CP_SDD+RDS:基于分行排序单向检测求解最近对

姚华传 王丽珍 陈红梅 胡 新
(云南大学信息学院计算机科学与工程系 昆明 650091)

摘 要 求解最近点对问题在诸如地理信息查询、空间数据库等领域应用广泛。但到目前为止,还没有一种高效的求解算法,如传统求解最近对的分治算法存在比较次数较多、阈值收敛速度慢、计算距离次数较多的缺点。基于网格技术的求解最近邻方法存在网格的大小难以确定和算法效率低的问题。据此,首先提出基于单向检测的最近对求解算法(CP_SDD),然后提出按行划分的排序算法(RDS),最后得到基于分行排序单向检测的最近对求解算法(CP_SDD+RDS)。该算法不仅克服了分治法存在的缺点,而且子算法(RDS)的分行思想还克服了划分网格过程中存在的弊端。大量实验表明,CP_SDD+RDS算法是高效和可行的。

关键词 最近对,直角距离,模糊距离,点密度,点密集区
中图法分类号 TP311 **文献标识码** A

CP_SDD+RDS: An Algorithm Based on Row-divided Sorting and Single-direction Detecting for Finding Closest Pair of Points

YAO Hua-chuan WANG Li-zhen CHEN Hong-mei HU Xin
(Department of Computer Science and Engineering, School of Information Science and Engineering, Yunnan University, Kunming 650091, China)

Abstract Finding out the closest pair of points has been widely applied in many areas, such as the geographic information query system and spatial databases etc. But so far, there is not an efficient algorithm for the problem. For example, the divide and conquer algorithm contains much comparison, slow convergence and much computing of the distance between two points. Grid-based methods used to solve the nearest neighbor problem can not determine the size of the grid reasonably, and they are inefficient. For this reason, a single-direction detecting algorithm (CP_SDD) was presented in the paper. Then a row-divided sorting algorithm was proposed. Finally, an algorithm based on row-divided sorting and single-direction detecting for finding the closest pair of points (CP_SDD+RDS) was formed. Compared with the divide and conquer algorithm, our algorithm not only efficiently overcomes these drawbacks that occur in the divide and conquer algorithm, the row-divided strategy of the RDS algorithm is also effective to solve these problems that occur in grid-based methods. A large number of experiments show that the CP_SDD+RDS algorithm is efficient and feasible.

Keywords Closest pair of points, Right-angled distance, Fuzzy distance, Point density, Point-intensive areas

1 引言

求最近点对,亦即从点集中找出距离最小的点对,是计算机科学中的经典问题,在地理信息查询系统、设施定位、知识发现、网络查询以及空间数据库等多个领域得到广泛的应用。

Shamos 和 Hoey 于 1975 年提出了分治算法[1],随后人们对该算法进行了不同程度的优化,比如文献[2]优化了计算两点距离的次数,使计算次数下降了 1/3;文献[3]又在此基础上将计算次数下降了 1/6,共下降了 4/9。但这些方法并未从根本上解决问题,如:算法中找中位数及对数据区域进行划分所需的比较次数较多,阈值收敛的消息传播慢和计算两点距离的次数仍较多等。

与最近对搜索问题有近亲关系的最近邻搜索问题,同样受到人们的高度关注,如基于网格的连续最近邻查询算法 SEA-CNN 算法[4]、YPK-CNN 算法[5]、CPM 算法[6] 和 T-CPM算法[7]。这些都是采用一个规则的网格来索引数据进而处理欧氏空间的精确 NN 问题。其中 CPM 算法采用一种"概念划分"方式对网格进行划分,提高了查询效率,改进了存取方式和连续查询。T-CP 算法是在"概念划分"网格的基础上采用树形结构来索引概念划分网格的连续最近邻查询算法。文献[8]提出了基于空间填充曲线网格划分的最近邻查询算法,但网格的大小难以确定,过大或过小都会直接影响算法的效率,同时可能存在大量空格而浪费大量的内存空间。

到稿日期:2012-09-13 返修日期:2013-01-14 本文受国家基金项目(61063008,61272126,61262069),云南省应用基础研究基金项目(2010CD025),云南省教育厅基金项目(2012C103)资助。
姚华传(1977—),男,硕士生,主要研究方向为数据挖掘,E-mail:djm13232@126.com;王丽珍(1962—),女,博士,教授,博士生导师,主要研究方向为数据库、数据挖掘和计算机算法,E-mail:lizhwang2005@126.com(通信作者);陈红梅(1976—),女,博士,讲师,主要研究方向为数据挖掘;胡 新(1988—),男,硕士生,主要研究方向为数据挖掘。

