一种基于速度的移动对象轨迹简化算法

杨 彪, 杨智应
(上海海事大学 信息工程学院, 上海 201306)

摘 要: 目前对移动对象轨迹简化问题分为离线简化和在线简化。以往的简化方法中许多依赖轨迹的几何特性,而针对移动对象的速度这一重要特征没有足够关注。基于速度研究移动对象轨迹的离线简化新方法,提出了移动对象轨迹离线简化的动态规划算法、基于广度优先搜索的轨迹简化算法及其优化算法、时间复杂度更低的近似算法。通过大量实验验证所提出的算法比基于方向的简化算法和基于位置的简化算法具有更好的简化效率。

关键词: 移动对象; 离线轨迹简化; 速度阈值; 近似算法
中图分类号: TP391 **文献标志码**: A **文章编号**: 1001-3695(2018)09-2666-05
doi:10.3969/j.issn.1001-3695.2018.09.024

Simplification algorithm of moving trajectory based on velocity

Yang Biao, Yang Zhiying
(*College of Information Engineering*, *Shanghai Maritime University*, *Shanghai* 201306, *China*)

Abstract: At present, the problem of moving object trajectory simplification is divided into offline simplification and online simplification. Many of the previous simplified methods rely on the geometric properties of the trajectory, and the important feature of the velocity of the moving object is not sufficiently concerned. This paper proposed a new method of moving object trajectories based on the new method of off-line simplification, and proposed a dynamic programming algorithm for moving object trajectory off-line, a trajectory simplification algorithm based on breadth-first search and its optimization algorithm, and an algorithm with lower time complexity. And through a large number of experiments to verify the proposed algorithm has a better simplification efficiency than the direction-based simplification algorithm and location-based simplification algorithm.

Key words: moving object; offline trajectory simplification; speed threshold; approximation algorithm

随着 GPS 嵌入式设备的激增(如智能电话和出租车),移动对象轨迹数据变得无处不在。在过去十多年中,人们对移动对象数据库(MOD)已进行了广泛的研究[1]。移动轨迹数据通常由周期性地收集移动物体的位置而获得。例如,一个拥有 100 万智能手机用户的城市,假定以每分钟的采样频率来收集用户的位置信息,一天的轨迹数据量在 30 GB 以上。

原始轨迹数据量的庞大,将导致存储成本剧增。另外,因大多数的查询处理和数据挖掘算法相当耗时,将整个收集到的原始轨迹直接使用,在效率上是不可行的。

虽然增大移动对象的采样时间间隔可以减小数据量,但对于移动对象的速度变化频繁的情况,将给跟踪分析带来不利影响。通常的做法是以较高的采样频率来记录移动对象的位置变化,然后在某一给定误差范围内,消除某些原始轨迹点来得到规模较小的简化轨迹,即移动对象轨迹简化。然而现在提出的轨迹简化算法要么去掉过多的轨迹特征点,使简化后的轨迹与原始轨迹偏差较大,精确度不高;要么保留了过多的轨迹点,使存储成本上升,简洁度不够。为了解决这个问题,本文提出的基于速度的轨迹简化算法能够在权衡轨迹的简洁性和精确性上有较大的优势,简化效果更好。

1 移动对象轨迹简化算法的研究现状

在最近的研究中,Long 等人在文献[2]中提出了基于方向的轨迹简化算法,通过给定的角度阈值来简化轨迹,使求出的简化轨迹最大角度值不大于给定的角度阈值。在此基础上,文献[3]提出了简化轨迹的数量在不大于给定值的情况下,使得简化轨迹与原始轨迹方向阈值最小的简化算法。Deng 等人[4]基于文献[2]提出了基于方向的实时简化算法。称这类轨迹简化算法为基于方向的轨迹简化算法(DPTS)。

另一类轨迹简化算法以垂线距离或同步欧氏距离为误差度量指标。Douglas 等人[5]提出的 DP 算法以垂线距离作为误差度量指标来简化轨迹。Keogh 等人基于垂线距离提出了实时压缩算法。Meratnia 等人[6]对 DP 算法作了改进,提出了以同步欧氏距离为误差度量指标的轨迹简化算法。称这类算法为基于距离的轨迹压缩算法(PPTS)。

此外,也有其他的轨迹简化算法。例如 Richter 等人[7]介绍了如何通过轨迹的语义信息来简化轨迹,并且允许在可接受的信息丢失范围内进行轨迹简化。文献[8~10]采用线性插值来简化轨迹,因为线性插值能够很好地在精度与计算成本之间取得平衡。文献[11~13]通过构建安全区域来预估下一个轨迹点的大致范围,然后根据下一个轨迹点的实际位置是否在该范围内来判断该点是否需要保留。文献[14,15]提出利用标准图形去逐段匹配整个轨迹,并用相应的误差度量方式来评价算法的优劣。

Ying 等人[16]提出了基于速度的轨迹简化算法,称这种算法为基于速度的轨迹压缩算法(VPTS)。本文的轨迹简化思想与文献[16]有很大的不同。文献[16]的轨迹简化核心度量指标是关于轨迹段平均速度的比值,而本文轨迹核心度量指标为轨迹段的最大速度差值。相较于文献[16]的度量指标,本文

收稿日期: 2017-04-20; **修回日期**: 2017-06-21

作者简介:杨彪(1992-),男,湖北荆州人,硕士,主要研究方向为移动对象轨迹化简(1980068026@qq.com);杨智应(1964-),男,教授,博士,主要研究方向为算法与复杂性、移动计算、分布式计算、软件工程.

