第 46 卷 第 7 期 计 算 机 工 程 2020 年 7 月
Vol. 46 No. 7 Computer Engineering July 2020

·人工智能与模式识别· 文章编号：1000-3428(2020)07-0072-06 文献标志码：A 中图分类号：TP18

OPTICS 与离线批处理在轨迹聚类中的应用

郭 雨[1,2]，陈金勇[2]，张新宇[1]，李 梁[1]，孙未未[1]

（1. 复旦大学 计算机科学技术学院，上海 201203；2. 中国电子科技集团公司航天信息应用技术重点实验室，石家庄 050081）

摘 要： 轨迹聚类是时空轨迹处理中的重要步骤，常用的轨迹聚类算法如 TRACLUS 算法，时间复杂度通常较高且对输入参数敏感，在寻找最优参数的过程中会消耗大量的时间。针对该问题，对 TRACLUS 算法运用离线批处理技术与 OPTICS 算法进行改进，在缓解输入参数敏感性的同时，减少对多组参数进行轨迹聚类的时间，从而减轻人为参数调试的工作量。实验结果表明，在最优参数未知需要对多组参数进行测试时，改进算法可使运行效率得到大幅提升。

关键词： 时空轨迹聚类；密度聚类；OPTICS 算法；离线批处理；TRACLUS 算法

开放科学（资源服务）标志码（OSID）：

中文引用格式： 郭雨，陈金勇，张新宇，等. OPTICS 与离线批处理在轨迹聚类中的应用[J]. 计算机工程，2020，46(7)：72-77，83.

英文引用格式： GUO Yu，CHEN Jinyong，ZHANG Xinyu，et al. Application of OPTICS and offline batch processing in trajectory clustering[J]. Computer Engineering，2020，46(7)：72-77，83.

Application of OPTICS and Offline Batch Processing in Trajectory Clustering

GUO Yu[1,2]，CHEN Jinyong[2]，ZHANG Xinyu[1]，LI Liang[1]，SUN Weiwei[1]

（1. School of Computer Science，Fudan University，Shanghai 201203，China；
2. CETC Key Laboratory of Aerospace Information Applications，Shijiazhuang 050081，China）

【Abstract】 Trajectory clustering is an important step in spatio-temporal trajectory processing. Common trajectory clustering algorithms，such as TRACLUS algorithm，usually have high time complexity and are sensitive to input parameters，thus consuming a lot of time to find optimal parameters. In order to solve this problem，this paper improves the TRACLUS algorithm by using offline batch processing technology and OPTICS algorithm. This optimization reduces the sensitivity of input parameters and the time for trajectory clustering of multiple sets of parameters，so the workload of the manual parameter debugging is reduced. Experimental results show that the time efficiency of the algorithm has been greatly improved when the optimal parameters are unknown and multiple sets of parameters need to be tested.

【Key words】 spatio-temporal trajectory clustering；density clustering；OPTICS algorithm；offline batch processing；TRACLUS algorithm

DOI： 10.19678/j.issn.1000-3428.0054309

0 概述

计算机网络技术在各个领域的广泛应用以及 GPS 定位、传感器、卫星遥感等先进定位技术的不断发展，形成了大量时空轨迹数据，如许多基于位置的服务（Location Based Service，LBS）应用积累了丰富的用户出行路线与移动轨迹数据。这些大规模的数据可以弱化数据质量低下的缺点[1]，使得从中提取出隐含信息并演化出新型应用成为可能。由于服务都建立在对大量时空轨迹数据简化分析和高效提取的基础上，因此如何高效地利用这些数据便相应成为数据分析的热点问题。

轨迹模式挖掘是通过大量的空间轨迹分析移动物体的移动模式，其由包含特定模式的单个轨迹或具有相似模式的一组轨迹来表示。为了找到由不同运动物体共享的代表性路径或共同趋势，通常需要