此外，还有通过对偶邻近对[9]求解最近对的方法，如文献[10]提出了基于SR树多对象最近邻查询方法。文献[11]研究了反向量最近邻查询方法。文献[12]给出了双数据集中计算最近对的方法，每个数据集的数据由R树索引结构进行存储索引，提出了5种不同的算法。文献[13,14]从不同的方面基于不同的方法解决数据集中k个最近对问题。

本文从如下方面着手，提出了一种性能更优的最近对求解算法：(1)求两点的距离时，尽量用加法运算代替平方与开方运算，这样比较两点的距离时，更节省计算机的运算时间；(2)对点集按x坐标(或y坐标)进行排序，再以单向检测的策略搜索最近对，力求做到最大限度的剪枝；(3)提出一种按行划分的高效排序算法，进一步改进新算法的执行效率。

2 CP_SDD算法

2.1 相关定义及性质

本节先对两点的直角距离、绝对距离以及模糊距离进行定义，然后给出在比较两点距离时尽量用加法运算代替求距离运算所依据的性质及定理。

定义1 设S_D为平面区域D上的点集，a_i_x，a_i_y分别为点a_i的横坐标和纵坐标，$\forall a_i,a_j\in S_D$，称$|a_i_x-a_j_x|+|a_i_y-a_j_y|$为$a_i,a_j$两点的直角距离，记作$|a_i\sim a_j|$；称$\sqrt{(a_i_x-a_j_x)^2+(a_i_y-a_j_y)^2}$为$a_i,a_j$两点的绝对距离，记作$|a_ia_j|$；$\forall |a_i\cong a_j|\in\{|a_i\sim a_j|,|a_ia_j|\}$，称$|a_i\cong a_j|$为$a_i,a_j$两点的模糊距离。

性质1 设$a_i,a_j,a_k\in S_D$，如果①$|a_i_x-a_j_x|\geqslant|a_ia_k|$，或②$|a_i_y-a_j_y|\geqslant|a_ia_k|$，则$|a_ia_j|\geqslant|a_ia_k|$。

证明：①如图1所示，由于$|a_ia_j|\geqslant|a_i_x-a_j_x|$，又$|a_i_x-a_j_x|\geqslant|a_ia_k|$，得$|a_ia_j|\geqslant|a_ia_k|$；②的证明与①类似，证毕。

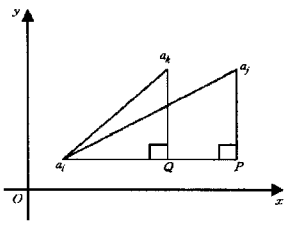

图1 相关定义及性质示例

定理1 设$a_i,a_j,a_k\in S_D$，如果$|a_i\sim a_j|<|a_ia_k|$，则$|a_ia_j|<|a_ia_k|$。

证明：因为$|a_ia_j|\leqslant|a_i\sim a_j|$，又$|a_i\sim a_j|<|a_ia_k|$，所以$|a_ia_j|<|a_ia_k|$，证毕。

定理2 设$a_i,a_j,a_k\in S_D$，如果①$|a_i\sim a_j|\geqslant\sqrt{2}|a_ia_k|$或②$|a_i\sim a_j|\geqslant\sqrt{2}|a_i\sim a_k|$或③$|a_i\sim a_j|\geqslant\sqrt{2}|a_i\cong a_k|$，则$|a_ia_j|\geqslant|a_ia_k|$。

证明：①由数学常识：$\forall a,b\in R$，都有$\sqrt{a^2+b^2}\geqslant\frac{a+b}{\sqrt{2}}$，得$|a_ia_j|\geqslant\frac{|a_i\sim a_j|}{\sqrt{2}}$，又$|a_i\sim a_j|\geqslant\sqrt{2}|a_ia_k|$，得$|a_ia_j|\geqslant|a_ia_k|$；②因为$|a_i\sim a_k|\geqslant|a_ia_k|$，又$|a_i\sim a_j|\geqslant\sqrt{2}|a_i\sim a_k|$，得$|a_i\sim a_j|\geqslant\sqrt{2}|a_ia_k|$，再由①的结论得$|a_ia_j|\geqslant|a_ia_k|$；③由①、②的结论得证，证毕。

