软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(Suppl.(2)):225−235

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

到达数据中时空异常聚簇发现*

刘俊岭[1,2], 魏茹玉[2], 于 戈[1], 孙焕良[2], 姚承伟[2]

[1](东北大学 信息科学与工程学院,辽宁 沈阳 110006)

[2](沈阳建筑大学 信息与控制工程学院,辽宁 沈阳 110015)

通讯作者: 刘俊岭, E-mail: liujl@sjzu.edu.cn, http://www.sjzu.edu.cn

摘 要: 在时空数据中有一类表示用户在某一时间到达某一地点的数据——到达数据,到达数据可以是社交网站的签到数据、轨迹数据中的停留点及公共交通中乘客抵达的位置数据,这些数据的聚簇可以反映用户的聚集行为.基于到达数据,提出一类新的时空数据查询——时空异常聚簇发现.将到达数据进行周期性划分,通过时空聚类算法对一个时间段的数据进行聚类,比较不同时间段内聚簇的差异度,发现具有最大簇异常度的前 k 个簇.通过该查询发现的时空异常聚簇可以应用于城市安全管理、基于位置的服务和交通调度等方面.定义了异常簇查询模型,提出了针对任意形状聚簇的簇差异度度量,将异常簇查询转化为二分图最大匹配问题,对二分图构建与匹配进行了优化并提出了高效的查询算法.利用真实数据集进行了充分实验,验证了查询结果的实际意义,评估了所提出的各查询算法在不同参数设置下的查询效率.

关键词: 到达数据;二分图最大匹配;聚类;时空异常聚簇;基于位置的服务

中文引用格式: 刘俊岭,魏茹玉,于戈,孙焕良,姚承伟.到达数据中时空异常聚簇发现.软件学报,2014,25(Suppl.(2)):225−235. http://www.jos.org.cn/1000-9825/14040.htm

英文引用格式: Liu JL, Wei RY, Yu G, Sun HL, Yao CW. Spatio-Temporal abnormal cluster discovery in arrival data. Ruan Jian Xue Bao/Journal of Software, 2014,25(Suppl.(2)):225−235 (in Chinese). http://www.jos.org.cn/1000-9825/14040.htm

Spatio-Temporal Abnormal Cluster Discovery in Arrival Data

LIU Jun-Ling[1,2], WEI Ru-Yu[2], YU Ge[1], SUN Huan-Liang[2], YAO Cheng-Wei[2]

[1](School of Information Science and Engineering, Northeastern University, Shenyang 110006, China)

[2](School of Information and Control Engineering, Shenyang Jianzhu University, Shenyang 110015, China)

Corresponding author: LIU Jun-Ling, E-mail: liujl@sjzu.edu.cn, http://www.sjzu.edu.cn

Abstract: Arrival data is a type of location related data which records the spots and time that users arrive. It can be the check-in data in the social network, stay points in the trajectory or the arrival locations of passengers in the public transport. The clusters of arrival data can reflect the aggregation behavior of users in a particular area. This paper presents a new spatio-temporal data query—Spatio-Temporal Abnormal Clusters Discovery. The new scheme partitions the arrival data into segments with equal timespan periodically. Then using spatio-temporal cluster algorithms, it clusters the data in every segment, and finds k most abnormal clusters by comparing the different degree of clusters. Finding abnormal clusters can be useful in areas such as urban safety management, location based service and transportation scheduling. This article defines the abnormal cluster query model, specifies the difference measurement for the clusters with arbitrary shape and transforms the query to the maximum matching problem of bipartite graphs. Algorithms are designed to improve the efficiency in constructing and matching of bipartite graphs. The experiments on the real datasets validate the application value of the query results and demonstrate the effectiveness of the query algorithm with different parameters.

Key words: arrival data; bipartite graph maximum matching; clustering; spatio-temporal abnormal clusters; location-based service

* 基金项目: 国家自然科学基金(61070024, 61272180); 教育部博士点基金(20120042110028); 教育部-英特尔信息技术专项科研基金(MOE-INTEL-2012-06); 本文部分工作是作者在访问中国人民大学萨师煊大数据研究中心时完成的,该中心获国家高等学校学科创新引智计划(111 计划)资助

收稿时间: 2014-05-07; 定稿时间: 2014-08-19

随着时空数据获取设备的快速发展,如装有 GPS 功能的智能手机、出租车、传感器网络等,轨迹数据[1]、带地理标签的媒体数据[2]及签到数据[3]等时空数据急剧增加[4].基于这些数据的查询与挖掘得到广泛关注,研究成果可以应用于基于位置的服务、选址等领域.

在时空相关数据中可以抽取出一类表示用户在某一时间到达某一地点的数据,包括社交网站的签到数据、轨迹数据中的停留点及公共交通中乘客抵达的位置等数据,我们称其为到达数据(arrival data).到达数据的时空聚簇可以反映对象在一段时间、一定空间范围内的聚集.观察发现,这些聚集往往具有周期性,如上下班时间写字楼区域的聚集、节假日商业区的聚集等.同时也存在一些异常聚集,如广场上临时的集会、非周期性的体育比赛现场、演唱会与招聘会附近的聚集等.发现这些异常聚集对于城市安全管理、基于位置的服务和交通调度等具有重要意义.

基于以上需求,本文提出一类新的时空数据查询:时空异常聚簇发现(SACD).该查询在给定时间和空间范围内发现具有最大簇异常度的前 k 个簇.

本文所提出的时空异常簇发现与现有的聚集模式发现、聚类及异常点检测等存在本质区别,将在第 1 节相关工作中加以讨论.由于时空数据的复杂性,有效发现时空异常簇成为一项具有挑战性的工作,具体体现在时空异常聚簇的定义、度量计算,到达数据的有效聚类和高效的查询算法等方面.

(1) 在异常簇的定义与度量方面,由于采用基于密度的聚类方法所发现的聚簇具有任意形状,现有聚簇间相似度度量包括 MIN、MAX、中心度量等不适用于度量具有任意形状的聚簇[5].因此,本文将聚簇的异常度计算转化为二分图最大匹配问题,利用最大匹配结果来计算聚簇异常度.

