软件学报 ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*,2018,29(8):2470−2484 [doi: 10.13328/j.cnki.jos.005289]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

# 面向位置大数据的快速密度聚类算法*

于彦伟[1], 贾召飞[1], 曹 磊[2], 赵金东[1], 刘兆伟[1], 刘惊雷[1]

[1](烟台大学 计算机与控制工程学院,山东 烟台 264005)

[2](Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge 02139, USA)

通讯作者: 于彦伟, E-mail: yuyanwei@ytu.edu.cn, http://www.ytu.edu.cn

**摘 要**: 面向位置大数据聚类,提出了一种简单但高效的快速密度聚类算法 CBSCAN,以快速发现位置大数据中任意形状的聚类簇模式和噪声.首先,定义了 Cell 网格概念,并提出了基于 Cell 的距离分析理论,利用该距离分析,无需距离计算,可快速确定高密度区域的核心点和密度相连关系;其次,给出了网格簇定义,将基于位置点的密度簇映射成基于网格的密度簇,利用排他网格与相邻网格的密度关系,可快速确定网格簇的包含网格;第三,利用基于 Cell 的距离分析理论和网格簇概念,实现了一个快速密度聚类算法,将DBSCAN基于数据点的密度扩展聚类转换成基于Cell的密度扩展聚类,极大地减少高密度区域的距离计算,利用位置数据的内在特性提高了聚类效率;最后,在基准测试数据上验证了所提算法的聚类效果,在位置大数据上的实验结果统计显示,与 DBSCAN、PR-Tree 索引和 Grid 索引优化的 DBSCAN 相比,CBSCAN 分别平均提升了 525 倍、30 倍和 11 倍效率.

**关键词**: 聚类分析;密度聚类;位置大数据;Cell 网格;网格簇

**中图法分类号**: TP311

中文引用格式: 于彦伟,贾召飞,曹磊,赵金东,刘兆伟,刘惊雷.面向位置大数据的快速密度聚类算法.软件学报,2018,29(8):2470−2484. http://www.jos.org.cn/1000-9825/5289.htm

英文引用格式: Yu YW, Jia ZF, Cao L, Zhao JD, Liu ZW, Liu JL. Fast density-based clustering algorithm for location big data. Ruan Jian Xue Bao/Journal of Software, 2018,29(8):2470−2484 (in Chinese). http://www.jos.org.cn/1000-9825/5289.htm

## Fast Density-Based Clustering Algorithm for Location Big Data

YU Yan-Wei[1], JIA Zhao-Fei[1], CAO Lei[2], ZHAO Jin-Dong[1], LIU Zhao-Wei[1], LIU Jing-Lei[1]

[1](School of Computer and Control Engineering, Yantai University, Yantai 264005, China)

[2](Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge 02139, USA)

**Abstract**: This paper proposes a simple but efficient density-based clustering, named CBSCAN, to fast discover cluster patterns with arbitrary shapes and noises from location big data effectively. Firstly, the notion of Cell is defined and a distance analysis principle based on Cell is proposed to quickly find core points in high density areas and density relationships with other points without distance computing. Secondly, a Cell-based cluster that maps point-based density cluster to grid-based density cluster is presented. By leveraging exclusion grids and relationships with their adjacent grids, all inclusion grids of Cell-based cluster can be rapidly determined. Furthermore, a fast density-based algorithm based on the distance analysis principle and Cell-base cluster is implemented to transform DBSCAN of point-based expansion to Cell-based expansion clustering. The proposed algorithm improves clustering efficiency significantly by using inherent property of location data to reduce huge number of distance calculations. Finally, comprehensive experiments on benchmark

* 基金项目: 国家自然科学基金(61403328, 61773331, 61572419, 61502410); 山东省重点研发计划(2015GSF115009); 山东省自然科学基金(ZR2013FM011, ZR2013FQ023, ZR2014FQ016)

Foundation item: National Natural Science Foundation of China (61403328, 61773331, 61572419, 61502410); Key Research and Development Program of Shandong Province (2015GSF115009); Shandong Provincial Natural Science Foundation (ZR2013FM011, ZR2013FQ023, ZR2014FQ016)

收稿时间: 2016-09-03; 修改时间: 2016-10-03; 采用时间: 2016-12-20; jos 在线出版时间: 2017-07-12

CNKI 网络优先出版: 2017-07-12 15:33:51, http://kns.cnki.net/kcms/detail/11.2560.TP.20170712.1533.009.html

© 中国科学院软件研究所 http://www.jos.org.cn

datasets demonstrate the clustering effectiveness of the proposed algorithm. Experimental results on massive-scale real and synthetic location datasets show that CBSCAN improves 525 fold, 30 fold and 11 fold of efficiency compared with DBSCAN, DBSCAN with PR-Tree and Grid index optimization respectively.

**Key words**: clustering analysis; density-based clustering; location big data; Cell grid; cell-based cluster

随着 GPS 终端、RFID、智能手机等位置感知设备的广泛普及,各类位置服务(location-based service,简称 LBS)和移动社交网络(mobile social network)应用不断涌现,如用户签到数据、车辆 GPS 轨迹、带有位置的图片、微博等海量的位置大数据被实时采集,而基于这些位置大数据的服务俨然已成为一大独具特色的新兴产业,尤其在交通调度与控制、推荐系统、广告投放、道路规划、公共设施选址与评估、商业决策等领域有着巨大应用价值.由于这些位置数据采集越来越便利,数据量正以爆炸式趋势增长.因此,如何有效存储、快速挖掘出有意义的潜在模式,成为快速处理位置大数据的主要挑战.

位置大数据记录了用户的时空轨迹信息,体现了移动对象的位置变化,如人的移动、车辆运动等.城市中的位置大数据主要来源于交通车辆轨迹数据、移动通信位置数据、移动社交网络签到数据、移动媒体位置数据、电子商务物流位置信息、公共交通刷卡位置记录等.每个位置数据除空间坐标信息外,还反映了在该地理位置上以及相应时间上的一个事件,包含了用户的行为活动、兴趣爱好以及其他未知的重要信息.海量位置数据的分析可以定量描述和估计人们的社会活动特征,发现人们在不同时空粒度下的行为规律,洞察群体整体移动趋势,识别人们感兴趣的路线和区域等,因此,通过对位置数据挖掘,可帮助我们理解和发现位置大数据中所隐含的巨大价值[1].例如,Zheng 等人[2]通过对用户的 GPS 轨迹挖掘来发现有关联的兴趣点模式,进而发现受欢迎的旅游路线;Zheng 等人[3]还对个人历史位置数据挖掘,实现对用户的好友推荐和景点推荐.在位置大数据分析处理应用中,聚类技术常常被用于对位置大数据进行预处理分析,以发现位置数据中的空间或时空分簇模式[4,5].典型聚类算法主要包括基于划分的方法(如 *K*-Means[6])、基于图模型的方法(如谱聚类[7])、基于建模的方法(如 EM 算法[8])、基于密度的方法(如 DBSCAN[9]和 GDBSCAN[10])和基于网格的方法(如 GDCA[11]和 GG[12]).由于位置大数据的海量性、空间分布任意性、更新速度快以及较大混杂性的特点,基于密度的 DBSCAN 算法可以发现任意形状的聚类簇,并且能处理噪声,常被用于位置大数据的模式发现问题.然而,$O(n^2)$的时间复杂度,使得 DBSCAN 算法很难在实际位置大数据系统中得到应用.

Sander 等人[10]提出了一种针对空间数据库聚类的 GDBSCAN 算法.该方法虽然实现了对多维空间数据的聚类,但仍然对聚类效率没有改善.为了扩展到大规模数据聚类,一种常用方法是利用空间索引技术(如 R-Tree[13])剪枝 DBSCAN 中邻域查询的搜索空间,以提高聚类效率,但优化算法效率仍然依赖数据规模.Zhao 等人[12]利用网格划分实现对地理位置数据的快速聚类(grid-growing clustering,简称 GG),该方法首先筛选 $m$ 个高密度网格作为初始种子,然后使用 8 相邻网格搜索对高密度种子进行迭代聚簇扩展,当某网格内数据点数量为 0 时,停止对其扩展.该方法仅仅利用网格和周围 8 邻居网格进行扩展聚类,完全没有考虑网格内数据点间的距离及其密度关系,因此无法获得精确的密度聚类结果,也不能有效识别噪声.Kumar 等人[14]提出了一种基于图结构的 Group 索引以加快邻居点搜索效率,进而提高 DBSCAN 效率,所提索引结构虽然提高了对数据维度的扩展性,但对算法效率提升空间不大.另外一类快速、高扩展性聚类研究热点关注在并行聚类方法上,Cao 等人[15]利用多核 GPU 处理器实现了并行 *K*-Means 聚类,之后,他们将基于 GPU 的方法扩展到数据流聚类[16].Böhm 等人[17]同样也提出了一种基于多核 GPU 的并行密度聚类方法.该方法利用 GPU 共享内存高带宽特性和密度链概念实现了高并行性的聚类算法,最后,冲突的密度链合并为一个聚类簇.He 等人[18]利用 MapReduce 平台实现了一种并行密度聚类算法,该方法利用数据的空间分布实现数据划分,然后在多个数据划分上实现并行聚类,最后实现边界区域的合并处理.Kim 等人[19]同样也提出了一种基于 MapReduce 的密度聚类算法 DBCURE-MR.他们首先提出了一种基于椭圆 $\tau$-邻居的密度聚类算法 DBCURE,然后在 Map 阶段并行实现数据点的椭圆 $\tau$-邻居搜索和核心簇的发现,最后串行地合并核心簇,得到最终结果.虽然这些方法达到了十几倍或数十倍的加速比,但都需要专用处理器或专用服务器集群的支持.