基金项目： 国家自然科学基金面上项目（61772138）；中国电子科技集团公司航天信息应用技术重点实验室开放基金。

作者简介： 郭 雨（1997—），男，硕士研究生，主研方向为空间数据处理、LBS 定位；陈金勇，研究员；张新宇、李 梁，硕士研究生；孙未未，教授。

收稿日期： 2019-03-20 **修回日期：** 2019-07-04 **E-mail：** yguo15@fudan.edu.cn

对类似的轨迹进行聚类。轨迹聚类在数据分析中发挥了至关重要的作用,因为它在揭示物体移动潜在趋势的同时为行为模式的刻画提供支持,并且可以用于旅行路线建议、出行模式理解以及后续位置预测等诸多方面[2-3]。

在轨迹聚类问题上,许多不同类型的聚类方法被提出应用于复杂的实际情况。文献[4-5]为在道路监控时检测异常车流提出的在线轨迹聚类方法,避免了传统的"两步聚类",而是在获取数据时使用树形结构在线更新聚类结果,构建概率转移模型以判断异常数据。也有研究者提出对数据进行二次筛选达到降载的目的[6]。为了对受道路网络约束的轨迹数据进行聚类,研究者又提出适应道路网络视点的距离度量法,并将其应用于新的轨迹聚类算法[7-9]。针对隐私保护需求,文献[10-11]对传统轨迹聚类算法进行改进,使其能够利用更多的信息。此外,也有将轨迹转换为特征向量[12]、加入隐马尔可夫模型[13-14]或者运用最长公共子序列(LCSS)进行聚类[15]的算法,且都在不同领域得到了应用。

关于适用于轨迹分析的聚类方法,研究者发现:与传统的 k-means 聚类相比,基于密度的聚类算法有明显优势[16],例如:聚类结果可以自然扩展形成任意形状;能够发现任意数量的类别以更好地适应源数据;能够很好地处理数据中的噪声。文献[17]提出基于密度的聚类算法 TRACLUS,通过轨迹分割、聚类、生成 3 个步骤进行轨迹聚类并得到对应的频繁模式。TRACLUS 算法不需要考虑轨迹的时间因素,即使一些运动对象从未同时在一起移动,也有可能被分在同一组聚类。该算法不以一条完整的轨迹为单位进行聚类,而是通过轨迹分割将轨迹转化为若干条线段,再对线段进行聚类,因此,生成的聚类结果更为灵活多变。TRACLUS 的算法思想也被用于轨迹离群点检测[18]、轨迹分类[19]以及增量聚类等工作[20]。

TRACLUS 在聚类时使用传统的 DBSCAN 算法,需要人为设定参数 ε 与 *MinLns*,并且由于实际需求的不同,可能会得到不同疏密程度的聚类结果,此时参数的设定几乎完全取决于工程师的经验。同时,由于每生成一次聚类结果都需要对所有数据运行一遍完整算法,这增加了在大规模数据集上调试参数的难度。也有类似 TRACLUS 的算法,如文献[21]算法,其先将轨迹分割为线段,再将线段投影至一维空间并做聚类,最后在聚类完成的线段中提取运动模式。该算法的聚类过程在一维空间中实现,避免了在二维空间中线段聚类的许多实际困难,但相应也会损失一定的可用信息。一些学者将改进的 Hausdorff 距离与离散 Fréchet 距离作为 TRACLUS 算法的距离度量,考虑了更多的影响因素,获得了更有效的聚类结果[22-24],但这些尝试都未关注运行速度的优化。

本文将 OPTICS[25] 这一基于 DBSCAN 的改进算法用于优化 TRACLUS 算法,同时通过引入离线批处理过程使其能够自动给出若干可选的参数并生成对应的聚类结果,从而减少工程师在调整参数上需要花费的时间与精力。

1　形式化表示及问题定义

1.1　形式化表示

轨迹 T_i 是一个多维点序列,通常表示为 $\{p_1, p_2, \cdots, p_{l_i}\}$,其中,$l_i$ 是轨迹 T_i 的长度,p_j 是一个多维点。本文算法不使用轨迹的时间信息,因此不需要特意记录 p_j 的时间信息。

轨迹序列集合 I 是轨迹的无序集合,通常表示为 $\{T_1, T_2, \cdots, T_{n_I}\}$,其中 n_I 是集合中轨迹的数量。

轨迹分割针对一个线段 p_ip_j,其中 p_i 和 p_j 是从同一轨迹中选择的点,通过从一条轨迹中选择若干个点(记为 n'个),取其中任意相邻两点可以得到共 $(n'-1)$ 个轨迹分割结果。