(2) 在有效聚类方面,现有的基于密度的方法聚类效果依赖于参数的选择,难以找到合适的参数用于异常簇发现.基于 DBSCAN 并结合 IDBSCAN[6]算法迭代思想,本文提出了 IDBSCAN_T 算法以支持异常簇发现.

(3) 在查询效率方面,由于计算簇异常度时需要进行二分图构建与最大匹配,当对象集较大时构建二分图与最大匹配代价较高.针对时空聚簇的特点,在二分图构建与匹配方面提出了相应的优化策略,并设计了高效的异常簇发现算法.

综上所述,本文主要贡献如下:

(1) 基于到达数据,提出一类新的时空数据查询——时空异常聚簇发现,用于发现人们在时空上的异常聚集.

(2) 设计了新的基于密度的时空聚簇异常度度量,有效衡量聚簇间的差异,并提出新的时空聚类算法 IDBSCAN_T,设计了优化二分图构建与匹配策略,提出了有效算法以解决时空异常聚簇发现问题.

(3) 采用真实数据集进行实验研究,验证了时空异常聚簇发现的有效性,在不同参数设置下对所提出的算法进行了查询效率评价.

1 相关工作

时空对象查询与挖掘广泛应用于地理信息系统、基于位置的服务及设施选址优化等领域[7].典型的时空对象查询与挖掘包括:传统的 KNN 与 RNN 查询[4]、密区域查询[8]、移动对象的聚集模式发现[9,10]、空间对象聚类与异常点检测[11]、基于轨迹的聚类[12,13]和异常轨迹检测[14]等.

移动对象的聚集模式发现是利用对象的轨迹,查询有共同行为与模式的对象组[9].为突破组模式大小和形状的严格限制,Convey[15]聚集模式要求在连续的 k 个时间戳中,利用基于密度的方法发现具有任意形状与范围的轨迹模式.Li 等人在文献[10]中提出了 Swarm 轨迹模式,用以发现在非连续的 k 个时间快照内,分布在同一聚簇中的多个移动对象.文献[9]提出了 Gathering 轨迹模式,要求对象在多个快照簇中出现,但不要求连续性.以上研究均为针对移动对象的轨迹数据,大多采用聚类算法将快照进行分组,检测在多个快照中同时出现的对象.与以上工作中发现聚集模式相比,本文提出的时空异常聚簇不关注组成聚簇的对象,而是发现该聚簇本身的异常特征.

空间对象的聚类与异常点检测是空间数据分析的主要任务之一[16],它将位置上相近的对象进行分组.现有

的簇间差异度度量,如簇间对象最小距离、最大距离等[5],对于偏好球形的聚簇有较好的适用性,但难以度量任意形状的簇间差异.本文在差异度度量方面,采用二分图匹配的方法,较好地适应了簇的形状任意性与密度差异性.异常点检测用于发现小的模式,分析与大多数对象不同的对象[17].文献[11]中,总结了基于聚类的异常点检测方法,它将非异常的数据归为簇,而异常点不属于或归为较小的簇.本文研究不同时间段内聚簇间的关系,通过对比聚簇的差异性来发现异常聚簇.

基于轨迹的时空聚类是时空数据分析的一个研究热点.Lee 等人提出一种聚类轨迹中公共子轨迹的划分与聚合框架[12].Rosswog 等人针对噪声环境中的移动对象,提出一种基于密度的聚类算法 DDBC,通过移动对象的历史关系增加聚类算法的准确性[13].上述研究均针对连续的轨迹数据,将相似的轨迹进行聚类.而本文研究的是表示对象停留特征的到达数据,更关注到达对象的异常聚集特征.

在异常轨迹检测方面,文献[11]用基于距离的异常点检测算法检测数据集中的异常轨迹.该方法适用于路径较短或较简单的轨迹.文献[14]利用 R-Tree 计算异常轨迹,提高了检测算法的查询效率.以上异常轨迹检测方法是通过比较轨迹之间的相似性找出异常轨迹,而本文所提出的时空异常聚簇发现是找出到达数据的异常聚簇,而不是异常轨迹对象.因此,异常轨迹检测方法难以应用于解决本文所提出的时空异常聚簇发现问题.

2 问题定义

到达对象 p 由一个三元组$\langle u,t,loc\rangle$表示,其中,u 为用户标识,t 为到达时间,loc 为空间位置,用经纬度表示.到达数据集 $DB=\{p_1,p_2,\ldots,p_n\}$.时空活动往往具有周期性,因此将整个时间跨度按等分方式进行划分(如自然天,周,月等),得到 m 个时间段,表示为 $\{t_1,t_2,...,t_m\}$,其中 t_i 为一个时段.时空异常簇查询可以在全数据空间进行,也可以指定查询时间段范围$[t_i,t_j]$及矩形空间查询区域 w,将落入该时间范围与空间区域的到达对象按时间段分别聚类,其中,t_i 时间段内聚簇 c_{t_i} 的异常度通过计算 c_{t_i} 与其他时间段中聚簇相异度计算获得.

对给定时间段 t_i 的到达数据对象集 DB_{t_i},用扩展基于密度的方法 DBSCAN 进行聚类,扩展的基于密度聚类算法在原有 DBSCAN 算法的基础上加入时间聚类参数,这里给出时空聚类定义.时间段 t_i 中的到达对象 $p\in DB_{t_i}$,ε'_d 为聚类半径,ε'_t 为聚类时间.到达对象 p 在 DB_{t_i} 中关于 ε'_d 和 ε'_t 的邻域集表示为 $N_{\varepsilon'_d,\varepsilon'_t}(p)=\{q\in DB_{t_j}|D(p.loc,q.loc)\leqslant\varepsilon'_d,T(p.t,q.t)\leqslant\varepsilon'_t\}$,其中,$D()$为欧几里德距离函数,$T()$为时间差函数.