本文针对位置大数据的高效密度聚类问题,从密度聚类算法内在特性优化视角,提出了一种简单但十分高

© 中国科学院软件研究所 http://www.jos.org.cn

效的快速密度聚类算法 CBSCAN(cell-based DBSCAN for location big data),可在一般计算平台实现对位置大数据的密度聚类任务.首先,我们定义了 Cell 网格概念,并提出了基于 Cell 的距离分析理论,利用该距离分析,无需距离计算,可快速确定高密度区域的核心点和密度相连关系,同时还给出了邻居点搜索区域的最大范围;其次,我们给出了网格簇概念(cell-based cluster),将基于位置点的密度簇定义映射到基于网格的密度簇,利用排他网格与相邻网格的密度关系,可快速确定网格簇的包含网格;第三,利用基于 Cell 的距离分析理论和网格簇概念,实现了一种快速密度聚类算法 CBSCAN,将 DBSCAN 基于数据点的密度扩展聚类转换成基于 Cell 的密度扩展聚类,极大地减少高密度区域的距离计算和位置点遍历,利用位置空间数据密度聚类的内在特性提高了聚类效率;最后,在多个基准测试数据和大规模位置大数据上进行了综合实验评估,验证了所提出算法的聚类效果和性能,并与最新密度聚类技术进行了对比分析.

## 1 问题定义

### 1.1 密度聚类概念

本节对 DBSCAN[9]的相关术语做出了进一步扩展定义.DBSCAN 涉及到两个参数:数据点邻域范围半径 *Eps* 和最小邻居点数目阈值 *MinPts*.设定位置大数据集合为 $D$, $N_{Eps}^{D}(p)$ 表示位置点 $p$ 在 *Eps* 邻域范围内的邻居点集合,其数量标记为 $|N_{Eps}^{D}(p)|$.

DBSCAN 算法将所有位置数据点分为 3 种类型:核心点、边界点和噪声点,它们的划分如图 1 所示.

- 核心点:如果位置点 $p$ 的 *Eps* 邻居点数量 $|N_{Eps}^{D}(p)| \geqslant MinPts$,则点 $p$ 是一个核心点;
- 边界点:如果位置点 $p$ 的 *Eps* 邻居点数量 $|N_{Eps}^{D}(p)| < MinPts$,但存在一个邻居点 $q(q \in N_{Eps}^{D}(p))$是核心点,则点 $p$ 是一个边界点;
- 噪声点:如果一个位置点 $p$ 既不是核心点也不是边界点,则它是一个噪声点.也就是,邻居点数量小于 *MinPts*,同时也不存在核心点邻居.

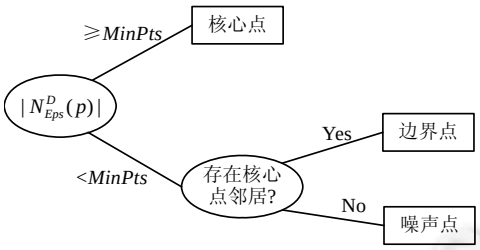



Fig.1 Decision tree for types of location points

图 1 位置点类型决策树

对于数据集 $D$ 的任意两个位置点,它们的密度关系都可分为 3 种:直接密度可达、密度可达和密度相连.

- 直接密度可达:对于任何一个位置点,其直接密度可达到它 *Eps* 邻域内的任一核心点;
- 密度可达:给定一系列的位置点 $p_1,p_2,\ldots,p_n$,若 $p_i$ 直接密度可达到 $p_{i+1}(i=1,2,\ldots,n-1)$,则称 $p_1$ 密度可达到 $p_n$;
- 密度相连:如果两个位置点 $p$ 和 $q$ 同时密度可达到点 $o$,则 $p$ 与 $q$ 密度相连.

很明显,直接密度可达与密度可达是不对称的,密度可达具有传递性.所有边界点直接密度可达到其 *Eps* 邻域内的核心点.而对于两个核心点,其直接密度可达和密度可达关系是对称的.密度相连关系是对称的,任何两点的密度相连都是相互的.

**定义 1(密度簇)**. 给定位置数据集 $D$,将所有密度相连的位置点聚集在同一集合,每一个非空子集称为一个密度簇.

© 中国科学院软件研究所 http://www.jos.org.cn

**推论 1**. 给定一个密度簇 $C$ 和它的一个核心点 $p$,若存在位置点 $q(q\in D)$密度可达到点 $p$,则点 $q\in C$.

推论 1 很容易由密度簇的定义和密度相连的概念推导得出.由推论 1 可知,任意一个密度簇可由它的一个核心点和所有密度可达到该核心点的位置点组成.

### 1.2 基于Cell的距离分析

本小节提出了一种基于网格划分的距离分析方法,称为 Cell 网格划分,如定义 2 所示.之后给出了基于 Cell 的距离关系分析方法.

**定义 2(cell)**. 给定参数 $Eps$,位置数据集 $D$ 的二维空间被划分成多个网格,网格边长为 $Eps/2\sqrt{2}$, 则每个划分称为一个 Cell 网格.下文简称网格,每个 Cell 网格被表示为 $G_{(d_1,d_2)}$,其中,$d_1=\lfloor 2\sqrt{2}\cdot x/Eps\rfloor$,$d_2=\lfloor 2\sqrt{2}\cdot y/Eps\rfloor$, $(x,y)$是该 Cell 的中心坐标.

由 Cell 网格的表示可知:对于任意落入 $G_{(d_1,d_2)}$ 中的位置点$(x_1,y_1)$,都有 $d_1=\lfloor 2\sqrt{2}\cdot x_1/Eps\rfloor$,$d_2=\lfloor 2\sqrt{2}\cdot y_1/Eps\rfloor$. 因此,根据位置点坐标,即可快速映射到相应的网格.

下面介绍网格与核心点之间的关系.$|G_{(d_1,d_2)}|$表示落入网格 $G_{(d_1,d_2)}$ 中位置点的数量.由定义 2 可知:对于同一网格内的任意两个位置点,它们的距离一定小于 $Eps$,也就是说,同一网格内的位置点一定互为邻居点.

**引理 1**. 给定网格 $G_{(d_1,d_2)}$ 中的一个位置点 $p$,如果 $\sum_{i=-1,j=-1}^{1,1}|G_{(d_1+i,d_2+j)}|\geqslant MinPts$ ,则点 $p$ 一定是核心点.

证明:引理 1 是显而易见的.如图 2 所示:首先,位置点 $p$ 与其所在网格 $G_{(d_1,d_2)}$ 内的任何位置点都互为邻居点;其次,对于 $G_{(d_1,d_2)}$ 中任意的位置点 $p$,其 $Eps$ 邻域一定覆盖了与 $G_{(d_1,d_2)}$ 相邻的 8 个网格,也就是说,对于这 8 个网格中任意位置点与 $G_{(d_1,d_2)}$ 中的位置点之间的距离一定不大于 $Eps$.如图中两个以 $G_{(d_1,d_2)}$ 顶点为圆心的 $Eps$ 区域(虚线和实线圆圈范围)所示.因此,$G_{(d_1,d_2)}$ 以及相邻的 8 个网格内的位置点都是点 $p$ 的邻居点.

所以,若 $\sum_{i=-1,j=-1}^{1,1}|G_{(d_1+i,d_2+j)}|\geqslant MinPts$ ,则点 $p$ 一定是核心点. □

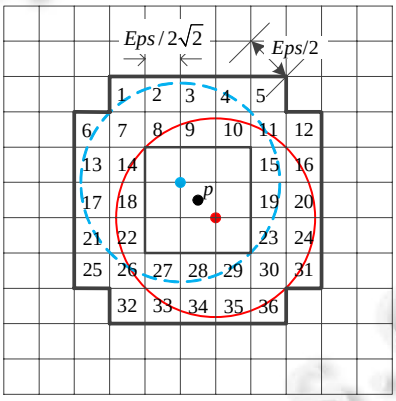



Fig.2 Cell-Based distance analysis

图 2 基于 Cell 网格划分的距离分析

进而根据引理 1,可得出推论 1.