2.2 基于单向检测求解最近对算法(Finding the Closest Pair of Points based on Single-Direction Detecting，简称CP_SDD)

CP_SDD的基本思想：

记全局变量min_d(初值为$|a_1a_2|$)为到目前为止找到的最近点对的距离。将平面区域D内的n个点按x坐标升序排列后得点列$a_1,a_2,\cdots,a_n$，对该点列中的每一点a_i依次执行如下操作：

首先考查$|a_i_x-a_{i+1}_x|$，如果$|a_i_x-a_{i+1}_x|<$min_d，则对min_d进行更新，即min_d=min{min_d，$|a_ia_{i+1}|$}，并继续以同样的方式考查$|a_i_x-a_{i+2}_x|$，如此进行下去，直到$|a_i_x-a_{i+s}_x|$(其中$|a_i_x-a_{i+s}_x|\geqslant$min_d或$i+s>n$)时终止。

此外，为了减少计算距离的次数，节省运算时间，上述过程所涉及的两点距离均使用模糊距离。

基于以上所述以及性质1和定理1、2，得到CP_SDD算法如下。

算法1 CP_SDD

输入：平面点集PS[n]

输出：最近点对

变量：max/min_x/y：n个点中x/y坐标的最大/小值；min_d：目前找到的最近点对的模糊距离；real_d：标识min_d是绝对距离还是直角距离

步骤：

1. if (max_x−min_x)>(max_y−min_y) then 按x坐标对n个点排序；//沿x轴方向搜索
   else
   按y坐标对n个点排序；//沿y轴方向搜索以下不妨设沿x轴方向搜索最近对，设按x坐标/排序后的点列为$a_1,a_2,\cdots,a_n$
2. min_d=$|a_1a_2|$；real_d=true；
3. for(i=1；i<n；i++)
4. for(j=i+1；$|a_i\_x-a_j\_x|$<min_d，j<n；j++)
   4.1. if ($|a_i\sim a_j|<\sqrt{2}$min_d) then //定理2
      4.1.1. if (real_d=true) then
         4.1.1.1. if ($|a_i\sim a_j|$<min_d) then //定理1
            min_d=$|a_i\sim a_j|$；real_d=false；
         4.1.1.2. else 求$|a_ia_j|$；min_d=min{min_d，$|a_ia_j|$}；
      4.1.2. else
         4.1.2.1. if ($\sqrt{2}|a_i\sim a_j|$<min_d) then //定理2
            min_d=$|a_i\sim a_j|$；
         4.1.2.2. else 置min_d为其代表的绝对距离；
            求$|a_ia_j|$；min_d=min{min_d，$|a_ia_j|$}；
            real_d=true；
5. return min_d对应的点对

例1 如图2所示，平面区域内有5个点a_1,a_2,a_3,a_4,a_5。求解最近点对的过程：先令min_d=$|a_1a_2|$，因为$|a_1_x-a_3_x|<$min_d，$|a_1\sim a_3|<\sqrt{2}$min_d，又min_d代表的是绝对距离，且$|a_1\sim a_3|<$min_d，故令min_d=$|a_1\sim a_3|$；而$|a_1_x-a_4_x|>$min_d，对a_1的操作终止。因为$|a_2_x-a_3_x|<$min_d，但$|a_2\sim a_3|>\sqrt{2}$min_d；接着因为$|a_2_x-a_4_x|>$min_d，对a_2的操作终止。如此进行，最后求得最近点对为(a_4,a_5)，距离为min_d=$|a_4a_5|$。