的度量指标更能突显轨迹速度变化的剧烈程度。

2　移动轨迹简化问题描述及算法

2.1　一些定义及记号

定义 1　位置点或轨迹点。位置点用 p 表示,p 是一个三元组,组内对应属性分别代表该点的经度、纬度、时间,$p=(x,y,t)$。

定义 2　原始轨迹。原始轨迹 T 是一些有序位置点的集合。定义 $T=(p_1,p_2,\cdots,p_i,\cdots,p_n)(i\in[1,n])$,其中 $p_i=(x_i,y_i,t_i)$,且 $t_1<t_2<\cdots<t_i<t_n$。

定义 3　原始轨迹的子轨迹。原始轨迹 T 的子轨迹用 $T[i,j]$ 表示,也是一些有序位置点的集合。定义 $T[i,j]=(p_i,p_{i+1},\cdots,p_j)(1\leqslant i\leqslant j\leqslant n)$。

定义 4　简化轨迹。简化轨迹 T_{sim} 是原始轨迹 T 去掉一些位置点的有序集合。定义 $T_{\text{sim}}=(p_{s_1},p_{s_2},\cdots,p_{s_m})(1\leqslant s_1<s_2<s_m\leqslant n$,且 $m\leqslant n)$。

定义 5　轨迹段。原始轨迹 T 或简化轨迹 T_{sim} 中,对位置点 p_i 和 $p_j(1\leqslant i<j\leqslant n)$,如果 $j-i=1$,则认为 p_i 和 p_j 是相邻的,相邻两点之间所连的线段称为轨迹 T 或轨迹 T_{sim} 中的轨迹段,用 $\overline{p_ip_j}$ 表示。

原始轨迹 T 中位置点的个数用 $|T|$ 表示,则 T 中轨迹段的个数为 $|T|-1$ 或 $n-1$。简化轨迹 T_{sim} 中位置点的个数用 $|T_{\text{sim}}|$ 表示,则 T_{sim} 中轨迹段的个数为 $|T_{\text{sim}}|-1$ 或 $m-1$。

如图 1 所示,假定原始轨迹 $T=(p_1,p_2,\cdots,p_8)$,简化后的轨迹 $T_{\text{sim}}=(p_1,p_4,p_6,p_8)$;原始轨迹 T 的轨迹段用实心黑线表示,简化轨迹 T_{sim} 的轨迹段用实心虚线表示;此时 $|T|=8$,$|T_{\text{sim}}|=4$。

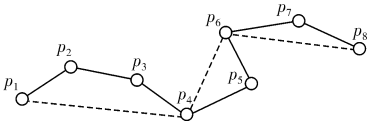

图 1　原始轨迹和简化轨迹

定义 6　轨迹段的长度。原始轨迹 T 或简化轨迹 T_{sim} 中,对位置点 p_i 和 $p_j(1\leqslant i<j\leqslant n)$,如果 p_i 和 p_j 相邻,则轨迹段的长度用 $d(\overline{p_ip_j})$ 表示。

定义 7　轨迹段的平均速度。原始轨迹 T 或简化轨迹 T_{sim} 中,对位置点 p_i 和 $p_j(1\leqslant i<j\leqslant n)$,如果 p_i 和 p_j 相邻,则轨迹段的平均速度用 $v(\overline{p_ip_j})$ 表示。

$$v(\overline{p_ip_j})=\frac{d(\overline{p_ip_j})}{p_j.t_j-p_i.t_i}$$

定义 8　简化轨迹 T_{sim} 某个轨迹段的最大速度差值。简化轨迹 T_{sim} 中,对位置点 p_{s_k} 和 $p_{s_{k+1}}$,$k\in[1,m)$,则在简化轨迹 T_{sim} 中的轨迹段 $\overline{p_{s_k}p_{s_{k+1}}}$ 在允许的误差范围内可以近似地替代原始轨迹 T 中位置点 p_{s_k} 与 $p_{s_{k+1}}$ 之间所有的轨迹段。简化轨迹 T_{sim} 中的轨迹段 $\overline{p_{s_k}p_{s_{k+1}}}$ 的最大速度差值用 $\varepsilon(\overline{p_{s_k}p_{s_{k+1}}})$ 表示。

$$\varepsilon(\overline{p_{s_k}p_{s_{k+1}}})=\max_{s_k\leqslant h<s_{k+1}}|v(\overline{p_{s_k}p_{s_{k+1}}})-v(\overline{p_hp_{h+1}})|$$

定义 9　简化轨迹 T_{sim} 的最大速度差值。对简化轨迹 T_{sim} 中,对任意位置点 p_{s_k},$k\in[1,m)$,简化轨迹 T_{sim} 的最大速度差值用 $\varepsilon(T_{\text{sim}})$ 表示。

$$\varepsilon(T_{\text{sim}})=\max_{1\leqslant k<m}|\varepsilon(\overline{p_{s_k}p_{s_{k+1}}})|$$

2.2　基于速度的轨迹简化问题定义

在本文中提出了一种称为基于速度的轨迹简化的新方法。用 ε_v 表示人为设定的速度阈值,用来表示简化轨迹与原始轨迹之间的近似程度范围。例如,对一条原始轨迹 T,假定人为设定速度阈值 $\varepsilon_v=3.0$ m/s,若 T 的一条简化轨迹为 T_{sim},其最大速度差值 $\varepsilon(T_{\text{sim}})\leqslant\varepsilon_v=3.0$ m/s,则表明简化轨迹与原始轨迹之间的速度偏差小、近似程度高,是原始轨迹 T 的简化轨迹;若求得的简化 T_{sim},其最大速度差值 $\varepsilon(T_{\text{sim}})>\varepsilon_v=3.0$ m/s,则表明简化轨迹与原始轨迹之间的速度偏差大、近似程度低,不是原始轨迹 T 的简化轨迹。

问题描述:T 是原始轨迹,T_{sim} 是 T 的简化轨迹,ε_v 是一个预先给定的速度阈值。如果 T 有多个简化轨迹满足 $\varepsilon(T_{\text{sim}})\leqslant\varepsilon_v$,如何找到一条简化轨迹 T_{sim},使简化轨迹 T_{sim} 的轨迹点最少?