**推论 1**. 给定一个网格 $G_{(d_1,d_2)}$,如果 $\sum_{i=-1,j=-1}^{1,1}|G_{(d_1+i,d_2+j)}|\geqslant MinPts$ ,则对于$\forall p(p\in G_{(d_1,d_2)})$都是核心点.

更直接地,如果$|G_{(d_1+i,d_2+j)}|\geqslant MinPts$ ,则 $G_{(d_1,d_2)}$ 中任意的位置点也都是核心点.

**引理 2**. 给定网格 $G_{(d_1,d_2)}$ 中的一个位置点 $p$,若验证点 $p$ 是否为一个核心点,则至多需要检查 36 个网格中位置点.

证明:对于 $G_{(d_1,d_2)}$ 中任意位置点 $p$,由引理 1 可知,$G_{(d_1,d_2)}$ 及相邻网格的位置点一定是点 $p$ 的邻居点,因此不需要检查.如图 2 所示:考虑极端情况,当 $p$ 位于 $G_{(d_1,d_2)}$ 的 4 个顶点上,标号 1~36 的网格覆盖了所有 4 个顶点的

© 中国科学院软件研究所 http://www.jos.org.cn

所有 *Eps* 邻域范围.因此,验证点 *p* 是否是一个核心点,只需计算这外围的 36 个网格中位置点到点 *p* 的距离即可.证明完毕. □

引理 2 给出了验证核心点最大的邻居点搜索范围.相似地,也可进一步得出推论 2.根据推论 2,可快速排除非核心点区域.

**推论 2**. 给定一个网格 $G_{(d_1,d_2)}$,如果 $\sum_{i=-3,j=-3}^{3,3}|G_{(d_1+i,d_2+j)}|-|G_{(d_1\pm3,d_2\pm3)}|<MinPts$,则 $G_{(d_1,d_2)}$ 中不存在核心点.

## 2 基于网格的密度聚类算法

本节将详细介绍基于 Cell 的密度聚类算法 CBSCAN.首先,提出了基于 Cell 的密度簇概念和基于 Cell 的网格密度相连关系;然后,给出了一种高效的 Cell 密度聚类算法实现和复杂度分析.

### 2.1 基于Cell的密度簇

**定义 3(包含网格)**. 给定一个网格 $G_{(d_1,d_2)}$ 和一个密度簇 *C*,如果对于$\forall p(p\in G_{(d_1,d_2)})$都能确定 $p\in C$,则称网格 $G_{(d_1,d_2)}$ 是密度簇 *C* 的包含网格(inclusion grid,简称 IG).

对于网格 $G_{(d_1,d_2)}$ 中某位置点 *p*,如果 *p* 是核心点并且属于密度簇 *C*,根据网格定义,$G_{(d_1,d_2)}$ 及相邻网格内的位置点都直接密度可达到点 *p*,也就是说,都确定属于簇 *C*.因此,只要网格内包含一个核心点,则该网格一定是某密度簇的一个包含网格.根据包含网格的定义,一个密度簇 *C* 中位置点可由所有包含网格内的位置点和不包括在任何包含网格内的边界点组成.

**定义 4(网格簇)**. 给定一个密度簇 *C*,二元组{*IGs*,*BPs*}被称作密度簇 *C* 所对应的网格簇(基于 Cell 的密度簇),其中,*IGs* 是 *C* 所有的包含网格集合,*BPs* 是不包括在 *IGs* 中的所有边界点集合.

网格簇定义将密度簇中密度相连的数据点简化表示成了包含网格 *IGs* 和不在 *IGs* 内的边界点两部分.根据密度簇中核心点与邻居点间的密度关系,我们进一步定义了排他网格.

**定义 5(排他网格)**. 给定一个网格 $G_{(d_1,d_2)}$ 和一个密度簇 *C*,如果$\exists p(p\in G_{(d_1,d_2)})$是核心点,并且 $p\in C$,则称网格 $G_{(d_1,d_2)}$ 是密度簇 *C* 的排他网格(exclusion grid,简称 XG).

显然,一个排他网格一定是一个包含网格,而一个包含网格却不一定是一个排他网格.根据引理 1 和引理 2,进一步将排他网格划分成 1 型排他网格 $XG^1$ 和 2 型排他网格 $XG^2$.

- $XG^1$:对于网格 $G_{(d_1,d_2)}$,如果 $\sum_{i=-1,j=-1}^{1,1}|G_{(d_1+i,d_2+j)}|\geqslant MinPts$,则 $G_{(d_1,d_2)}$ 为一个 $XG^1$;
- $XG^2$:对于网格 $G_{(d_1,d_2)}$,如果 $\sum_{i=-1,j=-1}^{1,1}|G_{(d_1+i,d_2+j)}|<MinPts$,但$\exists p(p\in G_{(d_1,d_2)})$是一个核心点,则 $G_{(d_1,d_2)}$ 为一个 $XG^2$.

**引理 3**. 给定一个网格 $G_{(d_1,d_2)}$ 和一个密度簇 *C*,如果 $G_{(d_1,d_2)}$ 是 *C* 的排他网格,则 $G_{(d_1,d_2)}$ 的相邻网格一定是 *C* 的包含网格.

证明:如果 $G_{(d_1,d_2)}$ 是密度簇 *C* 的排他网格,则一定$\exists p(p\in G_{(d_1,d_2)})$是核心点.根据网格的定义,所有在 $G_{(d_1,d_2)}$ 相邻网格内的位置点都直接密度可达到点 *p*,也就是说,一定属于簇 *C*.因此,根据包含网格的定义,$G_{(d_1,d_2)}$ 的相邻网格一定是 *C* 的包含网格. □

下一小节将详细介绍 CBSCAN 算法,利用引理 1,无需计算位置点间的距离,可快速找出密度较高处的核心点和 $XG^1$;利用引理 2,可最大范围地剪枝验证核心点的距离计算空间,快速搜索 $XG^2$;利用推论 2,也可快速排除掉低密度区域内核心点的可能;利用引理 3,CBSCAN 将对位置点的密度聚类转换成了对 Cell 网格的扩展聚类,对于高密度区域,无需计算距离,同时,对于网格簇边界区域又转换成了数据点聚类,保证了密度聚类的高精度和准确性.

### 2.2 CBSCAN

CBSCAN 聚类框架如算法 1 所示:首先,根据 *Eps* 划分位置数据空间,得到 Cell 网格索引(如第 1 行);然后,对于每个位置点 *p*,快速获取其所在的网格 *G*,仅当该网格没有被聚类且该点也没有被聚类时,才对点 *p* 所在的网

© 中国科学院软件研究所 http://www.jos.org.cn

格进行扩展聚类,即执行 *EXPANDCLUSTER*()函数.*cid* 表示网格簇的标识号,*CC* 为网格簇集合.

**算法 1**. CBSCAN 算法框架.

```
Parameter: Eps and MinPts
Input: Dataset D.
Output: cell-clusters CC.
Algorithm:
1:  Get set of Cells according to D; cid ← 1; CC ← ∅;
2:  for each p ∈ D do
3:    Get Cell G ← p.coordinate;
4:    if G.cid == UNCLASSIFIED && p.cid == UNCLASSIFIED then
5:      Create cell-cluster C ← Cluster(cid);
6:      if EXPANDCLUSTER(p,G,D,C,Eps,MinPts) then
7:        cid++;
8:      end if
9:    end if
10:   CC ← CC ∪ {C};
11: end for
12: ruturn CC;
```

算法 2 给出了 *EXPANDCLUSTER*()函数的实现过程,该过程将对位置点的密度聚类转换成了对排他网格的扩展聚类.首先,判断网格 *G* 是否为排他网格,如第 1 行所示.若 *G* 为排他网格,则进行密度扩展(如第 2 行~第 21 行);否则,将点 *p* 置为噪声,若点 *p* 并非真正噪声,则在扩展聚类中将被处理成边界点.

