引用格式:张政,华一新,张亚军,等.以节点为中心的关系边聚类与可视化算法[J].地球信息科学学报,2020,22(9):1779-1788. [Zhang Z, Hua Y X, Zhang Y J, et al. Node-centered edge clustering and visualization algorithm[J]. Journal of Geo-information Science, 2020,22(9): 1779-1788.] **DOI:**10.12082/dqxxkx.2020.190568

以节点为中心的关系边聚类与可视化算法

张　政[1],华一新[1],张亚军[2],曾梦熊[1],杨振凯[1]

1. 信息工程大学地理空间信息学院,郑州 450052; 2. 苏州中科蓝迪软件技术有限公司,苏州 215000

Node-centered Edge Clustering and Visualization Algorithm

ZHANG Zheng[1*], HUA Yixin[1], ZHANG Yajun[2], ZENG Mengxiong[1], YANG Zhenkai[1]

1. Institute of Geographic Space Information, Information Engineering University, Zhengzhou 450052, China;

2. SuZhou Bluethink Software Technology Company Limited of Chinese Academy of Science, Suzhou 215000, China

Abstract: The visualization of the associative relation between objects is mainly expressed by the edges of the graph. But a large number of edges will cause serious visual confusion due to the complexity of the associative relation between objects. Graph layout and edge bundling are both effective methods to solve the problem of visual confusion caused by complex edges. While the geo-location of some nodes has significant meaning, only edge bundling methods can be used to reduce the map load and reveal the potential association rules of graph. In the past, the edge bundling algorithms adjusted the position of the middle control points of the edge under the condition that the two end nodes of the edge were fixed. This would cause a large number of edges being gathered together, which would not only cause a secondary visual confusion, but also be difficult to reveal the potential rules of the graph at the node level. To solve this problem, this paper proposed a node-centered edge clustering and visualization algorithm. Firstly, the direction clustering algorithm was used to realize the clustering of direction edges. The direction clustering method proposed in this paper was about 13 times faster than K-means algorithm, and about 6 times faster than DBSCAN algorithm. Then, the interpolation of the control points was implemented for each edge. On this basis, FR model was used to prevent the occurrence of "excessive bending". Finally, we adjusted the transparency of the edges so that the result of the visualization would be able to highlight the portion of the edge near the end nodes. The experimental results show that the value of the NCEB algorithm's map load (L) and the mid-point distance change ($\triangle d$) were about half of the FDEB algorithm, which proved that the NCEB algorithm can move the binding position from the middle part of the edge to the node, thus not only solving the secondary visual confusion caused by traditional edge bundling methods, but also revealing the association rule and trend of the graph at the node level. The final distribution trend of the edge around the node was clear and readable, and the visualization result greatly reduced the map load, which effectively reduced visual errors and misunderstanding of information. The results of our experiments show that the proposed algorithm can reveal the potential associated trend of graphs at the node level and greatly reduce visual confusion.

收稿日期:2019-10-01;**修回日期:**2020-03-28.

基金项目:国家重点研发计划项目(2016YFB0502300);国家自然科学基金项目(41471336)。 [**Foundation items:** National Key Research and Development Program of China, No.2016YFB0502300; National Natural Science Foundation of China, No.41471336.]

作者简介:张　政(1990—),男,河南郑州人,博士生,研究方向为地理信息可视化与地理信息服务。E-mail: giser_zzy@163.com

Key words: edge bundling; direction clustering; force-directed; visual confusion; curvature control; associative relation; relation visualization; pan spatial information system; map load

***Corresponding author:** ZHANG Zheng, E-mail: giser_zzy@163.com

摘要:对象间的关联关系可视化主要是通过图的连边进行表达的,但是对象间的关联关系纷繁复杂,大量的连边交错会造成严重的视觉混乱,图布局和边捆绑都是解决复杂连边造成的视觉混乱问题的有效途径,然而某些节点的地理位置具有实际的含义,只能通过边捆绑方法来减少幅面载负量,进而揭示图的潜在关联规律。以往的边捆绑算法是在边的两端节点固定的前提下,调整边的中间控制点的位置,这样会使得大量边被聚集在一起,不仅会造成二次视觉混乱,且难以在节点级别揭示图的关联趋势。针对这一问题,本文提出了一种以节点为中心的关系边聚类与可视化算法。首先使用方向聚类算法实现隶属于同一个节点的连边的聚类,本文提出的方向聚类方法速度约是K-means算法的13倍,约是DBSCAN算法的6倍,然后对各个连边实现控制点的内插,在此基础上使用FR模型使得控制点位移,并通过弯曲度控制防止"过度弯曲"情况的出现,最后调整边的透明度,使得可视化的结果突出显示边靠近端点处的部分。实验结果表明,本文NCEB算法的幅面载负量L和中点距离变化量$\triangle d$的约为FDEB算法的二分之一,证明本文算法可以将捆绑位置从边的中间部位移动到节点端,不仅解决了传统边捆绑算法造成的二次视觉混乱问题,而且使得节点周围的连边分布趋势清晰可读,且视觉负载大大降低,有效减少了视觉误差和信息误判。

关键词:边捆绑;方向聚类;力引导;视觉混乱;弯曲度控制;关联关系;关系可视化;全空间信息系统;地图载负量

1 引言

随着地理信息技术的发展,除了表达和描述对象本身之外,人们越来越重视对象之间的关联关系研究[1-2]。在对象间关联关系的可视化研究中,普遍采用图模型中的节点和边来表示对象及对象间的关联关系。然而,当图的结构较为复杂时,图中会充满交互交叉的线,从而造成视觉混乱[3-4]。