本文采用了轨迹段的最大速度差值而不是平均速度差值,因为最大速度差值更能突显出轨迹速度变化的剧烈程度。为了解决上述问题,本文提出了以下多种算法来解决该问题。本文的创新点分别是:

a)提出基于速度的动态规划简化算法(v-dp 算法);

b)提出基于广度优先搜索的简化算法(v-bfs 算法),以及 v-bfs 算法的优化算法(v-bfs-imp 算法);

c)提出 v-bfs-imp 算法的近似算法(v-app 算法);

d)用实验验证了 v-bfs 以及 v-bfs-imp 算法在权衡轨迹的简洁性和精确性有较大优势,并且分析了 v-app 和 v-bfs-imp 算法的近似程度。

2.3　穷举法

一个本能的想法是罗列出原始轨迹 T 中所有位置点的可能的组合方式,除去首尾两点,所有可能的组合有 $2^{|T|-2}$ 种。对其中任一种组合方式 T_{any},若 $\varepsilon(T_{\text{any}})\leqslant\varepsilon_v$,则该简化轨迹是 T 的备选简化轨迹;若 $\varepsilon(T_{\text{any}})>\varepsilon_v$,则该轨迹不作为 T 的备选简化轨迹。再对 T 的所有备选简化轨迹中找到轨迹点个数最小的简化轨迹 T_{sim}。该算法的时间复杂度为 $\Omega(2^{|T|-2})$。当 $|T|$ 较大时,该算法不可行。

2.4　基于速度的动态规划简化算法(v-dp 算法)

设计一个带备忘机制的自顶向上的递归算法来求解上述描述的问题,称该算法为 v-dp 算法。求原始轨迹 T 在给定的速度阈值 ε_v 下的最少轨迹点的简化轨迹,等价于原始轨迹 T 在给定的阈值 ε_v 下求最少轨迹段的个数。子轨迹 $T[i,j]$ 在速度阈值 ε_v 下的简化轨迹的轨迹段的个数定义为 $S[i,j]$。

此动态规划算法如图 2 所示。当 $i=1$,$j=n$ 时,即可求出原始轨迹在速度阈值 ε_v 下最小的轨迹段 $S[1,n]$,此时 $|T_{\text{sim}}|=S[1,n]+1$。

$$S[i,j]=\begin{cases}1 & \varepsilon(\overline{p_ip_j})\leqslant\varepsilon_v\\ \min_{i<k<j}\begin{cases}1+S[k,j] & \text{if } \varepsilon(\overline{p_ip_k})\leqslant\varepsilon_v,\ \varepsilon(\overline{p_kp_j})>\varepsilon_v\\ S[i,k]+1 & \text{if } \varepsilon(\overline{p_ip_k})>\varepsilon_v,\ \varepsilon(\overline{p_kp_j})\leqslant\varepsilon_v\\ S[i,k]+S[k,j] & \text{if } \varepsilon(\overline{p_ip_k})>\varepsilon_v,\ \varepsilon(\overline{p_kp_j})<\varepsilon_v\end{cases}\end{cases}$$

图 2　v-dp 算法的 $S[i,j]$ 关系表达式

当 $\varepsilon(\overline{p_ip_j})\leqslant\varepsilon_v$ 时,表明子轨迹 $T[i,j]=(p_i,p_{i+1},\cdots,p_j)$ 的轨迹段 $\overline{p_ip_j}$ 的最大速度差值 $\varepsilon(\overline{p_ip_j})$ 都不大于 ε_v,则无须再进行计算,则存在一条最短的轨迹段为 $\overline{p_ip_j}$ 满足条件,此时轨迹段的个数为 1,满足条件的轨迹点为 p_i 和 p_j 两点;当 $\varepsilon(\overline{p_ip_j})>\varepsilon_v$ 时,则需要继续对子轨迹 $T[i,j]=(p_i,p_{i+1},\cdots,p_j)$ 的两个子轨迹 $T[i,k]$ 和 $T[k,j]$ 进行判断($T[i,k]=(p_i,p_{i+1},\cdots,p_k)$、$T[k,j]=(p_k,p_{i+1},\cdots,p_j)(i+1\leqslant k\leqslant j-1)$),当 k 值从 $i+1$ 变化到 $j-1$ 时,则使 $S[i,j]$ 值最小的对应 k 值是所求的值。

算法复杂度分析:对于一个具体的实例 $T[i,j]$,首先验证 $\varepsilon(\overline{p_ip_j})\leqslant\varepsilon_v$ 是否成立,该步骤的最坏时间复杂度为 $O(n)$,当 $\varepsilon(\overline{p_ip_j})\leqslant\varepsilon_v$ 成立时,$S[i,j]=1$;当 $\varepsilon(\overline{p_ip_j})\leqslant\varepsilon_v$ 不成立时,则需对 $T[i,j]$ 的子轨迹进行判断,变成求另外的具体的实例。总的来说,有 $O(n^2)$ 个具体的轨迹对 (p_i,p_j) 需要去求 $\varepsilon(\overline{p_ip_j})$,求 $\varepsilon(\overline{p_ip_j})$ 的最坏时间复杂度为 $O(n)$,因此总的时间复杂度为 $O(n^2)\times O(n)=O(n^3)$。

通过上述分析,此动态规划算法的时间复杂度为 $O(n^3)$,空间复杂度为 $\Theta(n^2)$[17]。然而该动态算法的时间复杂度和空

间复杂度依旧较高。为了能进一步提高简化效率,提出基于广度优先搜索的简化轨迹模型及其相应算法。

2.5 基于广度优先算法的简化算法(v-bfs 算法)