**算法 2**. EXPANDCLUSTER 函数.

```
Function EXPANDCLUSTER(p,G,D,C,Eps,MinPts).
1:  if ∑|G| ≥ MinPts || ∑|innerCells(G)| ≥ MinPts || p is a core point then
2:    EXPANDCELLS(G,D,C,Eps,MinPts);
3:    outCells.add(outerCells(C.XGs));
4:    while outCells <> Empty do
5:      G ← outerCells.getFirst();
6:      if G.cid <> C.cid then
7:        for each p ∈ G do
8:          if dist(p,C.XGs) ≤ Eps then
9:            if p is a core point then
10:             EXPANDCELLS(G,D,C,Eps,MinPts);
11:             outCells.add(outerCells(C.XGs));
12:             break;
13:           else
14:             C.BPs.add(p); p.cid ← C.cid;
15:           end if
16:         end if
17:       end for
18:     end if
19:     outCells.remove(G);
20:   end while
21:   return true;
22: else
23:   p.cid ← Noise; return false;
24: end if
```

第 2 行~第 21 行展示了网格簇的扩展聚类过程:首先,对排他网格 *G* 进行扩展,如 *EXPANDCELLS*()函数所示;然后,对边界网格进行处理,*C.XGs* 表示网格簇 *C* 的所有排他网格,*outerCells*(*C.XGs*)表示 *C* 的所有边界网格,即所有排他网格的外围网格集合.*outCells* 用于存储需要检测的边界网格.对于边界网格中的位置点 *p*,如果点 *p* 是排他网格中核心点的邻居点(第 8 行),则 *p* 一定属于网格簇 *C*,若点 *p* 非核心点,则 *p* 是 *C* 的边界点(第 14 行);若点 *p* 是核心点,则还需进一步对 *p* 进行排他网格扩展,即执行 *EXAPNDCELLS*()函数、更新 *outCells* 列表,如第

© 中国科学院软件研究所 http://www.jos.org.cn

9 行~第 12 行所示.

EXPANDCELLS 函数如算法 3 所示,*innerCells*(*G*)表示网格 *G* 的 9 个相邻网格(包含 *G* 自身),*Gseeds* 为扩展网格列表.若 *G* 为排他网格($XG^1$ 或 $XG^2$),则继续扩展相邻网格(第 5 行~第 11 行);否则,*G* 为包含网格,不再向周边扩展.

**算法 3**. EXPANDCELLS 函数.

Function *EXPANDCELLS*(*p*,*G*,*D*,*C*,*Eps*,*MinPts*).

1: *C.IGs.add*(*G*);*C.XGs.add*(*G*);*Gseeds.add*(*innerCells*(*G*));
2: *Gseeds.remove*(*G*);
3: while *Gseeds* <> *Empty* do
4: $G \leftarrow$ *Gseeds.getFirst*();
5: if $\sum|$ *innerCells*(*G*)$| \geqslant$ *MinPts* then // $G \in XG^1$
6: *G.cid* $\leftarrow$ *C.cid*;*G.core* $\leftarrow$ true;*G.XGs.add*(*G*);
7: *C.IGs.add*(*G*);*Gseeds.add*(*innerCells*(*G*));
8: else if $\exists p \in G$, *p* is a core point then // $G \in XG^2$
9: *G.cid* $\leftarrow$ *C.cid*; *p.cid* $\leftarrow$ *C.cid*; *p.core* $\leftarrow$ true;
10: *G.XGs.add*(*G*);*G.IGs.add*(*G*);
11: *G.seeds.add*(*innerCells*(*G*));
12: else
13: *G.cid* $\leftarrow$ *Cid*,*C.IGs.add*(*G*);
14: end if
15: *Gseeds.remove*(*G*);
16: end while

### 2.3 复杂度分析

设定位置点数量为 *n*,依据 Cell 定义,划分 Cell 网格数量为 *m*,噪声和边界点比例为$\alpha$,因此,噪声和边界点数量为$\alpha \cdot n$.首先,划分网格的时间复杂度为 $O(n)$;其次,对于扫描到的噪声及边界点,无需再进行扩展,则扫描这些噪声和边界点至多只需扫描 36 个 Cell 网格,所以最坏时间复杂度为 $O(\alpha \cdot n \cdot 36)$.对于排他网格的扩展聚类过程,需要对网格进行遍历扩展,最坏情况下的时间复杂度为 $O(c \cdot m)$,其中,常量 *c* 表示为检测每个网格是否为排他网格的计算成本,对于 $XG^1$ 来说,*c* 仅为统计 9 个网格内位置点数量;对于 $XG^2$ 来说,*c* 最坏情况下为扫描 36 个外围网格.因此,总的来说,CBSCAN 的时间复杂度为 $O(n)+O(c \cdot m)+O(\alpha \cdot n \cdot 36)$.由于高密度区域的排他网格检测采用 Cell 距离分析,统计网格内位置点数量即可判断,极大地降低了计算成本 *c*;对于边界网格内边界点与噪声点,一般数量较少,又利用引理 2 限定了最大搜索区域,减少了邻居点查找时间.第 4.3 节的效率实验结果也验证了该时间复杂度分析.

## 3 实验测试及分析

本节对 CBSCAN 的聚类效果和效率进行了综合实验评估,并与 DBSCAN、采用 PR-tree[20]索引优化的 DBSCAN(记为 RDBSCAN)、采用 Grid[21]索引优化的 DBSCAN(记为 GDBSCAN)、*k*-means[22]、最小生成树聚类 MST[23,24]和 GG[12]进行了对比分析.所有算法由 Java 实现,实验平台采用 2.2GHz 至强 E5-2660 处理器,16G 内存,Windows Server 2012 操作系统.

### 3.1 实验数据和测试方法

- 实验数据

实验采用的数据见表 1,Aggregation~t4.8k 为 7 个包括任意形状、不同密度和数量聚簇的二维位置数据集,其中,Aggregation[25]包括 7 个非高斯分布的聚类簇;Compound[26]包含 6 个不同形状的复杂簇结构,涵盖了相连、内嵌和不同密度重叠的情况;Flame[27]包含了 2 个相连的密度簇结构;Pathbased[28]包括 1 个环形簇和 2 个内嵌的

© 中国科学院软件研究所 http://www.jos.org.cn

高斯密度簇;R15[29]由 15 个大小相似的高斯密度簇组成,D31[29]由 31 个高斯密度簇构成;t4.8k[30]由 7 个不同形状、互相嵌套,并且掺杂了大量噪声的密度簇构成.

**Table 1** Description of experimental datasets

**表 1** 实验数据描述

| Dataset | # of points | Dimension | Classes | Parameters |
|---|---|---|---|---|
| Aggregation | 788 | 2 | 7 | *Eps*=1.7, *MinPts*=10 |
| Compound | 399 | 2 | 6 | *Eps*=1.4, *MinPts*=3 |
| Flame | 240 | 2 | 2 | *Eps*=1.4, *MinPts*=10 |
| Pathbased | 300 | 2 | 4 | *Eps*=1.9, *MinPts*=3 |
| R15 | 600 | 2 | 15 | *Eps*=0.5, *MinPts*=15 |
| D31 | 3 100 | 2 | 31 | *Eps*=0.6, *MinPts*=15 |
| t4.8k | 8 000 | 2 | 7 | *Eps*=10, *MinPts*=20 |
| MG1 | 41 650 | 2 | 12 | *Eps*=20, *MinPts*=20 |
| MG2 | 120 000 | 2 | 50 | *Eps*=20, *MinPts*=20 |
| Road | 434 874 | 2 | Unknown | *Eps*=0.05, *MinPts*=100 |
| Taxi1 | 1 194 976 | 2 | Unknown | *Eps*=0.005, *MinPts*=20 |
| Taxi2 | 2 148 225 | 2 | Unknown | *Eps*=0.005, *MinPts*=20 |
| Gowalla | 3 635 468 | 2 | Unknown | *Eps*=0.04, *MinPts*=200 |
| Taxi3 | 9 872 768 | 2 | Unknown | *Eps*=0.003, *MinPts*=200 |

MG1 和 MG2 是采用混合高斯模型生成的两个数据集[22],分别包含 4.16 万和 12 万个数据点.MG1 由 12 个数量不同、部分相连的数据簇构成,MG2 由 50 个任意形状、部分边界重叠的密度簇组成.ROAD[31]包含约 43 万个位置点数据,来自丹麦北日德兰地区道路网位置采集数据.Taxi1、Taxi2 和 Taxi3 数据来自微软亚洲研究院的 T-Drive 项目[32],T-Drive 采集了北京 10 357 辆出租车一周内的 GPS 数据,Taxi1 提取了北京五环以内 5 千辆出租车 8 小时的位置数据,Taxi2 提取了 1 万辆出租车 8 小时的位置数据,Taxi3 提取了所有出租车两天的位置数据.Gowalla 数据[33]来自移动社交网站 Gowalla 的 20 万用户在 2009 年 2 月~2010 年 10 月间的签到位置数据,共包括 644 万个位置点,我们提取了在美国境内的位置数据,大约 363 万位置点.

