ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.2, February 2007, pp.332−344
DOI: 10.1360/jos180332

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

基于道路网络的对象聚类*

陈继东+, 孟小峰, 赖彩凤

(中国人民大学 信息学院,北京 100872)

Clustering Objects in a Road Network

CHEN Ji-Dong+, MENG Xiao-Feng, LAI Cai-Feng

(Information School, Renmin University of China, Beijing 100872, China)

+ Corresponding author: Phn: +86-10-62514994, Fax: +86-10-62519453, E-mail: chenjd@ruc.edu.cn, http://www.ruc.edu.cn

Chen JD, Meng XF, Lai CF. Clustering objects in a road network. *Journal of Software*, 2007,18(2):332−344.
http://www.jos.org.cn/1000-9825/18/332.htm

Abstract: Most spatial clustering algorithms deal with the objects in Euclidean space. In many real applications, however, the accessibility of spatial objects is constrained by spatial networks (e.g. road network). It is therefore more realistic to work on clustering objects in a road network. The distance metric in such setting is redefined by the network distance, which has to be computed by the expensive shortest path distance over the network. The existing methods are not applicable to such cases. Therefore, by exploiting unique features of road networks, two new clustering algorithms are presented, which use the information of nodes and edges in the network to prune the search space and avoid some unnecessary distance computations. The experimental results indicate that the algorithms achieve high efficiency for clustering objects in real road network.

Key words: data mining; clustering; road network; spatial object; network distance; shortest path

摘 要: 大多数的空间聚类算法主要针对欧几何空间中的数据对象.然而在大多真实的应用中,空间对象的访问主要受限于空间网络(如道路网络),因此,对道路网络中的对象进行聚类分析更具有现实意义.道路网络中对象之间的距离度量需要通过基于网络的最短路径距离来重新定义,其计算代价高,这使得已有的基于欧几何距离的聚类算法不能直接运用到这种环境中.因此,通过开发道路网络的特征提出了两种新的聚类算法.算法使用网络中的边和结点信息来缩减搜索空间,避免了一些不必要的距离计算.实验结果表明,算法对于真实道路网络中的对象聚类是高效的.

关键词: 数据挖掘;聚类;道路网络;空间对象;网络距离;最短路径
中图法分类号: TP311 文献标识码: A

* Supported by the National Natural Science Foundation of China under Grant Nos.60273018, 60573091 (国家自然科学基金); the National Grand Fundamental Research 973 Program of China under Grant No.2003CB317000 (国家重点基础研究发展计划(973)); the Key Project of the Ministry of Education of China under Grant No.03044 (国家教育部科学技术重点项目); the Program of the Ministry of Education of China for New Century Excellent Talents in University (NCET) (国家教育部新世纪优秀人才支持计划); the Creative Ph.D. Thesis in Renmin University of China (中国人民大学博士学位论文创新资助计划)

Received 2005-12-21; Accepted 2006-06-01

聚类分析是对数据对象进行分组,使得同一个组中对象之间具有较高的相似度,而不同组中的对象差别较大.聚类分析作为数据挖掘的核心技术,已经应用于多个领域.其中,对空间数据的聚类分析一直是研究热点之一.在空间数据库中,通过聚类可以识别出密集和稀疏的区域,因而发现全局的分布模式以及数据属性之间的相互关系.国内外的研究者已经提出了大量的空间聚类算法[1−11].文献[1,12]对现有算法进行了总结和比较,主要分类为划分方法[1,2,11]、层次方法[3−6,10]和基于密度的方法[7−9].目前,大多数空间聚类算法主要针对欧几何空间中的数据对象.然而真实的应用中,空间对象的访问都受限于空间网络(如道路网络),因此,对道路网络中对象的聚类研究更具应用价值[13].例如,在交通管理系统中,通过对车辆的聚类分析得到的车辆密集区域可用于标识道路中出现交通堵塞的潜在区域.本文主要针对交通堵塞检测这种应用来研究道路网络中的对象聚类问题.

在空间数据库中,数据对象以其欧几何空间的位置为主要特征,使得对象之间的距离相似度也是通过计算它们之间的欧几何距离(Euclidean distance)而得到的.然而在道路网络中,对象之间的距离相似度需要通过基于网络的最短路径距离即网络距离(network distance)来重新定义,其计算代价高.当经典的 Dijkstra 最短路径算法用于网络中的结点时,其时间复杂度为 $O(|V|\log|V|)$,其中,V 为网络中的结点数.为了加速最短路径计算,许多工作使用实体化技术将预计算的距离存储在实体化视图中,通过空间代价来换取时间代价.当网络结点的数目增加时,需要基于磁盘的最短路径算法.文献[14]提出了 CCAM(connectivity-clustered access method)的磁盘存储结构,基于连接性和访问频率把相邻的结点以及它们的邻接链表一起存储在相同的磁盘页上,以此来最小化最短路径计算过程中的 IO 访问.为了解决带约束限制的最短路径计算问题,文献[15]通过利用空间存取结构和欧几何距离进行过滤的预处理技术来加速基于磁盘的最短路径计算.另外,文献[16]提出了一种基于编码的方法来对道路网络结点进行编码,然后直接计算结点间的海明码距离就可以快速地得到它们之间的最短路径距离,但编码是基于对网络边的预先分割和离散化,这使得最短路径距离的计算结果可能存在偏差.这些最短路径计算的改进工作在某种程度上减少了两个对象之间的网络距离计算代价,但将基于欧几何空间的聚类算法直接用网络距离替换后,需要大量重复的对象间最短路径计算,使得总的网络距离计算代价依然很高.此外,道路网络中的对象不仅分布在网络的结点上,也可能是网络边上的任意位置,它们之间的网络距离计算还需要调整传统的最短路径算法.另一方面,有许多工作研究道路网络上的查询处理问题,文献[17]配置了一种基于磁盘的网络表示方法,集成网络连接和欧几何位置信息以支持基于网络距离的最邻近(nearest neighbor,简称 NN)查询、范围(range)查询和最近对(closest pair)查询.文献[18]分别使用欧几何距离边界属性、空间存取方法和网络距离实体化技术来处理道路网络中的聚集最近邻(aggregate nearest neighbor,简称 ANN)查询问题.文献[19]也利用一个基本的图论原理和查询结果实体化技术减少 Dijkstra 算法中的网络扩展,以加速网络图中的反向最近邻(reverse nearest neighbor,简称 RNN)查询处理.我们的主要思路与这些工作相似,即结合网络的特征和具体的问题,以尽量减少对象间最短路径距离的计算及网络扩展.但我们所要解决的问题不同,我们关注道路网络中对象的聚类分析而不是查询处理问题.

另外,现有的几种空间聚类方法中的一些概念,如聚类中心[2]、聚类特征[3]等,难以在道路网络中进行定义.具体来说,给定道路网络中的一组对象,由于可能在网络边上不存在一个唯一的点,与这组中所有对象之间的平均距离最小,所以不能使用网络距离来定义它们的聚类中心.而且即使有这样的中心点,找到中心点的代价也很高.综上所述,考虑到网络距离的计算代价和聚类中心等概念难以定义的问题,不能直接将已有的空间聚类算法运用到道路网络的环境.文献[20]提出了带障碍(obstacle)的空间对象聚类方法,考虑的是当空间网络中的道路、河流等作为障碍物的情况下对象的聚类.在这种情况下,聚类的对象不在网络边上,网络中的对象数目也较少.而本文则关注大量位于网络边上的对象(如城市道路网络中的车辆).Yiu 等人在 SIGMOD2004 上首次提出空间网络中对象聚类的问题[13],并提出相应的解决方案来扩展存在的聚类方法,将其运用到空间网络中的对象上.他们提出的聚类算法对于网络中对象稀疏的情况较为适用,而对于网络中对象密集的情况,算法的效率较低.

