文章编号:1674-2974(2017)02-0149-08 DOI:10.16339/j.cnki.hdxbzkb.2017.02.021

基于回溯的移动对象时序轨迹在线化简方法*

李想†,章登义
(武汉大学 计算机学院,湖北 武汉 430072)

摘 要:针对从移动端采集到的移动对象原始轨迹序列的化简,定义了一种回溯化简框架,通过线性预测来控制化简的时机,对当前时刻到回溯的历史轨迹的起始时刻之间的原始轨迹进行离线化简,化简采用时态距离作为误差度量方法.在回溯化简框架下,首先利用每次离线化简后新产生的化简点构建多个向量,通过向量计算出预测速度方向,旨在缩小预测方向与未来真实速度方向的差异;然后利用点集合存储有向无环图中必需访问边来降低最优线化简算法的时间复杂度.第1组实验表明,相对于直接使用最近两个位置点计算速度方向,抖动较为剧烈的原始轨迹在新的预测速度方向下的化简率更高,说明预测速度方向比切线速度方向更接近移动对象的未来运动方向;第2组实验表明,优化后离线化简算法的时间性能有所提高,说明减少边的访问量确实能够降低算法的时间开销.

关键词:移动对象数据库;轨迹;化简;回溯;线性预测;时态距离
中图分类号:TP301 **文献标志码**:A

Backtracking Based Method for On-line Trajectory Simplification of Moving Objects

LI Xiang†, ZHANG Dengyi
(School of Computer, Wuhan University, Wuhan 430072, China)

Abstract: For the simplification for the original trajectory sequence of the moving object collected from the mobile devices, this paper defined a kind of backtracking based simplification framework, which used the linear prediction and length of simplification queue to dominate the time of simplification, and simplified the original trajectory sequence between the present moment and starting time of a retrospective historical trajectory adopting the method of temporal distance as the error metric. In the backtracking based simplification framework, this paper first utilized the new reduced points to construct several vectors and predicted the velocity, which could narrow the gap between the prediction and actual velocity in the future. This paper then utilized the point sets to store the edges in the directed acyclic graph needed in the access to reduce time complexity of the algorithm. The first experiment shows that the reduction rate using the optimal velocity prediction is greater than that of the original velocity prediction with the high fluctuant trajectory data. It suggests that the predicted velocity is closer to the actual velocity in the future moving

* **收稿日期**:2016-05-20
基金项目:国家自然科学基金资助项目(60903035, 41001296),National Natural Science Foundation of China(60903035, 41001296);国家高技术研究发展计划(863计划)资助项目(2013AA12A301),National High Technology Research and Development Program of China (863 Program) (2013AA12A301)
作者简介:李想(1987—),男,湖北黄冈人,武汉大学博士研究生
†通讯联系人,E-mail: lixiang1987@whu.edu.cn

direction than that in the tangent direction. The second experiment shows that the time performances of the optimized simplification algorithm are improved. This study shows that the reduction of the visits of the edges can decrease the time overhead of the algorithm.

Key words: moving object database; trajectories; simplification; backtracking; linear prediction; temporal distance

如今 GPS 设备的普及使得基于位置的服务(Location Based Services,LBS)市场迅速增长,催生了大量的基于位置的应用.例如在车辆导航系统中,新的路线规划服务需要根据环境、车辆运行状态和道路交通规则[1],综合考虑多个行车成本(时间、距离、油耗等),通过收集和分析车辆的历史轨迹得到驾驶员的行车偏好,然后为其定制个性化的行车路线[2]. 而"未来 1 小时路况预测系统"将高速公路通行状况的历史数据、实时数据与路网状况结合,预测未来一小时内高速公路的拥堵状况.其次在商业动线设计中,通过分析大多数顾客在大型超市或者购物中心内的行进轨迹,找到不同类型顾客的兴趣点,对不同商品的摆放区域或不同商铺的位置进行精心设计,让顾客在商业体内部停留时间更久,在购物过程中尽可能经过更多有效区域,提升销售额.上述应用都需要使用大量的车辆、人的时序轨迹数据,但是,目前在使用轨迹数据中存在着三大问题,第一,通过网络传输大量的原始轨迹数据的代价十分高昂;第二,由于轨迹数据的低价值密度和存储设备限制,数据库无法保存全部轨迹数据[3];第三,不断增长的轨迹数据规模使得在其中发现有用的模式变得更加困难,因此对原始轨迹进行化简和压缩具有重要的研究价值和实际意义.