- 测试方法

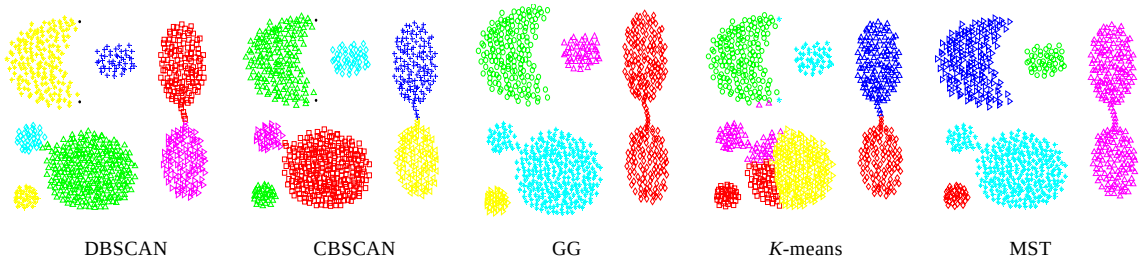
为了验证算法的有效性,在 Aggregation~t4.8k 等 7 个基准数据集上测试了 DBSCAN、GG、*k*-means、MST 和 CBSCAN 方法的聚类效果,评估方法采用与 DBSCAN 聚类结果相比较的方式.为了验证算法的高效性,在 MG1~Taxi3 这 7 个大规模位置数据集上测量了所提出方法和对比算法的聚类时间指标(CPU 时间)和内存消耗,评估了算法的效率和对输入参数的敏感性.

### 3.2 效果评估

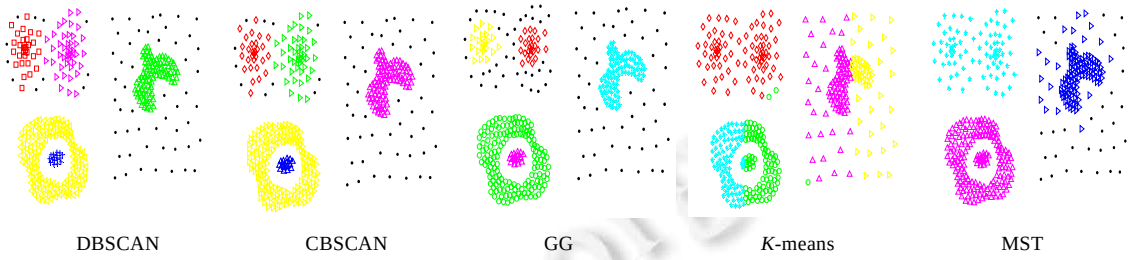
在基准数据集上的聚类结果如图 3 所示,CBSCAN 与 DBSCAN 在同一数据集上采用相同参数,见表 1,*k*-means 的参数 *k* 设置为表 1 所示的类数量,最小生成树算法的 *cut-off* 参数与 *Eps* 相同,GG 在表 1 基准数据集上参数依次如下:(40,60),(40,60),(30,60),(26,60),(76,40),(120,200),(115,40),第 1 个参数 *n* 为每维划分网格的数量,第 2 个参数 *m* 为选取高密度网格的数量.

为了便于分辨,使用不同颜色和形状表示不同的簇.从图中可以看出,CBSCAN 可以发现和 DBSCAN 相同的簇结构,并且能识别噪声点.这是因为 CBSCAN 虽然利用网格扩展,但是遵循密度相连的关系,属于精确的密度聚类方法.GG 算法选取 *m* 个较高密度的网格,然后采用简单的相邻网格扩展,聚类效果严重依赖于网格划分粒度以及数据分布,如在 Aggregation 数据集,很难将右侧两个簇分离;在 R15 上,同样很难将内部相邻簇区域分开;在 D31 上表现更为明显;在 Compound 数据集上,想要区分左上角的两个相连的簇和左下角相嵌套的簇时,需设置粒度较小的网格,但只能识别出中心两个簇,而把周围位置点识别成了噪声;在掺有噪声的 t4.8k 数据上,GG 也很难区分相连的聚类簇和边界的噪声数据.*K*-means 无法发现任意形状的聚类簇,如图 3 所示,得到聚类簇形状倾向于凸形或类球形.MST 将数据点间距离表示成树结构,利用 *cut-off* 参数得到多个簇,它同样很难区分相邻的簇结构,如 Aggregation,Compound 数据集.若要区分开相邻的簇结构,则需设置较小的 *cut-off* 值,但同时会产生较多个节点非常少的树(形成噪声)或将一个簇分成多个部分,如 R15 和 Pathbased 所示.此外,MST 也无法处理好 t4.8k 中的噪声数据.