本文也试图解决同样的问题,对于网络中对象密集的情况,利用更多道路网络的信息,开发出网络中对象聚类的特征来缩减搜索空间,避免更多不必要的距离计算.通过使用对象表示中包含的边和结点的信息,在保证聚类效果的情况下改进聚类的效率,提出两种新的道路网络中对象的聚类方法:基于边的聚类和基于结点的聚类.

理论分析和对比实验结果都表明,所提出的算法是可行的、高效的.

本文第 1 节给出基本模型和问题定义.第 2 节详细描述所提出的聚类算法.第 3 节对算法进行性能分析和实验.最后是结论.

1 基本模型和问题定义

本节首先引入文献[13]中的网络定义和位于这个网络中的对象以及对象之间的网络距离定义,然后根据道路网络的聚类特点,增加了聚类间的网络距离定义.最后,通过提出一个新的聚类块的概念来定义道路网络中的对象聚类问题.

定义 **1**. 一个网络可以表示为一个无向带权图 $G=(V,E,W)$,其中:V 是顶点(结点)的集合;E 是边的集合;W 为正的实数集合($W:E\to IR^+$),表示边所对应的权值.网络中的一个对象位于网络中的边 $e(e\in E)$上,对象在网络中的位置可以表示为一个三元组(n_i,n_j,pos)(取代欧几何的空间坐标表示),其中:n_i 和 n_j 是对象所在网络边的两个结点;$pos\in[0,W(e)]$是对象与结点 n_i 之间的距离.

一个对象位于且仅位于一条网络边上(在实际的应用中,一些对象可能不在网络边上或者由于位置获取的误差不能定位在网络边上,这样的对象可以通过与其最近的网络边来近似表示).对于一个道路网络,为了降低其复杂性,可以将道路网络的交叉口建模为网络图的结点,而将交叉口之间的路段建模为网络图的边.

当聚类道路网络中的对象时,需要定义对象之间的距离相似度为网络距离,而取代欧几何距离.假设 p 和 q 为网络中的两个对象,其位置分别为(n_a,n_b,pos_p)和(n_c,n_d,pos_q),定义对象间网络距离度量如下:

定义 **2**. 对象之间的网络距离:

1) 同一网络边上的对象之间的直接距离:$Dd(p,q)=|pos_p-pos_q|$(其中,对象 p 和 q 在同一条网络边上,即 $n_a=n_b$ 且 $n_c=n_d$);否则,对象 p 和 q 不在同一网络边上,其直接距离为无穷大.

2) 同一网络边上的对象和结点之间的直接距离:$Dd(p,n_a)=pos_p$;$Dd(p,n_b)=W(n_a,n_b)-pos_p$.

3) 结点之间的网络距离:$Dn(n_i,n_j)$为从结点 n_i 到结点 n_j 的最短路径距离.

4) 不同网络边上的对象之间的网络距离:$Dn(p,q)=\min_{x\in\{a,b\},y\in\{c,d\}}(Dd(p,n_x)+Dn(n_x,n_y)+Dd(n_y,q))$.

由定义可以看出,道路网络中对象之间的网络距离相似度可以分为 4 种情况:前两种情况的网络距离定义为直接距离,不需要两结点之间的最短路径距离计算,可以在常量时间内得到;后面两种情况定义为网络距离,需要两结点之间的最短路径距离计算,计算代价高.

定义 **3**. 聚类之间的网络距离为聚类不同边界对象之间最小的网络距离.假设聚类 C_x 的边界对象为 p_{x1}, $p_{x2},\ldots,p_{xm}$,聚类 C_y 的边界对象为 $p_{y1},p_{y2},\ldots,p_{yn}$,则聚类 C_x 和 C_y 的相似度 $M(C_x,C_y)=\min_{i\in\{1,m\},j\in\{1,n\}}Dn(p_{xi},p_{yj})$.

聚类之间的网络距离计算需要先确定聚类的边界对象.在道路网络中,聚类的边界与网络中的结点相关,我们把聚类所包含的对象中与结点最近的对象作为聚类的边界对象.但如果某结点相连的所有边都包含在聚类中,则不把与该结点最近的对象作为聚类的边界对象.

道路网络中的对象聚类通常由位于多个网络边上的对象集合构成,因此,为了更好地表示聚类结果,减少对象之间网络距离的计算,我们提出了聚类块的概念.

定义 **4**. 聚类块(clustering block,简称 CB)为根据阈值ε构建的某个网络边上的一个微小聚类,CB 表示为$(O,n_a,n_b,posc,len)$.其中:O 为对象的集合 $O=\{o_1,o_2,\ldots,o_i,\ldots,o_n\}$($o_i=(n_a,n_b,pos_i)$,$pos_i<pos_{i+1}(1\le i\le n-1)$),满足 $Dd(o_i,o_{i+1})\le\varepsilon(1\le i\le n-1)$,即任意两个相邻对象之间的直接距离小于等于阈值$\varepsilon$;$(n_a,n_b)$为聚类块所在的边;$posc$ 为聚类块的中心位置;len 为聚类块的长度,等于所包含的两个相距最远的对象之间的距离.

由定义可以看出:在网络的某个边上可能有多个聚类块,但某个聚类块仅仅属于一条边,聚类块本身也是一个聚类.因此,网络中的聚类由一个或多个相邻的聚类块组成.给定位于一个网络中 N 个对象点的集合,对这些对象进行聚类的过程转换为根据它们的直接距离形成聚类块,并根据聚类块之间的网络距离合并聚类块的过程.基于边的聚类方法本质上就是利用聚类块,通过这个过程对网络中的对象进行聚类.

定义 **5**. 聚类块的ε有效结点为聚类块所在边的两个结点中与聚类块的距离小于等于ε的结点.对于聚类块

$CB(\{o_1,o_2,\ldots,o_m\},n_a,n_b)$,假设 o_1 和 o_m 分别为聚类块相对于结点 n_a 和 n_b 的两个边界对象,若 $Dd(o_1,n_a)\leq\varepsilon$,则 n_a 是 CB 的有效结点;若 $Dd(o_m,n_b)>\varepsilon$,则 n_b 不是 CB 的有效结点.聚类的ε有效结点为相邻结点和所包含的结点中与该聚类之间的距离小于等于ε的结点.由于一个聚类可以由若干个聚类块组成,所以,一个聚类的ε有效结点可以通过合并其所包含的聚类块的ε有效结点得到.

引入聚类和聚类块的ε有效结点可以加快道路网络聚类算法中聚类(块)之间的合并过程.我们将在算法描述中详细说明聚类的ε有效结点对算法效率的改进.

2 基于道路网络的对象聚类算法

现有的空间聚类方法由于难以开发出对象的其他可用信息,仅仅通过对象自身的空间相似性如欧几何距离来进行粗糙的分组,需要反复扫描所有的对象信息,代价较高.而对于道路网络的情况,可以观察到对象的聚类结果与网络的边和结点相关.具体来说,在相同或相邻边上的对象更可能被分组到同一个聚类中,而结点周围的对象划分到一起的可能性也更大(如交通阻塞通常发生在城市道路的交叉口).基于此,本文使用对象表示中包含的边和结点的信息来改进聚类效率,提出两种聚类方法:基于边的聚类和基于结点的聚类.下面分别加以描述.