该算法运用图的广度优先搜索算法(BFS)来求解简化轨迹大小$|T_{sim}|$最小值问题,称该算法为 v-bfs 算法。其构造步骤如下:

a)图的构造。构造一个图称为 $G_{\varepsilon_v}(V,E)$,其中 V 表示顶点的集合,E 表示边的集合。对任意的 p_i 点($1\leqslant i\leqslant n$),$p_i\in V$;对任意的位置点 p_i 和 p_j($1\leqslant i<j\leqslant n$),如果 $\varepsilon(\overline{p_ip_j})\leqslant\varepsilon_v$,则边 $(p_i,p_j)\in E$。

b)寻找最短路径。在图 $G_{\varepsilon_v}(V,E)$ 中用广度优先搜索算法(BFS)寻找一条从 p_1 到 p_n 的最短路径。

c)找到图的简化轨迹。根据步骤 b)的路径关系找到一条简化轨迹。

v-bfs 算法时间复杂度分析:步骤 a)最多有 $O(n^2)$ 对轨迹点(p_i,p_j)需要比较,每次比较 $\varepsilon(\overline{p_ip_j})\leqslant\varepsilon_v$ 是否成立,最坏情况下需要 $O(n)$,则步骤 a)的时间复杂度为 $O(n^3)$;步骤 b)BFS 算法的时间复杂度和空间复杂度均为 $O(V+E)$,($|V|=O(n)$,$|E|=O(n^2)$);步骤 c)寻找图简化轨迹点只需 $O(n)$的时间,则该算法需要的时间复杂度为 $O(n^3)$,空间复杂度为 $O(V+E)$。

为了进一步降低该算法的时间和空间复杂度,提出 v-bfs 算法的优化算法。该优化算法通过简化 v-bfs 算法步骤 a)图的构造过程,来减小该算法的时间复杂度。

2.6 v-bfs 算法的优化(v-bfs-imp 算法)

优化算法的目的是为了降低算法的时间复杂度。在提出优化算法之前先介绍几个概念。

定义 10 轨迹段的速度变化范围。原始轨迹 T 中,对位置点 p_i 和 p_{i+1}($1\leqslant i<i+1\leqslant n$),基于速度阈值 ε_v 的轨迹段 $\overline{p_ip_j}$ 的速度变化范围用 $fvr(\overline{p_ip_{i+1}}\,|\,\varepsilon_v)$ 表示,$fvr(\overline{p_ip_{i+1}}\,|\,\varepsilon_v)=[v(\overline{p_ip_{i+1}})-\varepsilon_v,v(\overline{p_ip_{i+1}})+\varepsilon_v]$。

在图 1 中,假定 $v(\overline{p_3p_4})=20$ m/s,$\varepsilon_v=3.0$ m/s,则 $fvr(\overline{p_3p_4}\,|\,3.0)=[17.0,23.0]$。

定义 11 子轨迹的速度变化范围。在子轨迹 $T[i,j]$ 中,$fvr(T[i,j]\,|\,\varepsilon_v)$ 为子轨迹中各个轨迹段的速度变化范围的交集,子轨迹 $T[i,j]$ 的速度变化范围用 $fvr(T[i,j]\,|\,\varepsilon_v)$ 表示。

$$fvr(T[i,j]\,|\,\varepsilon_v)=\cap_{i\leqslant h<j}fvr(\overline{p_hp_{h+1}}\,|\,\varepsilon_v)$$

定义 12 $fvr(T[i,j]\,|\,\varepsilon_v)$ 中区间的个数。$fvr(T[i,j]\,|\,\varepsilon_v)$ 中区间的个数用 $\|fvr(T[i,j]\,|\,\varepsilon_v)\|$ 表示。

在图 1 中,又假定 $v(\overline{p_1p_2})=18$ m/s,$v(\overline{p_2p_3})=19$ m/s,$v(\overline{p_4p_5})=27$ m/s,则 $fvr(T[1,4]\,|3.0)=fvr(\overline{p_1p_2}\,|3.0)\cap fvr(\overline{p_2p_3}\,|3.0)\cap fvr(\overline{p_3p_4}\,|3.0)=[17.0,21.0]$,$fvr(T[3,5]\,|3.0)=fvr(\overline{p_3p_4}\,|3.0)\cap fvr(\overline{p_4p_5}\,|3.0)=[17.0,23.0]\cap[24.0,30.0]=\varnothing$。此时 $\|fvr(T[1,4]\,|3.0)\|=1$,$\|fvr(T[3,5]\,|3.0)\|=0$,$\|fvr(T[1,5]\,|3.0)\|=0$。

引理 1 在原始轨迹 T 中,$T=(p_1,\cdots,p_i,\cdots,p_j,\cdots,p_n)$($1\leqslant i<j\leqslant n$),$\|fvr(T[i,j]\,|\,\varepsilon_v)\|=0$ 或 $\|fvr(T[i,j]\,|\,\varepsilon_v)\|=1$。

引理 2 在原始轨迹 T 中,$T=(p_1,\cdots,p_i,\cdots,p_j,\cdots,p_n)$($1\leqslant i<j\leqslant n$),$\varepsilon(\overline{p_ip_j})\leqslant\varepsilon_v$ 当且仅当 $v(\overline{p_ip_j})\in fvr(T[i,j]\,|\,\varepsilon_v)$。

引理 3 在原始轨迹 T 中,$T=(p_1,\cdots,p_i,\cdots,p_j,\cdots,p_n)$($1\leqslant i<j\leqslant n$),如果 $\varepsilon(\overline{p_ip_j})\leqslant\varepsilon_v$,则 $\|fvr(T[i,j]\,|\,\varepsilon_v)\|=1$。