给定到达对象 p,如果$|N_{\varepsilon'_d,\varepsilon'_t}(p)|\geqslant\min Pts$,则 p 为核心对象,其中 $\min Pts$ 为一正整数,表示成为核心对象需要的最小对象数阈值.通过时空可达、时空相连等概念定义时空聚类.

定义 1. 时空聚类.给定到达数据集 DB 中一个时间段 t_i 中的到达对象,距离参数ε'_d,时间参数ε'_t 和 $\min Pts$,时间段内的时空聚类 c_{t_i} 满足:(1) c_{t_i} 中至少存在一个核心对象;(2) $\forall p,q\in c_{t_i},p(t_i)$与 $q(t_i)$关于$\varepsilon'_d,\varepsilon'_t$ 和 $\min Pts$ 时空相连;(3) c_{t_i} 是基于时空可达性的最大时空相连对象集合.

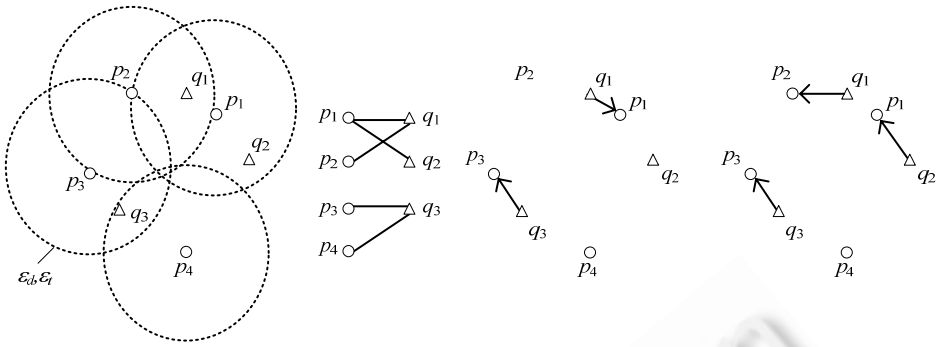
由于采用基于密度的方法得到的聚簇具有任意形状,采用传统的类间相异度[15]的方法难以度量簇的异常度.本文基于到达对象“再现”的概念定义时空聚簇的异常度.

定义 2. 到达对象再现.给定时间段 t_i 中到达对象 p,时间段 t_j 的对象集 DB_{t_j},如果 $N_{\varepsilon_d,\varepsilon_t}(p)$非空,则称 p 在时间段 t_j 中再现,其中,ε_d 为到达对象再现的距离阈值,ε_t 为到达对象再现的时间阈值.

定义 3. 到达对象的异常度.到达对象 p 的异常度为查询时间$[t_i,t_j]$内 p 未再现的次数的比率.

参数ε'_d和ε'_t是判断核心对象的条件,用于聚类;ε_d和ε_t为再现条件,用于异常度计算.通常,$\varepsilon_d\leqslant\varepsilon'_d$且$\varepsilon_t\leqslant\varepsilon'_t$.

簇的异常度表示一个簇相对于其他时间段内的簇分布的特殊程度.本文提出一种二分图最大匹配方法[18]计算簇的异常度.如图 1(a)所示,存在簇 $c_{t_i}\{p_1,p_2,p_3,p_4\}$与 $c_{t_j}\{q_1,q_2,q_3\}$,要计算簇 c_{t_i} 相对于簇 c_{t_j} 的异常度.首先求得簇 c_{t_i} 内所有对象 p_1,p_2,p_3,p_4 关于ε'_d 和ε'_t 的邻域集,分别为$\{q_1,q_2\},\{q_1\},\{q_3\},\{q_3\}$,在此基础上生成二分图,如图 1(b)所示,图中用边将对象与其邻域集内对象相连.邻域集中的任一对象都是 c_{t_j} 中相应对象的再现,但其只能对应 c_{t_i} 中的一个对象.图 1(c)及图 1(d)描述了两种二分图匹配方式.前者采用最近邻匹配,由于在 p_2 的邻域集中仅包含对象 q_1,而 q_1 已经与 p_1 匹配,所以 q_2 未在 c_{t_i} 中形成匹配,因此在图 1(c)中仅有两对对象形成匹配.图 1(d)使用二分图的最大匹配方法,共匹配了 3 对.



(a) 两个聚簇 (b) 二分图 (c) 二分图最近邻匹配 (d) 二分图最大匹配

Fig.1 Calculation of the abnormal value for two clusters

图 1 两个聚类的异常度计算

二分图中的每一对匹配表示了两个对象之间的再现关系,削弱了簇的异常程度,我们称其为一次缩减.本文计算簇相对于簇的异常度时要进行缩减最大化,因此使用最大匹配方法.

定义 4. 聚簇 c_{t_i} 相对于 t_j 时间段的异常度.已知聚簇 c_{t_i} 与一个时间段 t_j 内所有簇的集合 $C_{t_j}\{c_{t_{j1}},c_{t_{j2}},\ldots,c_{t_{jw}}\}$,聚簇 c_{t_i} 相对于 t_j 的异常度$|d_{t_j}(c_{t_i})|$定义为 c_{t_i} 未在簇集 C_{t_j} 中再现的对象个数.

将聚簇 c_{t_i} 与簇集 C_{t_j} 中的所有簇的对象进行ε'_d 和ε'_t 邻域计算并生成二分图.$|d_{t_j}(c_{t_i})|=|c_{t_i}|-M(c_{t_i})$,其中 $M(c_{t_i})$ 为二分图最大匹配对数,$t_i\neq t_j$.

定义 5. 聚簇异常度.已知聚簇 c_{t_i} 与时间范围$[t_1,t_m]$,聚簇 c_{t_i} 的异常度 $d(c_{t_i})$定义为 c_{t_i} 在时间范围$[t_1,t_m]$内相对于各个时间段的异常度之和,即 $d(c_{t_i})=\sum_{t_j=t_1,t_j\neq t_i}^{t_m}\left|d_{c_{t_j}}(c_{t_i})\right|$.