图可视化布局作为近年来可视化领域一项蓬勃发展的技术,经常被用于对象间关联关系的可视化[5]。图可视化布局技术主要通过移动节点的位置,尽量避免边的交叉,进而减少视觉混乱。经典的算法有FDA算法[6]、KK算法[7]和FR算法[8],这些算法基本上是在借鉴物理学原理基础上,兼顾美学原则,通过模拟退火等算法不断迭代调整节点的位置,进而实现图可视化布局。但是对于某一类数据,如航班数据、移民数据、交通路线数据等网状结构数据而言,节点的空间位置往往具有一定的含义,无法通过移动节点位置来实现可视化布局。对于这类数据的可视化,节点位置的重规划是不合适的,边的路径重规划(边捆绑)则成为另外一种可选方案。

对于图可视化的边捆绑问题,早期有Holten[9]提出的用于处理具有层次结构关系的对象间的边捆绑算法(Hierarchical Edge Bundles,HEB),该算法根据对象间的层次关系规划路径,使用B样条曲线拟合,进而实现边捆绑。Cui等[10]提出的基于几何的边捆绑算法(Geometry-Based Edge Bundling,GBEB),是通过生成控制网格来引导路径的重规划,该算法的捆绑效率较高,但是捆绑后图中会存在大量弯曲曲线,较难观察出边的走向。Holten等[11]在2009年提出了基于力引导的边捆绑算法(Force-Directed Edge Bundling,FDEB),该算法通过控制点将边分解为多段,分别计算控制点所受到的库伦斥力和胡克引力,多次迭代中不断调整控制点的位置,进而实现边捆绑,该算法通过计算边之间的角度、长度、节点距离等指标来判断节点间的相容性。Ersoy等[12]提出的基于骨架线的边捆绑算法(Skeleton-based Edge Bundling,SBEB),该算法利用具有位置相似性的边的中轴线作为骨架线,结合边聚类、距离场、二维骨架化,并通过迭代渐进式地实现边捆绑。为了能够在减少视觉混乱的同时,兼顾边的重要性差异,路强等[13]提出了基于内容重要性的边捆绑算法(Content Importance Based Edge Bundling,CIBEB),该算法通过关联边提取与关联度估计算法提取出高等级结构的边簇信息,然后在改进力导引算法的基础上使得较重要的边捆绑到独立的边簇中。Zwan等[14]提出了CUBu框架来解决大规模边捆绑的问题,百万量级的连边捆绑速度比同类方法提高了近50倍。Zielasko等[15]在FDEB算法的基础上,提出了面向三维交互场景的边捆绑算法。

上述边捆绑算法基本上都是基于一定的美学或科学标准,采用不同的算法将具有相似性的边进行聚类,通过边的路径重规划进而形成图的骨架线

结构，以达到减少视觉混乱和揭示潜在规律的目的。由于边捆绑算法的思路是固定两端节点，通过算法调整边的中间控制点位置，捆绑的结果会使得大量连边在中间部位聚集，这些聚集可能会覆盖部分节点，造成节点周围连边的二次视觉混乱问题。针对这一问题，本文提出了以节点为中心的边捆绑算法（Node-centered Edge Bundling，NCEB），通过连边方向聚类、边捆绑、渲染处理等过程，突出显示边在节点附近的部分，有效地解决了传统边捆绑算法造成的二次视觉混乱问题，使得节点周围的连边分布情况清晰可读，聚类簇中的主方向一定程度上也代表了节点在该方向上的连边分布趋势。

2　研究方法

为了能够在节点级别揭示关联趋势，减少由于连边捆绑而造成的二次视觉混乱问题，本文设计了NCEB算法的基本流程（图1），其基本思想是将对边的捆绑点从中间部位转移到节点两端，主要分为3个部分：边方向聚类、边捆绑和渲染处理。

在进行边聚类时，大多数算法都是采用类似FDEB中的方法，将具有相似方向、相似尺度、相近位置等的连边划归至同一个聚类簇，并不考虑簇中的连边是否属于同一个节点，这种方法虽然可以有效地将大量具有相似性的连边聚集在一起，但连边捆绑后的结果无法突出节点部分，甚至会由于大量的连边捆绑而造成部分节点被遮挡。为了能够避免这一问题，本文的边聚类算法则以节点为中心，首先按照节点重新组织边的集合，将具有同一源节点或端节点的连边组织在一个集合中，再按照边的方向角对连边进行排序，然后以连边之间的夹角作为输入参数，通过聚类算法实现连边的方向聚类。

边方向聚类完成后，需要对聚类边进行路径重规划。首先，循环迭代对边进行控制点内插，每次迭代完成后分别对内插点应用FR模型（胡克弹力、库伦引力模型），直至满足循环退出条件后为止。由于各个边的长度和位置不一，所以在进行控制点内插时，需要对内插点的位置做一定的限制，以防止出现过度弯曲曲线。

为了能够更加清晰地在节点级别显示其聚类特性，本文NCEB算法通过透明度计算模型对于连边进行渲染处理，着重显示边的头部或尾部，弱化边的中间部分，进而解决由于边捆绑算法造成的二次视觉混乱问题。

2.1　边方向聚类

边聚类是指将具有相似性的连边划归至同一聚类簇中的过程，目前常用的方法是判断连边的相容性[11]，即通过连边的夹角、连边的长度、节点的位置等方面判断连边是否属于同一聚类簇。但是本文的NCEB算法是对隶属于同一节点的连边进行聚类，而非对全局范围内的连边进行聚类，通过判断连边相容性的方法可能会导致聚类簇中的连边个数过少，从而影响边捆绑后可视化的最终效果。

针对这一问题，本文使用边方向聚类来解决隶属于同一节点的连边聚类问题。边方向聚类的实质是对连边的方向角进行处理，使得方向角差异较小的连边划归至同一聚类簇中。现有的聚类算法[16]并不能直接适用于边方向聚类，即使对传统聚类算法进行改进，也会由于算法的复杂性较高而导致大量连边的方向聚类难以处理。本文提出了一种基于角度邻域的边方向聚类方法，能够以较小的计算代价处理大量连边的方向聚类问题。