聚类 O 是一个轨迹分割的集合,属于同一个聚类的轨迹分割在设定的距离下相互接近。

代表轨迹 R_i 是一个多维点的序列 $\{p_1, p_2, \cdots, p_{l'_i}\}$,其中 l'_i 是轨迹 R_i 的长度,用于表示一个聚类中所有线段主要行为虚构的轨迹。

1.2　问题定义

根据给定的轨迹序列集合 I,算法需要生成一个聚类的集合 $O=\{C_1, C_2, \cdots, C_{n_I}\}$,以及每个聚类对应的一个代表轨迹。

2　TRACLUS 算法

TRACLUS 算法分轨迹分割、聚类和生成 3 个子部分进行轨迹聚类[17]。首先应用最小描述长度(MDL)原理,将原序列分解成代表原轨迹序列的特征线段集合;然后通过 DBSCAN 得到线段的聚类;最后对每一个聚类生成一个对应的代表轨迹。算法第一部分轨迹划分的时间复杂度为 $O(N)$,第二部分线段聚类的时间复杂度为 $O(N^2)$,如果使用合理索引如 R-tree,可将时间复杂度降至平均 $O(N\log_a N)$,第三部分特征轨迹的生成需要经过排序,因此,复杂度为 $O(N\log_a N)$。显然,在不使用高效的索引时,第二部分为整个算法的复杂度瓶颈,达到了 $O(N^2)$,而当使用高效的索引时,虽然第二和第三部分复杂度一致,但由于复杂度常数上 R-tree 远大于排序,因此第二部分依然是整个算法的瓶颈。

此外,由于 DBSCAN 包含 2 个人为控制的参数 *MinLns* 与 ε,分别代表邻域内的对象个数与邻域大小,用于共同决定邻域的密度。它们的确定需要依靠数据分析任务的实际需求和数据分析工程师的经验,但实践中工程师可能会在这其中耗费大量无意义的时间而又难以保证其设置的合理性。为了

在大规模数据上运用该算法,本文将对该部分进行优化。

由于本文的目的在于对 TRACLUS 算法第二部分(使用 DBSCAN 对线段进行密度聚类)的优化,因此算法第一部分(轨迹分割)与第三部分(代表轨迹的生成)[17]的具体实现本文不再赘述。

3 聚类算法优化

3.1 基于 OPTICS 的密度聚类优化

DBSCAN 存在一个一般化的改进版本,被称为 OPTICS[25]。OPTICS 并不直接生成数据集聚类,而是先为聚类分析生成一个有序排列,再对比有序排列进行后续处理以得到聚类结果。这个有序排列代表了各样本点基于密度的聚类结构,数据越接近,越有可能被分在同一聚类的样本,在排列中的位置就越靠近。以样本点输出次序为横轴,以可达距离为纵轴,可通过实验得到如图 1 所示的输出结果。

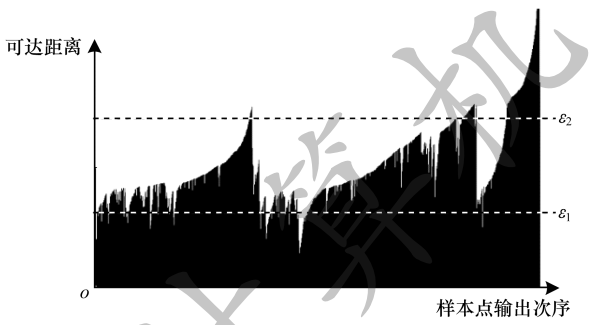

图 1 OPTICS 有序排列输出结果

Fig. 1 Ordered arrangement output result of OPTICS

样本的聚类在图 1 中体现为低谷,当以参数 ε_2 为阈值时,有序排列被划分为 3 个小于 ε_2 的低谷,每个低谷对应一个样本聚类;而当以更小的参数 ε_1 为阈值时,有序排列被划分为多个小于 ε_1 的低谷,同样一个低谷对应一个样本聚类。因为在对多个不同的参数 ε 生成聚类时,所使用的有序排列是完全相同的,所以如果根据有序排列生成聚类的运行时间小于传统 DBSCAN 算法的运行时间,那么当 ε 阈值数量增多时,算法的运行速度就会得到改善。