2.1 基于边的聚类算法

2.1.1 算法的思路以及与相关文献的比较

根据聚类的形成方式,层次聚类方法可以分为自顶向下(分裂的层次聚类)和自底向上(凝聚的层次聚类)两种.其中,凝聚的层次聚类[3−5]最为常见.这种方法首先将每个对象作为一个聚类,然后合并这些聚类,直到所有的对象都在一个聚类中或者某个终止条件得到满足为止.当对象的数目很多时,由于开始把每个对象作为一个聚类,因此,需要合并的聚类数目多、代价大.另外,不论是自底向上的凝聚方法还是自顶向下的分裂方法,其合并和分裂的结果都不能被撤消,可能会导致低质量的聚类结果.文献[13]的 single-link 方法实际上是一种层次聚类方法,它在聚类空间网络中的对象时,首先将每个对象作为一个聚类,并初始化一个很大的堆结构来存储这些要合并的候选聚类对,再逐步合并距离最近的聚类对.这在道路网上对象稀疏时其代价还可以接受,但当对象数目增多、道路网络对象密集时,将导致算法的时间消耗和存储代价都很高.因此,我们考虑利用对象包含的边信息,从中间某个层次将对象划分为初始聚类,再分别向底层和顶层进行层次的分裂和合并,调整聚类的结果.基于这种思想,提出了基于边的聚类方法,它实际上也是一种层次聚类方法.

2.1.2 算法说明

基于边的聚类方法分为 3 个阶段:初始化阶段、分裂阶段和合并阶段.

1) 初始化阶段(initiation phase):按照对象所在的网络边将其分组,即位于同一个边上的对象被指定到同一个聚类中.初始聚类的个数为网络边的个数.这个阶段可以过滤掉那些没有对象的网络边,从而减少不必要的网络搜索过程.

2) 分裂阶段(splitting phase):根据对象之间的距离相似度,依次分裂大的聚类为细粒度的聚类块,得到聚类的中间结果.具体来说,对于每个初始的聚类,若其中某两个相邻对象之间的网络距离大于预先给定的阈值ε,则在这两个相邻的对象之间将聚类划分开来,分裂为两个更细粒度的聚类块;否则,可将其作为一个单独的聚类块.这个过程保证了聚类块内部的紧凑性,即聚类块内部任意两个相邻对象之间的网络距离均不大于ε.另外,为了优化接下来的聚类合并,还需要在这个过程中为每个聚类块指定其ε有效结点.

3) 合并阶段(merging phase):根据聚类块之间的距离相似度循环合并相邻的聚类(块),以形成最终的聚类结果.具体来说,利用聚类块的有效结点链表,根据阈值ε和聚类块之间的网络距离,对结点周围相邻的聚类块进行合并(同一个边上分裂后的聚类块之间不需要合并),反复合并每个结点周围相邻的聚类(块),直到不能再合并为止,得到最后的聚类结果.合并的具体细节将在算法的关键技术部分予以阐述.

基于边的聚类算法其伪码如下:

算法 **1**. 基于边的聚类算法 *Edge-based-cluster*().

输入:基于一个网络的对象集合,网络结点和边链表,距离阈值ε参数.

输出:对象的聚类结果.

Q:优先队列,用来存储结点周围待合并的有序聚类(块).

Begin

1. **for** 网络边链表中有对象存在的每一个边(n_x,n_y) **do** /*初始化阶段*/
2. 为边(n_x,n_y)创建一个新的聚类 *C*,分配 *cid*;
3. **for** 每一个创建的聚类 C_i **do** /*分裂阶段*/
4. *o* 为 C_i 对应的边(n_x,n_y)的第 1 个对象;*o.cid*=C_i.*cid*;
5. **if** *Dd*(*o.pos*,n_x)≤ε **then** 将〈n_x,*Dd*(*o.pos*,n_x)〉插入到 C_i 的有效结点链表 *nodelist* 中;
6. *nexto* 为边(n_x,n_y)从 *o* 到结点 n_y 的下一个对象;*C*=C_i;
7. **while** *nexto*<>null **do**
8. **if** *Dd*(*o.pos*,*nexto.pos*)>ε **then**
9. 将 *C* 分裂为 CB_1 和 CB_2;*C*=CB_2;
10. **if** *nexto* 是边(n_x,n_y)的最后一个对象 **AND** *Dd*(*nexto.pos*,n_y)≤ε **then**
11. 将〈n_y,*Dd*(*nexto.pos*,n_y)〉插入到 *C* 的有效结点链表 *nodelist* 中;
12. *o*=*nexto*;*nexto* 为边(n_x,n_y)从 *o* 到结点 n_y 的下一个对象;
13. *o.cid*=*C.cid*;
14. **for** 网络结点链表中的每一个结点 n_i **do** /*合并阶段*/
15. 初始化优先队列 *Q* 为空;
16. 对于其 *nodelist* 包含结点 n_i 的聚类(块),将其按照与 n_i 的距离排序后插入到优先队列 *Q* 中;
17. **if** *Q* 不为空 **then**
18. C_i 为 *Q* 中取出的一个聚类;C_j 为 *Q* 中取出的下一个聚类;
19. **while** *Dd*(C_i,n_i)+*Dd*(C_j,n_i)≤ε **do**
20. 将聚类 C_j 合并到 C_i,将 C_j 的有效结点链表 *nodelist* 合并到 C_i 的有效结点链表 *nodelist* 中;
21. **if** *Q* 不为空 **then** C_j 为 *Q* 中取出的下一个聚类;
22. **else break**;
23. **for** n_i 的每个相邻结点 n_z **do**
24. **if** 边(n_i,n_z)上没有聚类 **AND** *Dd*(C_i,n_i)+*W*(n_i,n_z)≤ε **then**
25. 将〈n_z,*Dd*(C_i,n_i)+*W*(n_i,n_z)〉插入到 C_i 的有效结点链表 *nodelist* 中;

End

为了减少网络中对象的访问次数,算法在初始化阶段只是为有对象存在的每个边创建一个聚类,并不指定边上的对象到相应聚类中;而在分裂阶段遍历聚类对应边上的对象时,将其指定到分裂后的聚类块中(第 4 行~第 13 行).算法中的聚类(块)合并过程将在下一节详细加以解释.

2.1.3 算法的关键技术

算法的关键部分是聚类(块)的合并过程(为叙述方便,下面将聚类块也统称为聚类).一般来说,聚类的合并过程需要反复计算两个聚类之间的网络距离相似度,以判断它们是否需要合并.在道路网络中,当聚类不相邻时,聚类之间的网络距离相似度必须通过计算其边界对象的最小网络距离得到,计算代价高.然而,根据道路网络中对象的聚类特点,当算法的分裂阶段结束之后,由于同一个边上分裂后的聚类之间的网络距离大于阈值ε,不需要再合并,因此,只有两种情况下的聚类可能合并:1) 与结点相邻的聚类之间;2) 跨越较短边的聚类之间,即聚类不相邻,但它们之间的边上没有任何聚类,并且边的长度短,聚类之间的网络距离(聚类分别与相邻边两个结点的距离与边的长度之和)小于等于阈值ε,这种情况也需要合并.基于这两种情况,算法在合并时从每个结点出发,只需合并与结点相邻的聚类,并在合并后对结点进行简单的网络扩展,以判断是否有跨越较短边的聚类需

要合并.