2.1.1　聚类思想

基于角度邻域的边方向聚类的基本思想是：设定连边的搜索角度邻域 ε_s 和限制角度邻域 ε_l，在隶属于同一节点的连边集合中，找到夹角最小的两条连边，将它们作为第一个聚类簇，然后不断向该聚

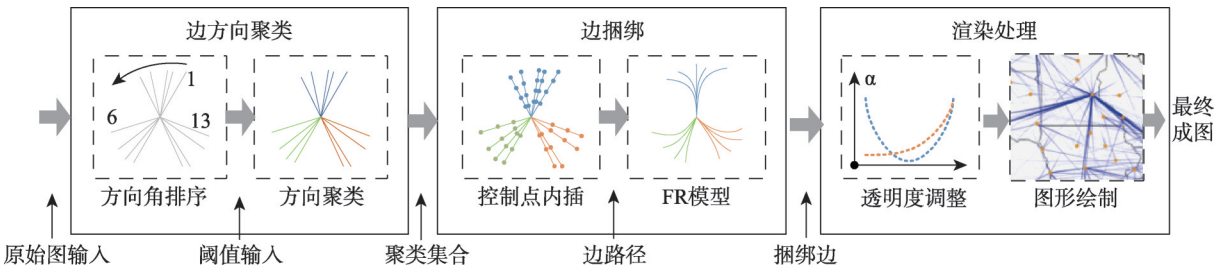


图1　NCEB算法的基本流程设计

Fig. 1　The basic procedure design of NCEB Algorithm

类簇中添加连边，如果该连边满足“与聚类簇中的连边最小夹角小于搜索角度邻域 ε_s，且与聚类簇中的连边最大夹角小于限制角度邻域 ε_l”的聚类条件，则该连边属于该聚类簇，否则，以该连边作为新的聚类簇。如图2所示为基于角度邻域的边方向聚类原理示意图，其中，连边 oa 和 ob 的夹角最小，所以将它们作为第一个聚类簇 c_1，然后按照连边方向角顺序添加连边 oc，由于 oc 满足聚类条件的要求，所以将它划归至聚类簇 c_1，按照同样的方法将所有的连边划分完毕，最终形成3个聚类簇 c_1、c_2、c_3。

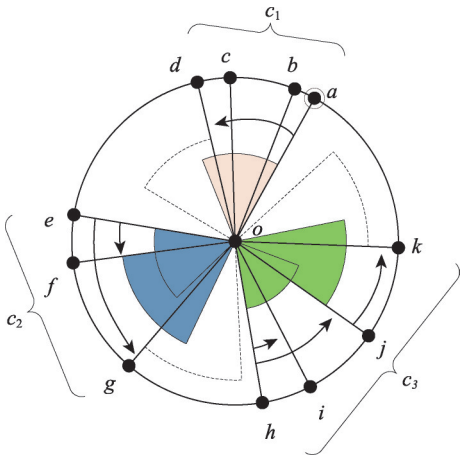


图2 基于角度邻域的边方向聚类原理

Fig. 2 The principle of angular neighborhood based edge clustering

2.1.2 聚类步骤

基于角度邻域的边方向聚类具体步骤为：

(1)将边按照节点 $N=\{n_1, n_2, \cdots, n_m\}$ 进行组织，具有同一个源节点或端节点的边组织在一个集合中，对于某个节点 i，得到连边集合 $E_i=\{e_1, e_2, \cdots, e_n\}$；

(2)对于隶属于同一个节点 i 的连边集合 E_i，分别计算每个连边的方向角 α，计算方法如式(1)所示。

$$\alpha=\arctan\left(\frac{y_{\text{target}}-y_{\text{source}}}{x_{\text{target}}-x_{\text{source}}}\right)\cdot\frac{180^\circ}{\pi} \quad (1)$$

式中：$(x_{\text{source}}, y_{\text{source}})$ 为源节点的坐标，$(x_{\text{target}}, y_{\text{target}})$ 为端节点的坐标。为了便于处理，将角度 α 的值域范围控制在 $[0°, 360°)$。

(3)连边集合 E_i 中各个边的方向角计算完成后，将隶属于该节点 i 的边按照方向角大小进行排序；

(4)按照顺序计算集合中前后两两连边的夹角，设连边 e_i 和 e_{i-1} 的夹角 $\alpha_{\min}$ 最小，若 $\alpha_{\min}\leqslant\varepsilon_l$，则将连边 e_i 和 e_{i-1} 划归为一个聚类簇 c_1；

(5)按照顺序选取下一个连边 e_{i+1}，计算其与聚类簇 c_1 中连边的最小夹角 $\alpha'_{\min}$ 和最大夹角 $\alpha'_{\max}$，若 $\alpha'_{\min}\leqslant\varepsilon_s$，且 $\alpha'_{\max}\leqslant\varepsilon_l$，则将连边 e_{i+1} 划归至聚类簇 c_1 中，否则，建立新的聚类簇 c_2，并将连边 e_{i+1} 划归至 c_2 中；

(6)循环步骤(2)—步骤(5)，直到集合 E_i 中的所有连边都被划归至相应聚类簇中；

(7)将节点集合 $N=\{n_1, n_2, \cdots, n_m\}$ 中的每个节点都循环执行上述过程。

2.1.3 算法对比

为了能够验证提出的基于角度邻域的边方向聚类算法的效率，本文与传统的K-means聚类算法和DBSCAN聚类算法进行了对比。但是传统聚类算法并不能直接适用于边方向的聚类，因此需要对其进行一定的改进，即将算法中的距离函数改为夹角计算函数，由于K-means算法和DBSCAN算法都相对比较成熟，限于篇幅限制，具体的算法步骤不再赘述。

本文选用美国的航班数据，其中有235个节点，2101条边，实验结果如表1所示。对于K-means算法，选择角度阈值 $\varepsilon=90°$，测试10组求平均值，平均每个节点的所属边集合的连边聚类数约为2.208，平均聚类时间约为15.167 s；DBSCAN算法中，选择角度邻域 $\varepsilon=15°$，$\min Pts=1$，测试10组求平均值，平均每个节点的所属边集合的连边聚类数约为2.979，平均聚类时间约为6.267 s；对于本文基于角度邻域的边方向聚类算法，选择搜索角度