定义 6. Top-k 时空异常聚簇发现.给定一个到达数据集 DB,一个时间范围$[t_i,t_j]$,发现发生在时间范围$[t_i,t_j]$内具有最大簇异常度的前 k 个簇.

3 查询算法

本节介绍了时空聚类算法、异常簇发现算法及两种优化策略.时空异常簇发现包括到达数据的划分、时空聚类及异常簇发现 3 个阶段.第 1 阶段可利用领域知识(如按自然天、周、月等)或采用等宽分箱的方法将数据按时间分成跨度相等的段,每个时间段表示对象的一个活动周期.第 2 阶段采用基于 DBSCAN 的 IDBSCAN_T 算法对对象进行聚类,将聚类后的聚簇进行 R*-tree 索引并存入磁盘.第 3 阶段采用二分图匹配方法,利用定义 5 计算每个簇的异常度值,返回具有最大异常度的前 k 个簇.

3.1 时空聚类算法

有效的聚类结果对异常簇的发现具有重要的意义.DBSCAN 算法的聚类结果依赖于输入的聚类参数,不同的参数对结果影响较大.观察发现,到达数据大多分布在交通路网上,采用基于密度的算法通过密度相连容易产生较大的聚簇.因此,文献[6]提出了基于迭代的聚类算法 IDBSCAN,借助一个区域面积约束,采用迭代过程实现对大聚簇的分割.基于 DBSCAN 并结合 IDBSCAN 算法迭代思想,本文提出了 IDBSCAN_T 算法,可以支持异常簇发现.首先,在聚类过程中加入了时间维,在确定核心对象时增加了时间参数ε'_t,如定义 1 所示.其次,设计了两个用于迭代聚类的参数:影响区域与时间跨度,这两个参数限定了聚簇在空间与时间上的范围.影响区域参数的设定与查询区域中兴趣点(POI)的影响范围相关.其中,一个 POI 点的影响范围为其所在区域的外包道路围成的面积.影响区域参数取查询区域中最大的影响范围.

3.2 异常聚簇发现算法

异常聚簇发现算法主要分为二分图构建与二分图匹配两个阶段,在这两个阶段分别进行了优化,设计了基

本的两阶段算法,优化的两阶段算法与动态建图匹配算法,提高了异常聚簇发现算法的效率.当计算一个时间段 t_i 中簇 c_{t_i} 的异常度时,需要找到所有与 t_i 不同的时间段 t_j 中 c_{t_i} 在 ε_d 和 ε_t 邻域内的候选集.将簇 c_{t_i} 中对象的均值作为簇中心,取簇中最远对象到簇中心的距离作为簇半径,在时间段 t_j 的索引中进行范围查询,得到候选集 C'_{t_j}.

3.2.1 基本的两阶段的算法

基本的两阶段算法(basic two stages,简称 BTS),采用现有的方法实现二分图构建、最大匹配及异常度计算.

第 1 步,获得聚簇 c_{t_i} 的候选集 C'_{t_j},对簇 c_{t_i} 中每个对象 p,在 C'_{t_j} 中计算其关于 ε_d 和 ε_t 的邻域集,即 $N_{\varepsilon_d,\varepsilon_t}(p)$,为 p 与 $N_{\varepsilon_d,\varepsilon_t}(p)$中的每个对象建立边,生成二分图.

第 2 步,采用匈牙利算法[18]进行二分图最大匹配得到最大匹配对数.

第 3 步,利用定义 4 的计算方法得到 c_{t_i} 相对于一个时间段 t_j 的异常度,根据定义 5 计算出簇 c_{t_i} 在$[t_1,t_m]$时间范围内的异常度.

匈牙利算法采用深度优先搜索寻找增广路径.当找不到任何增广路径时,就得到了一个最大匹配.由于本文的异常度计算利用最大匹配对象对聚簇进行缩减,具有较多对象的簇倾向获得较大的异常度.因此我们考虑先从包含对象多的簇开始计算,动态维护 k 个结果.由定义 5 可知,簇的可能最大异常度值为$(m-1)|c_{t_i}|$,其中,$|c_{t_i}|$为当前访问的类 c_{t_i} 的对象数,m 为时间段范围.因此,当第 k 大的结果的异常度不小于$(m-1)|c_{t_i}|$时,其他包含对象数小于$|c_{t_i}|$的簇可以被剪枝.

BTS 包括二分图构建与最大匹配两个阶段.第 1 阶段,查询类中每个对象都要对候选集进行一次遍历,获取关于 ε_d 和 ε_t 的邻域集,代价为 ln.第 2 阶段运用匈牙利算法求出二分图最大匹配,若二分图采用邻接表的存储结构,其时间复杂度为 $O((n+l)e)$,所以 BTS 算法的时间复杂度为 $O(ln+ne+le)$.

3.2.2 优化的两阶段算法

本节将分别对二分图构建、二分图匹配两个阶段进行优化,提出了优化的两阶段算法(optimized two stages,简称 OTS).

对聚簇 c_{t_i} 及 C'_{t_j} 构建二分图时,可以利用到达对象的时间属性对构建集合进行排序,在有序集上进行邻域查询实现对构建过程的优化.将聚簇 c_{t_i} 及候选集 C'_{t_j} 中的对象按时间由早到晚排序.确定对象在 C'_{t_j} 中匹配的上界,当对其进行建边时,可直接从其上界向下搜索,从而不必遍历 C'_{t_j} 中的全部对象.算法 1 中的步骤 4 找到查询对象在候选集中的时间上界位置,步骤 5~步骤 7 向下搜索,为满足条件的两个对象之间建立一条边.

算法 1. Construction(c_{t_i},C'_{t_j},ε_d,ε_t).

输入:当前聚簇 c_{t_i},候选对象集 C'_{t_j},ε_d,ε_t;

输出:二分图 $G(v,e)$的邻接表 *m_node*.