引理 4 在原始轨迹 T 中,$T=(p_1,\cdots,p_i,\cdots,p_j,\cdots,p_n)$($1\leqslant i<j\leqslant n$),如果 $\|fvr(T[i,j]\,|\,\varepsilon_v)\|=0$,则 $\varepsilon(\overline{p_ip_j})>\varepsilon_v$。

引理 5 在原始轨迹 T 中,$T=(p_1,\cdots,p_i,\cdots,p_j,\cdots,p_n)$($1\leqslant i<j\leqslant n$),如果 $\|fvr(T[i,j]\,|\,\varepsilon_v)\|=1$,$v(\overline{p_ip_j})\notin fvr(T[i,j]\,|\,\varepsilon_v)$,则 $\varepsilon(\overline{p_ip_j})>\varepsilon_v$。

因为 v-bfs 算法步骤 a)图构造的时间复杂度为 $O(n^3)$,只有降低步骤 a)的时间复杂度才能降低整个算法的复杂度。将 $\varepsilon(\overline{p_ip_j})\leqslant\varepsilon_v$ 的计算转换为 $v(\overline{p_ip_j})\in fvr(T[i,j]\,|\,\varepsilon_v)$ 的计算可减小时间复杂度,称此优化算法为 v-bfs-imp 算法。

例如,当 $p_i=p_1$ 时,$j\in[2,n]$,j 逐渐增大时,则有 $n-1$ 对轨迹点(p_i,p_j)需要计算 $\varepsilon(\overline{p_ip_j})\leqslant\varepsilon_v$,以算法 v-bfs 来计算,则需要时间复杂度为 $O(n^2)$。然而该计算有很多冗余的过程,引理 2 的提出就是为了减小冗余。

求 $v(\overline{p_ip_j})\in fvr(T[i,j]\,|\,\varepsilon_v)$($1\leqslant i<j\leqslant n$)的时间复杂度如下:

首先计算 $fvr(\overline{p_jp_{j+1}})$ 的时间复杂度为 $O(1)$;$fvr(T[i,j]\,|\,\varepsilon_v)=\cap_{k\in[i,j)}fvr(\overline{p_kp_{k+1}})$,则有 $m-1(m=j-i)$个区间相交$m-2$ 次得到 $fvr(T[i,j]\,|\,\varepsilon_v)$,根据引理 1 有 $\|fvr(T[i,j]\,|\,\varepsilon_v)\|\leqslant 1$,则求 $fvr(T[i,j]\,|\,\varepsilon_v)$的时间复杂度为 $O(m)$;然后根据引理 2~5,判断 $v(\overline{p_ip_j})$是否属于 $fvr(T[1,n]\,|\,\varepsilon_v)$的时间复杂度变为 $O(1)$,则计算 $v(\overline{p_ip_j})\in fvr(T[i,j]\,|\,\varepsilon_v)$总的时间复杂度为 $O(m)$。

当 $p_i=p_1$ 时,$j\in[2,n]$,j 逐渐增大时,按上述计算方法,计算 $v(\overline{p_ip_j})\in fvr(T[i,j]\,|\,\varepsilon_v)$时间复杂度为 $O(n)$,时间复杂度比原来少了一个数量级。

根据引理 2 可以证明,相同的轨迹经过 v-bfs 和 v-bfs-imp 算法得到相同的简化轨迹,只是 v-bfs-imp 算法时间复杂度降低了一个数量级,空间复杂度还是 $O(V+E)$。

然而当 n 很大时,该算法的复杂度仍然不让人满意。现提出近似算法进一步简化 v-bfs-imp 算法的时间复杂度和空间复杂度。

2.7 近似算法(v-app 算法)

在提出近似算法前,先提出子轨迹的关键位置点概念。

定义 13 子轨迹的关键位置点。在子轨迹 $T[i,j]$($1\leqslant i<j\leqslant n$)中,当 $i<k<k+1\leqslant j$,如果 $\varepsilon(\overline{p_ip_k})\leqslant\varepsilon_v$ 而 $\varepsilon(\overline{p_ip_{k+1}})>\varepsilon_v$ 时,则位置点 p_k 为子轨迹 $T[i,j]$的关键位置点。

运用引理 2 和关键位置点的思想,提出近似算法 v-app。算法伪代码如下:

```
输入:T_sim = ∅,ε_v,T = (p_1,…,p_i,…,p_j,…,p_n)。
{
  T_sim = T_sim ∪ (p_1); //将起始点 p_1 加入 T_sim 中
  i = 1;j = 2;
  while(j≤n)
  { while(j≤n and v(p_ip_j) ∈fvr(T[i,j] | ε_v))
    {
        j ++;
    }
      T_sim = T_sim ∪p_{j-1}; //v(p_ip_{j-1}) ∈fvr(T[i,j-1] | ε_v)
    //v(p_ip_j) ∉fvr(T[i,j] | ε_v)
      i = j - 1; //点 p_{j-1}是关键点
  }
  return T_sim;
}
```

近似算法的时间复杂度为 $O(n)$,空间复杂度为 $O(n)$。通过后续实验比较结果可以发现,当速度阈值 ε_v 较小时,经过 v-app 与 v-bfs-imp 算法简化后的轨迹近似程度非常高。

3 实验比较

为了分析与评估算法的性能,实验选取笔记本电脑作为硬件测试平台。具体配置如下:处理器为英特尔 Pentium P6200 @2.13 GHz 双核,内存为 6 GB;实验环境为 Windows 7 操作系统和 Eclipse 开发系统,Java 语言实现。实验数据从项目 INFATI 得到[18],INFATI 的位置数据由 20 个有效的 GPS 数据集