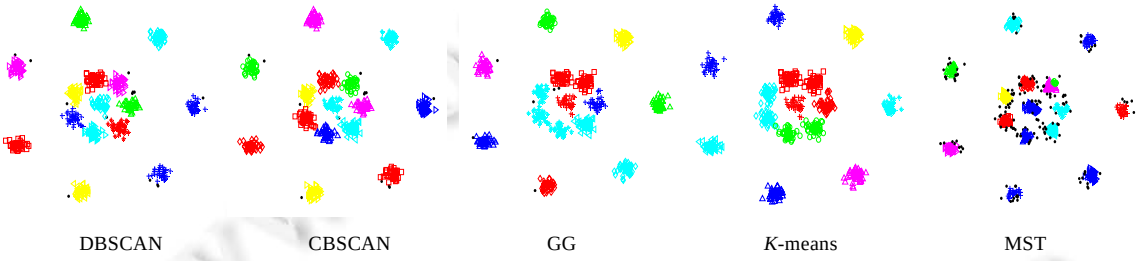
© 中国科学院软件研究所 http://www.jos.org.cn



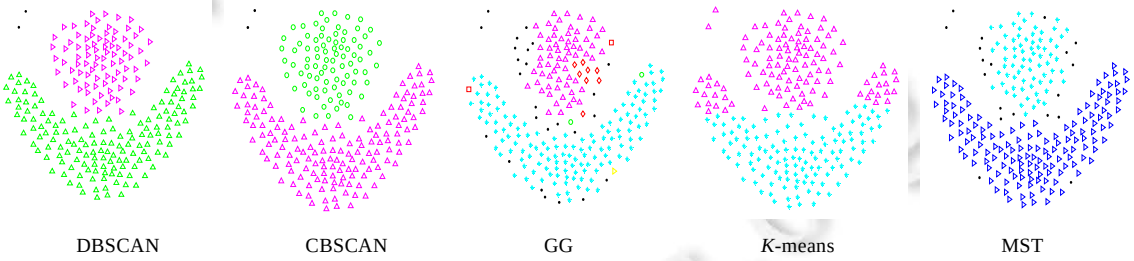

(a) Aggregation



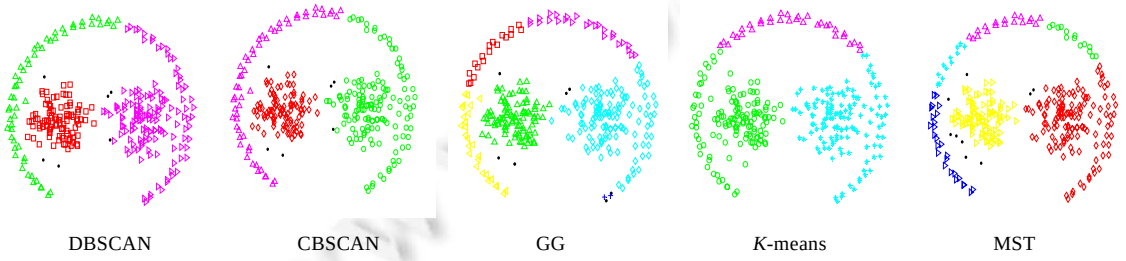

(b) Compound




(c) R15




(d) Flame

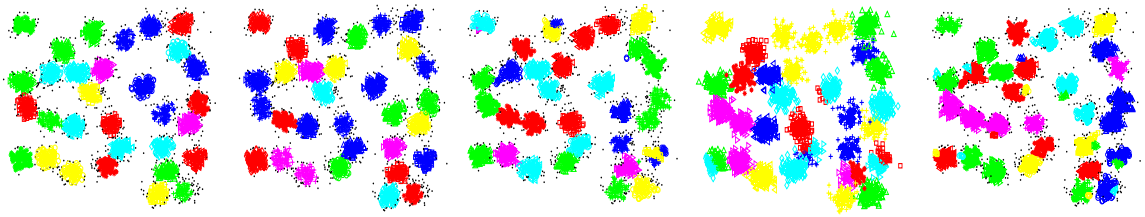



(e) Pathbased

Fig.3 Comparison of clustering results

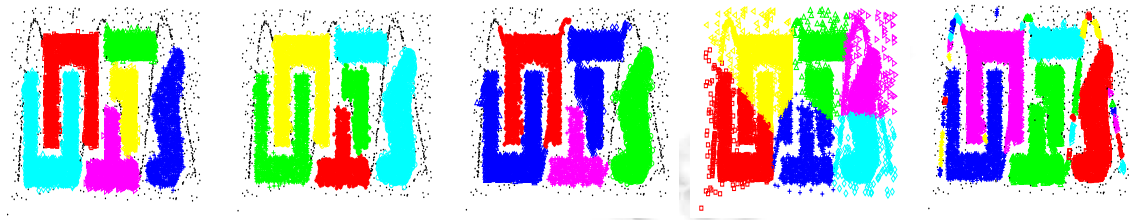
图 3 聚类效果对比

© 中国科学院软件研究所 http://www.jos.org.cn



DBSCAN CBSCAN GG *K*-means MST

(f) D31



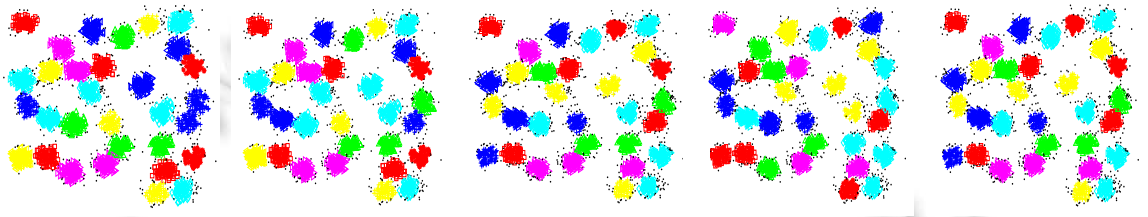
DBSCAN CBSCAN GG *K*-means MST

(g) t4.8k

Fig.3 Comparison of clustering results (Continued)

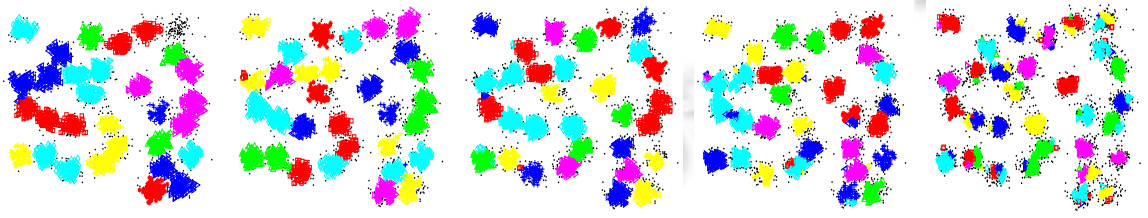
图 3 聚类效果对比(续)

在 D31 数据集上测试了网格大小对 CBSCAN 和 GG 算法聚类效果的影响,固定 *MinPts*=15(*m*=200)不变,结果如图 4 所示.



*Eps*=0.66 *Eps*=0.64 *Eps*=0.62 *Eps*=0.60 *Eps*=0.58

(a) CBSCAN



*n*=100 *n*=110 *n*=120 *n*=130 *n*=140

(b) GG

Fig.4 Comparison of clustering results w.r.t. grid size

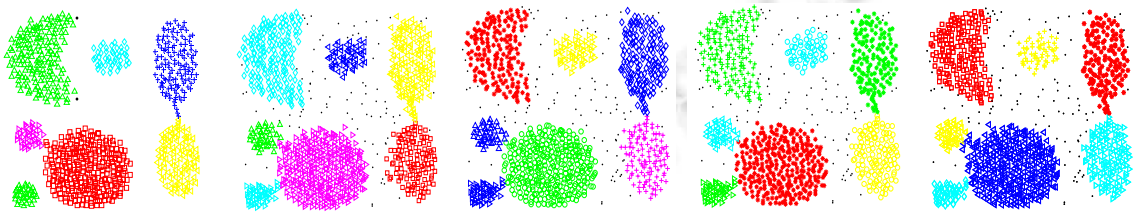
图 4 相对于网格大小的聚类效果对比

可以看出,CBSCAN 在较大的 *Eps* 范围内都能获取到较好的聚类结果.随着 *Eps* 的减少,较高密度区域的位置点聚为一簇,低于该密度阈值的位置点识别为噪声.而 GG 在该数据集上很难获取到较好的聚类效果,当网格粒度较大时,会将相邻的簇聚为一类;当网格粒度较小时,不仅将各簇划分开,还会产生较多的子簇,同时还将某

© 中国科学院软件研究所 http://www.jos.org.cn

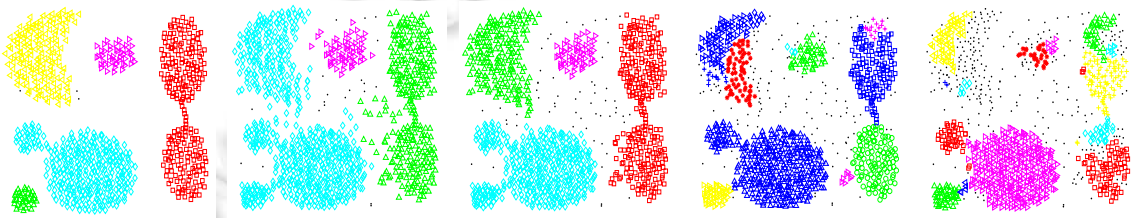
些较密区域的位置点识别为噪声.这是因为 GG 划分网格后,仅通过相邻 8 网格扩展,一旦网格内位置点数量为 0,则停止扩展.因此,网格粒度较大时,容易将相邻簇聚为一类;网格粒度较小时,容易分散聚类簇.除此之外,GG 还仅对前 $m$ 个较高密度网格展开聚类,所以未被扩展到的第 $m$ 个之后的网格数据可能未被处理到,导致识别成假噪声.

在 Aggregation 数据上测试了 CBSCAN 与 GG 算法对噪声识别的效果,我们在 Aggregation 数据上添加了 200 随机分布的噪声点.图 5 中第 1 列为加噪声之前的聚类效果,后面 4 列为在加噪声数据上变化网格大小的聚类效果.CBSCAN 通过调整参数可以较好地识别出噪声数据,并且能够获取到与加噪声之前相似的聚类簇结果.而 GG 算法很难在原数据上区分开相连的聚类簇,在噪声数据上,很容易通过噪声点将相近的簇连成一个簇,也很难识别出噪声数据.通过降低网格粒度,虽然能够识别部分噪声,但无法获取到有意义的聚类簇结果.原因与上一个实验分析相同,GG 仅扩展相邻的位置点数量不为 0 的网格,完全忽略数据点间的密度关系.



*Eps*=1.7, *MinPts*=10 *Eps*=1.58, *MinPts*=12 *Eps*=1.64, *MinPts*=14 *Eps*=1.7, *MinPts*=15 *Eps*=1.76, *MinPts*=16

(a) CBSCAN



$n$=40 $n$=30 $n$=40 $n$=50 $n$=60

(b) GG

Fig.5 Comparison of noise identification w.r.t. grid size

图 5 相对于网格大小的噪声识别效果对比

### 3.3 性能测试

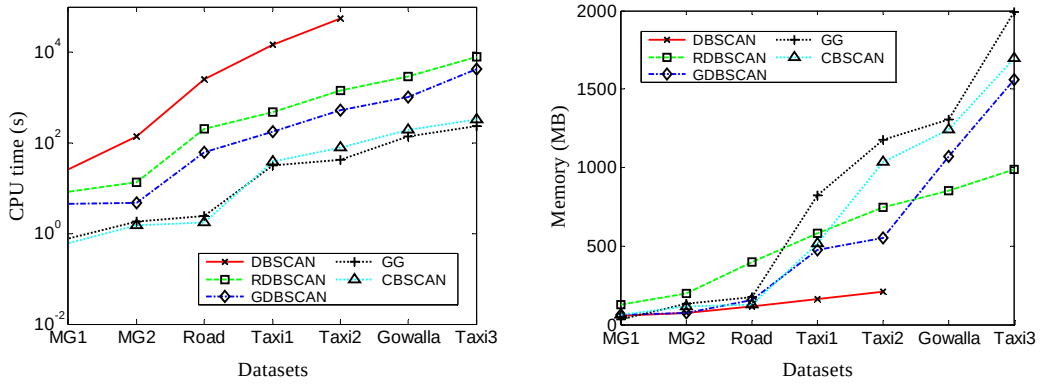
本节在位置大数据上测试了 CBSCAN 算法的聚类效率和空间上的内存消耗,并和 DBSCAN、RDBSCAN、GDBSCAN、GG 和 MST 进行了比较.各算法在同一数据集上设置相同的参数,见表 1,GG 在各数据集上的参数依次为(100,1000)、(100,1000)、(1000,1000)、(2000,20000)、(2000,20000)、(5000,20000)、(3000,20000).各算法所消耗的 CPU 时间见表 2,趋势图如图 6(a)所示,消耗内存如图 6(b)所示.