(1) 对 c_{t_i} 与 C'_{t_j} 按时间由早到晚排序;

(2) 用 c_{t_i} 对象初始化邻接表 *m_node* 头节点;

(3) **for** $p_l \in c_{t_i}$ **do**

(4) 根据 p_{l-1} 的时间上界找到 p_l 的时间上界在 C'_{t_j} 中的下标 *index*;

(5) **while** (*index*++<$|C'_{t_j}|$且 $C'_{t_j}[index].t<=p_i.t+\varepsilon_t$)

(6) **if** ($D(p_k,C'_{t_j}[index])<=\varepsilon_d$)

(7) 为 p_l 与 $C'_{t_j}[index]$建立一条边加入 G 中;

(8) **return** G;

定理 1. 算法 1 可以正确构建二分图.

证明:设查询对象为 p_i,匹配上界为 q_i,即 $p_i.t-\varepsilon_t <=q_i.t<=p_i.t+\varepsilon_t$,$p_i.t-\varepsilon_t>q_{i-1}.t$.由于查询类中对象已按时间排序,$p_{i+1}.t>=q_i.t$,则 $p_{i+1}.t-\varepsilon_t>q_{i-1}.t$.所以对查询对象 p_{i+1} 进行匹配时,只需从 q_i 向下搜索即可.由于 $q_j.t<=q_{j+1}.t$,当遇到满足 $p_{i+1}.t+\varepsilon_t<q_j.t$ 的候选对象 q_j 时即可停止向下搜索,其余候选点均不满足时间条件,可知算法 1 能够产生正确的二分图.得证. □

定理 2. 算法 1 的时间复杂度为 $O(n\log n+l\log l+\delta n)$.

证明:略. □

聚簇表示到达对象分布密集区域,在二分图构建阶段,聚簇及其候选集的交叠部分会生成边,且一个对象会对应多条边.二分图匹配过程中如果采用宽度优先搜索策略可以尽早完成匹配,但如果仅采用宽度优先搜索会在寻找增广路径时增加时间花费.所以,本文针对所生成二分图的特点提出宽度与深度优先结合的最大匹配优化策略.算法 2 实现了优化匹配过程.算法中,Q 用于存储候选集中每个对象 q 的匹配结果与访问状态,其中,$q.result$ 用来记录 q 的匹配结果,$q.state$ 记录 q 的访问状态,m_num 用来表示成功匹配的节点对数.其中步骤 4 通过调用过程 1,从节点 l 尝试扩展,若找到可增广路径,则 m_num 加 1,算法最终返回成功匹配对数 m_num.

算法 2. Optimal_match(*m_node*,*Q*).

输入:二分图 $G(v,e)$的邻接表 *m_node*,*Q*;

输出:最大匹配对数*m_num*.

```
(1) m_num=0,l=0;
(2) for l<|m_node| do
(3)     for q∈Q do q.state=false;
(4)     if (search(m_node,l)) m_num++;
(5) return m_num;
```

过程 1 中,步骤 2~步骤 6 按宽度优先策略搜索是否存在没有被匹配的邻接点,若有,则直接匹配.步骤 7~步骤 12 采用深度搜索策略寻找增广路径,递归地寻找匹配结果.

过程 1. Search(*m_node*,*l*).

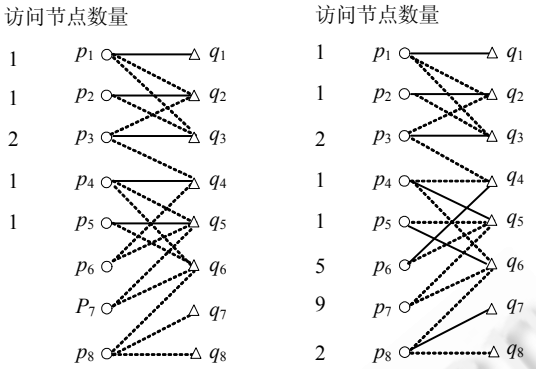
```
(1) pp=m_node[l].first   //从首个邻接点访问
(2) while (pp!=null)   //宽度优先搜索
(3)     if (pp->result=false)
(4)         pp->result=l; return true;
(5)     pp=pp->next;
(6) pp=m_node[l].first;   //从首个邻接点搜索
(7) while (pp!=null)   //深度优先搜索
(8)     if (pp->state=false)
(9)         pp->state=true;
(10)        if (search(m_node, pp->result))
(11)            pp->result=l; return true;
(12)    pp=pp->next;
(13) return false;
```

定理 3. 利用算法 2 可以完成最大匹配.

证明:算法 2 对查询类中每个对象进行匹配,首先进行宽度优先搜索,判别查询对象是否存在没有被匹配的邻接点.若有,则进行匹配,否则,调用匈牙利算法寻找可增广路径.该过程在匈牙利算法搜索的基础上增加了宽度优先搜索,是匈牙利算法的覆盖,则算法 2 能够得到正确的匹配结果.得证. □

由定理 3 可知,其时间复杂度仍为 $O((n+l)e)$.

图 2(a)为成功匹配前 5 个对象的状态图.在匹配对象 p_3 时,并没有通过 q_2 节点递归地寻找增广路径,而是继续访问其邻接点 q_3,由于 q_3 未被匹配,对 p_3 进行匹配时,采用算法 3 需要访问节点数为 2,而使用匈牙利算法需访问节点数为 3.图 2(b)为运用算法 3 后最终的匹配状态图,共需访问节点数为 22,而匈牙利算法访问节点数为 29.



(a) p_5及之前对象成功匹配 (b) p_6, p_7 与p_8匹配

Fig.2 Final matching state of Algorithm 2

图 2 算法 2 最终匹配状态

3.2.3 动态建图匹配算法

本节介绍一种动态建图匹配算法(dynamic construction and matching,简称 DCM),该算法将二分图构建与二分图匹配结合起来,在匹配过程中按需动态建图.如算法 3 所示,步骤 5 调用过程 2,对查询对象进行匹配,若成功匹配,则 *m_num* 加 1,算法最终返回成功匹配对数 *m_num*.