具体来说,算法为每个聚类维护一个ε有效结点链表ε-*nodelist*,链表每一项为(ε-*node*,ε-*dist*),其中:ε-*node* 为此聚类包含的ε有效结点;ε-*dist* 为此聚类与ε-*node* 的距离.它可用于计算与该结点相邻的聚类之间的网络距离(聚类到该结点的距离之和).ε-*dist* 为聚类中离ε-*node* 最近的边界对象与该结点的距离.然而,聚类的边界对象随着聚类的合并而发生变化,聚类与结点的距离也随之改变.为了避免由于边界对象的变化而需要重新合并聚类的过程,在算法实现过程中,当结点相邻的多个聚类合并时,首先通过聚类的ε-*nodelist* 过滤掉与该结点的距离大于ε的聚类,再对满足条件的聚类按其离结点的距离来排序,按顺序依次计算相邻聚类间的网络距离,若判断出距离大于ε,则停止比较之后的聚类,它们不能合并.对于ε-*nodelist* 为空的聚类,它们也不与其他聚类进行合并.因此,使用聚类的ε有效结点链表并将聚类按离结点的距离预先排序,减少了很多不必要的聚类之间网络距离的计算,也避免了由于合并后聚类与结点的距离改变而导致的聚类之间网络距离的重复计算.

聚类的ε有效结点链表的构造和维护比较简单.其构造可在算法的分裂阶段扫描第 1 个对象和最后一个对象时判断其与结点的距离而得到(第 5,11,12 行).在聚类合并后,其相应的ε有效结点链表也进行更新(第 20 行).另外,当结点扩展到相邻边时,若判断出有跨越较短边的聚类需要合并,则将相邻边上的对应结点也加入到当前聚类的ε有效结点链表中(第 23~25 行),使得在对那个结点周围的聚类合并时也考虑到当前聚类.

图 1 举例说明道路网络中聚类合并的具体过程.图 1 为道路网络图的一部分,假设ε设为 10,算法在分裂阶段遍历对象时计算聚类的有效结点,得到聚类 *CB*1,*CB*3,*CB*4 的有效结点为{*J*1};*CB*2 的有效结点为{*J*1,*J*2};*CB*5 的有效结点为{*J*2};*CB*6 的有效结点为{*J*2,*J*3}.在合并阶段,对于结点 *J*1,由于聚类 *CB*1,*CB*2,*CB*3,*CB*4 的有效结点都包括 *J*1,按照其与 *J*1 的距离排序,插入到优先队列 *Q* 中,即{*CB*2,*CB*1,*CB*3,*CB*4},依次取出队列中的聚类判断是否合并,由于 *CB*2 与 *CB*1 的距离(2+1=3)小于 10,首先将其合并为 *C*1,其有效结点更新为{*J*1,*J*2},且 *C*1 与 *J*1 的距离为 1,与 *J*2 的距离为 2.然后计算 *C*1 与 *Q* 的下一个聚类 *CB*3 的距离为 3+1=4 小于 10,则将 *CB*3 合并到 *C*1,且 *C*1 与 *J*1 的距离仍为 1,继续计算 *C*1 与 *Q* 的下一个聚类 *CB*4 的距离,结果将 *CB*4 也合并到 *C*1 中,此时,队列 *Q* 为空,该结点周围的聚类合并过程结束.再对结点 *J*2 合并其周围的聚类,由于 *CB*5,*CB*6 和 *C*1 的有效结点都包含 *J*2,按照其与 *J*2 的距离排序后插入到优先队列,得到 *Q* 为{*C*1,*CB*5,*CB*6},依次合并 *CB*5 和 *CB*6 到 *C*1 中,结点 *J*2 周围的聚类完成了合并过程.同理,对于其他结点反复执行这个过程,直到所有结点周围的聚类不能再合并为止,形成最后的聚类结果.

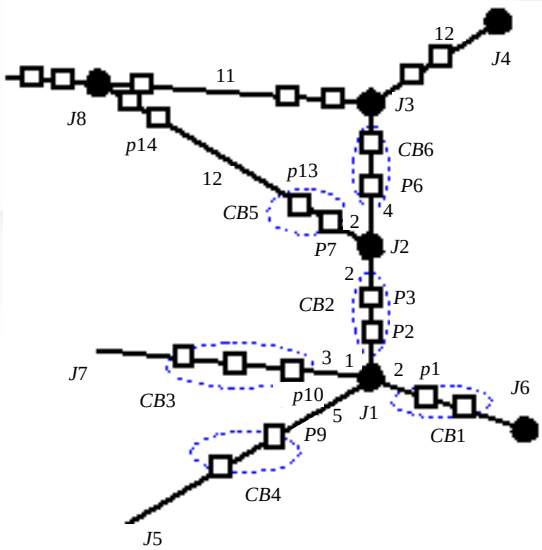

Fig.1 The merging of clusters

图 1 聚类的合并

2.2 基于结点的聚类算法

2.2.1 算法的思路以及与相关文献的比较

基于密度的聚类算法(如 DBSCAN(density based spatial clustering of applications with noise)方法[7])首先任意选择一个对象执行ε范围查询,如果该范围内至少包含 Min*Pts* 个对象,则为这个对象创建一个聚类,并将该范围内的对象都归到这个聚类中.然后,反复地对聚类中新的对象执行范围查询,直到聚类不再扩张为止.将这个算法用于道路网络中的对象后,其主要部分为查找网络中对象的ε范围内的相邻对象,需要对每个对象点执行一次基于网络距离的范围查询,当对象数量大时,代价很高.文献[13]中的ε-link 方法实际上是一种基于密度的聚类算法,它在网络遍历过程中插入新的对象点到当前聚类时,结点与聚类的距离发生变化,需要重复地访问结点的相邻边及边上的对象点,尤其是在聚类包含多个边时,遍历过程需要更新每个结点与当前聚类的距离.当网络中对象密集时,相邻边上对象的大量重复访问会产生很多不必要的代价.另外,该算法对于网络中的每个还未遍历到的对象都需要执行一次ε-link 算法进行网络的遍历,当对象数目大时,代价也很高.

基于结点的对象聚类本质上也是一种基于密度的聚类算法,但它利用了网络的结点信息和网络拓扑结构来进行扩展,避免了随意扩展带来的冗余距离计算,以及传统的基于密度算法中大量的范围查询.算法按照对象与结点之间距离的顺序遍历边,以解决结点相邻边以及边上对象的重复访问问题.

2.2.2 算法说明

基于结点的对象聚类首先从某个结点出发,把结点周围的对象聚类在一起(满足相邻对象之间的距离小于等于阈值ε的条件),并扩展到相邻结点,将相邻结点周围满足条件的对象也包含在该聚类中,直到聚类不能再扩展为止;然后,对没有遍历过的结点循环这个过程,直到结点周围的对象都已指定给某个聚类为止;最后对网络边上未形成聚类的对象,根据对象的聚类条件来判断是否能单独形成聚类.下面对算法的主要部分,即一个结点周围对象的聚类形成及扩展过程进行详细说明.

基于某个结点的聚类过程可以分为两步:初始化阶段和扩展阶段.初始化阶段先判断每个相邻边上与结点相邻的对象与该结点距离是否小于等于阈值ε,以此来过滤掉不必遍历的边,然后对满足条件的相邻边按照对象与结点的距离排序,将结点最相邻的对象作为初始聚类;在扩展阶段按顺序依次遍历相邻边上的对象来扩展初始聚类,继续扩展到相邻结点,再对相邻结点周围的相邻边进行排序,按顺序遍历相邻边上的对象,直到相邻对象之间的距离大于阈值ε,聚类不能再扩展为止.算法的伪码如下:

算法 **2**. 基于结点的聚类算法,找某个结点周围的聚类 *Find-cluster*(n_i).