一些研究引入压缩和线化简算法对移动对象的历史轨迹进行化简,实际是一个折线近似过程,首先由获取到的移动对象的轨迹点之间的连线构成时空折线来表达移动对象的原始轨迹,然后找到一条新的轨迹使其包含的原始轨迹尽量少点且尽可能接近原始轨迹,这一类方法的压缩率高,但是时间复杂度高,不适用于实时化简.一些研究者提出基于推算定位的化简方法,这一类方法需要根据现有的轨迹对移动对象未来的运动速度矢量进行估计,实际是对原始轨迹进行分段离线化简的过程,速度矢量估计的精确度直接影响到化简的质量,离线化简的效率直接影响到化简的时效性.另一些研究者提出了基于区域过滤的方法,该算法不同于用一条折线来近似原始轨迹的方法,通过参考运动速度、方向和时间构建安全区域来对原始轨迹点进行过滤,安全区域的构建代价高.

本文的研究基于推测定位通过回溯部分历史轨迹点来预测移动对象在未来一段时间内的运动趋势,在实际位置点与预测位置点的距离超过化简精度阈值时,对当前时刻到回溯的历史轨迹的起始时刻之间的轨迹进行化简.针对上述化简过程,本文对两个步骤进行了优化,首先针对整体抖动较为剧烈的轨迹,改进速度矢量的预测方法,减少离线化简的次数,提升轨迹的化简率;然后对离线化简算法的实现过程进行优化,提升化简的时间性能.

1 相关工作

在线化简的含义是在通过移动设备不断获取移动对象的轨迹点时对移动端累积的轨迹进行化简,旨在减少轨迹点从移动设备传输到服务器过程中的通讯代价,同时降低存储轨迹的代价.目前,已有的研究中,移动对象轨迹在线化简方法是根据其是否需要累积部分历史轨迹来对后续的轨迹点进行化简,化简方法可分为部分在线化简和完全在线化简.

1.1 部分在线化简

部分在线化简方法的核心思想在于不断累积和抛弃原始轨迹点,将化简转化为对无数个轨迹段的离线化简.第 1 类为基于推算定位[4]方法,该类方法是根据当前轨迹点和预测速度来估计下一个轨迹点,当下一个轨迹点的实际位置与估计位置的距离超过化简误差时,将该轨迹点放入化简轨迹,具有代表性的有线性推测定位(Linear Dead Reckoning)、连接保持推测定位(Connection-Preserving Dead Reckoning)和 GRTS (Generic Remote Trajectory Simplification)[5].后者在前两者的基础之上将轨迹分为稳定部分、可变部分以及预测部分,通过预测部分推算预测轨迹点的位置,一旦预测轨迹点与实际轨迹点的距离大于化简误差,则对可变部分和预测部分的原始轨迹点进行离线化简,采用的离线化简方法主要是最优线化简方法 Opt(optimal line simplification)[6]、段启发式方法 Sec(segment heuristic)[7]以及道格拉斯-普客算法 DP(Douglas-Peucker)[8].基于推测定位方法的关键在于预测速度矢量的精度,其采用的速度矢量预测方法是直接使用预

测起始点和其之前一点的向量进行减法运算后除以两点的时间间隔得到,即近似轨迹在预测起始点的切线方向.该方法对于较为平稳的轨迹能够保证在较长一段时间内移动对象的预测位置与实际位置的距离不超过化简精度阈值,但是对于抖动剧烈的轨迹,该方法得到的速度矢量与移动对象未来运动方向的差距较大,使得化简过程中频繁触发离线化简,化简率降低.另一类是基于区域过滤方法,国内的一些研究者利用最小边界扇形[9-10]来近似简化移动对象的原始轨迹,在角度和距离两个层面上对简化误差进行控制,另一些研究者[11]通过引入速率和偏离阈值,构造分别适应于局部和总体速度的安全区域,实现轨迹简化.对于移动对象的运动速率和方向波动频繁的情况,基于区域过滤的方法需要频繁重新构建安全区域,计算代价较高.