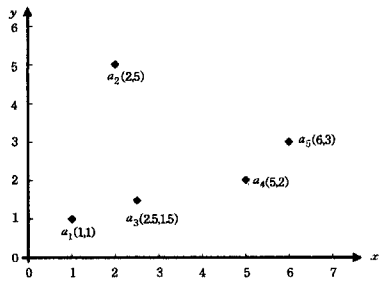

图 2　例 1 的图

2.3　CP_SDD 算法分析

当 CP_SDD 算法第 1 步对数据排序取用快速排序算法时，该步的平均时间复杂度为 $O(n\log_2 n)$；第 3、4 步的时间复杂度为 $cn=O(n)$，c 是与 min_d 相关的常数，代表某个 a_i 需要作距离比较的次数，故其时间复杂度为 $O(n)$。因此算法的平均时间复杂度为 $O(n\log_2 n)$。算法在最坏情况下的时间复杂度为 $O(n^2)$，当有序点列 $a_1,a_2,\cdots,a_n$ 中的任一点 a_i 到 $a_{i+1},\cdots,a_n$ 各点的距离都需比较时，算法呈最坏情形。此时，点 a_i 需做 $n-i$ 次的距离比较，故 n 个点共做了 $\sum_{i=1}^{n}(n-i)=\frac{n(n-1)}{2}$ 次的距离比较。

由于快速排序算法(QS)的空间复杂度为 $O(n)$，存储一对与 min_d 对应的点对所需空间为 $O(1)$，因此，CP_SDD 算法的空间复杂度为 $O(n)$。

大量实验表明，CP_SDD 算法消耗时间的主部来自对点列进行排序(约占算法总耗时的 80%)，因此，要提高算法的效率，应该从优化排序算法入手。下面提出一种按行划分的排序算法，称之为“分行排序算法”(Row-Divided Sorting)，简称为 RDS。

3　分行排序算法(RDS)

3.1　RDS 算法

基本思想：首先将点列所在的平面区域划分成宽度一致且与 x 轴垂直的若干行，然后将每一点根据其 x 坐标的大小放到相应的行上，这时点列关于 x 坐标大致有序，最后对各行中的点作精确排序，便可生成一个关于 x 坐标有序的点列。

但行宽的设定是该算法的关键和难点。行宽定得太大，每行集中的点太多；反之，大量的行连一个点都没有，算法均显低效。针对这个问题，下面提出解决办法。

3.2　行宽确定方法及相关定义、性质

确定行宽的步骤如下：

(1)找出一条垂直于 x 轴且尽可能穿过点分布较为集中区域的直线，定为 $x=\frac{1}{n}\sum_{i=1}^{n}a_i\_x$(记作 $x=\bar{x}$)；

(2)找出距离直线 $x=\bar{x}$ 最近的若干个点(不妨设为 k 个)，然后求出这 k 个点中 x 坐标的最大值 max_x_P 及最小值 min_x_P，行宽定为(max_x_P−min_x_P)/(k−1)；

(3)当行宽过大或过小时，步骤(1)和(2)改沿 y 坐标方向进行。

下面通过性质 2、3 说明选择 $x=\bar{x}$ 作为穿过点分布较为集中区域的一条直线的合理性。

定义 2　点密度是指在某平面区域 D 中，单位面积内所含的点数；如果平面区域 D 内有一点集，其中区域 $D_1,D_2,\cdots,D_k$ 具有相同的点密度，且其点密度远远大于区域 $D-(D_1\cup D_2\cup\cdots\cup D_k)$ 的点密度，这里 $D_i\subseteq D,D_i\cap D_j=\varnothing,i,j=1,2,\cdots,k,i\neq j$，则称 $D_1,D_2,\cdots,D_k$ 为平面区域 D 中的 k 个点密集区。

不失普遍性，以下设这样的点密集区均为圆形区域。

性质 2　如果某平面区域 D 内有 n 个点，D 中含有一个点密集区 A(如图 3 所示)，则直线 $x=\bar{x}$ 必定穿过点密集区 A。

证明：记 A 的圆心为 $A(a,b)$，半径为 R，面积为 S_A，并用 S_A 衡量区域 A 内的点数。假设圆 A 内有 m 个点，则这 m 个点的 x 坐标平均值为

$$\begin{aligned}\frac{1}{m}\sum_{i=1}^{m}a_i\_x&=\frac{1}{S_A}\int_{a-R}^{a+R}x\cdot 2\sqrt{R^2-(a-x)^2}\,\mathrm{d}x\\&=\frac{1}{S_A}\Big[\int_{a-R}^{a+R}\sqrt{R^2-(x-a)^2}\,\mathrm{d}(x-a)^2+2a\int_{a-R}^{a+R}\sqrt{R^2-(x-a)^2}\,\mathrm{d}(x-a)\Big]\\&=\frac{1}{S_A}\Big\{-\frac{2}{3}\big[R^2-(x-a)^2\big]^{\frac{3}{2}}\Big|_{a-R}^{a+R}+2a\int_{-R}^{R}\sqrt{R^2-u^2}\,\mathrm{d}u\Big\}\\&=\frac{1}{S_A}\Big\{-\frac{2}{3}\big[R^2-(x-a)^2\big]^{\frac{3}{2}}\Big|_{a-R}^{a+R}+2a\cdot\frac{1}{2}\Big(R^2\arcsin\frac{x}{R}+x\sqrt{R^2-x^2}\Big)\Big|_{-R}^{R}\Big\}\\&=\frac{1}{S_A}(0+\pi aR^2)=\frac{\pi aR^2}{\pi R^2}=a\end{aligned}$$