**Table 2** Comparison of clustering time (s)

**表 2** 聚类时间对比 (秒)

| Dataset | DBSCAN | RDBSCAN | GDBSCAN | GG | MST | CBSCAN |
|---|---|---|---|---|---|---|
| MG1 | 25.96 | 8.49 | 4.56 | 0.78 | 3.95 | 0.62 |
| MG2 | 136.75 | 13.31 | 4.83 | 1.82 | 内存溢出 | 1.49 |
| Road | 2 460.97 | 201.24 | 60.28 | 2.41 | 内存溢出 | 1.73 |
| Taxi1 | 14 449.47 | 482.34 | 178.68 | 30.98 | 内存溢出 | 38.62 |
| Taxi2 | 54 706.86 | 1 421.49 | 516.98 | 41.59 | 内存溢出 | 78.71 |
| Gowalla | × | 2 865.52 | 1 020.71 | 138.06 | 内存溢出 | 192.47 |
| Taxi3 | × | 7 941.51 | 4 262.01 | 233.81 | 内存溢出 | 322.02 |

© 中国科学院软件研究所 http://www.jos.org.cn




(a) CPU 时间 (Log) (b) 内存

Fig.6 Performance comparison w.r.t. datasets

图 6 在各数据集上的性能对比

从图 6(a)可以看出,随着数据量的增大,所有算法消耗时间也相应增加.CBSCAN 和 GG 算法消耗了最少的 CPU 时间,在 200 万以上数据集,CBSCAN 比 GG 消耗稍多时间.这是因为 CBSCAN 除了进行密度网格扩展聚类,还需要对大量边界点进行处理;而 GG 方法没有采用密度聚类,仅对网格进行扩展,并不能获取到精确的密度簇结果.可以发现,随着数据规模增加,CBSCAN 减少的聚类时间越来越多,展现了较高的可扩展性.根据 7 个数据集上的实验结果统计,PR-Tree 索引[20]对 DBSCAN 算法平均提升了 18 倍,Grid 索引[21]对 DBSCAN 算法平均提升了 49 倍.而相对于 DBSCAN,RDBSCAN 和 GDBSCAN,我们的 CBSCAN 算法分别平均提升了 525 倍、30 倍和 11 倍效率.

图 6(b)展示了各算法消耗的内存结果,MST 占用最多内存,在 4 万数据集上内存已达 10GB,在其他数据集上内存溢出.DBSCAN 占用最少的内存,RDBSCAN 使用 PR-Tree 索引,在 100 万以下数据集上消耗最多内存,而随着数量增加,占用内存量呈现线性增加.这是因为 PR-tree 的树状层次结构一旦覆盖了数据空间,数据量的增加则仅增加叶子节点数量.GDBSCAN 与 CBSCAN 都依据 *Eps* 划分网格,随着数据空间增大,网格数量快速增加,Cell 比 Grid 划分的网格粒度更细,因此占用更多内存.而 GG 若想获取到具有一定意义的聚类结果,需要划分较细粒度的网格,因此占用的内存最多.

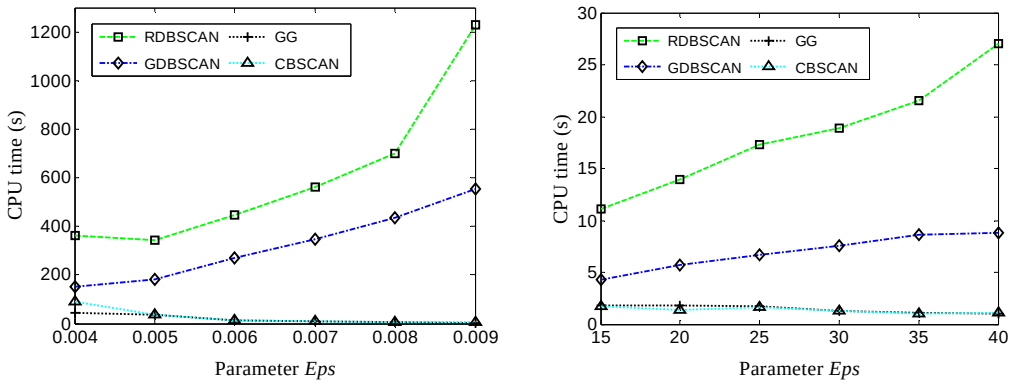
### 3.4 参数敏感性分析

本节在 Taxi1 和 MG2 上评估了 CBSCAN 聚类效率对输入参数的敏感性.在 MG2 和 Taxi1 上固定 *MinPts*=20,变化 *Eps* 的实验结果如图 7 所示.可以看出,CBSCAN 算法在所有测试参数下效率都明显优于 RDBSCAN 和 GDBSCAN,在两个数据集上平均提高了 45 倍和 22 倍效率.随着 *Eps* 的增加,CBSCAN 聚类时间逐渐减少,而 RDBSCAN 和 GDBSCAN 都相应地增加.这是因为当 *MinPts* 不变时,增加 *Eps* 使得更多的 Cell 网格变成了排他网格或包含网格,加快了网格扩展聚类,进而减少了聚类时间;而对 PR-Tree 和 Grid 索引,*Eps* 的增加使得每个位置点的邻居搜索区域变大,距离计算量增加导致聚类时间也相应地增加.

图 8 展示了各算法的效率随 *MinPts* 变化的实验结果,在 MG2 上固定 *Eps*=20,Taxi1 上固定 *Eps*=0.005,可以发现,RDBSCAN 和 GDBSCAN 基本不受 *MinPts* 影响.这是因为变化 *MinPts* 不影响 PR-Tree 和 Grid 索引对位置点的邻居搜索区域,距离计算量基本不变.相对而言,*MinPts* 对 CBSCAN 效率影响也比较小,但当 *Eps* 固定时,较小的 *MinPts* 导致聚类簇较大,进而需要处理的边界区域也较大,导致 CPU 时间较大,而较大的 *MinPts* 又使得排他网格数量减少,网格扩展聚类部分减少,边界点和噪声点增加,总的聚类时间也相应增加.从图 8(a)和图 8(b)上可以验证,随着 *MinPts* 从较小到较大变化,CBSCAN 聚类时间有一个先减小后又增加的趋势变化.当 *MinPts* 达到一定大小时,大多位置点变成噪声点,网格扩展部分占较小比例,CBSCAN 也就演化成了一个采用 Cell 索引的聚类.尽管如此,当 *MinPts* 过大时,利用推论 2 可快速排除非核心点,确定噪声数据,保证 CBSCAN 算法快速完

© 中国科学院软件研究所 http://www.jos.org.cn

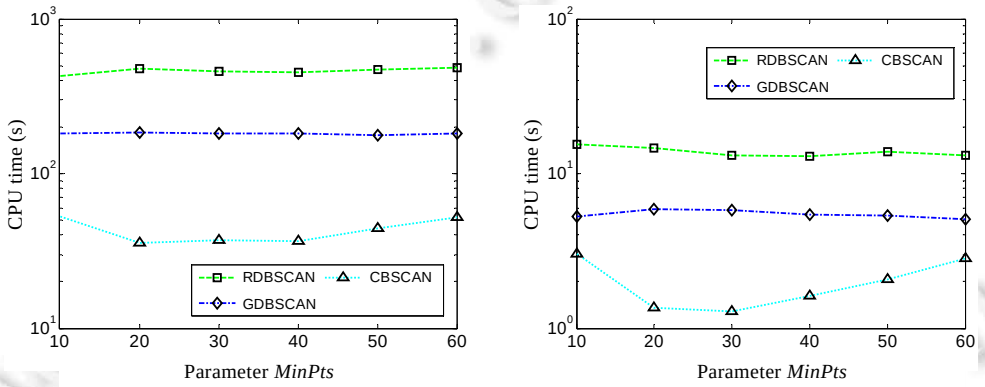
成聚类.




(a) Taxi1 数据集 (b) MG2 数据集

Fig.7 Efficiency comparison of algorithms w.r.t. *Eps*

图 7 相对于 *Eps* 的算法效率对比




(a) Taxi1 数据集(Log) (b) MG2 数据集(Log)

Fig.8 Efficiency comparison of algorithms w.r.t. *MinPts*

图 8 相对于 *MinPts* 的算法效率对比

## 4 总 结

针对位置大数据的密度聚类问题,本文实现了一种快速的基于密度和网格划分的聚类算法.与其他基于网格的密度聚类不同,提出的 Cell 距离分析模型无需计算距离,即可快速确定高密度区域的核心点和密度相连关系,利用排他网格与相邻网格关系,实现了基于密度的网格扩展算法,极大地减少距离计算量和对位置点的遍历.最后,在位置大数据上的综合评估实验验证了所提出算法具有较好的聚类效果和较高的效率,相对于 DBSCAN、PR-Tree 索引和 Grid 索引技术,分别平均提升了 525 倍、30 倍和 11 倍效率.