1.2　完全在线化简

完全在线化简与部分在线化简的区别在于前者在化简过程不回溯历史轨迹点,只判断新到来的轨迹点是否是化简点.最常见的在线化简方法为均匀采样法(Uniform Sampling),即每间隔相同数量的原始轨迹点采样一个点放入化简轨迹,该方法简便快速,但是化简误差大.另一类经典的在线化简方法为 OPW-TR[7],其核心思想是首先以原始轨迹的第一个点为起始点开始维护一个窗口,不断将新的原始轨迹点放入窗口中,直到原始轨迹到起始点与窗口中某一点连线的同步欧氏距离超过阈值,此时将该点或该点之前的点放入化简轨迹中,并且以该点或该点之前的点为起始点重新维护窗口,该方法每接收到一个新的轨迹点后需要进行多次距离计算,时间复杂度较高.为了更好地保留轨迹的位置、时间和速度信息,有研究者提出了一种在线化简方法 SQUISH[12-13],使用优先队列过滤轨迹点实现化简,对于优先队列中除起始点外的任意一点,其优先级为该点到与该点相邻的前后两点之间连线的同步欧氏距离,一旦新进的点的到来导致优先队列溢出,则将优先级最低的点从队列中移除,并调整该点相邻两点的优先级.该方法与推测定位和道格拉斯普克算法相比在压缩率较小的情况下,化简误差较小,该化简方法的优先级计算方法不适用于高压缩率的化简.

2　回溯化简

本文以连续获取移动对象的传感轨迹为背景,将轨迹化简过程放置在客户端,客户端上的位置传感器不断感知新的位置,同时客户端通过化简框架对获取到的轨迹数据进行回溯化简,化简产生的化简点即时发送给移动对象数据库(Moving Object Databases,MOD),保证 MOD 接收的化简轨迹经过插值后与原始轨迹的误差小于给定的化简阈值.化简框架的符号说明如表 1 所示.

表 1　符号说明

Tab. 1　The symbol description

符号	说明
$S:\{s_1,s_2,\cdots,s_i,\cdots\}$	原始轨迹
$U:\{u_1,u_2,\cdots,u_j,\cdots\}$	化简轨迹
$s_i:(p,t):(x,y,t)$	原始轨迹点
$u_j:(p,t):(x,y,t)$	化简轨迹点
Δt	s_i 与 s_{i-1} 之间的时间间隔
$o:(p,t):(x,y,t)$	预测起始点
v_p	预测速度
s_a'	原始轨迹点在 s_a 对应的化简轨迹段上时态插值点
ε	化简精度阈值
$S_{\text{history}}:\{s_{i-m-1},\cdots,s_{i-1},s_i\}$	从当前时刻开始回溯的部分历史轨迹点
m	回溯历史轨迹点的总数量

我们对回溯化简有如下定义:

定义 1　回溯化简. 在当前时刻 t_c,原始轨迹序列为 $S:\{s_1,s_2,\cdots,s_i\}$,MOD 维护的化简轨迹序列 $U=\{u_1,u_2,\cdots,u_j\}\subseteq\{s_1,s_2,\cdots,s_i\}$,保留一部分历史轨迹序列 $S_{\text{history}}:\{s_{i-l},\cdots,s_{i-1},s_i\}$,时刻 $t_c+k\times\Delta t(k\in N^+)$,位置服务器接收到新的位置点 s_{i+k},一旦 s_{i+k} 与线性预测位置点的距离大于 ε 或者 S_{history} 的元素个数达到最大值,则对历史轨迹序列 $S_{\text{history}}:\{s_{i-l},\cdots,s_{i-1},s_i,s_{i+1},\cdots,s_{i+k}\}$ 则进行离线化简.化简结果为 ΔU,然后删除 MOD 维护的时间范围在 $[s_{i-l}.t,s_i.t)$ 内的化简轨迹点后得到集合 U',新的化简轨迹序列为 $U=U'\cup\Delta U$.