区域 A 外的离散点极少，对直线 $x=\bar{x}$ 的位置影响极其有限，因此

$$\begin{aligned}\bar{x}&=\frac{sum\_x}{n}=\frac{1}{n}\sum_{i=1}^{n}a_i\_x\\&=\frac{1}{n}\Big(\sum_{i=1}^{m}a_i\_x+\sum_{i=m+1}^{n}a_i\_x\Big)\\&\approx\frac{1}{S_A+(n-m)\cdot 0}\Big[\int_{a-R}^{a+R}x\cdot 2\sqrt{R^2-(x-a)^2}\,\mathrm{d}x+0\Big]\\&=a\end{aligned}$$

即直线 $x=\bar{x}$ 穿过点密集区 A，证毕。

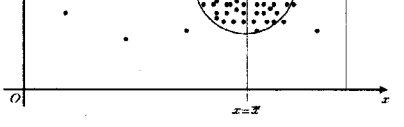

图 3　平面区域内含有一个点密集区

性质 3　如果某平面区域 D 内有 n 个点，D 中含有两个点密集区 A 和 B(如图 4 所示)，则只要 A 的面积相对于 B 的面积足够小，直线 $x=\bar{x}$ 必定会与 B 有公共点，即该直线会穿过点分布更集中的区域。

证明：记 A 的圆心为 $A(x_A,y_A)$，半径为 R_A，面积为 S_A，B 的圆心为 $B(x_B,y_B)$，半径为 R_B，面积为 S_B，$S_{总}=S_A+S_B$，不妨设 $0<x_A<x_B,R_A<R_B$。由性质 2 的证明过程可得 $\bar{x}=\frac{1}{n}\sum_{i=1}^{n}a_i\_x=\frac{1}{S_A+S_B}\Big[\int_{a-R_A}^{a+R_A}x\cdot 2\sqrt{R_A^2-(x-x_A)^2}\,\mathrm{d}x+\int_{a-R_B}^{a+R_B}x\cdot 2\sqrt{R_B^2-(x-x_B)^2}\,\mathrm{d}x\Big]=\frac{R_A^2x_A+R_B^2x_B}{R_A^2+R_B^2}$，原命题等价于

$|\bar{x}-x_B|\leqslant R_B$，而 $S_A+S_B=S_{总}$，$|\bar{x}-x_B|=\frac{R_A^2|x_A-x_B|}{R_A^2+R_B^2}$，要使 $|\bar{x}-x_B|\leqslant R_B$ 成立，即 $|x_A-x_B|\leqslant\frac{R_B(R_A^2+R_B^2)}{R_A^2}=S_{总}/(\frac{S_{总}}{R_B}-\pi R_B)$，只要 $\exists R_B$，使得 $|x_A-x_B|\leqslant S_{总}/(\frac{S_{总}}{R_B}-\pi R_B)$ 成立即可。由 $S_A+S_B=S_{总}$，得 $\pi R_A^2+\pi R_B^2=S_{总}$，$R_B=\sqrt{\frac{S_{总}}{\pi}-R_A^2}$，当 $R_A\to 0$ 时，$R_B=\sqrt{\frac{S_{总}}{\pi}-R_A^2}\to\sqrt{\frac{S_{总}}{\pi}}$，所以 $0<R_B<\sqrt{\frac{S_{总}}{\pi}}$。不等式 $|x_A-x_B|\leqslant S_{总}/(\frac{S_{总}}{R_B}-\pi R_B)$ 中，$|x_A-x_B|$，$S_{总}$ 均为常数，且 $\frac{S_{总}}{R_B}-\pi R_B>0$，所以 $S_{总}/(\frac{S_{总}}{R_B}-\pi R_B)$ 关于 R_B 单调递增，当 $R_B\to\sqrt{\frac{S_{总}}{\pi}}$ 时，有 $S_{总}/(\frac{S_{总}}{R_B}-\pi R_B)\to+\infty$，从而存在某个 R_B，使得 $|x_A-x_B|\leqslant S_{总}/(\frac{S_{总}}{R_B}-\pi R_B)$ 成立，证毕。