构成。

3.1　度量指标

轨迹简化应该具有以下两个期望的性质:简洁性和精确性。简洁性意味着简化后的轨迹数量尽可能少,精确性意味着简化轨迹应该与原始轨迹差异尽可能小。然而既要高精度又要高简洁度是不可能的,精确性与简洁性之间是相互矛盾的,只能在两者之间找平衡点。文献[19]用 $L(H)$ 表示简化轨迹与原轨迹之间简洁度的偏差,用 $L(D|H)$ 表示简化轨迹与原轨迹之间精确度的偏差。$L(D|H)$_position 表示精确度偏差中基于位置信息上的偏差,$L(D|H)$_direction 表示精确度偏差中基于方向信息上的偏差。*MDL* 表示简化轨迹与原始轨迹总偏差。根据文献[19]中式(1)(3)(6)(7),分别罗列出这五个变量的定义:

$$L(H)=\sum_{i=1}^{s_m}\log_2(d(\overline{p_ip_{i+1}}))$$

$$L(D|H)=L(D|H)\_\text{position}+L(D|H)\_\text{direction}$$

$$L(D|H)\_\text{position}=\sum_{j=s_1}^{s_m}\sum_{i=j}^{j+1}\log_2(d_\perp(p_ip_{i+1},p_jp_{j+1}))$$

$$L(D|H)\_\text{direction}=\sum_{j=s_1}^{s_m}\sum_{i=j}^{j+1}\log_2(d_\theta(p_ip_{i+1},p_jp_{j+1}))$$

$$MDL=L(H)+L(D|H)$$

3.2　不同轨迹数据对轨迹压缩率的影响

选取4 条不同的轨迹,每条轨迹选取 300 个连续的 GPS 轨迹点,采用相同的 v-bfs-imp 算法进行轨迹简化。当速度阈值 ε_v 从 0.5 m/s 逐渐增大到 4.5 m/s 时,简化轨迹轨迹点与原始轨迹轨迹点的比值变化情况如图 3 所示。

从图 3 中可以看出,随着 ε_v 逐渐增大,简化后的轨迹数逐渐减小;其中任意两条轨迹,在相同的速度阈值下,轨迹压缩率各不相同,压缩率的差值也在不断变化。

3.3　算法比较(VPTS vs DPTS vs PPTS)

用本文提出的 v-bfs-imp 算法来与基于方向信息的轨迹简化算法(DPTS)、基于位置信息的轨迹简化算法(PPTS)作比较,用以上五个指标来度量算法的优劣性。

PPTS 采用 DP 算法[5]来作为比较对象,因为该算法是纯粹的基于位置信息来简化轨迹,同时也是非常著名的算法。DPTS 采用 SP 算法[2]作为比较对象,因为该算法纯粹基于方向信息来简化轨迹。

对每一个速度阈值 ε_v,将用算法 v-bfs-imp 得到的简化轨迹 T_{sim},根据文献[2]中式(2)求出简化轨迹 T_{sim} 与原始轨迹 T 的角度偏差,作为 SP 算法的输入值。

同样将相同速度阈值得到的简化轨迹 T_{sim},对简化轨迹 T_{sim} 中的位置点 p_{s_k} 和 $p_{s_{k+1}}$ ($k\in[1,m)$),则点 p_{s_k} 和 $p_{s_{k+1}}$ 在简化轨迹 T_{sim} 中是相邻的;在简化轨迹 T_{sim} 中的轨迹段 $\overline{p_{s_k}p_{s_{k+1}}}$ 的最大垂线距离用 $d_\perp(\overline{p_{s_k}p_{s_{k+1}}})$ 表示。

$$d_\perp(\overline{p_{s_k}p_{s_{k+1}}})=\max_{s_k\leqslant h\leqslant s_{k+1}}(d_\perp(p_h,\overline{p_{s_k}p_{s_{k+1}}}))$$

其中:$d_\perp(p_h,\overline{p_{s_k}p_{s_{k+1}}})$ 表示轨迹点 p_h 到直线 $\overline{p_{s_k}p_{s_{k+1}}}$ 的垂直距离。简化轨迹 T_{sim} 与原始轨迹 T 的垂直距离定义为 $d_\perp(T_{\text{sim}})$,$d_\perp(T_{\text{sim}})=\max_{1\leqslant k<m}|d_\perp(\overline{p_{s_k}p_{s_{k+1}}})|$,将 $d_\perp(T_{\text{sim}})$ 作为 DP 算法的输入值。

$L(H)$ 的值变化幅度如图 4 所示。v-bfs-imp 和 DP 算法都随着 ε_v 的增大而 $L(H)$ 的值逐渐变小,当 $\varepsilon_v=6.0$ m/s 时,$L(H)$ 的值近似相等;然而 SP 算法随着 ε_v 的增大 $L(H)$ 的值先变小然后保持不变,在相同速度阈值 ε_v 下,$L(H)$ 的值最小。

实验表明 SP 算法相较于 v-bfs-imp、DP 算法简洁性更好;在 $\varepsilon_v<4.0$ m/s 时,DP 算法的简洁性同样要比 v-bfs-imp 算法要好。

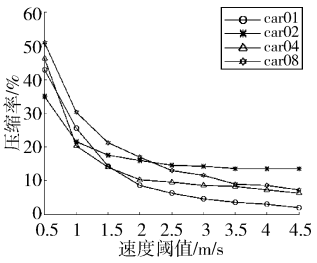

图 3　不同速度阈值的简化轨迹轨迹点与原始轨迹轨迹点的比值

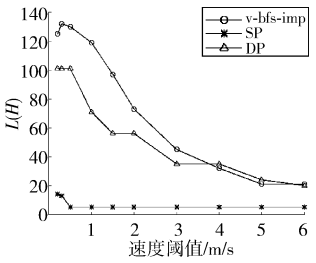

图 4　不同速度阈值的不同 $L(H)$ 值

在图 5 中,v-bfs-imp 和 DP 算法随着 ε_v 的增大 $L(D|H)$_position 的值先增大后减小,SP 算法随着 ε_v 的增大 $L(D|H)$_position 的值先减小后保持不变。

v-bfs-imp 算法的轨迹一直都处在 DP 算法轨迹的下方,说明 v-bfs-imp 比 DP 算法保留了更多原始轨迹位置上的信息;但是当 $\varepsilon_v>0.7$ m/s 时,v-bfs-imp 算法的轨迹一直都处在 SP 算法轨迹的上方,说明 v-bfs-imp 算法在保留原始轨迹位置上的信息方面比 SP 算法差。