表1 几种方向聚类算法的结果统计

Tab. 1 Statistics of the Result of the Some Direction Clustering Algorithms

算法名称	节点个数/个	边个数/个	平均聚类数/个	平均聚类时间/s
改进K-means的边聚类算法	235	2101	2.208	15.167
改进DBSCAN的边聚类算法			2.979	6.267
本文边聚类算法			3.132	1.087

$\varepsilon_s=30°$，限制角度邻域 $\varepsilon_l=90°$，测试10组求平均值，平均每个节点的所属边集合的连边聚类数约为3.132，平均聚类时间为1.087 s。可以看出，本文算法在聚类数目大致相同的条件下，聚类速度比起K-means算法和DBSCAN算法要快很多。这是因为，不论是K-means算法还是DBSCAN算法都属于无监督学习算法，往往需要通过多次迭代才可以得到聚类的结果，而且搜索的范围是全局范围内所有的连边，但本文提出的基于角度邻域的边方向聚类算法只需要在隶属于同一节点的连边范围内搜索和遍历即可，不需要进行迭代计算，因此其计算的时间复杂度要远远低于K-means算法和DBSCAN算法。

2.2 边捆绑

通过边方向聚类将隶属于同一节点且方向相近的连边划归至同一个聚类簇，可以采用路径重规划将这些方向相近的连边捆绑在一起，这样可以在保证连边整体方向趋势的前提下减少连边所占的幅面空间，进而达到减少视觉混乱的目的。本文借鉴节点布局的FR算法[8]对边的路径进行重规划，其基本过程为：首先，在连边的起始节点和终止节点之间内插控制点，然后使用力引导算法分别计算同一聚类簇中的每个连边的内插控制点所受到的胡克引力和库伦斥力，在合力作用下迭代计算控制点的新位置。

2.2.1 控制点内插

边捆绑算法要求不可以改变边的源节点和端节点的位置，因此，需要在源节点和端节点之间内插控制点，通过改变控制点的位置进而实现边路径的重规划。设节点增长率 ε，节点个数为 p，每循环一次节点个数变为 $\varepsilon\cdot p$，需要进行内插的边为 E，源节点为 P_0，端节点为 P_1，则内插控制点的基本过程是：

（1）初始化控制点个数 p，令 $p=1$，取 P_0 和 P_1 的中点作为第一个内插节点；

（2）执行力引导算法改变内插点的位置；

（3）根据式（2）计算由内插节点组成的内插线段的平均长度 $\bar{d}$；

$$\bar{d}=\frac{\sum_{i=1}^{n} distance(p_i,p_{-1})}{p+1} \tag{2}$$

（4）遍历内插线段，若内插线段的长度 $d_i>\bar{d}$，则根据式（3）计算新的内插节点的位置，并令 $d_i=d_i-\bar{d}$；若 $d_i\leqslant\bar{d}$，则令 $\bar{d}=\bar{d}-d_i$，并退出当前循环。

$$\begin{cases} x'=\dfrac{\bar{d}}{d_i}\cdot(x_i-x_{i-1}) \\ y'=\dfrac{\bar{d}}{d_i}\cdot(y_i-y_{i-1}) \end{cases} \tag{3}$$

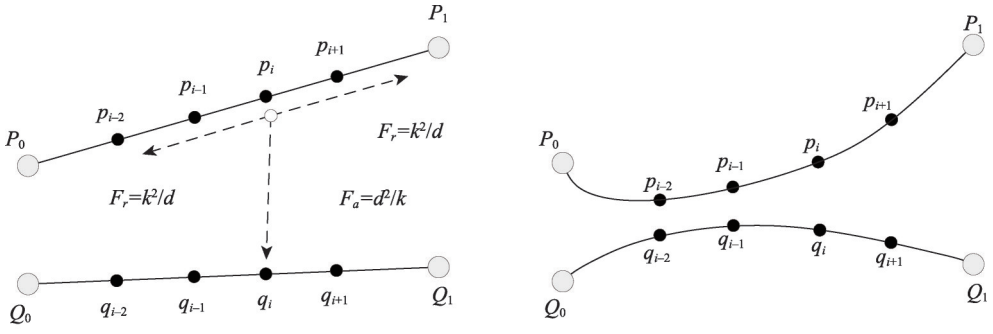
（5）重复步骤（2）至步骤（4），直到循环结束。

2.2.2 力引导算法

每次内插控制点完成后，需要利用力引导算法重新计算控制点的位置，其基本原理如图3所示：对于每一个控制点，其受到来自当前边上邻近控制点的斥力以及来自聚类边上对应节点的引力，其计算公式如下：

$$\begin{cases} F_a=\dfrac{d^2}{k} \\ F_r=\dfrac{k^2}{d} \end{cases} \tag{4}$$

式中：d 为节点间的距离，对于引力 F_a 而言，d 为当前边上 p_i 与相邻控制点的距离；对于斥力 F_r 而言，d 为 p_i 同聚类边上对应点 q_i 的距离。在引力



