而对后续的轨迹数据分析产生负面影响. 根据隐性位置异常轨迹数据的局部聚集性特征, 本文提出了面向隐性位置异常轨迹数据的检测方法并实现了并行化处理, 测试结果显示了所提方法的有效性. 异常轨迹数据检测是轨迹数据分析的一个基本环节, 本文所提方法是对常规的异常轨迹数据检测方法的一种有益补充. 同时, 本文所提方法仅利用了轨迹数据的时空特征进行异常检测, 结合更多的语义特征以实现更为高效的异常轨迹检测是下一步的研究方向.

参考文献

1 Belhadi A, Djenouri Y, Lin JCW, *et al*. Trajectory outlier detection: Algorithms, taxonomies, evaluation, and open challenges. ACM Transactions on Management Information Systems, 2020, 11(3): 16. [doi: 10.1145/3399631]

2 康军, 杜锦光, 段宗涛, 等. 基于浮动网格的路段检索方法. 计算机系统应用, 2022, 31(12): 259–265. [doi: 10.15888/j.cnki.csa.008856]

3 周洋, 杨超. 基于时空聚类算法的轨迹停驻点识别研究. 交通运输系统工程与信息, 2018, 18(4): 88–95. [doi: 10.16097/j.cnki.1009-6744.2018.04.014]

4 Hsueh YL, Chen HC, Huang WJ. A hidden Markov model-based map-matching approach for low-sampling-rate GPS trajectories. Proceedings of the 7th IEEE International Symposium on Cloud and Service Computing (SC2). Kanazawa: IEEE, 2017. 271–274. [doi: 10.1109/SC2.2017.52]

5 Hsueh YL, Chen HC. Map matching for low-sampling-rate GPS trajectories by exploring real-time moving directions. Information Sciences, 2018, 433–434: 55–69. [doi: 10.1016/j.ins.2017.12.031]

6 Ebel P, Göl IE, Lingenfelder C, *et al*. Destination prediction based on partial trajectory data. Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV). Las Vegas: IEEE, 2020. 1149–1155. [doi: 10.1109/IV47402.2020.9304734]

7 Mohamed R, Aly H, Youssef M. Accurate real-time map matching for challenging environments. IEEE Transactions on Intelligent Transportation Systems, 2017, 18(4): 847–857. [doi: 10.1109/TITS.2016.2591958]

8 Patil V, Singh P, Parikh S, *et al*. GeoSClean: Secure cleaning of GPS trajectory data using anomaly detection. Proceedings of the 2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR). Miami: IEEE, 2018. 166–169. [doi: 10.1109/MIPR.2018.00037]

9 Ma BD, Liu HQ, Fang HE. Trajectory prediction method of millimeter-wave radar based on markov model for roadside installation scenario. Proceedings of the 2022 IEEE Conference on Telecommunications, Optics and Computer Science (TOCS). Dalian: IEEE, 2022. 1206–1211. [doi: 10.1109/TOCS56154.2022.10016148]

10 Geohash.org. Tips & Tricks. http://geohash.org/site/tips.html. (2023-03-15).

11 康军, 郭佳豪, 段宗涛, 等. 大规模轨迹数据并行化地图匹配算法. 测控技术, 2019, 38(2): 98–102. [doi: 10.19708/j.ckjs.2019.02.021]

12 Goutte C, Gaussier E. A probabilistic interpretation of precision, recall and *F*-score, with implication for evaluation. Proceedings of the 27th European Conference on Information Retrieval. Santiago de Compostela: Springer, 2005. 345–359. [doi: 10.1007/978-3-540-31865-1_25]

(校对责编: 孙君艳)