传统 DBSCAN 算法[26]十分经典,本文不再赘述。OPTICS 在 DBSCAN 的基础上,增加了核心距离和可达距离[25]2 个新的定义:

定义 1(核心距离) 一个对象 p 在参数 ε 与 *MinLns* 下的核心距离,等于其第 *MinLns* 近的对象到对象 p 的距离。如果对象 p 的 ε 邻域中对象个数小于 *MinLns*,那么对象 p 在参数 ε 与 *MinLns* 下的核心距离为 UNDEFINED。

定义 2(可达距离) 对象 o 和对象 p 在参数 ε 与 *MinLns* 下的可达距离为 max(核心距离(o),*distance*(o,p))。如果对象 p 的 ε 邻域中对象个数小于 *MinLns*,那么对象 p 在参数 ε 与 *MinLns* 下的可达距离为 UNDEFINED。

生成有序排列的函数伪代码如下:

算法 1 ExpandClusterOrder

```
输入  objects[ ]
输出  ResultList[ ],ReachDist[ ],CoreDist[ ]
Initialize:
Visited[ ] = [ false]
temp = 1
while temp < = objects. size do
m = null
i = 1
while i < = objects. size do
if visited[ i] = = false then
m = i
end if
end while
visited[ m] = true
ResultList[ temp] = m
CoreDist[ m] = FindMinLnsthDistance( )
for each i in EpsNeighborhood( m) do
ReachDist_ = max( CoreDist[ m] ,dist( i,m) )
if ReachDist[ i] > ReachDist_ then
if visited[ i] = = false then
ReachDist[ x] = ReachDist_
end if
end if
end for
temp = temp + 1
end while
```

ExpandClusterOrder 函数每次选择一个未被选择的对象 m,且满足 *ReachDist*[m](之前已被选择的对象到对象 m 的最近距离)为最小值。将对象 m 添加到结果序列,并标注其已被选择。FindMinLnsthDistance( )函数会计算对象 m 的核心距离,随后枚举对象 m 的 ε 邻域,用 m 到其他对象的可达距离更新 *ReachDist* 数组。如此循环直到所有对象都被选择。

根据定义 1 与定义 2 可知,OPTICS 使用的参数 ε 是一个限制邻域大小上限的宽松上界,过大的参数 ε 仅可能拖慢运行时间,而不会影响之后生成的聚类质量。如果将参数 ε 设置为正无穷,那么核心距离与可达距离均不会出现 UNDEFINED,此时需要枚举所有对象对并计算它们的间距以得到核心距离并更新可达距离,复杂度退化到 $O(N^2)$,与不使用高效空间索引的 DBSCAN 相同。如果根据实际需求确定了参数 ε 的上限(并不需要十分精确),那么 OPTICS 也可以使用高效空间索引,在 $O(\log_a N)$ 的时间内得到第 *MinLns* 近的对象并更新邻域的可达距离,从而将总的时间复杂度降低到 $O(N\log_a N)$。因此,相对于 DBSCAN 而言,OPTICS 的时间复杂度并没有增加,参数 ε 设置不再敏感。

3.2　基于有序排列的聚类

虽然已经获得了对聚类元素的有序排列,但为了得到最终需要的密度聚类结果,仍需要对此排列进行分析。如果已知一个 DBSCAN 参数 ε 的取值,通过算法 2 可以得到对应的密度聚类结果。

算法 2　ExtractDBSCANClustering

输入　*ResultList*,*ReachDist*,*CoreDist*

输出　*mark*[ ]

```
Initialize:
mark[ ] = [UNCLASSIFIED]
clusterID = 0
i = 1
while i < = ResultList do
x = ResultList[i]
if ReachDist[x] > eps then
if CoreDist[x] < = eps then
clusterID = clusterID + 1
mark[x] = clusterID
else
mark[x] = NOISE
end if
else
mark[x] = clusterID
end if
end while
```