(a) 控制点受力分析　　(b) 控制点位移后的结果

图3　基于力引导的边捆绑算法原理

Fig. 3　The principle of force-directed edge bundling algorithm

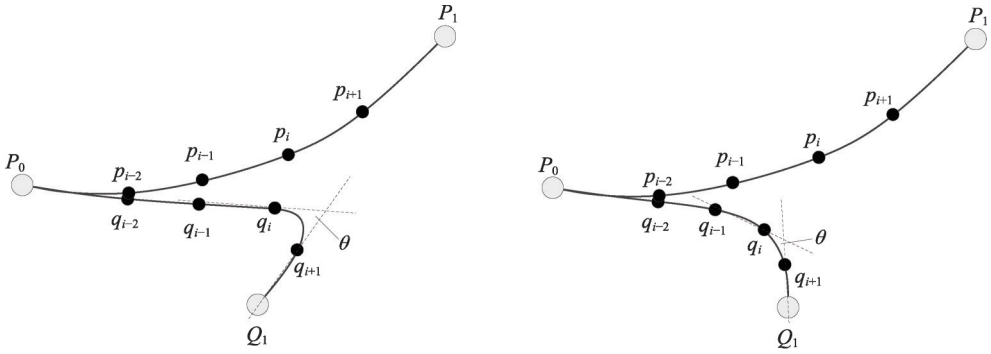
和斥力的作用下，通过迭代使得控制点位移，最终由图3(a)变为图3(b)。

2.2.3 弯曲度控制

FDEB算法中，会根据边的长度、方向、位置等对边的相容性进行判断，结果会使得具有相似长度、方向和位置的边被捆绑在一起，但是本文的算法并不对边的相容性进行判断，所以有可能出现“过度弯曲”现象，如图4(a)所示，q_i 切线与 q_{i+1} 切线的夹角 θ 过小，导致边弯曲过度，这样的结果不仅不会减轻视觉混乱反而会增加视觉负担，甚至造成信息误传。理想的结果是如图4(b)所示，边的弯曲较为平滑，美观的同时也减少了视觉混乱。

本文在计算控制点 q_i 的位置后，根据式(5)的余弦定理计算其切线与下一个控制点切线之间的夹角 θ，如果夹角 θ 的值小于阈值，则不再移动控制点 q_i。

$$\theta=\arccos\left(\frac{(x_{i-1}-x_i)\cdot(x_{i+2}-x_{i+1})+(y_{i-1}-y_i)\cdot(y_{i+2}-y_{i+1})}{\sqrt{(x_{i-1}-x_i)^2+(y_{i-1}-y_i)^2}\cdot\sqrt{(x_{i+2}-x_{i+1})^2+(y_{i+2}-y_{i+1})^2}}\right) \tag{5}$$



(a) 过度弯曲曲线　　(b) 正常弯曲曲线

图4 “过度弯曲”现象

Fig. 4 "Excessive bending" phenomenon

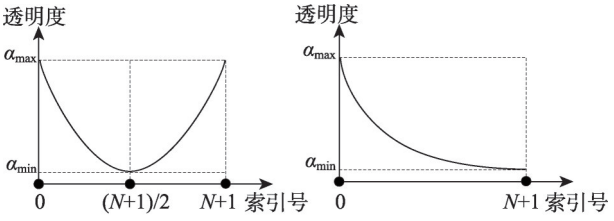
2.3 渲染处理

本文的NCEB算法目的是在节点级别揭示图的内在规律。相比于其他边捆绑算法而言，NCEB算法将边的捆绑位置从边的中间部分移动到边的两端，这样可以弱化在边的中间部分的捆绑效果，进而突出节点附近的边聚合情况。但是，实验证明仅仅依靠捆绑位置的偏移并不能有效地实现突出节点附近边捆绑情况的目的，因此，还需要结合透明度的调整算法对边进行渲染处理。

在第2.2节中的边捆绑算法实施之前，先对边进行了控制点内插处理，最终的结果是将仅有源节点和端节点的边内插为拥有 N 个控制点的边，这样，一条边就可以看作是由 N+1条零散线段组成的曲线。如果我们通过一定的算法控制，调节每条零散线段的透明度，就可以实现突出节点附近边捆绑情况的目的。对于透明度的调整主要有2种方案，一种是只突出源节点或端节点(Single Node，SN)，另一种是同时突出源节点和端节点(Double Node，DN)。

对于DN这种情况而言，我们希望其透明度的曲线图像如图5(a)所示，在第一条和最后一条曲线时，透明度达到最大值，在中间的曲线透明度逐渐减小至最小值。对于透明度曲线而言，并没有限定其具体的函数实现形式，只要保证其在中间控制点之前为减函数，而中间控制点之后为增函数即可。本文直接给出DN情况下的透明度计算公式，设 $\alpha_{\min}$ 和 $\alpha_{\max}$ 为透明度的最大值和最小值，N 为控制点的索引号的最大值，i 为当前控制点的索引号，则第 i 条零散线段的透明度计算公式为：

$$\alpha_i=(\alpha_{\max}-\alpha_{\min})\cdot\frac{\left[i-int\left(\frac{N+1}{2}\right)\right]^2}{int\left(\frac{N+1}{2}\right)^2}+\alpha_{\min} \tag{6}$$



(a) DN的透明度曲线　　(b) SN的透明度曲线

图5 DN和SN的透明度曲线

Fig. 5 The transparency curve of DN and SN

对于SN这种情况而言，我们希望其透明度的曲线图像如图5(b)所示，在第一条曲线透明度到达最大值，而在最后一条曲线时达到最小值，或者在第一条曲线透明度达到最小值，而在最后一条曲线时达到最大值，其计算公式为：

$$\alpha_i=(\alpha_{\max}-\alpha_{\min})\times\frac{[i-(N+1)]^2}{(N+1)^2}+\alpha_{\min} \tag{7}$$

3　实验及结果分析

3.1　实验数据来源

本文使用文献[11]中的美国航班数据以及移民数据来进行相关实验。FDEB算法的易用性、高效性，使得该算法得到了相当广泛的应用，有许多边聚类算法都是基于FDEB算法在不同应用场景下的改进，因此本文以经典的FDEB算法作为对比算法来说明NCEB算法的效果。使用JavaScript语言分别实现了NCEB算法和FDEB算法，并利用d3.js实现了矢量地图数据的读取与显示，使得边处理后的结果可以与地理底图叠加显示。