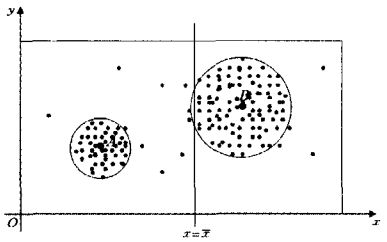

图 4　平面区域内含有两个点密集区

如果某平面区域 D 内含有 $N(N\geqslant 3)$ 个点密集区，则可将那些在 x 坐标方向上距离较近的点密集区看成一个较大的点密集区，从而尽可能地将这 N 个点密集区看成一个或两个较大的点密集区，如此即可使用性质 2 或性质 3。

3.3　分行排序算法(RDS)的实现

这一小节先给出 RDS 算法的实现及复杂性分析，然后用 RDS 算法对 CP_SDD 算法进行优化，得 CP_SDD+RDS 算法。

算法 2　RDS

输入：平面点集 PS[n]

输出：有序点列 $a_1,a_2,\cdots\cdots,a_n$

步骤：

1. 求 max_x，min_x 和 $\bar{x}$；
   //max/min _x 表示 n 个点中 x 坐标的最大/小值
   //$\bar{x}$ 表示 n 个点中 x 坐标的平均值
2. 找出距离直线 $x=\bar{x}$ 最近的若干点(不妨定为 k 个)CP[k]；
3. max_x_P=max{p_i_x|$p_i\in$CP[k],i=1,2,…,k}；
   min_x_P=min{p_i_x|$p_i\in$CP[k],i=1,2,…,k }；
4. 行宽 LW =(max_x_P−min_x_P)/(k−1)；
5. 行数 LC =(max_x−min_x)/LW+1；
6. 如果行数过大或过小，则返回 1，并改沿 y 坐标方向进行；
7. 新建一动态数组 DA[LC]，数组的元素为单链表，其中单链表 DA[j]的长度为第 j 行所含点数；
8. for(i =0;i<n;i++)
   将第 i 个点 PS[i]插入单链表 DA[(PS[i]_x−min_x)/LW]中；
9. 分别对数组 DA[LC]中各单链表内的点进行排序后便得到一个按 x 坐标有序排列的点列 $a_1,a_2,\cdots,a_n$；
10. return $a_1,a_2,\cdots,a_n$。

定理 3　RDS 算法的平均时间复杂度为 O(n)。

证明：算法第 1，2 步的时间复杂度均为 O(n)；第 3 至 7 步的时间复杂度均为 O(1)，其中 k 为常数，本文实验取 $k=6$ 至 24 不等；第 8 步的时间复杂度为 O(n)；第 9 步所需时间为 $\sum_{i=1}^{line\_count}L^2(i)$，其中 $L(i)$ 为第 i 个单链表 $DA[i]$ 内所含的点数，$L(i)\leqslant\mu$，$LC=\lambda n$，λ，μ 均为常量，λ 是动态数组 $DA[LC]$ 的长度与平面点集 $PS[n]$ 的点数 n 之比，μ 是单链表的最大长度，因此，$\sum_{i=1}^{line\_count}L^2(i)\leqslant\sum_{i=1}^{\lambda n}\mu^2=\lambda\mu^2 n=O(n)$；第 10 步的时间复杂度为 O(1)。综上所述，算法的平均时间复杂度为 O(n)，证毕。

RDS 算法在最坏情况下的时间复杂度为 O(n^2)，当 n 个点几乎全部分布在平面区域 D 内半径足够小、点密度一致的若干个点密集区中，且直线 $x=\bar{x}$(或 $y=\bar{y}$)距离近任一点密集区都足够远，则行宽难以确定，算法呈最坏情形。

RDS 算法需要一个动态数组 $DA[LC]$ 来保存中间结果，如定理 3 所述，$LC=\lambda n$，λ 为常量，数组 $DA[LC]$ 的空间容量小于 $LC+n=\lambda n+n=(\lambda+1)n=O(n)$，因此，算法的空间复杂度为 O($n$)。