例 1　如图 1 所示,假定 S_{history} 中元素的个数 m 最大值为 8,在 $s_6.t$ 时刻,MOD 维护的化简轨迹序列 $U=\{u_1,u_2,u_3'\}$,保留的历史轨迹序列为 $S_{\text{history}}:\{s_4,s_5,s_6\}$,此时线性预测的起始点为 $u_3'=s_6$,根据 S_{history} 推算出的预测速度为 v_p,$s_8.t$ 时刻,轨迹点 s_8 与线性预测位置点的距离有 $|s_8.p-(u_3'.p+v_p\times(s_8.t-u_3'.t))|>\varepsilon$,则对历史轨迹序列 $S_{\text{history}}:\{s_4,s_5,s_6,s_7,s_8\}$ 进行线化简,化简结果为 $\Delta U=\{u_2,u_3\}$,因此 MOD 维护的新的化简轨迹序列 $U=\{u_1,u_2,u_3\}$,线性预测的起始点修改为 $u_3=s_8$,根据 S_{history} 重新推算预测速度 v_p,$s_{11}.t$ 时刻,S_{history} 中元素的总数量达到了 8,于是对 $S_{\text{history}}:\{s_4,s_5,s_6,s_7,s_8,s_9,s_{10},s_{11}\}$ 进行线化简,化简结果为 $\{u_2,u_3,u_4'\}$,对 MOD 维护的化简轨迹序列进行更新得到 $\{u_1,u_2,u_3,u_4'\}$,对 S_{history} 中时间范围在 $[u_2.t,u_3.t)$

的元素进行清理得到 $S_{\text{history}}:\{s_8,s_9,s_{10},s_{11}\}$，修改预测起始点为 $u_4'=s_{11}$，并重新推算预测速度 v_{p}，继续进行回溯化简.

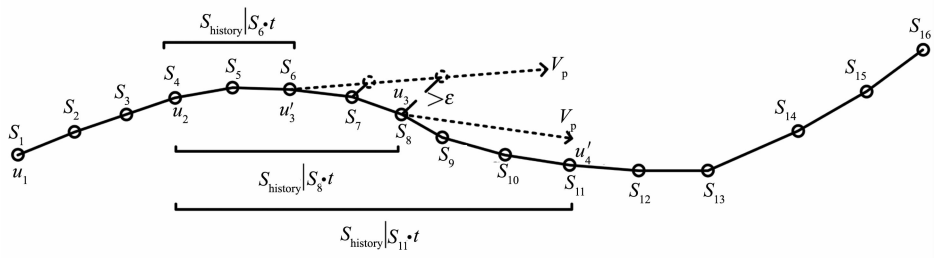


图 1 回溯化简示例

Fig. 1 Example of simplification based on backtracking

定理 1 当回溯的历史轨迹序列长度达到阈值而触发离线化简的次数忽略不计时，回溯化简过程中根据历史轨迹序列推算的预测速度方向越接近移动对象在未来的实际运动速度方向，则轨迹的化简率越高.

证明 以图 1 为例，假定 S_{history} 的长度阈值为 100，若 $s_8.t$ 时刻的另一个预测速度 v_{p}' 比 v_{p} 更接近移动对象的未来速度，使得 $|s_8.p-(u_3'.p+v_{\text{p}}'\times(s_8.t-u_3'.t))|<\varepsilon$，则服务器将继续接收新的位置点，假定直到 $s_{13}.t$ 时刻，$|s_{13}.p-(u_3'.p+v_{\text{p}}'\times(s_{13}.t-u_3'.t))|>\varepsilon$ 触发了离线化简，对轨迹段 $\{s_4,s_5,\cdots,s_{13}\}$ 而言，离线化简次数至少比原来减少了一次，扩展至整条轨迹，更接近实际速度方向的预测速度使得总体的离线化简次数减少，化简结果点数量减少，从而使得总体的化简率提高.

证毕.