由于本文和FDEB算法都应用了力引导算法来实现边的捆绑计算，因此需要对力引导算法涉及的参数进行实验标定。主要的参数有：迭代次数 C；循环一次控制点增长的速率 ε 以及控制点个数 P；控制点在力引导算法的影响下移动的距离 S 及其初始距离 S_0；每次迭代计算的轮数 I 、以及初始轮数 I_0 以及轮数的减少率 ω。经过多次实验，本文最终确定的参数方案为 $C=6$，$\varepsilon=2$，$S_0=0.1$，$I_0=90$，$\omega=0.6667$，其迭代结果如表2所示。

表2　力引导算法计算方案

Tab. 2　Calculation scheme of force-directed algorithm

迭代次数	P	S	I
0	2	0.05	60
1	4	0.025	40
2	8	0.0125	26
3	16	0.00625	17
4	32	0.003125	11
5	64	0.0015625	7

在本文参数的设置下，该方案共执行了161轮计算，计算完成后，每条边包含64个控制点，实验的结果也表明在该参数设置下可以获得较好的实验效果，同时，64个控制点足够满足曲线绘制的需求。

3.2　对比实验

本文使用的美国航班数据[11]中共有节点235个，边2101条。对于美国航班数据而言，其节点数据和连边数据是分开组织的，首先需要以节点为中心，将边组织成相应的集合，即将源节点或端节点相同的边组织在一个集合中；然后，使用表2中的参数方案利用力引导算法对边数据进行路径重规划，但在规划过程中要判断边的弯曲程度，防止“过度弯曲”情况的出现；最后，通过透明度算法对各条边的透明度进行调整，使得节点附近被突出显示，最终的结果如图6所示。从结果中可以看出，就减少视觉混乱效果上来说，本文的NCEB算法比FDEB算法要有效的多，FDEB算法中存在大量的弯曲曲线，而由于本文的NCEB算法对曲线弯曲度进行了一定的约束和控制，不存在过度弯曲的曲线，一定程度上限制了连边的形变。除此之外，如图6(b)的红色虚线框标注位置，FDEB算法会造成在边的中间位置大量捆绑重叠，这些重叠区域覆盖了节点，造成节点周围的连边情况难以判别，不但没有有效地减轻视觉混乱，还容易引起节点周围连边信息的误判；而图6(c)的红色虚线框标注位置，本文的NCEB算法将边的捆绑从中间位置移到节点端，和FDEB算法相比，节点周围的连边主要分布方向趋势清晰可读，被突出显示的部分由隶属于同一节点的方向相近、分布密集的连边聚合在一起形成的，代表着该连边聚类簇中的主方向(平均方向角所指方向)，它意味着该节点沿此方向的连边较多且密集，一定程度上可以表示节点在该方向上的连边分布趋势，而FDEB算法则较难在节点一侧观察出连边的主要分布方向趋势，甚至会由于连边的捆绑而造成部分节点被遮挡。除此之外，通过对边的渲染处理，NCEB算法在重点显示了连边在节点周围的部分的同时，大大减轻了幅面载负量，减少了由于边捆绑而造成的二次视觉混乱问题。

就处理速度而言，FDEB算法处理美国航班数据的平均时间约为35 s，而NCEB算法处理美国航班数据的平均时间约为28 s，这主要是由于本文的NCEB算法会将边分为更多的簇，而在簇中的边进行路径重规划时边越多耗费的时间越长。设NCEB算法将所有的边分为 n 个簇，而每个簇中的边大致为 k 条，那么其需要进行力引导计算的次数为 nC_k^2；设FDEB算法将所有的边分为 m 个簇($m<n$)，则平均每个簇中的边大致为 nk/m 条，而

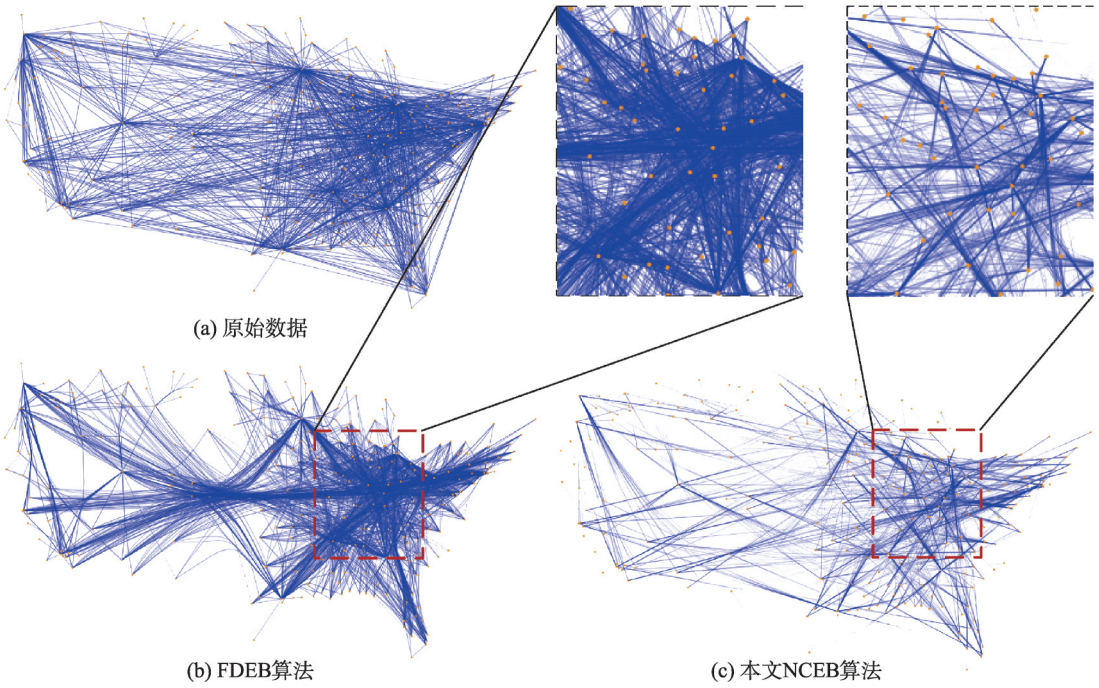


图6 2种算法对美国航班数据的处理结果

Fig. 6 Processing results of US airlines data by two algorithms

FDEB算法需要进行力引导计算的次数为 $C_{nk/m}^{2}$，FDEB算法和NCEB算法的力引导计算次数差值为 $(n-m)\cdot k^2/2m$，可以看出，这个值是大于0的，且 n 和 m 的差值越大，这个值越大，计算次数也就越多。