至此，可用 RDS 算法对 CP_SDD 算法进行改进，最终得 CP_SDD+RDS 算法。CP_SDD+RDS 算法与 RDS 算法的空间复杂度相同；除 2.3 节所述的极端情况外，其时间复杂度也相同。同时 CP_SDD+RDS 算法对点集的搜索方向须与 RDS 算法对点集的具体排序方向一致。

4　实验评估

实验目的：评估与分治法(CP_DCM)相比，CP_SDD 算法在计算距离次数方面的剪枝效果及其总体性能；验证与快速排序算法(QS)相比，RDS 算法的高效性及其稳健性；验证与 CP_DCM 算法相比，CP_SDD 算法和 CP_SDD+RDS 算法的高效性和可行性。

实验环境：算法采用 C＃编写，在 AMD Athlon(tm) 64X2 Dual Core Processor 4200+ 2.2GHz，1G memory 的计算机上运行。实验数据随机产生，并分布在[0，1000]×[0，1000]的平面区域内，数据类型是 double。

4.1　CP_SDD 算法与 CP_DCM 算法的计算距离次数比较

从图 5 可以看出，在点数相同的情况下，CP_SDD 算法比 CP_ DCM 算法所做的计算距离次数更少，在这方面的剪枝效果明显。

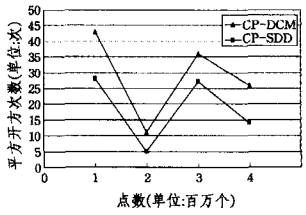

图 5　CP_SDD 算法与 CP_DCM 算法所做的计算距离次数比较

4.2　RDS 算法中取距离直线 $x=\bar{x}$ 最近点数的敏感度分析

RDS 算法中的 k 为找出的距离直线 $x=\bar{x}$ 最近的点数。从图 6 可以看出，k 越大，算法所耗时间越长，但增长缓慢。另一方面，分别对 2 百万和 4 百万个点进行 30 次试验，统计每种情况出现行宽不合理的次数(见图 7)，随着 k 的增大，行宽不合理的次数均呈下降趋势，有效地解决了由于点集呈扎堆分布所带来的行宽难以确定的问题。因此，须在算法的高效性及其稳健性之间取得一个平衡。

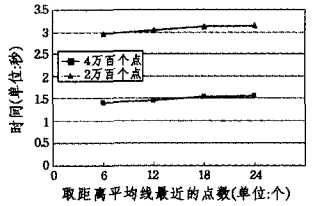

图 6 RDS 算法中取距离平均线最近的点数对算法效率的影响

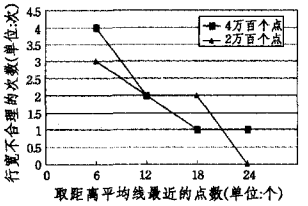

图 7 RDS 算法关于取距离直线 $x=\bar{x}$ 最近点数的敏感度

4.3 RDS 算法与快速排序算法(QS)的效率比较

考虑到 RDS 算法所具有的随机因素，我们选取经典的 QS 算法与之比较，结果如图 8 所示。从图中可以看出，随着实验点数的不断增加，两个算法的运行时间都在相应增加，在实验点数相同的情况下，QS 算法比 RDS 算法所耗时间明显要多。

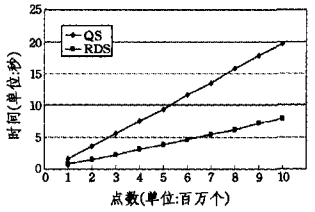

图 8 RDS 算法与 QS 算法的比较

4.4 CP_SDD 算法、CP_SDD+RDS 算法和 CP_DCM 算法的效率比较

从图 9 可以看出，随着实验点数的不断增加，两个算法的运行时间都在相应增加，CP_SDD 算法比 CP_SDD+RDS 算法所耗时间明显要多。

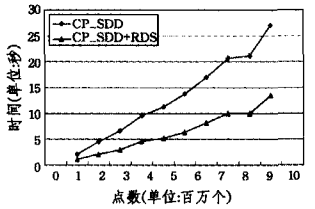

图 9 CP_SDD 算法与 CP_SDD+RDS 算法的比较

由于原始与改进后的 CP_DCM 算法[2,3]在效率上无根本区别，后者只对前者在计算距离次数方面作了一些改进，如计算一百次的两点距离 $|a_ia_j|$，用时约为 0.0286ms，再结合图 5、图 10 可知，原始的 CP_DCM 算法用于计算两点距离的时间不足 0.1%，耗费时间的主部来自多次找中位数及对数据区域进行划分。因此，新算法只需与原始的 CP_DCM 算法作比较。