输入:基于一个网络的对象集合,结点 n_i,距离阈值ε参数.

输出:结点 n_i 周围对象的聚类结果.

Q:优先队列,每一项 *B* 为$\langle node_1,node_2,dist\rangle$.表示边($node_1$,$node_2$)上与 $node_1$ 相邻的对象距离为 *dist*.

Ndist[n_i]:存储每个结点相邻边上离结点最近的对象的距离,初始值为无穷大.

Begin

1. **for** n_i 的每个相邻结点 n_z **do** /*初始化阶段*/
2. *o* 为边(n_i,n_z)的第 1 个对象;
3. **if** (*o*<>null **AND** *Dd*(*o.pos*,n_i)≤ε) **OR** (*o*==null **AND** *W*(n_i,n_z)≤ε) **then**
4. 按照 *o* 或 n_z 与 n_i 的距离顺序将$\langle n_i,n_z,Dd(o.pos,n_i)\rangle$或$\langle n_i,n_z,W(n_i,n_z)\rangle$插入到 *Q* 中;
5. **if** *Q* 不为空 **then** 创建一个新的聚类 *C*,分配 *cid*;
6. **while** *Q* 不为空 **do** /*扩展阶段*/
7. 取出 *Q* 的队首第 1 项 *B*;n_x=*B.node*$_1$;n_y=*B.node*$_2$;
8. **if** *B.dist*<*Ndist*[n_x] **then** *Ndist*[n_x]=*B.dist*;
9. *o* 为边(n_x,n_y)的第 1 个对象;
10. **if** *o*==null **then** *Ndist*[n_y]=*B.dist*; **Goto** 19;
11. **if** *Dd*(*o.pos*,n_x)+*Ndist*[n_x]≤ε **OR** *o* 为与 n_x 距离最近的对象 **then**
12. *o.cid*=*C.cid*;*clustered*(*o*)=true;

13. *nexto* 为边(n_x,n_y)从 *o* 到结点 n_y 的下一个对象;
14. **while** *nexto*<>null **AND** *Dd*(*o.pos*,*nexto.pos*)≤ε
15. *nexto.cid*=*C.cid*;*clustered*(*nexto*)=true;
16. *o*=*nexto*;*nexto* 为边(n_x,n_y)从 *o* 到结点 n_y 的下一个对象;
17. **if** *Dd*(*o.pos*,n_y)≤ε **then**
18. **if** *Dd*(*o.pos*,n_y)<*Ndist*[n_y] **then** *Ndist*[n_y]=*Dd*(*o.pos*,n_y);
19. *visited*(n_y)=true;
20. **for** n_y 的每个相邻结点 n_z(除了结点 n_x) **do**
21. *o* 为边(n_y,n_z)的第 1 个对象;
22. **if** (*o*<>null **AND** *Dd*(*o.pos*,n_y)≤ε) OR (*o*==null **AND** *Ndist*[n_y]+*W*(n_y,n_z)≤ε) **then**
23. 按照 *o* 或 n_z 与 n_y 的距离顺序将⟨n_y,n_z,*Dd*(*o.pos*,n_y)⟩或⟨n_y,n_z,*Ndist*[n_y]+*W*(n_y,n_z)⟩插入到 *Q* 的队首;

End

Find-cluster()算法需要一个优先队列 *Q* 顺序保存结点周围需遍历的相邻边以及边上与结点相邻的对象距离.*Q* 中按结点分组排序其周围的边,扩展遍历的结点插入到 *Q* 的队首.数组 *Ndist*[n_i]保存与结点 n_i 最近的对象的距离,用于判断结点周围除最近的对象所在边之外其他相邻边上的对象是否需要遍历,并加入到由该结点最近对象构造的初始聚类中(第 11 行).算法也处理了某个相邻边上没有对象,但边很短且跨越该边的两个对象之间的网络距离仍小于ε的特殊情况(第 3,4,10,22,23 行),这种情况也需要将对象加入到初始聚类中并继续扩展.

结点 *J*1 的聚类形成及扩展过程如图 1 所示,假设阈值ε为 10,首先找到与结点 *J*1 相邻的对象 *p*1,*p*2,*p*9, *p*10,由于它们与 *J*1 的距离都小于 10,因此按照它们与 *J*1 的距离排序,将这些对象所在的边和距离按顺序插入到优先队列 *Q* 中,即(⟨*J*1,*J*2,1⟩,⟨*J*1,*J*6,2⟩,⟨*J*1,*J*7,3⟩,⟨*J*1,*J*5,5⟩);然后,取出 *Q* 队首的边(*J*1,*J*2),把边上的第 1 个对象 *p*2(即离 *J*1 最近的对象)作为初始聚类 *C*1,其到 *J*1 的距离 1 存入 *Ndist*[*J*1]中,再遍历该边的下一个对象 *p*3,与 *p*2 的距离小于 10,因此将 *p*3 加入到 *C*1 中.由于 *p*3 与结点 *J*2 的距离小于阈值 10,再扩展到结点 *J*2 并将与 *J*2 的距离存入 *Ndist*[*J*2].同理,排序结点 *J*2 周围的其他相邻边,插入到 *Q* 队首之后得到队列 *Q* 为(⟨*J*2,*J*8,2⟩,⟨*J*2,*J*3,4⟩,⟨*J*1,*J*6,2⟩,⟨*J*1,*J*7,3⟩,⟨*J*1,*J*5,5⟩),接着取出 *Q* 队首的边(*J*2,*J*8)进行遍历,由于边上第 1 个对象 *p*7 与 *Ndist*[*J*2]的距离之和为 4,小于 10,则将 *p*7 加入到 *C*1 中,并继续遍历下一个对象 *p*13,与 *p*7 的距离也小于 10,因此将 *p*13 也加入到 *C*1 中.由于 *p*14 与 *p*13 的距离为 12,大于 10,所以停止边(*J*2,*J*8)的遍历.取出 *Q* 队首的边(*J*2,*J*3)继续遍历,直到相邻结点 *J*3.同理,处理结点 *J*3.把边(*J*3,*J*8)和(*J*3,*J*4)插入 *Q* 队首,再按顺序分别取出后遍历边上的对象,将相邻距离小于 10 的对象都归入到 *C*1 中,直到沿边(*J*1,*J*2)的方向不能再扩展为止.此时,队列 *Q* 为(⟨*J*1,*J*6,2⟩,⟨*J*1,*J*7,3⟩,⟨*J*1,*J*5,5⟩).用同样的方法取出 *Q* 队首的边(*J*1,*J*6),由于 *p*1 与 *Ndist*[*J*1]的距离之和为 3,小于 10,因此继续遍历该边(*J*1,*J*6)的下一个对象加入到 *C*1 中,沿着该边的方向扩展.重复此过程,直到队列 *Q* 为空为止,即结点 *J*1 周围的对象及相邻结点的对象都加入到 *C*1 中且不能再扩展为止.

上述两个算法中引入的ε参数是一个用户定义的阈值,其取值决定了最后的聚类结果(聚类的大小和数目).与文献[7]中的 DBSCAN 算法及文献[13]中的ε-link 方法类似,ε参数的取值可以根据实际应用,由用户经验或在道路网络边上进行取样而得到.ε阈值可以控制聚类内对象之间的凝聚性以及聚类的独立性.如果网络中的任意两个相邻对象之间的网络距离小于ε,并且它们被指定在某个聚类内,那么这两个对象必定属于同一个聚类.