为了能够进一步说明本文NCEB算法相比于FDEB算法的优势，使用中国的航班数据[18]进行了实验，数据有160个节点，1769条连边，实验结果如图7所示，可以看出，FDEB算法对连边进行捆绑处理后相比于原始数据而言并没有有效地减少视觉混乱，大量的连边捆绑反而造成节点附近的连边情况难以分辨，而本文的NCEB算法有效地减轻了幅面载负量，突出显示了连边靠近节点附近的部分。FDEB算法基本失效的原因主要是由于机场分布情况造成的，连边主要呈近似垂直的交错分布状态，平行或相近的连边较少，FDEB算法判断的相容性连边较少，进而导致FDEB算法对连边的捆绑处理效果不佳，但是本文的NCEB算法并不受连边分布情况的局限，即使大量连边呈近似垂直的交错分布状态，也不会影响到最后的处理结果。

3.3 算法分析评述

为了能够定量地证明NCEB算法与FDEB算法的差异，本文从视觉载负量和边捆绑位置2个方面来进行评价。

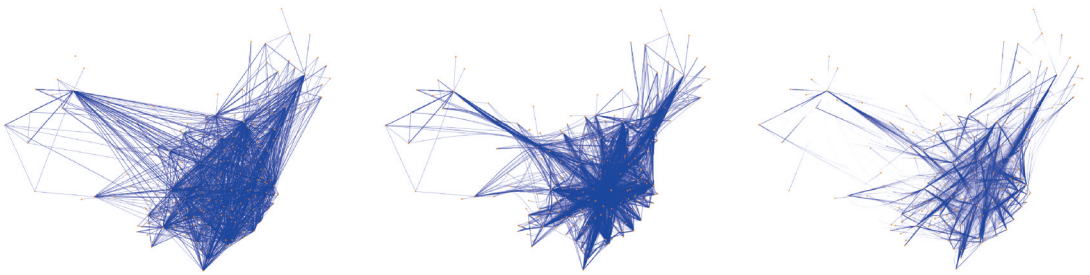


图7 2种算法对中国航班数据的处理结果

Fig. 7 Processing results of China airlines data by two algorithms

视觉载负量是通过计算图上要素符号所占面积与图幅总面积的比值，就是计算要素所占像素的个数与幅面的总像素个数之比[19]，其取值范围为[0.1]。但是，对于透明度不同的要素，载负量的值应有所区别，即需要考虑透明度叠加的影响。为了量化方便，本文将研究区域按照经纬度划分为 $m \times n$ 的格网，设索引号为第 (i,j) 个格网中，有 k 条边穿过该格网，且穿过该格网的线段的透明度为 α_s，则整个图幅范围内的视觉载负量计算公式为：

$$L=\frac{\sum_{i,j=0}^{m,n}\left[1-\prod_{s=0}^{k}(1-\alpha_s)\right]}{m\cdot n} \tag{8}$$

受文献[20]的启发，本文使用图中所有边与其具有相容性的边中点的平均欧氏距离之和在捆绑前后的变化量 Δd 来评价算法的边捆绑位置，如果变化量较小，说明边在中间位置捆绑的较少，如果变化量较大，说明边大量在中间位置捆绑。

本文利用美国航班数据以及移民数据[11]进行了算法评价实验，实验结果如表3所示，从表中可以看出，本文NCEB算法的视觉载负量明显小于FDEB算法，且 Δd 的值也远远小于FDEB算法，这是由于本文算法并没有在边的中间位置对相容性的边进行捆绑，而是在节点处对其进行捆绑，这样可以在节点级别显示出节点间的关联趋势，此外，不在边的中间部位进行捆绑同时也减少了信息误判的可能。

表3　算法评价实验的统计结果

Tab. 3　Statistical results of the algorithm evaluation experiment

数据	算法	L	Δd
航班	FDEB	0.32	0.33
	本文算法	0.18	0.12
移民	FDEB	0.57	0.37
	本文算法	0.29	0.14

4　结论与讨论

4.1　结论

现有的边捆绑算法主要是基于美学标准或科学标准，将具有一定相似性的连边聚合在一起，并通过边捆绑算法实现连边路径的重规划，进而形成图的骨架线结构，达到减少图的视觉载负量和揭示图的潜在规律的目的。然而，传统边捆绑算法的主要思想是固定两端节点，通过调整连边的中间控制点位置，这样会造成大量连边在中间部位聚集，有可能会覆盖部分节点，大大降低了节点周围的连边情况的易读性，进而造成二次视觉混乱。针对这一问题，本文提出了NCEB算法，该算法通过对隶属于同一个节点的连边进行方向聚类、边捆绑、渲染处理等，使得靠近节点的连边部分被突出显示，节点周围的连边分布趋势清晰可读。为了验证NCEB算法的可行性和有效性，本文分别采用美国航班数据、中国航班数据，与经典的FDEB算法进行了对比实验，从可视化结果可以看出，NCEB算法的幅面载负量远远小于FDEB算法，节点周围的连边情况也更加清晰。尤其是对于中国航班数据而言，由于大多数连边几乎呈垂直分布状态，导致FDEB算法的处理结果失效，而本文NCEB算法依然具有良好的数据适应性。由于本文NCEB对隶属于同一节点的连边进行了方向聚类，被突出显示的连边部分一定程度上可以表示节点在该方向上的连边分布趋势。为了能够量化这一结果，本文采用美国航班数据和移民数据，分别通过视觉载负量L和中点距离变化量 Δd 对算法的结果进行评估，从数据结果可以看出，本文NCEB算法的视觉载负量明显小于FDEB算法，且 Δd 的值也远远小于FDEB算法。