在图 6 中,三种算法均随着 ε_v 的增大 $L(D|H)$_direction 的值先增大后保持不变。

v-bfs-imp 算法的轨迹一直处于另外两条轨迹的下方,说明 v-bfs 比 DP 和 SP 算法保留了更多原始轨迹方向上的信息。

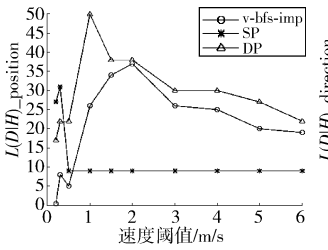

图 5　不同速度阈值的不同 $L(D|H)$_position 值

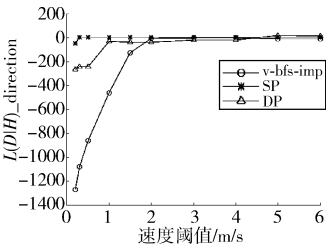

图 6　不同速度阈值的不同 $L(D|H)$_direction 值

图 7 与图 6 相近,表明 v-bfs-imp 算法简化后的轨迹与原始轨迹的精确度更高,与原始轨迹的偏差更小。

在图 8 中,三条轨迹的变化非常明显,当速度阈值 $\varepsilon_v<1.5$ m/s时,v-bfs-imp 算法在均衡简洁性和精确性上要比 DP、SP 算法要好;当速度阈值 $\varepsilon_v>1.5$ m/s 时,随着速度阈值 ε_v 的增大,经过三种算法后的简化轨迹与原始轨迹的总偏差大体上相差不大,表明这三种算法均衡效果相近。

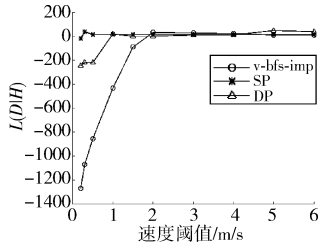

图 7　不同速度阈值的不同 $L(D|H)$ 值

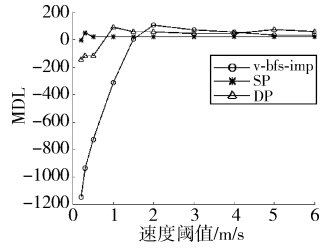

图 8　不同速度阈值的不同 MDL 值

总的来说,v-bfs-imp 算法在简洁度上比 SP、DP 算法差;但是 v-bfs-imp 算法在精确度上有明显优势,在权衡简洁性和精确性上比 SP、DP 算法也要好。

3.4　优化算法与近似算法的比较

本文研究轨迹数量 T 和速度阈值 ε_v 对 v-bfs-imp 和 v-app 算法的影响。原始轨迹 T 通过 v-bfs-imp 算法简化后的轨迹数为 $|T_{\text{bfs}}|$,原始轨迹 T 通过 v-app 算法简化后的轨迹数为 $|T_{\text{app}}|$,则定义变量 degree_app($|T|$,ε_v) 来描述这两种算法的

近似程度，degree_app($|T|$, ε_v) = $\frac{|T_{bfs}|}{|T_{app}|}\times 100\%$ 。

a) 当 ε_v 设定为 1.0 m/s 时，$|T|$ 从 100 逐渐增大到 1 000 时，两种算法的近似度变化幅度如图 9 所示。

b) 当 $|T|$ = 300 时，ε_v 从 0.2 m/s 逐渐增大到 4.0 m/s 时，两种算法的近似度变化幅度如图 10 所示。

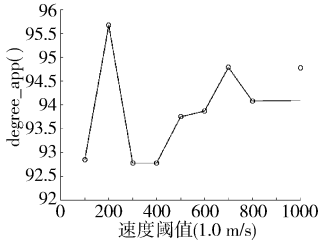

图 9 速度阈值固定，轨迹点数目变化时近似度的变化情况

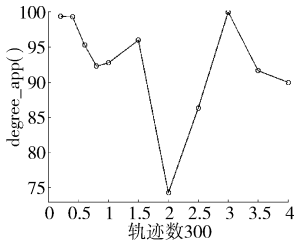

图 10 轨迹数量固定，速度阈值变化时近似度的变化情况

从图 9 中可以看出，近似度在 92% ~96% 波动，近似程度非常高。

从图 10 中可以看出，当速度阈值 $\varepsilon_v \leqslant 1.5$ m/s 时，近似度在 92% ~100% 波动；然而当速度阈值 $\varepsilon_v > 1.5$ m/s 时，个别点近似度波动相对较大，近似度最低达到 74.28%，但是大部分点近似度都在 85% 以上。

从以上的实验可看出，轨迹数量 $|T|$ 对近似度的影响小，速度阈值 ε_v 对近似度的影响大。当速度阈值较小时，v-app 与 v-bfs-imp 算法的近似度高。