ExtractDBSCANClustering 函数依序枚举有序排列中的每一个对象,如果当前被枚举对象 x 的可达距离小于参数 ε,那么将对象 x 与前一个对象合并到同一个聚类中;否则,如果对象 x 的核心距离小于参数 ε,则将对象 x 标记为一个新的聚类。当可以合并时,将对象 x 与其前一个对象合并是合理的[25],具体证明此处不再赘述。由于 ExtractDBSCANClustering 函数只对有序排列进行一次扫描,因此时间复杂度为 $O(N)$。

3.3　参数 ε 的生成

使用 OPTICS 替代 DBSCAN 用于密度聚类,可以大幅缩短多组参数存在时的运行时间,但在 ExtractDBSCANClustering 阶段依然需要手动指定参数 ε。如果能够根据中间结果自动生成可供参考的参数 ε,则能大幅减少参数调整的工作量。

考虑密度聚类中参数 ε 的定义[26],即用于确定聚类对象的邻域大小,聚类对象间的距离整体越大,通常就需要越大的邻域用以判断疏密(过小的邻域将难以包含足够的对象以至于随机性过大),而聚类对象间的距离整体越小,就需要越小的邻域(过大的邻域将包含过多的对象以至于密度趋近于整体平均密度,从而失去代表性)。

通过分析算法 2 可以发现,两个相邻的对象是否被合并到同一个聚类,仅与它们的可达距离有关,而核心距离仅影响了在未被合并时是否将其判断为噪声。因此,参数 ε 与聚类对象可达距离的分布有关。可达距离的分布示例如图 2 所示,其中横轴为可达距离,纵轴为等于可达距离的聚类对象数量。

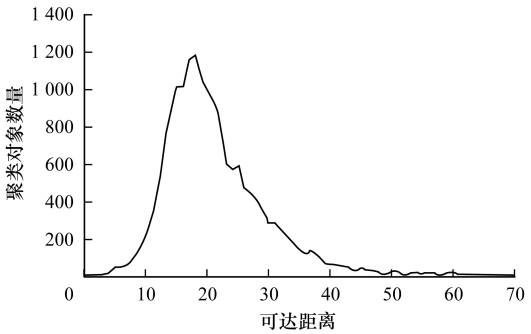

图 2　聚类对象可达距离分布

Fig. 2　Reachable distance distribution of clustering objects

本文用正态分布拟合此分布,并计算 N 等分点的数值(N 为任意设定的整数),将它们作为一系列可选的参数 ε 返回。虽然越靠近边界(如 10% 与 90% 处)的数值通常被需要的可能性越小,但本文依然将其作为参数纳入考虑,因为 ExtractDBSCANClustering 的时间复杂度仅为 $O(N)$,相对于 ExpandClusterOrder 的时间复杂度 $O(N^2)$ 或 $O(N\log_a N)$ 显得微不足道。

3.4　离线批量聚类

OPTICS 算法的 ExtractDBSCANClustering 函数运行一次只能对一组参数生成对应的聚类结果,面对需要对多组参数分别生成聚类结果的需求,仍有一定的优化空间。

上文 3.2 节已经提到,在 OPTICS 的有序排列中相邻的两个对象,后者或者与前者合并到同一个聚类,或者成为一个新的聚类,或者成为噪声。如果将噪声视为仅含有一个对象的聚类(在正常的密度聚类中,不会出现只含有一个对象的聚类,它无法扩展也无法被扩展,因此没有分析意义),那么相邻的两个对象是否被合并仅与可达距离有关。随着参数 ε 的逐渐增大,可达距离越小的对象将会越早被合并,已经被合并到同一个聚类内的对象不会再被拆分至两个聚类。如图 1 所示,在参数 ε 从 ε_1 逐渐增加的过程中,原先被几个峰分隔的许多个聚类逐渐合并,直至增加到 ε_2 时只剩下 3 个聚类。

正是由于只存在集合合并的操作,并查集很适合被用于维护这些聚类之间的关系。在将可达距离与参数 ε 按升序排序后,当参数 ε 增大后,小于参数 ε 的可达距离也相应增加,只需要将这些新增的相邻对象合并,即可得到对应新参数 ε 的聚类结果。上述操作的算法伪代码如下:

算法 3　offlineExtractClustering

输入　*ResultList*,*ReachDist*

输出　*cluster*[ ]

```
Initialize:
DistList = sort(ReachDist)
i = 1
while i < = size do
mark[i] = i
end while
```

```
cur = 1
for each eps in EpsList do
cur = RenewClustering(cur,eps,DistList)
cluster[eps] = mark
end for
```

算法 4 RenewClustering

输入 *cur*,*eps*,*DistList*

输出 *cur*

```
while cur < DistList. size do
if DistList[cur] > eps then
break;
end if
x = DistList[cur]. from
mark. merge(x,x - 1)
cur = cur + 1
end while
```

上述代码对 *ReachDist* 数组进行从小到大排序，并存储在 *DistList* 数组中。从小到大枚举每一个候选的 ε，每次调用 RenewClustering 函数以更新 *mark* 数组。RenewClustering 函数从当前 *cur* 位置开始扫描 DistList 内的对象，如果当前对象的 *ReachDist* 小于参数 ε，那么说明对应的位置需要进行一次合并，调用并查集 merge 函数。如果可达距离大于参数 ε，那么之后的可达距离也均大于 ε，此次更新就到此结束，返回移动后的 *cur*。

可以看到，该部分对所有待计算的 ε 取值都计算出了对应的聚类结果，且所有对象仅被单调扫描了一次，因此其复杂度为 $O(\alpha N)$（其中 α 为并查集效率常数）。而在传统 OPTICS 的 ExtractDBSCANClustering 过程中，该部分的复杂度为 O(运行次数 $\times N$)。

3.5 时间复杂度分析

传统 TRACLUS 算法时间复杂度瓶颈在第二部分的线段密度聚类上，因为使用了 DBSCAN，所以在对多组不同的参数进行密度聚类时，时间复杂度为 O(参数组数 $\times N^2$)，即使使用了合理的空间索引，时间复杂度也为 O(参数组数 $\times CN\log_a N$)（其中 C 为空间索引带来的巨大常数）。当数据规模较大且参数组数较多时，未经优化的算法会严重拖长运行时间。

经过优化后，DBSCAN 被 OPTICS 替代，原本对每组参数需要扫描所有数据并计算它们的间距，现在可以一次完成，复杂度降为 $O(CN\log_a N$ + 参数组数 $\times N)$，其中前部分是生成有序排列的代价，后部分是从排列中对每组参数生成对应聚类结果的代价。经过离线批处理优化，从有序排列生成对应聚类结果的过程被进一步合并，复杂度降为 $O(CN\log_a N + \alpha N$ + 参数组数)（其中 α 为并查集的时间复杂度常数，通常小于 5）。可以发现，优化后第二部分的时间复杂度几乎不受参数组数影响，这在需要对多组参数进行计算的情况下是较大的改进。

4 实验与结果分析

本文实验环境为单个计算机，处理器型号 Intel(R) Core(TM) i5-4690K CPU @ 3.50 GHz，内存大小为 16 GB，操作系统为 Win10(64 位)，使用相同软件环境。

本文在 Starkey 项目提供的 1995 年鹿类动物迁徙轨迹数据集(http://www.fs.fed.us/pnw/starkey/data/tables/)上进行对比实验，该数据集是相关问题的经典数据集，TRACLUS 原论文也在该数据集下进行实验。实验结果表明，在生成相同质量聚类结果的同时，本文所提出的优化算法花费时间小于原算法，由此说明本文的优化确实对其有所改进。为了对比，2 种算法都没有使用诸如 R-tree 数据结构等额外的优化结构。

2 种算法一次性生成 N 张聚类结果所需要消耗的时间比较如表 1 所示(当参数 ε 组数 N 大于 33 时，传统 TRACLUS 算法的运行时间过长，不再具有比较的意义)。可以看出，TRACLUS 所花费的时间与参数 ε 组数近似呈正比例关系，而使用离线与 OPTICS 优化后的 TRACLUS，虽然在仅有一组 ε 参数时花费的时间高于 TRACLUS(因为有更大的常数)，但随着参数 ε 的取值数量增长，算法运行的时间增长缓慢，远低于直接使用 TRACLUS 进行聚类的耗费。正如之前的分析，改进后的算法节省了大量重复计算过程，在同时对多组 ε 参数进行计算时具有明显的速度优势。