算法 3. DCM(c_{t_i},C'_{t_j},ε_d,ε_t).

输入:当前聚簇 c_{t_i},候选对象集 C'_{t_j},匹配半径参数 ε_d,匹配时间参数 ε_t;

输出:最大匹配次数 *m_num*.

(1) 用 c_{t_i} 初始化邻接表 *m_node* 头节点,用 C'_{t_j} 初始化 *Q*;

(2) *m_num*=0,*l*=0;

(3) **for** *l*<*m_node.size* **do**

(4) **for** $q \in Q$ **do** *q.state*=false;

(5) **if** (*breadth_search*(*m_node*,*l*))

(6) *m_num*++;

(7) **return** *m_num*;

在过程 2 中,步骤 2~步骤 6 按宽度优先搜索策略一边建图一边匹配,若匹配成功,则停止建图.步骤 9~步骤 14 采用深度优先搜索策略进行匹配.其中,步骤 12 调用过程 3 寻找可增广路径进行匹配.

过程 2. breadth_search(*m_node*,*l*).

(1) 根据 p_{l-1} 的时间上界找到 p_l 的时间上界在 C'_{t_j} 中的下标 *index*; //如果 *l*=0,则 *index*=0

(2) **while** (*index*++<$|C'_{t_j}|$且 $C'_{t_j}[index].t$<=*m_node*[*i*].*t*+ε_t)

(3) **if** ($D(m\_node[l],C'_{t_j}[index]) \leqslant \varepsilon_d$)

(4) *m_node*[*l*]中插入一个节点 *q_new*; //建立一条边

(5) **if** (*q_new.result*=false)

(6) *q_new.result*=*l*;**return** true;

(7) *m_node*[*l*].*end*=true //标记邻接边是否全部建立

(8) *pp*=*m_node*[*l*].*first*;

(9) **while** (*pp*!=*null*) //深度优先搜索;

(10) **if** (*pp*–>*state*=false)

(11) *pp*–>*state*=true;

(12) **if** (depth_search(*m_node*,*pp*–>*result*))

```
(13)          pp–>result=l; return true;
(14)     pp=pp–>next;
(15)  return false;
```

在过程 3 中,步骤 2~步骤 7 在已建好的邻接点中寻找增广路径,步骤 8~步骤 18 建立新的邻接点继续寻找增广路径,若成功找到,则对其进行匹配.

过程 3. depth_search(*m_node*,*l*).

```
(1) pp=m_node[l].first;
(2) while (pp!=null)
(3)     if (pp–>state=false)
(4)        pp–>state=true;
(5)        if (pp–>result=false 或 depth_search(m_node,pp–>result))
(6)           pp–>result=l; return true;
(7)     pp=pp–>next;
(8) if (!m_node[l].end)
(9)     while (有边可建)
(10)       m_node[l]中插入新节点 q_new;
(11)       if (q_new.result=false)
(12)          q_new.result=l; return true;
(13)       if (q_new.state=false)
(14)          q_new.state=true;
(15)          if (depth_search(m_node,q_new.result))
(16)             q_new.result=l; return true;
(17) m_node[l].end=true;
(18) return false;
```

图 3(a)为查询聚簇的初始存储结构图,算法采用按需动态建图思想,只在需要时建立邻接点.图 3(b)为运用动态建图匹配算法成功匹配前 5 个对象时的存储状态图,图 3(c)为匹配结束时,最终存储状态图.共建立邻接点 15 个,访问节点数为 20.与前述算法相比较,少建立了 5 个节点.因此动态建图匹配算法减少了运行开销与存储开销,进一步提高了效率.

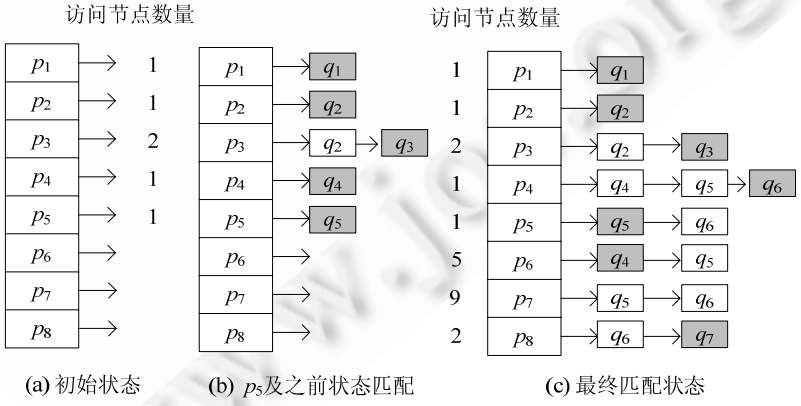


Fig.3 Matching state of DCM

图 3 DCM 算法的匹配状态

定理 4. 算法 3 可以实现最大匹配.

证明:DCM 算法运用了按需动态建图的思想并结合优化建图策略与向后优先匹配策略,仅对需要被访问的邻接点进行构建,未被建立的邻接点没有参与整个匹配过程,这些未构建的边对匹配结果无影响.同时,定理 1 与整个匹配过程,这些未构建的边对匹配结果无影响.另外,定理 2 与定理 3 保证了优化策略的正确性,从而 DCM 算法可以实现最大匹配.得证. □

引理 1. 算法 3 中匹配比较次数不小于构建边数.

定理 5. DCM 时间复杂度为 $O(ne'+le')$.

4 实验结果与分析

本节评价查询的有效性及本文所提出算法的效率.所有算法由 C++语言实现,实验平台为 4 个 3.3GHz Core(TM)i3 CPU 和 4.0 GB 的内存,操作系统为 Windows 7 的 HP PC.实验使用 2012 年 10 月与 11 月北京 1.2 万辆出租车的 GPS 数据,大小为 36.1GB.根据数据中的触发事件与运营状态对上下车记录进行提取,将下车数据作为到达数据.经过噪声处理后,获得每天约 10 万条下车记录.