3.5 优化算法和动态规划算法的比较

原始轨迹 T 通过 v-bfs-imp 算法简化后的轨迹数为 $|T_{bfs}|$，原始轨迹 T 通过 v-dp 算法简化后的轨迹数为 $|T_{dp}|$，则这两种算法的近似程度为 degree_app($|T|$, ε_v) = $\frac{|T_{bfs}|}{|T_{dp}|}\times 100\%$ 。

a) 当 ε_v 设定为 1.0 m/s 时，$|T|$ 从 100 逐渐增大到 1 000 时，两种算法的近似度变化幅度如图 11 所示，近似度一直为 100% 。

b) 当 $|T|$ = 300 时，ε_v 从 0.2 m/s 逐渐增大到 4.0 m/s 时，两个算法的近似度变化幅度如图 12 所示，近似度也一直为 100% 。

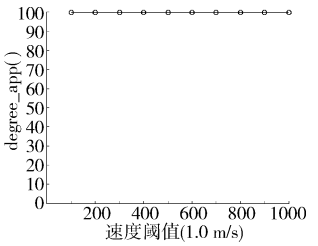

图 11 速度阈值固定，轨迹点数目变化时近似度的变化情况

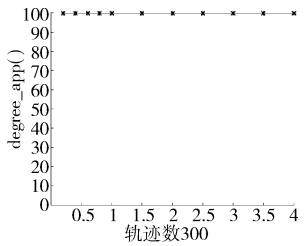

图 12 轨迹数量固定，速度阈值变化时近似度的变化情况

从以上的实验可以看出，轨迹数量 $|T|$ 、速度阈值 ε_v 对近似度没有影响，v-bfs-imp 和 v-dp 算法得到相同的简化轨迹。

4 结束语

本文提出了基于速度的轨迹简化新思路，为使简化轨迹规模最小，提出了四种不同的算法。通过实验比较得出，在速度阈值较小时，v-bfs-imp 算法相较于基于方向的或基于位置的轨迹简化算法在精确性上有较大优势，v-bfs-imp 算法在权衡简洁性和精确性上相较于基于方向的或基于位置的轨迹简化算法更好；然后比较 v-app 与 v-bfs-imp 算法得出，在速度阈值较小时近似度非常高。为了进一步缩短轨迹压缩时间，将考虑如何实现基于速度的轨迹压缩实时简化算法。

参考文献：

[1] Patel D, Sheng Chang, Hsu W, *et al.* Incorporating duration information for trajectory classification [C]//Proc of the 28th IEEE International Conference on Data Engineering. Washington DC: IEEE Computer Society, 2012: 1132-1143.

[2] Long Cheng, Wong R C W, Jagadish H V. Direction-preserving trajectory simplification [J]. Proceedings of the VLDB Endowment, 2013, 6(10): 949-960.

[3] Long Cheng, Wong R C W, Jagadish H V. Trajectory simplification: on minimizing the directionbased error [J]. Proceedings of the VLDB Endowment, 2014, 8(1): 49-60.

[4] Deng Ze, Han Wei, Wang Lizhe, *et al.* An efficient online direction-preserving compression approach for trajectory streaming data [J]. Future Generation Computer Systems, 2017, 68(5): 150-162.

[5] Douglas D H, Peucker T K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature [J]. Cartographica the International Journal for Geographic Information & Geovisualization, 1973, 10(2): 112-122.

[6] Meratnia N, De By R A. Spatiotemporal compression techniques for moving point objects [C]//Proc of the 9th International Conference on Extending Database Technology. Berlin: Springer, 2004: 765-782.

[7] Richter K F, Schmid F, Laube P. Semantic trajectory compression: representing urban movement in a nutshell [J]. Journal of Spatial Information Science, 2012, 4(4): 3-30.

[8] Cao Hu, Wolfson O. Nonmaterialized motion information in transport networks [C]//Proc of the 10th International Conference on Database Theory. Berlin: Springer, 2005: 173-188.

[9] Cao Hu, Wolfson O, Trajcevski G. Spatio-temporal data reduction with deterministic error bounds [J]. The VLDB Journal, 2006, 15(3): 211-228.

[10] Civilis A, Jensen C S, Pakalnis S. Techniques for efficient road-network-based tracking of moving objects [J]. IEEE Trans on Knowledge & Data Engineering, 2005, 17(5): 698-712.

[11] Potamias M, Patroumpas K, Sellis T. Sampling trajectory streams with spatiotemporal criteria [C]//Proc of the 18th International Conference on Scientific and Statistical Database Management. Washington DC: IEEE Computer Society, 2006: 275-284.

[12] Soroush E, Wu Kui, Pei Jian. Fast and quality-guaranteed data streaming in resource-constrained sensor networks [C]//Proc of the 9th ACM International Symposium on Mobile Ad hoc Networking and Computing. New York: ACM Press, 2008: 391-400.

[13] Trajcevski G, Cao Hu, Scheuermanny P, *et al.* On-line data reduction and the quality of history in moving objects databases [C]//Proc of the 5th ACM International Workshop on Data Engineering for Wireless and Mobile Access. New York: ACM Press, 2006: 19-26.

[14] Liu Guangwen, Iwai M, Sezaki K. An online method for trajectory simplification under uncertainty of GPS [J]. IPSJ Online Transactions, 2013, 6: 65-74.

[15] 王欣然，杨智应. 基于最小边界扇形的移动对象轨迹实时化简算法 [J]. 计算机应用，2014，34(8)：2409-2414.

[16] Ying J J C, Su J H. On velocity-preserving trajectory simplification [C]//Proc of the 8th Asian Conference on Intelligent Information and Database Systems. Berlin: Springer, 2016: 241-250.

[17] Cormen T H, Leiserson C E, Rivest R L, *et al.* 算法导论（原书第3版）[M]. 北京：机械工业出版社，2012.

[18] Jensen C S, Lahrmann H, Pakalnis S, *et al.* The infati data [J]. Computer Science, 2004, 29(3): 449-473.

[19] Lee J G, Han Jiawei, Whang K Y. Trajectory clustering: a partition-and-group framework [C]//Proc of ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 2007: 593-604.