上述化简过程中的距离度量方法定义如下：

定义 2 时态距离. 在二维空间中，给定原始轨迹 S 上的任意子轨迹段 $\{s_i,s_{i+1},\cdots,s_{i+l}\}$，其时态映射化简轨迹段为 $\overline{u_ju_{j+1}}$（$u_j.t\leqslant s_i.t<s_{i+l}.t\leqslant u_{j+1}.t$），子轨迹段上一点 $s_a\in\{s_i,s_{i+1},\cdots,_{i+l}\}$ 在 $\overline{u_ju_{j+1}}$ 上的时态插值点 s_a'（即 $s_a'.t=s_a.t$）的位置为：

$$s_a'.p=\frac{(s_a.t-u_j.t)}{u_{j+1}.t-u_j.t}(u_{j+1}.p-u_j.p)+u_j.p \tag{1}$$

s_a 到化简轨迹段 $\overline{u_ju_{j+1}}$ 的时态距离为 s_a 到 s_a' 的欧氏距离：

$$\text{dist}_{\text{temporal}}(s_a,\overline{u_ju_{j+1}})=\text{dist}_{\text{Euclidean}}(s_a,s_a')=\left|s_a.p-\frac{(s_a.t-u_j.t)}{u_{j+1}.t-u_j.t}(u_{j+1}.p-u_j.p)-u_j.p\right| \tag{2}$$

引理 1 原始轨迹 S 上任意子轨迹段 $\{s_i,s_{i+1},\cdots,s_{i+l}\}$ 在化简轨迹 U 上的时态映射轨迹段为 $\overline{u_ju_{j+1}}$，如图 2 所示原始轨迹点 $s_i,s_{i+1},\cdots,s_{i+l}$ 在轨迹段 $\overline{u_ju_{j+1}}$ 上的时态插值点分别为 $u_j,s_i',s_{i+1}',\cdots,s_{i+l}',u_{j+1}$，由式(2)得到原始轨迹段 $\{s_i,s_{i+1},\cdots,s_{i+l}\}$ 的化简误差之和为 $\sum_{a=i}^{i+l}\text{dist}_{\text{Euclidean}}(s_a,s_a')$，根据定积分的几何意义，当 Δt 趋近于零时，原始轨迹段 $\{s_i,s_{i+1},\cdots,s_{i+l}\}$ 的化简误差之和近似为直线 $\overline{u_ju_{j+1}}$ 与曲线 $\overset{\wedge}{s_is_{i+1}\cdots s_{i+l}}$ 围成的几何图形的面积.

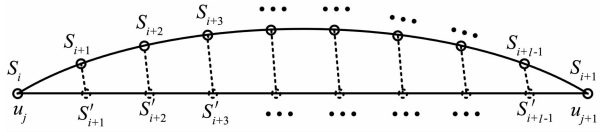


图 2 原始轨迹映射到化简轨迹示意图

Fig. 2 Diagram for the original trajectory mapping to the simplified trajectory

由引理 1 可知，对于任意一段原始轨迹，若其对应化简轨迹的轨迹点越多，则该段原始轨迹的化简误差越小.

定义 3 平均化简误差. 在二维空间中，原始轨迹序列为 $S:\{s_1,s_2,\cdots,s_n\}$，化简轨迹序列 $U=\{u_1,u_2,\cdots,u_l\}\subseteq\{s_1,s_2,\cdots,s_n\}$ 且 U 是按照化简精度阈值 ε 对 S 进行化简得到的，根据定义 1，对于 S 上任意一轨迹点 s_1 的时态距离为 $\text{dist}_{\text{Euclidean}}(s_i,s_i')$，其中 s_i' 为 s_i 在化简轨迹段 $\overline{u_ju_{j+1}}$ 上的时态插值点，这个时态距离又称作轨迹点 s_i 的化简误差，那么原始轨迹序列 S 的平均化简误差为：

$$\frac{1}{n}\sum_{i=1}^{n}\text{dist}_{\text{Euclidean}}(s_i,s_i')=\frac{1}{n}\sum_{i=1}^{n}|s_i.p-s_i'.p| \tag{3}$$

定义 4 轨迹抖动系数 J. 原始轨迹序列 $S:\{s_1,s_2,\cdots,s_n\}$ 中任意连续三点 s_{i-1},s_i,s_{i+1}，组成的两个向量 $\boldsymbol{s}_{i-1}\boldsymbol{s}_i$ 和 $\boldsymbol{s}_i\boldsymbol{s}_{i+1}$ 的夹角余弦值为 s_i 的抖动系数，描述轨迹在 s_i 的抖动程度，则轨迹的抖动系数 J 为轨迹中所有连续三点组成的向量的夹角余弦值的平均值：

$$J=\frac{1}{n-2}\sum_{i=2}^{n-1}\cos\langle\boldsymbol{s}_{i-1}\boldsymbol{s}_i,\boldsymbol{s}_i\boldsymbol{s}_{i+1}\rangle \tag{4}$$