表 1 多组参数情况下的聚类时间比较

Table 1 Comparison of clustering time with multiple parameters ms

N(参数 ε 组数)	原 TRACLUS 算法	本文优化算法
1	23 441	29 316
2	44 966	28 803
3	66 280	29 895
5	109 078	30 135
9	197 894	31 383
17	372 399	33 322
25	547 752	35 604
33	—	37 850
41	—	39 547
57	—	44 754
81	—	50 323

可以看出，因为 TRACLUS 的第三部分仍要对聚类结果进行再加工，所以优化后的算法聚类时间依然随着参数 ε 组数增加而保持缓慢增长，但这个增长十分缓慢，以至于生成对应 81 组参数的聚类结果所花费的时间，也未超过生成对应 1 组参数的聚类结果所花费时间的两倍。

5　结束语

本文运用 OPTICS 算法与离线批处理技术改进 TRACLUS 轨迹聚类算法。优化后的算法从需要两个外部参数降低到仅需要其中一个,即能够自动生成可供参考的参数组并输出对应多种密度的聚类结果,同时只额外消耗极少的时间,从而提高效率。实验结果表明,在最优参数未知需要对多组参数进行轨迹聚类时,改进算法在运行速度上较原算法具有明显优势。本文仅对 TRACLUS 的密度聚类部分进行改进,在未来工作中,将对轨迹划分和代表轨迹生成部分进行优化,进一步提升轨迹聚类的效率及质量。

参考文献

[1] GIANNOTTI F, NANNI M, PEDRESCHI D, et al. Unveiling the complexity of human mobility by querying and mining massive trajectory data[J]. The VLDB Journal, 2011, 20(5): 695-719.

[2] YU Zheng. Trajectory data mining: an overview[J]. ACM Transactions on Intelligent Systems and Technology, 2015, 6(3): 1-29.

[3] PARENT C, SPACCAPIETRA S, RENSO C, et al. Semantic trajectories modeling and analysis[J]. ACM Computing Surveys, 2013, 45(4): 1-42.

[4] PICIARELLI C, FORESTI G L. On-line trajectory clustering for anomalous events detection[J]. Pattern Recognition Letters, 2006, 27(15): 1835-1842.

[5] FU Zhouyu, HU Weiming, TAN Tieniu. Similarity based vehicle trajectory clustering and anomaly detection[C]// Proceedings of IEEE International Conference on Image Processing. Washington D. C., USA: IEEE Press, 2005: 602-605.

[6] WU Peili, LIU Kui'en, HAO Shengang, et al. Rapid traffic congestion monitoring based on floating car data[J]. Journal of Computer Research and Development, 2014, 51(1): 189-198. (in Chinese)
吴佩莉,刘奎恩,郝身刚,等. 基于浮动车数据的快速交通拥堵监控[J]. 计算机研究与发展,2014,51(1):189-198.

[7] ROH G P, HWANG S W. NNCluster: an efficient clustering algorithm for road network trajectories[C]// Proceedings of International Conference on Database Systems for Advanced Applications. Berlin, Germany: Springer, 2010: 47-61.

[8] WON J I, KIM S W, BAEK J H, et al. Trajectory clustering in road network environment[C]//Proceedings of IEEE Symposium on Computational Intelligence and Data Mining. Washington D. C., USA: IEEE Press, 2009: 299-305.

[9] HAN B, LIU L, OMIECINSKI E. NEAT: road network aware trajectory clustering[C]//Proceedings of the 32nd IEEE International Conference on Distributed Computing Systems. Washington D. C., USA: IEEE Press, 2012: 142-151.

[10] WU Yingjie, TANG Qingming, NI Weiwei, et al. A clustering hybrid based algorithm for privacy preserving trajectory data publishing[J]. Journal of Computer Research and Development, 2013, 50(3): 578-593. (in Chinese)
吴英杰,唐庆明,倪巍伟,等. 基于聚类杂交的隐私保护轨迹数据发布算法[J]. 计算机研究与发展,2013,50(3):578-593.