3 性能分析与实验

3.1 算法分析

基于边和基于结点的聚类算法的代价都线性于对象数目 *N*,算法需要遍历对象所在的网络边,其最差情况下(所有的网络边上都有对象,聚类结果跨越了整个网络)需要遍历整个网络图,因此,算法的时间复杂度为 *O*(|*V*|log|*V*|+*N*),其中,*V* 为网络中结点的个数,而 *N* 为网络中对象的个数.

分析基于边的聚类方法,其高效性主要表现在以下 3 点:

1) 由于算法从某个合适的层次分组对象进行聚类的初始化,使得仅需少数的分裂和合并就能得到聚类的结果,从而减少了将每个对象作为聚类的代价高的初始化以及中间合并过程.先分裂后合并的方法也避免了传统的层次算法合并后难以调整结果的聚类准确度问题.

2) 通过引入参数ε阈值,不仅控制了聚类内对象之间的相似度,而且控制了聚类之间的相似度,使得能够在聚类的合并过程中尽早地发现合适的聚类结果;聚类的ε有效结点链表使得算法在合并聚类时提前过滤掉了不必合并的聚类,减少了聚类之间合并的次数,提高了聚类合并的效率.

3) 利用网络中聚类的特征,引入聚类块的概念来简化网络中聚类的表示和计算,并且在合并时只对结点相邻的聚类(块)合并,减少了大量不同边上对象之间网络距离的计算,提高了聚类算法性能.

分析基于结点的聚类方法,其高效性主要表现为以下 3 点:

1) 算法通过一种图遍历方法在遍历网络中对象的同时对其进行聚类,只需计算同一边上相邻对象或者与结点相邻的对象之间的网络距离(利用结点来计算距离),避免了任意两个对象之间网络距离的重复计算.

2) 遍历时不是从任意对象开始,而是从结点开始,按照相邻对象与结点之间距离的顺序依次向周围边扩展.遍历到相邻结点时,也是先对该结点相邻的对象排序再按照顺序遍历相邻对象所在的边.这样,每个相邻边只需访问一次,避免了相邻边及边上对象的重复访问,对于网络中对象密集的情况,将极大地提高聚类的效率.

3) 通过基于结点的网络扩展,算法避免了传统的基于密度的聚类算法中对每个对象必须执行一次范围查询的代价,同时,与基于边的算法相比,避免了聚类之间的合并过程.

对于道路网络中对象密集、对象数目多的情况,基于边的聚类算法先对每个边上的所有对象构成一个聚类,而不是将每个对象作为初始聚类,这可以使聚类初始化和合并的代价大为减小;另一方面,对象密集使得对象之间的距离减小,聚类块中对象数目增多,但网络中聚类块本身数目并没有增加,聚类块的合并代价并没有增加.然而,文献[13]中的 single-link 方法的初始聚类数量会大量增加,合并代价也随之增加,因此,在网络中对象密集的情况下,基于边的聚类算法更高效.基于结点的方法从结点开始遍历网络,而不是从某个对象开始,同时,在遍历多个相邻边时,也是按照各个边的对象与结点之间距离的顺序进行,避免了文献[13]中的ε-link 方法重复访问相邻边及边上对象的过程,在边上的对象数目较多时,会减小更多的代价开销,因此更适用于道路网络中对象密集的情况.

3.2 实验评估

3.2.1 实验配置和数据集

实验的硬件环境配置为 Pentium(R)4 2.39GHz CPU 和 256MB 内存的 PC,软件环境为 Linux 平台,所有代码均用 C++实现.为了与 Yiu 等人在文献[13]中提出的空间网络聚类算法比较,我们使用与其相同的数据集.数据集包括两部分:道路网络和位于其中的对象集合.道路网络数据集分别使用 Oldenburg 城市和 San Francisco 城市的真实道路数据,并在此基础上对道路网络进行了变换,将其转换为图模型,即边和结点的表示形式.

道路网络上的对象数据集通过 Yiu 的对象数据生成器得到.这个生成器可以通过在一个网络中产生大量的对象来模拟真实世界的聚类形态,有利于比较聚类的效果.数据集的特点是对象的分布使得产生的聚类有一个密集中心,其边界的对象逐渐稀疏.在使用这个对象生成器时,必须给定以下几个参数:K,*ptNum* 和 *olratio*,其中:K 表示产生的聚类总数;*ptNum* 表示对象总数;*olratio* 表示孤立点的比率.对象生成器产生 $ptNum\times(1-olratio)$个对象,平均分布在 K 个密集中心周围(即每个聚类包含 $ptNum\times(1-olratio)/k$ 个对象).另外,$ptNum\times olratio$ 个对象作为孤立点随机分布在网络中.例如,设 K=10,*ptNum*=100000,*olratio*=1%,则产生 10 个聚类,每个聚类包含 9 900 个对象,其他 1 000 个对象随机分布.这样的数据集有利于我们直接评估聚类算法的效果.由于我们的模型将道路网络上的对象表示为边和相对位置,实验前还需要将每一个对象由(x,y)坐标的表示转换为$(edge,pos)$的形式.

3.2.2 实验内容和结果分析

为了评估聚类算法的有效性和性能,我们将基于边的算法(表示为 eb-cls)和基于结点的算法(表示为 nb-cls)分别与传统的 DBSCAN[7]算法和 Yiu 等人提出的ε-link(图中表示为 *e*-link)和 single-link 算法[13]进行比较(考虑

到他们提出的另一个 k-medoids 算法在聚类效果和效率上明显不如这两种方法,省略了与它的比较).首先,对于同一个道路网络,我们改变数据集的对象分布(聚类数目为 K)来观察并测量聚类的效果和聚类算法的时间消耗,再改变对象数目($ptNum$),使得网络中对象由稀疏逐渐密集,评估聚类算法的效率;然后,我们选择了一个更为复杂的道路网络来比较聚类算法随对象数变化的性能;最后,比较我们的算法中ε参数对聚类效果和效率的影响.

根据对象生成器的特点,我们统计出算法得到的聚类总数以及每个聚类中的对象总数,比较不同算法的聚类效果(注意:只统计对象数大于 $ptNum\times olratio$ 的聚类,其他的作为异常点).在实验中,对象总数 $ptNum$ 设为 100 000,$olratio$ 设为 1%,K 分别为 10,20,50,100,1 000,以产生不同数目的聚类.调整聚类算法的参数使算法能够找出数据集中预先产生的聚类结果.最后,统计每个算法得到的聚类数(记为 K')和每个聚类中的对象数($ObjNum_i$)并与数据集产生的聚类数和对象数作比较,计算出对象数平均绝对百分比误差(记为 MAPE(mean absolute percent error)).这个度量值可衡量聚类效果.

$$MAPE=\frac{1}{k'}\sum_{i=1}^{k'}\left|1-\frac{ObjNum_i}{ptNum\times(1-olratio)/k}\right| \tag{1}$$

实验统计结果见表 1.我们可以看出,eb-cls 和 nb-cls 算法可以较精确地找到数据集产生的聚类数,而且当产生的聚类数大于 10 时,聚类结果的平均相对误差均小于 DBSCAN,ε-link 和 single-link 算法,聚类效果更好.