本文将轨迹抖动系数 $J=\sqrt{2}/2$ 作为轨迹抖动

剧烈与否的分界线,抖动系数小于$\sqrt{2}/2$的轨迹为抖动轨迹,即上述两个向量的角度差为 45°～180°,抖动系数大于$\sqrt{2}/2$的轨迹为平稳轨迹,即上述两个向量的角度差为 0～45°,抖动系数越小则抖动越剧烈.

3　优化策略

本节将详细描述针对回溯化简框架下的在线化简算法的两个优化策略.

3.1　速度预测模型的优化

由于移动对象的运动具有惯性,而化简轨迹序列的更新反映了移动对象的运动方向发生了显著变化,因此我们通过 MOD 服务器接收到的化简点(包括后来被替换掉的)对速度矢量进行预测.

情形Ⅰ　当前离线化简后新产生的化简点数量大于等于 2 时,说明移动对象在最近的几个化简点所覆盖的时间区域内发生较为显著的运动方向变化,此时通过 MOD 服务器最近接收到的 4 个化简点 $u_{i-3},u_{i-2},u_{i-1},u_i$ 构建如下 3 个向量:

$$\boldsymbol{v}_1=\frac{u_{i-2}.p-u_{i-3}.p}{u_{i-2}.t-u_{i-3}.t},\boldsymbol{v}_2=\frac{u_{i-1}.p-u_{i-2}.p}{u_{i-1}.t-u_{i-2}.t},$$

$$\boldsymbol{v}_3=\frac{u_i.p-u_{i-1}.p}{u_i.t-u_{i-1}.t} \tag{5}$$

判断第 1 个向量到第 2 个向量的变化方向是否与第 2 个向量到第 3 个向量的变化方向是否同时向下或向上,若为肯定,则如图 3(a)和(b)所示,此时轨迹呈现上扬趋势或下降趋势,预测速度的方向为:

$$\boldsymbol{v}_{\mathrm{p}}=2\,\boldsymbol{v}_3-\boldsymbol{v}_2=2\,\frac{u_i.p-u_{i-1}.p}{u_i.t-u_{i-1}.t}-\frac{u_{i-1}.p-u_{i-2}.p}{u_{i-1}.t-u_{i-2}.t} \tag{6}$$

预测速度的大小为 u_{i-1} 和 u_i 之间的平均速率,若不是,则如图 3(c)和(d)所示,此时轨迹呈现波浪变化,预测速度的方向为:

$$\boldsymbol{v}_{\mathrm{p}}=\frac{1}{2}(\boldsymbol{v}_2+\boldsymbol{v}_3)=\frac{1}{2}(\frac{u_{i-1}.p-u_{i-2}.p}{u_{i-1}.t-u_{i-2}.t}+\frac{u_i.p-u_{i-1}.p}{u_i.t-u_{i-1}.t}) \tag{7}$$

预测速度的大小为 u_{i-1} 和 u_i 之间的平均速率.

情形Ⅱ　当前离线化简后新产生的化简点数量小于 2 时,说明移动对象在历史轨迹序列覆盖的时间范围内运动方向的变化不显著,此时通过 MOD 最近接收到的两个化简点 u_{i-1} 和 u_i 所覆盖的时间区域 $[u_{i-1}.t,u_i.t]$ 内的原始轨迹序列的三等分点和 u_{i-1},u_i 构建与情形Ⅰ类似的 3 个向量,然后采用与情形Ⅰ相同的方式计算预测速度方向,预测速度的大小为 u_{i-1} 和 u_i 之间的平均速率.