4.1 算法结果分析

在 5 个不同的区域时间范围为 15 天的情况下,执行 Top-10 异常簇查询,将每个查询结果与实际地点和时间可能发生的事件进行对照,表 1 中列出了 5 个查询对应的事件.由表 1 可以看出,异常簇所对应的事件为临时的、非周期性的对象聚集.

Table 1 Results for SACD

表 1 SACD 查询结果

序号	簇中心时间	簇中心坐标	地点	事件
1	10-11 18:13	116.28,39.91	万事达中心	NBA 中国赛
2	11-24 19:16	116.45,39.93	工人体育馆	丁当演唱会
3	11-23 10:16	116.31,39.98	北京大学	北大招聘会
4	11-02 19:20	116.40,39.99	奥体中心	伊莲佩姬演唱会
5	10-01 05:31	116.39,39.90	天安门广场	国庆升旗

4.2 算法效率分析

本节将评价各异常聚簇发现算法在不同参数设置下的运行效率.表 2 列出了查询时的各参数设置.其中查询时间间隔为查询区域的时间跨度.由于算法的大部分代价花费在二分图构建与匹配过程中,实验仅对上述两个过程进行效率分析.

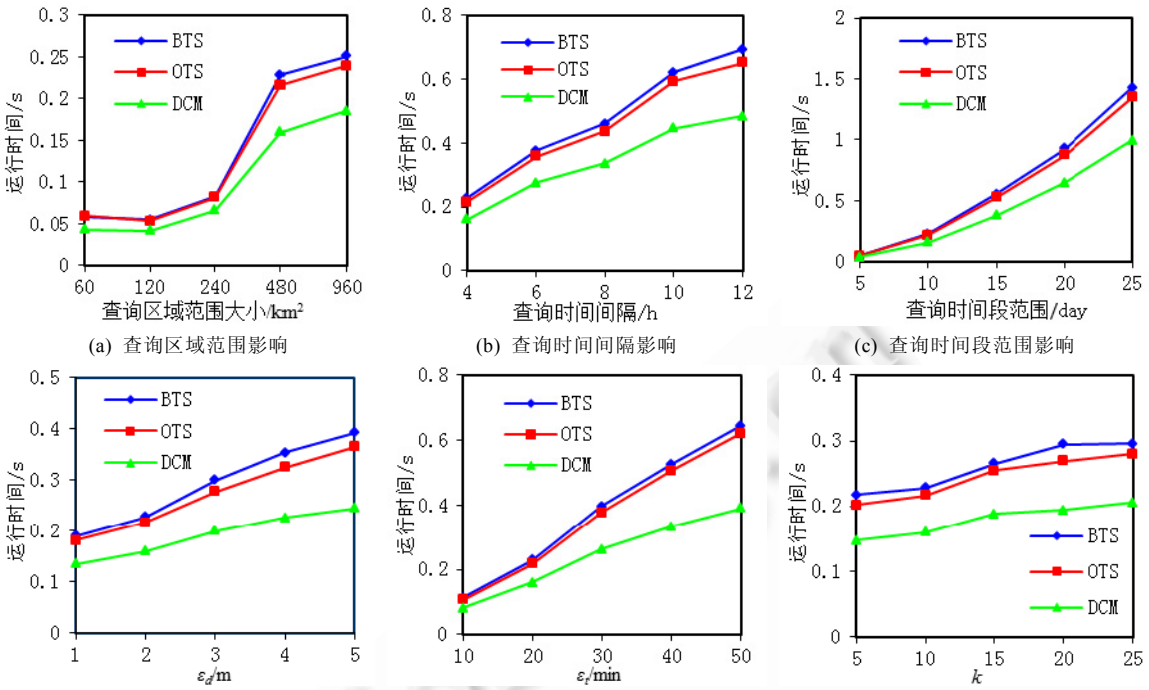
Table 2 Parameter settings

表 2 参数设置

Parameter	Settings	Default
查询区域/平方千米	60,120,240,480,960	480
查询时间间隔/小时	4,6,8,10,12	4
查询段范围/天	5,10,15,20,25	10
ε_d/m	1, 2, 3, 4, 5	2
ε_t/minute	10,20,30,40,50	20
k/个	5,10,15,20,25	10

图 4 给出算法效率对比情况.

图 4(a)~图 4(f)给出查询区域、查询时间间隔、时间段范围、ε_d,ε_t,k 变化时各算法的效率.从图中可以看出,算法 DCM 运行时间明显少于另外两种算法,这是由于 DCM 算法采用动态建图方法造成的.算法 BTS 与 OTS 具有相似的运行时间,其原因是满足再现时间阈值的候选对象较多,而匹配过程需要的边数较少.从图中还可以看出,算法 BTS 与 OTS 具有相似的运行时间,而算法 DCM 运行时间明显少于前二者.在图 4(a)~图 4(e)中,当参数取值较小时,3 种算法运行速度相当.随着参数的增大,DCM 算法较其他两种算法效率明显提高.图 4(f)表明,参数 k 的改变对各算法性能的影响较小.总的来说,DCM 的查询效率较其他两种算法提高约 30%,原因在于它采用单阶段策略,按需动态建图,提高了查询效率.



(a) 查询区域范围影响 (b) 查询时间间隔影响 (c) 查询时间段范围影响

(d) ε_d 影响 (e) ε_t 影响 (f) k 影响

Fig.4 The comparison of algorithms efficiency

图 4 算法效率对比

5 结 论

本文提出一类新的时空数据查询:时空异常类发现(SACD).给出 SACD 相关定义,提出新的时空聚类算法 IDBSCAN_T 和 3 种优化策略并设计了 3 种有效的查询算法.利用真实的到达数据集,对算法的有效性和效率进行分析,实验结果表明,异常簇发现算法能够有效地发现异常簇,并且 DCM 算法具有最高的查询效率.下一步工作将在到达数据的自动周期性划分、参数的自动选择及在线异常簇检测方面进行深入研究.