Table 1 Comparison of clustering effectiveness

表 1 不同算法的聚类效果比较

K	Cluster effectiveness measure	DBSCAN	ε-link	single-link	eb-cls	nb-cls
10	K'	10	10	10	10	10
	$MAPE$	0.000 27	0.000 27	0.001 57	0.004 93	0.005 76
20	K'	18	18	18	19	19
	$MAPE$	0.111 56	0.111 56	0.107 90	0.047 84	0.045 54
50	K'	48	48	48	48	48
	$MAPE$	0.042 49	0.042 49	0.040 17	0.022 69	0.020 57
100	K'	95	95	95	98	98
	$MAPE$	0.053 85	0.053 85	0.052 66	0.014 50	0.013 31
1000	K'	958	957	961	952	956
	$MAPE$	0.056 75	0.055 78	0.050 42	0.008 02	0.002 62

我们调整算法的参数使得不同算法产生相近的聚类效果,然后比较聚类算法的时间效率.图 2 和图 3 分别显示了在不同聚类数目 K 的情况下,我们提出的聚类算法与 DBSCAN 以及与ε-link 和 single-link 算法聚类过程所花费的时间比较.从图中可以看出:eb-cls 和 nb-cls 算法效率大大高于 DBSCAN 算法,这是由于 DBSCAN 算法需要对每个对象执行一个范围查询,其时间代价很高;而与ε-link 和 single-link 算法相比,eb-cls 和 nb-cls 算法的聚类性能也提高了 1~3 倍.从图中还可以看出:在道路网络中,随着聚类数目的增加,eb-cls 和 nb-cls 算法具有较好的伸缩性.另外,由于 nb-cls 算法不需要聚类的合并过程,当聚类结果数目 K 较大时,其性能较好于 eb-cls 算法.

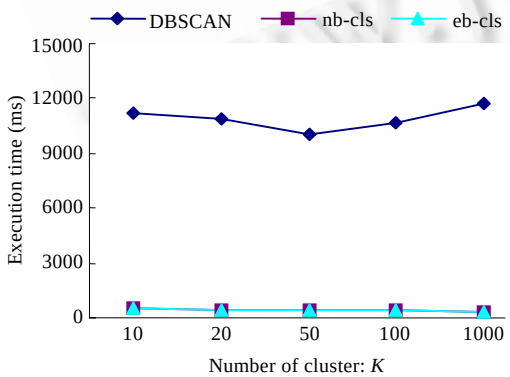

Fig.2 Comparison with DBSCAN in different K

图 2 不同 K 下与 DBSCAN 比较

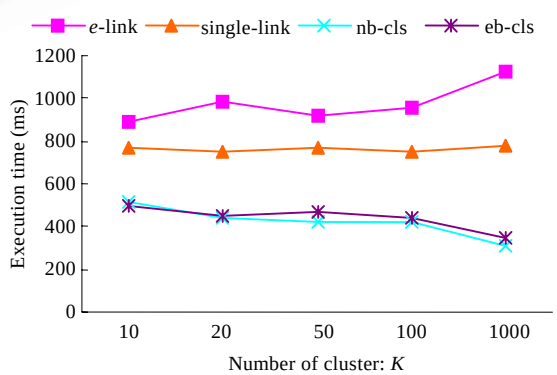

Fig.3 Comparison with ε-link, single-link in different K

图 3 不同 K 下与ε-link,single-link 比较

为了评估不同的聚类算法在网络中对象密度不同时的聚类效率,我们对 Oldenburg 城市道路网络(结点数为 6 105,边数为 7 035)增加对象的数目,使得网络中对象由稀疏变得密集,以评估不同算法的时间效率.聚类数 K 设为 50,图 4 和图 5 分别显示了对象数目 *ptNum* 分别为 20k,50k,100k,200k,500k,1000k 时,eb-cls 和 nb-cls 算法与 DBSCAN 以及ε-link 和 single-link 算法的性能比较.从图中可以看出:当聚类结果数目固定时,对不同的对象数,DBSCAN 以及ε-link 和 single-link 聚类算法的效率都低于 eb-cls 和 nb-cls 算法;而且随着对象数目的增大,DBSCAN 算法性能急剧恶化.而对于其他 4 种算法,single-link 聚类的时间花费随着对象数增加较快,而ε-link,eb-cls 和 nb-cls 算法的时间增加趋势较为缓慢.当网络对象密集时(例如实验中 1000k 个对象的情况),eb-cls 和 nb-cls 算法的效率明显高于其他方法.另外,nb-cls 算法由于不需要合并过程,其性能在对象数目增大时也略高于 eb-cls 算法.我们在另一个更复杂的道路网络 San Francisco(结点数为 174 956,边数为 223 001)中也产生不同数目的对象并进行相同的实验,评估不同聚类算法的时间效率,得到了相同的结论.实验结果如图 6 所示.另外,由于算法需要遍历网络图,当使用这个更复杂的道路网络时(网络的边和结点数增加),聚类代价增加.

在我们提出的算法中,ε参数是一个用户定义的阈值,其取值决定了聚类结果.在最后的实验中改变参数ε取值,测试其对 eb-cls 和 nb-cls 算法的聚类效率和聚类结果的影响.我们设置对象数 *ptNum* 为 100k,聚类数 K 为 20,使用 San Francisco 道路网络,其中边的平均长度为 1 422,最大边的长度为 86 069,参数ε分别取 50,100~500,图 7 显示了算法的时间消耗.可以看出,参数ε对我们的聚类算法的效率影响较小.表 2 显示了算法随着参数ε变化的聚类效果.不同ε得到的聚类结果精度不同,对于此实验环境,当ε取 200 时,算法得到较好的聚类结果.