**致谢** 在此,特别感谢 Robert Olofsson 提供的 PR-Tree 索引源码;同时,对本文工作给予支持和建议的同行及评审老师也深表感谢.

**References**:

[1] Zheng Y. Trajectory data mining: An overview. ACM Trans. on Intelligent Systems & Technology, 2015,6(3):1−41.

© 中国科学院软件研究所 http://www.jos.org.cn

[2] Zheng Y, Zhang LZ, Xie X, Ma WY. Mining interesting locations and travel sequences from GPS trajectories. In: Proc. of the Int'l Conf. on World Wide Web (WWW 2009). Madrid, 2009. 791−800.

[3] Zheng Y, Zhang L, Ma ZX, Xie X, Ma WY. Recommending friends and locations based on individual location history. ACM Trans. on the Web, 2010,5(1):99−111.

[4] Guo C, Liu JN, Fang Y, Luo M, Cui JS. Value extraction and collaborative mining methods for location big data. Ruan Jian Xue Bao/Journal of Software, 2014,25(4):713−730 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4570.htm [doi: 10.13328/j.cnki.jos.004570]

[5] Liu JN, Fang Y, Guo C, Gao KF. Research progress in location big data analysis and processing. Geomatics and Information Science of Wuhan University, 2014,39(4):379−385 (in Chinese with English abstract).

[6] Macqueen J. Some methods for classification and analysis of multivariate observations. In: Proc. of the Berkeley Symp. on Mathematical Statistics and Probability. 1967. 281−297.

[7] Chen XL, Cai D. Large scale spectral clustering with landmark-based representation. In: Proc. of the AAAI Conf. on Artificial Intelligence (AAAI 2011). 2011.

[8] Vlassis N, Likas A. A greedy EM algorithm for Gaussian mixture learning. Neural Processing Letters, 2002,15(1):77−87.

[9] Ester M, Kriegel HP, Sander J, Xu XW. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. of the KDD. Portland: AAAI Press, 1996. 226−231.

[10] Sander J, Ester M, Kriegel HP, Xu XW. Density-Based clustering in spatial databases: The algorithm GDBSCAN and its applications. Data Mining & Knowledge Discovery, 2010,2(2):169−194.

[11] Chen N, Chen A, Zhou LX. An incremental grid density-based clustering algorithm. Ruan Jian Xue Bao/Journal of Software, 2002, 13(1):1−7 (in Chinese with English abstract). http://www.jos.org.cn/jos/ch/reader/create_pdf.aspx?file_no=20020101&year_id=2002&quarter_id=1&falg=1

[12] Zhao QP, Shi Y, Liu Q, Fränti P. A grid-growing clustering algorithm for geo-spatial data. Pattern Recognition Letters, 2014,53: 77−84.

[13] Guttman A. R-Trees: A dynamic index structure for spatial searching. In: Proc. of the Int'l Conf. on Management of Data. 1984. 47−57.

[14] Kumar KM, Reddy ARM. A fast DBSCAN clustering algorithm by accelerating neighbor searching using groups method. Pattern Recognition, 2016,58:39−48.

[15] Cao F, Tung AKH, Zhou AY. Scalable clustering using graphics processors. In: Proc. of the Int'l Conf. on Advances in Web-Age Information Management (WAIM 2006). 2006. 372−384.

[16] Cao F, Zhou AY. Fast clustering of data streams using graphics processors. Ruan Jian Xue Bao/Journal of Software, 2007,18(2): 291−302 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/291.htm [doi: 10.1360/jos180291]

[17] Böhm C, Noll R, Plant C, Wackersreuther B. Density-Based clustering using graphics processors. In: Proc. of the ACM Conf. on Information and Knowledge Management (CIKM 2009). Hong Kong, 2009. 661−670.

[18] He YB, Tan HY, Luo WM, Mao HJ, Ma D, Feng SZ, Fan JP. MR-DBSCAN: An efficient parallel density-based clustering algorithm using MapReduce. In: Proc. of the 2011 IEEE 17th Int'l Conf. on Parallel and Distributed Systems. IEEE Computer Society, 2011. 473−480.

[19] Kim Y, Shim K, Kim MS, Lee JS. DBCURE-MR: An efficient density-based clustering algorithm for large data using MapReduce. Information Systems, 2014,42(2):15−35.

[20] Arge L, De Berg M, Haverkort H, Yi K. The priority R-tree: A practically efficient and worst-case optimal R-tree. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. 2004. 442−450.

[21] Yu YW, Wang Q, Kuang J, He J. An on-line density-based clustering algorithm for spatial data stream. Acta Automatica Sinica, 2012,38(6):1051−1059 (in Chinese with English abstract).

[22] Qi JP, Yu YW, Wang LH, Liu JL. K*-Means: An effective and efficient *K*-means clustering algorithm. In: Proc. of the IEEE Int'l Conf. on Social Computing and Networking (SocialCom). Atlanta: IEEE Computer Society, 2016.

[23] Oleksandr G, Zhou Y, Zach J. Minimum spanning tree based clustering algorithms. In: Proc. of the 18th IEEE Int'l Conf. on Tools with Artificial Intelligence. 2006. 73−81.

© 中国科学院软件研究所 http://www.jos.org.cn

[24] Zhong CM, Miao DQ, Wang RZ. A graph-theoretical clustering method based on two rounds of minimum spanning trees. Pattern Recognition, 2010,43(3):752−766.

[25] Gionis A, Mannila H, Tsaparas P. Clustering aggregation. ACM Trans. on Knowledge Discovery from Data (TKDD), 2007,1(1): 1−30.

[26] Zahn CT. Graph-Theoretical methods for detecting and describing gestalt clusters. IEEE Trans. on Computers, 1971,100(1):68−86.

[27] Fu LM, Medico E. FLAME: A novel fuzzy clustering method for the analysis of DNA microarray data. BMC Bioinformatics, 2007, 8(1):1−15.

[28] Chang H, Yeung D. Robust path-based spectral clustering. Pattern Recognition, 2008,41(1):191−203.

[29] Veenman CJ, Reinders MJT, Backer E. A maximum variance cluster algorithm. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2002,24(9):1273−1280.

[30] Karypis G, Han EH, Kumar V. CHAMELEON: A hierarchical clustering algorithm using dynamic modeling. IEEE Trans. on Computers, 1999,32(8):68−75.

[31] Kaul M, Yang B, Jensen CS. Building accurate 3D spatial networks to enable next generation intelligent transportation systems. In: Proc. of the Int'l Conf. on Mobile Data Management. IEEE Computer Society, 2013. 137−146.

[32] Yuan J, Zheng Y, Xie X, Sun GZ. T-Drive: Enhancing driving directions with taxi drivers' intelligence. IEEE Trans. on Knowledge & Data Engineering, 2013,25(1):220−232.

[33] Cho E, Myers SA, Leskovec J. Friendship and mobility: User movement in location-based social networks. In: Proc. of the ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD). 2011.

**附中文参考文献:**

[4] 郭迟,刘经南,方媛,罗梦,崔竞松.位置大数据的价值提取与协同挖掘方法.软件学报,2014,25(4):713−730. http://www.jos.org.cn/1000-9825/4570.htm [doi: 10.13328/j.cnki.jos.004570]

[5] 刘经南,方媛,郭迟,高柯夫.位置大数据的分析处理研究进展.武汉大学学报:信息科学版,2014,39(4):379−385.

[11] 陈宁,陈安,周龙骧.基于密度的增量式网格聚类算法.软件学报,2002,13(1):1−7. http://www.jos.org.cn/jos/ch/reader/create_pdf.aspx?file_no=20020101&year_id= 2002&quarter_id=1&falg=1

[16] 曹锋,周傲英.基于图形处理器的数据流快速聚类.软件学报,2007,18(2):291−302. http://www.jos.org.cn/1000-9825/18/291.htm [doi: 10.1360/jos180291]

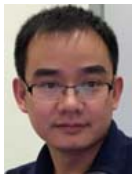
[21] 于彦伟,王沁,邝俊,何杰.一种基于密度的空间数据流在线聚类算法.自动化学报,2012,38(6):1051−1059.

**于彦伟**(1986－),男,山东菏泽人,博士,副教授,CCF 专业会员,主要研究领域为数据挖掘,机器学习,分布式计算.

**赵金东**(1974－),男,博士,副教授,主要研究领域为智能数据处理,无线传感器网络.

**贾召飞**(1995－),男,学士,主要研究领域为聚类分析.

**刘兆伟**(1979－),男,副教授,CCF 专业会员,主要研究领域为机器学习.



**曹磊**(1983－),男,博士,研究员,主要研究领域为数据挖掘,数据库系统.

**刘惊雷**(1971－),男,博士,教授,CCF 专业会员,主要研究领域为机器学习,数据挖掘.

© 中国科学院软件研究所 http://www.jos.org.cn