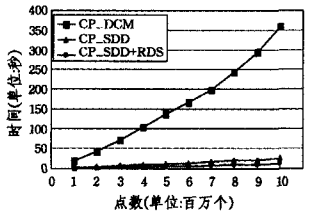

图 10 CP_SDD 算法、CP_SDD+RDS 算法与 CP_DCM 算法的比较

从图 10 可以看出，在实验点数相同的情况下，原始的 CP_DCM 算法比 CP_SDD 算法和 CP_SDD +RDS 算法所耗时间要多得多，同时 CP_SDD+RDS 算法所耗时间最少。

结束语 本文提出了一种新的求解最近对算法 CP_SDD+RDS，与经典的 CP_DCM 算法相比，该算法在不增加空间开销的前提下，平均时间复杂度下降到 $O(n)$，算法效率显著提高。另外，新算法 RDS 的提出，不仅为数据排序提供了一种新颖而高效的方法，同时，该算法所提出的分行思想及算法本身亦可移植到划分网格的策略上来，为基于网格划分的最近邻搜索方法在高效性和稳定性方面的改进提供了思想方法与理论支持。

未来工作将在此论文的基础上，利用 CP_SDD+RDS 算法对最近对及基于网格划分的相关问题进行优化。

参 考 文 献

[1] Shamos M I, Hoey D. Closest-point problems[C]// Proc. 16th IEEE Annu. Symp. Found. Comput. Sci. Berkeley, US, 1975: 151-162

[2] Zhou Yu-lin, Xiong Peng-rong, Zhu Hong. An improved algorithm about the closest pair of points on plane set[J]. Computer Reachearch and Development, 1998, 35(10): 957-960

[3] Ge Qi, Wang Hai-tao, Zhu Hong. An Improved Algorithm for Finding the Closest Pair of Points[J]. J. Comput. Sci. & Technol., 2006, 21(1): 27-31

[4] Xiong X P, Mohamed F, et al. SEA-CNN: Scalable processing of continuous K-nearest neighbor queries in spatio-temporal databases[C]// Proc of ICDE2005. Piscataway, NJ: IEEE, 2005: 643-654

[5] Yu X H, Pu K Q, et al. Monitoring K-nearest neighbor queries over moving objects [C] // Proc of Data Engineering (ICDE2005). Piscataway, NJ: IEEE, 2005: 631-642

[6] Mouratidis K, Hadjieleftheriou M. Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring[C]// Proc of ACM SIGMOD2005. New York: ACM, 2005: 634-645

[7] Zhang Xiao-feng, Wang Li-zhen, Xiao Qing, et al. Continuous Nearest Neighbor Query Based on Conceptual Partitioning[J]. Journal of Computer Research and Development, 2010, 47(Suppl.): 154-160

[8] Xu Hong-bo, Hao Zhong-xiao. Nearest-neighbor Query Algorithm Based on Grid Partition of Space-filling Curve[J]. Computer Science, 2010, 37(1)

[9] 张丽平，李松. 数据集中强邻近对的查询方法[J]. 计算机工程与设计，2008，29(16)：4353-4355

[10] 张奋，潘梅生，邹北骥. 基于 SR-树的空间对象最近邻查询[J]. 计算机工程与应用，2007，43(4)：173-175

[11] 李松，郝忠孝. 移动对象的动态反向最近邻查询技术[J]. 计算机工程，2008，34(10)：40-42

[12] Sack J R, Urrutia J. Closest-point problems in computational geometry[M]. Handbook on Computational Geometry. Ottawa: Elesvier Science, 2000: 877-935

[13] Corral A, Manolopoulos Y, Theodoridis Y, et al. Algorithms for processing k-closest-pair queries in spatial databases[J]. Data and Knowledge Engineering, 2004, 49(1): 67-104

[14] Fabrizio A, Clara P. Approximate k-closest-pairs in large high dimensional data sets[J]. Journal of Mathematical Modeling and Algorithms Springer Netherlands, 2005, 4(2): 149-179