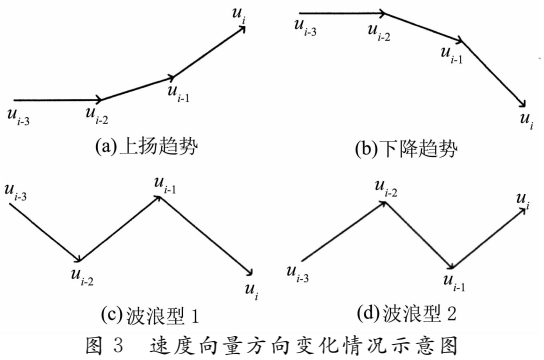


图 3　速度向量方向变化情况示意图

Fig. 3　Diagram for direction changes of velocity vector

3.2　离线化简算法的优化

本文针对推测定位中离线化简通常采用的最优化线化简算法的实现过程进行优化,原算法的化简过程分为 3 步:第 1 步,根据回溯的历史轨迹点构建有向无环图;第 2 步,求有向无环图中起点 s_1 到终点 s_m 的最短路径;第 3 步,返回最短路径为化简结果.

本文在两个步骤中采用优化措施降低算法的时间复杂度的同时降低空间复杂度,第一是针对最优化线化简算法在第 1 步中构建有向无环图时需要构建所有可能存在的边,从而导致时间复杂度高的特点,仅仅构建广度优先搜索的过程中需要访问到的边,减少时态距离的计算量,且通过集合来代替邻接表或邻接矩阵存储边.具体做法是在广度优先搜索过程中将所有轨迹点进行分类,分类的原则是按照它们到起始点的距离进行判断,距离为 0 的点是起始点 s_1 本身,因此将 s_1 单独作为一个集合,然后访问到 s_1 距离为 1 的所有点,把它们放入一个集合.再访问到这个集合中的每一个点距离为 1 且没有被放入任何一个集合的点,将这些点放入到一个新的集合中,然后对新的集合进行上述相同的处理,直到终点 s_m 放入某个集合中.第二是每当访问到一个与当前集合 H_{c-1} 中的点之间存在边的点,就直接判断其与终点 s_m 之间是否有边,若存在边,则最短路径已经得到,避免扫描一部分不相关的点.具体算法如下:

算法 1.　Opt⁺算法.

输入:回溯的历史轨迹序列 S_{history}:$\{s_1,s_2,\cdots,s_{m-1},s_m\}$,化简精度阈值 ε

输出:化简结果 u

1. $H_0=\{s_1\}$; $B=\{s_2,\cdots,s_{m-1},s_m\}$; $c=1$;
2. WHILE ($B\neq\varnothing$) DO {
3.　　$H_c\leftarrow\varnothing$;
4. FOR EACH s_i in H_{c-1} and EACH s_j in B DO {//逆序访问 H_{c-1} 和 U 中的元素
5. cond = TRUE;

6. FOR EACH s_k in S_{history} WHERE $s_i.t < s_k.t < s_j.t$ DO {//判断 s_i 到 s_j 的边是否存在
7. IF ($\text{dist}_{\text{temporal}}(s_k, \overline{s_i s_j}) \leqslant \varepsilon$) THEN
8. cond = cond & TRUE;
9. ELSE
10. cond = cond &FALSE;
11. BREAK;
12. }
13. IF (cond) THEN // s_i 到 s_j 的边存在
14. IF ($s_j == s_m$) THEN
15. RETURN; //已找到最短路径
16. con = TRUE;
17. FOR EACH s_q in S_{history} WHERE $s_j.t < s_q.t < s_m.t$ DO { //判断 s_j 到 s_m 是否存在边
18. IF ($\text{dist}_{\text{temporal}}(s_q, \overline{s_j s_m}) \leqslant \varepsilon$) THEN
19. con = con & TRUE;
20. ELSE
21. con = con & FALSE;BREAK;
22. }
23. IF (con) THEN // s_j 到 s_m 的边存在
24. RETURN; //已找到最短路径
25. $B \leftarrow B \backslash \{s_j\}$;
26. $H_c \leftarrow H_c \cup \{s_j\}$;
27. }
28. $c = c + 1$;
29. }