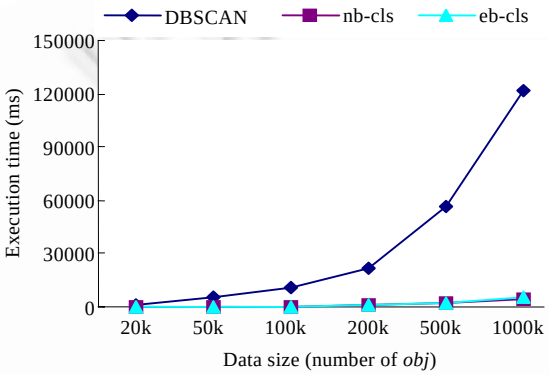

Fig.4 Comparison with DBSCAN in different #*obj*

图 4 不同对象数目下与 DBSCAN 比较

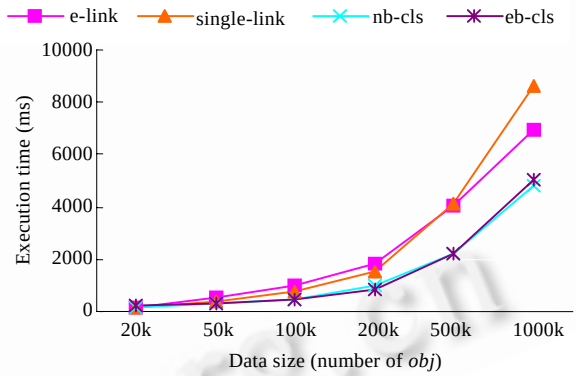

Fig.5 Comparison with ε-link, single-link in different #*obj*

图 5 不同对象数目下与ε-link 和 single-link 比较

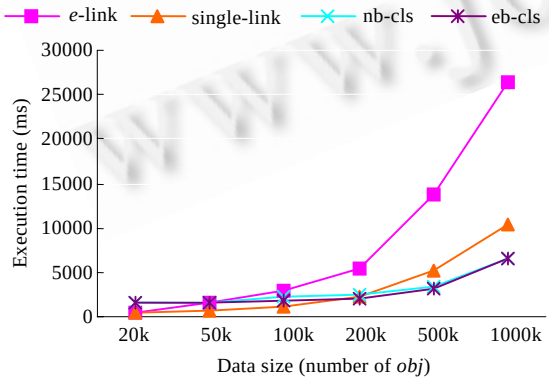

Fig.6 Comparison with ε-link, single-link in different #*obj*

图 6 不同对象数目下与ε-link 和 single-link 比较

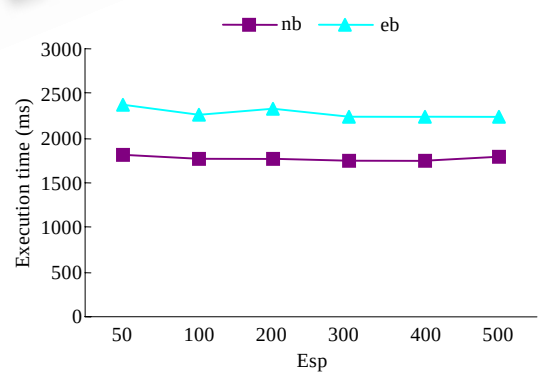

Fig.7 Effect of different ε on clustering efficiency

图 7 不同ε对聚类效率的影响

Table 2 Effect of different ε on clustering effectiveness
表 2 参数ε对聚类效果的影响

ε	Cluster effectiveness measure	eb-cls	nb-cls
50	K'	10	10
	MAPE	0.721 39	0.577 69
100	K'	19	19
	MAPE	0.346 49	0.510 70
200	K'	19	19
	MAPE	0.034 93	0.002 41
300	K'	19	19
	MAPE	0.047 85	0.045 55
400	K'	19	19
	MAPE	0.047 86	0.047 58
500	K'	19	19
	MAPE	0.047 84	0.047 67

通过上述比较可以看出:对于道路网络中的对象聚类,基于边和基于结点的聚类方法可以有效地找出对象聚类的结果;而且算法的效率较高、伸缩性良好,是两种较为高效、实用的聚类算法.

4 结 论

本文主要对道路网络中的对象聚类方法进行研究,根据网络距离重新定义道路网络中对象的聚类问题.我们开发出网络的特征,提出新的基于网络距离的对象聚类方法——基于边和基于结点的聚类方法.实验表明,新的聚类方法是可用且高效的.聚类道路网络中的对象可以应用于交通系统中对城市道路网络中车辆的堵塞情况进行检测.

References:

[1] Han JW, Kambr M. Data Mining Concepts and Techniques. 2nd ed., Beijing: China Machine Press, 2001. 335−395.

[2] Ng RT, Han J. Efficient and effective clustering methods for spatial data mining. In: Bocca JB, Jarke M, Zaniolo C, eds. Proc. of the 20th Int'l Conf. on Very Large Data Bases (VLDB). San Fransisco: Morgan Kaufmann Publishers, 1994. 144−155.

[3] Zhang T, Ramakrishnan R, Livny M. BIRCH: An effcient data clustering method for very large databases. In: Jagadish HV, Mumick IS, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 1996. 103−114.

[4] Guha S, Rastogi R, Shim K. CURE: An efficient clustering algorithm for large databases. In: Haas LM, Tiwary A, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 1998. 73−84.

[5] Nanopoulos A, Theodoridis Y, Manolopoulos Y. C2P: Clustering based on closest pairs. In: Apers PMG, Atzeni P, Ceri S, eds. Proc. of the 27th Int'l Conf. on Very Large Data Bases (VLDB). San Fransisco: Morgan Kaufmann Publishers, 2001. 331−340.

[6] Karypis G, Han EH, Kumar V. Chameleon: Hierarchical clustering using dynamic modeling. IEEE Computer, 1999,32(8):68−75.

[7] Martin E, 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 2nd Int'l Conf. on Knowledge Discovery and Data Mining (KDD). Portland: AAAI Press, 1996. 226−231.

[8] Ankerst M, Breunig M, Kriegel HP, Sander J. OPTICS: Ordering points to identify the clustering structure. In: Delis A, Faloutsos C, Ghandeharizadeh S, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 1999. 49−60.

[9] Agrawal R, Gehrke J, Gunopulos D, Raghavan P. Automatic subspace clustering of high dimensional data for data mining applications. In: Haas LM, Tiwary A, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 1998. 94−105.

[10] Ma S, Wang TJ, Tang SW, Yang DQ, Gao J. Dynamic reorganization of location databases based on clustering. Journal of Software, 2003,14(5):963−969 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/963.htm

[11] Zhang M, Gan J. Fuzzy partitional clustering algorithms. Journal of Software, 2004,15(6):858−868 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/858.htm

[12] Qiang WN, Zhou AY. Analyzing popular clustering algorithms from different viewpoints. Journal of Software, 2002,13(8):

1382−1394 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/13/1382.pdf

[13] Yiu ML, Mamoulis N. Clustering objects on a spatial network. In: Weikum G, König AC, Deβloch S, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 2004. 443−454.

[14] Shekhar S, Liu D. CCAM: A connectivity clustered access method for networks and network computations. IEEE Trans. on Knowledge and Data Engineering, 1997,19(1):102−119.

[15] Terrovitis M, Bakiras S, Papadias D, Mouratidis K. Constrained shortest path computation. In: Medeiros CB, Egenhofer MJ, Bertino E, eds. Proc. of the 9th Int'l Symp. on Spatial and Temporal Databases (SSTD). LNCS 3633, Heidelberg: Springer-Verlag, 2005. 181−199.

[16] Gupta S, Kopparty S, Ravishankar C. Roads, codes, and spatiotemporal queries. In: Deutsch A, ed. Proc. of the 23rd ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems (PODS). New York: ACM Press, 2004. 115−124.

[17] Papadias D, Zhang J, Mamoulis N, Tao Y. Query processing in spatial network databases. In: Freytag JC, Lockemann PC, Abiteboul S, Carey MJ, Selinger PG, Heuer A, eds. Proc. of the 29th Int'l Conf. on Very Large Data Bases (VLDB). San Fransisco: Morgan Kaufmann Publishers, 2003. 790−801.

[18] Yiu ML, Mamoulis N, Papadias D. Aggregate nearest neighbor queries in road networks. IEEE Trans. on Knowledge and Data Engineering, 2005,17(6):820−833.

[19] Yiu ML, Papadias D, Mamoulis N, Tao Y. Reverse nearest neighbors in large graphs. IEEE Trans. on Knowledge and Data Engineering, 2006,18(4):540−553.

[20] Tung AKH, Hou J, Han JW. Spatial clustering in the presence of obstacles. In: Proc. of the 17th Int'l Conf. on Data Engineering (ICDE). Heidelberg: IEEE Computer Society, 2001. 359−367.

附中文参考文献:

[10] 马帅,王腾蛟,唐世渭,杨冬青,高军.基于聚类的位置数据库动态重组.软件学报,2003,14(5):963−969. http://www.jos.org.cn/1000-9825/14/963.htm

[11] 张敏,于剑.基于划分的模糊聚类算法.软件学报,2004,15(6):858−868. http://www.jos.org.cn/1000-9825/15/858.htm

[12] 钱卫宁,周傲英.从多角度分析现有聚类算法.软件学报,2002,13(8):1382−1394. http://www.jos.org.cn/1000-9825/13/1382.pdf

陈继东(1978－),男,四川彭州人,博士生,主要研究领域为移动数据管理,数据库,数据挖掘.

赖彩凤(1981－),女,硕士生,主要研究领域为移动数据管理,数据库,数据挖掘.

孟小峰(1964－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据库与知识库系统,Web 数据管理,移动数据管理.