总结来讲，本文的主要研究成果为：

（1）提出的基于角度邻域的连边方向聚类方法，平均聚类速度约为K-means算法的13倍，约为DBSCAN算法的6倍，处理速度更快，平均聚类数更多；

（2）提出的NCEB算法与经典的FDEB算法进行实验对比，幅面载负量L和重点聚类变化量 Δd 约为FDEB算法的1/2，说明NCEB算法可以有效地较少幅面载负量，并使得连边的捆绑部位靠向节点一端。

4.2　讨论

利用本文提出的NCEB算法可以有效地解决传统边捆绑算法造成的二次视觉混乱问题，在节点级别揭示图的规律，但是也存在着以下几点需要努力的方向：

（1）三维场景中的边捆绑处理及可视化问题。目前，不论是本文的NCEB算法，还是传统的边捆绑算法，都是相对于二维可视场景而言的，但在三维可视场景中算法是否有效则需要进一步的实验和研究。

（2）海量数据条件下的连边快速捆绑问题。虽

然本文NCEB算法在整体速度和效率上都比传统的边捆绑算法有了较大的提升，但是海量数据条件下，算法的速度依然需要较大提升，后续研究可以将本文NCEB算法改进为并行计算方法，以提高算法对海量数据的处理能力。

参考文献(References):

[1] 华一新,周成虎.面向全空间信息系统的多粒度时空对象数据模型描述框架[J].地球信息科学学报,2017,19(9):1142-1149. [Hua Y X, Zhou C H. Description frame of data model of multi-granularity spatio-temporal object for Pan-spatial Information System[J]. Journal of Geo-information Science, 2017,19(9):1142-1149.]

[2] 张政,华一新,张晓楠,等.多粒度时空对象关联关系基本问题初探[J].地球信息科学学报,2017,19(9):1158-1163. [Zhang Z, Hua Y X, Zhang X N, et al. The basic issues of associative relationship of spatial-temporal objects of multi-granularity[J]. Journal of Geo-information Science, 2017,19(9):1158-1163.]

[3] Qu H M, Zhou H, Wu Y C. Controllable and progressive edge clustering for large networks[M]. Lecture Notes in Computer Science. Heidelberg: Springer, 2006,4372:399-404.

[4] Wong N, Carpendale S, Greenberg S. Edgelens: An interactive method for managing edge congestion in graphs [C]. Proceedings of the IEEE Symposium on Information Visualization. Los Alamitos: IEEE Computer Society Press, 2003:51-58.

[5] Battista G D, Eades P, Tamassia R, et al. Graph drawing: Algorithms for the visualization of graphs[M]. New Jersey: Prentice Hall PTR Upper Saddle River, 1998.

[6] Eades P. A heuristics for graph drawing[J]. Congressus Numerantium, 1984,42:149-160.

[7] Kamada T, Kawai S. An algorithm for drawing general undirected graphs[J]. Information Processing Letters, 1989, 31(1):7-15.

[8] Fruchterman T M J, Reingold E M. Graph drawing by force-directed placement[J]. Software: Practice and Experience, 1991,21(11):1129-1164.

[9] Holten D. Hierarchical edge bundles: visualization of adjacency relations in hierarchical data[J]. IEEE Transactions on Visualization and Computer Graphics, 2006,12 (5):741-748.

[10] Cui W W, Zhou H, Qu H M, et al. Geometry-based edge clustering for graph visualization[J]. IEEE Transactions on Visualization and Computer Graphics, 2008,14(6): 1277-1284.

[11] Holten D, Van Wijk J J. Force-directed edge bundling for graph visualization[J]. Computer Graphics Forum, 2009, 28(3):983-990.

[12] Ersoy O, Hurter C, Paulovich F, et al. Skeleton-based edge bundling for graph visualization[J]. IEEE Transactions on Visualization and Computer Graphics, 2011,17 (12):2364-2373.

[13] 路强,马坤乐.基于内容重要性边捆绑的图可视化算法[J].计算机辅助设计与图形学学报,2016,28(11):1899-1905. [Lu Q, Ma K L. Content importance based edge bundling for graph visualization[J]. Journal of Computer-Aided Design & Computer Graphics, 2016,28(11):1899-1905.].

[14] Zwan M V D, Codreanu V, Telea A. CUBu: Universal real-time bundling for large graphs[J]. IEEE Transactions on Visualization & Computer Graphics, 2016,22(12):1-1.

[15] Zielasko D, Weyers B, Hentschel B, et al. Poster: Interactive 3D force-directed edge bundling on clustered edges [J]. Computer Graphics Forum, 2016,35(3):51-60.

[16] 李志林,刘启亮,唐建波.尺度驱动的空间聚类理论[J].测绘学报,2017,46(10):1534-1548. [Li Z L, Liu Q L, Tang J B. Towards a scale-driven theory for spatial clustering [J]. Acta Geodaetica et Cartographica Sinica, 2017,46 (10):1534-1548.].

[17] 伍育红.聚类算法综述[J].计算机科学,2015,42(6A):491-499. [Wu Y H. General overview on clustering algorithm [J]. Computer Science, 2015,42(6A):491-499.].

[18] 牛健平. 全国航班数据与可视化[EB/OL]. https://cloud.tencent.com/developer/article/1357025, 2018-10-23. [Niu J P. National flight data and visualization[EB/OL]. https://cloud.tencent.com/developer/article/1357025, 2018-10-23.]

[19] 江南,曹亚妮,孙庆辉,等.双峰型基础电子地图载负量变化规律的探究[J].测绘学报,2014,43(3):306-313. [Jiang N, Cao Y N, Sun Q H, et al. Exploration of two-peak changing law of basic electronic map load[J]. 2014,43(3): 306-313.]

[20] Liu X T, Shen H W, Hu Y F. Supporting multifaceted viewing of word clouds with focus context display[J]. Information Visualization, 2014,14(2):168-180.