[11] CHI Xiangsong, PI Dechang, GUAN Peng. High-density sub-trajectory clustering algorithm for moving objects[J]. Journal of Chinese Computer Systems, 2016, 37(9): 2014-2018. (in Chinese)
迟相松,皮德常,关鹏. 移动对象高密度子轨迹聚类算法[J]. 小型微型计算机系统,2016,37(9):2014-2018.

[12] ANJUM N, CAVALLARO A. Multifeature object trajectory clustering for video analysis[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2008, 18(11): 1555-1564.

[13] ALON J, SCLAROFF S, KOLLIOS G, et al. Discovering clusters in motion time-series data[C]//Proceedings of 2003 IEEE Computer Society Conference on Computer Vision and Pattern. Washington D. C., USA: IEEE Press, 2003: 1-5.

[14] SUN Hong, CHEN Suo. Spatio-temporal trajectory prediction algorithm based on clustering based hidden markov model[J]. Journal of Chinese Computer Systems, 2019, 40(3): 472-476. (in Chinese)
孙红,陈锁. 一种聚类隐马尔可夫模型的时空轨迹预测算法[J]. 小型微型计算机系统,2019,40(3):472-476.

[15] BUZAN D, SCLAROFF S, KOLLIOS G. Extraction and clustering of motion trajectories in video[C]// Proceedings of the 17th International Conference on Pattern Recognition. Washington D. C., USA: IEEE Press, 2004: 521-524.

[16] NANNI M, PEDRESCHI D. Time-focused clustering of trajectories of moving objects[J]. Journal of Intelligent Information Systems, 2006, 27(3): 267-289.

[17] LEE J G, HAN J W, WHANG K Y. Trajectory clustering: a partition-and-group framework[C]// Proceedings of 2007 ACM SIGMOD International Conference on Management of Data. New York, USA: ACM Press, 2007: 593-604.

[18] LEE J G, HAN J, LI X L. Trajectory outlier detection: a partition-and-detect framework[C]//Proceedings of the 24th IEEE International Conference on Data Engineering. Washington D. C., USA: IEEE Press, 2008: 140-149.

[19] LEE J G, HAN J W, LI X L, et al. TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering[J]. Proceedings of the VLDB Endowment, 2008, 1(1): 1081-1094.

[20] LI Z H, LEE J G, LI X L, et al. Incremental clustering for trajectories[C]//Proceedings of IEEE International Conference on Database Systems for Advanced Applications. Washington D. C., USA: IEEE Press, 2010: 32-46.

[21] SUNG C, FELDMAN D, RUS D. Trajectory clustering for motion prediction[C]//Proceedings of 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. Washington D. C., USA: IEEE Press, 2012: 1547-1552.

(下转第 83 页)

(上接第 77 页)

[22] CHEN Jinyang, SONG Jiatao, LIU Liangxu. Trajectory clustering algorithm based on improved Hausdorff distance[J]. Computer Engineering, 2012, 38(17): 157-161. (in Chinese)
陈锦阳,宋加涛,刘良旭. 基于改进 Hausdorff 距离的轨迹聚类算法[J]. 计算机工程,2012,38(17):157-161.

[23] JIANG Yuling, XIONG Zhennan, TANG Jihong. Ship trajectory clustering algorithm based on DBSCAN[J]. Navigation of China, 2019, 42(3): 1-5. (in Chinese)
江玉玲,熊振南,唐基宏. 基于轨迹段 DBSCAN 的船舶轨迹聚类算法[J]. 中国航海,2019,42(3):1-5.

[24] SHI Chonglin, GAN Wenyan, WU Lin, et al. Clustering trajectories of entities in computer wargames[J]. Journal of Software, 2013, 24(3): 465-475. (in Chinese)
石崇林,淦文燕,吴琳,等. 计算机兵棋作战实体轨迹聚类算法[J]. 软件学报,2013,24(3):465-475.

[25] ANKERST M, BREUNIG M M, KRIEGEL H P, et al. Optics: ordering points to identify the clustering structure[J]. ACM SIGMOD Record, 1999, 28(2): 49-60.

[26] ESTER M, KRIEGEL H P, SANDER J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise[C]//Proceedings of the 2nd International Conference on Knowledge Discovery and Data. New York, USA: ACM Press, 1996: 226-231.

编辑　金胡考