References:

[1] Ben CZ, Shen HT, Zhou XF, Zheng Y, Xie X. Searching trajectories by locations: An efficiency study. In: Elmagarmid AK, Agrawal D, eds. Proc. of the Special Interest Group on Management of Data Conf. New York: ACM Press, 2010. 255−266.

[2] Yin ZJ, Cao LL, Han JW, Luo JB, Huang T. Diversified trajectory pattern ranking in geo-tagged social media. In: Agarwal N, ed. Proc. of the Society for Industrial and Applied Mathematics Int'l Conf. on Data Mining. Phoenix: SIAM Press, 2011. 980−991.

[3] Hsieh HP, Li CT. Composing traveling paths from location-based services. In: Hoffmann J, Selman B, eds. Proc. of the Association for the Advancement of Artificial Intelligence Conf. India: AAAI Press, 2012. 618−619.

[4] Lee KCK, Lee WC, Zheng BH, Tian Y. ROAD: A new spatial object search framework for road networks. IEEE Trans. on Knowledge and Data Engineering, 2012,24(3):547−560.

[5] Huttenlocher DP, Klanderman G, Rucklidge WJ. Comparing images using the hausdorff distance. IEEE Trans. on Pattern Analysis and Machine Intelligence, 1993,15(9):850−863.

[6] Pan G, Qi GD, Wu ZH, Zhang DQ, Li SJ. Land-Use classification using taxi GPS traces. IEEE Intelligent Transportation Systems Trans. and Magazine, 2013,14(1):113−123.

[7] Zhou AY, Yang B, Jin CQ, Ma Q. Location-Based services: Architecture and progress. Chinese Journal of Computers, 2011,34(7): 1155−1171 (in Chinese with English abstract).

[8] Ni J, Ravishankar CV. Pointwise-Dense region queries in spatio-temporal database. In: Chirkova R, Dogac A, Tamer M, Sellis TK, eds. Proc. of the Institute of Electrical and Electronics Engineers Int'l Conf. on Data Engineering. Piscataway: IEEE Press, 2007. 1066−1075.

[9] Zheng K, Zheng Y, Yuan J, Shang S. On discovery of gathering patterns from trajectories. In: Jensen CS, Jermaine CM, Zhou XF, eds. Proc. of the Institute of Electrical and Electronics Engineers Int'l Conf. on Data Engineering. Piscataway: IEEE Press, 2013. 242−253.

[10] Li ZH, Ding B, Han JW, Kays R. Swarm: Mining relaxed temporal moving object clusters. In: Bordawekar R, Lang CA, eds. Proc. of the Int'l Conf. on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers, 2010. 723−734.

[11] Chandola V, Banerjee A, Kumar V. Anomaly detection: A survey. ACM Computing Surveys (CSUR), 2009,41(3):1−72.

[12] Lee JG, Han JW, Whang KY. Trajectory clustering: A partition-and-group framework. In: Chan CY, Ooi BC, Zhou A, eds. Proc. of the Special Interest Group on Management of Data Conference. New York: ACM Press, 2007. 593−604.

[13] Rosswog J, Ghose K. Efficiently detecting clusters of mobile objects in the presence of dense noise. In: Biryukov A, Gong G, Stinson DR, eds. Proc. of the Association for Computing Machinery Symp. on Access Control Model and Technology. New York: Springer-Verlag, 2010. 1095−1102.

[14] Liu LX, Qiao SJ, Liu B, Le JJ, Tang CJ. Efficient trajectory outlier detection algorithm based on R-Tree. Ruan Jian Xue Bao/ Journal of Software, 2009,20(9):2426−2435 (in Chinese with English abstrat). http://www.jos.org.cn/1000-9825/3580.htm [doi: 10.3724/SP.J.1001.2009.03580]

[15] Jeung H, Shen HT, Zhou X. Convoy queries in spatio-temporal databases. In: Alonso G, Blakeley JA, Chen ALP, eds. Proc. of the Institute of Electrical and Electronics Engineers Int'l Conf. on Data Engineering. Piscataway: IEEE Press, 2008. 1457−1459.

[16] Ester M, Kriegel HP, Sander J, Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Simoudis E, Han JW, Fayyad UM, eds. Proc. of the Association for Computing Machinery Conf. on Knowledge Discovery and Data Mining. Portland: AAAI Press, 1996. 226−231.

[17] Breunig MM, Kriegel HP, Ng RT, Sander J. LOF: Identifying density-based local outliers. In: Chen WD, Naughton JF, Bernstein PA, eds. Proc. of the Special Interest Group on Management of Data Conf. New York: ACM Press, 2000. 93−104.

[18] Kuhn HW. The Hungarian method for the assignment problem. Naval Research Logistics Quarterly, 1955,2(1):83−97.

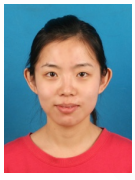
附中文参考文献:

[7] 周傲英,杨彬,金澈清,马强.基于位置的服务:架构与进展.计算机学报,2011,34(7):1155−1171.

[14] 刘良旭,乔少杰,刘宾,乐嘉锦,唐常杰.基于 R-Tree 的高效异常轨迹检测算法.软件学报,2009,20(9):2426−2435. http://www.jos.org.cn/1000-9825/3580.htm [doi: 10.3724/SP.J.1001.2009.03580]

刘俊岭(1972－),女,辽宁沈阳人,博士生,副教授,CCF 会员,主要研究领域为空间数据库.
E-mail: liujl@sjzu.edu.cn

孙焕良(1969－),男,博士,教授,CCF 高级会员,主要研究领域为空间数据库,数据挖掘.
E-mail: sunhl@sjzu.edu.cn



魏茹玉(1990－),女,硕士生,主要研究领域为空间数据库.
E-mail: weiruy@126.com

姚承伟(1988－),男,硕士生,主要研究领域为空间数据库.
E-mail: yaochengwei2008@163.com

于戈(1962－),男,博士,教授,博士生导师,CCF 杰出会员,主要研究领域为数据库理论与技术.
E-mail: yuge@mail.neu.edu.cn