该算法维护若干集合,其中 $H_c(c = 0,1,2,\cdots)$ 保存有向无环图中起始点到其路径已知的点,c 表示从起始点到 H_c 中的点的路径长度,而集合 B 用来保存有向无环图中起始点到其路径未知的点.因为 s_1 作为最短路径的起点,行 1 首先用轨迹序列 S_{history} 中的第一个轨迹点 s_1 初始化 H_0 ,B 初始化为轨迹序列 S_{history} 中的除第一点外的其他轨迹点,接着逆序访问 H_{c-1} 和 B 中的轨迹点;行 5－10 通过计算 S_{history} 中 s_i 到 s_j 之间的每一个轨迹点 s_k 到线段 $\overline{s_i s_j}$ 的时态距离 $\text{dist}_{\text{temporal}}(s_k, \overline{s_i s_j})$,并且判断此距离是否小于等于化简精度阈值 ε ;如行 13,如果 S_{history} 中 s_i 到 s_j 之间所有的轨迹点的 $\text{dist}_{\text{temporal}}(s_k, \overline{s_i s_j})$ 小于化简精度阈值 ε ,即有向无环图中边 $\overrightarrow{s_i s_j}$ 存在,则在误差范围内可以用线段 $\overline{s_i s_j}$ 近似 S_{history} 中 s_i 到 s_j 的轨迹;在此基础上行 14 判断 s_j 是否是 S_{history} 最后一个轨迹点 s_m ,若 s_j 等于 s_m ,则最短路径已经找到;最短路径存入 u 中,算法结束,否则继续执行下一步;行 16－24 对于当前的轨迹点 s_j ,判断其与 S_{history} 的终点 s_m 是否存在边,若存在则最短路径已经找到,算法结束,否则继续执行下一步,于是将 s_j 从 B 中删除,添加至 H_c ,继续逆序访问 H_{c-1} 和 B 中的轨迹点,直至 H_{c-1} 和 B 中的轨迹点都访问完毕;行 28 将 c 自加 1,重复 4－27 行的过程,直至集合 B 为空(行 2).

Opt 算法的步骤 1 在构建有向无环图中,顶点个数为回溯的历史轨迹点数量 m ,考虑图中任意两个顶点 s_i 和 s_j ,若 $s_i.t < s_j.t$,则判断两点之间是否存在边,还需要判断 s_i 和 s_j 之间所有的轨迹点到线段 $\overline{s_i s_j}$ 的时态距离是否小于等于化简精度阈值 ε ,时间复杂度为 $O(m)$,这样两点组合的数量有 $\sum_1^m k = \frac{m(m-1)}{2}$ 个,因此该步骤的时间复杂度为 $O(m^3)$.步骤 2 中求两点之间的最短路径,假设采用最基本的图的最短路径算法 Dijkstra 算法,时间复杂度为 $O(m^2)$.步骤 3 中返回化简结果的时间复杂度为 $O(m)$.因此步骤 1 决定了整个算法的时间复杂度为 $O(m^3)$,而使用邻接表或邻近矩阵维护一个有向无环图的空间复杂度为 $O(m^2)$.

改进后的 Opt^+ 算法在采取了两种优化措施之后,避免了构建一些不会访问到的边的计算和一些顶点的访问,若化简结果中仅有起始点 s_1 和终止点 s_m ,则此时能够达到最好的时间复杂度 $O(m)$.由于 Opt^+ 算法通过集合关系来表示在有向无环图中顶点之间边的关系且只需要保存每个顶点,因此其空间复杂度为 $O(m)$.

4 实验结果

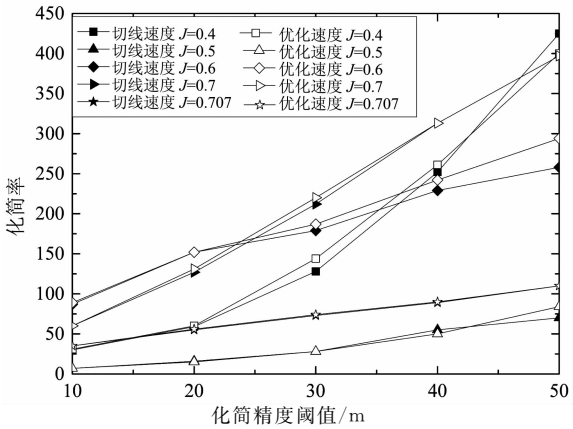
为了验证优化策略对轨迹化简性能的提升,本文在回溯化简框架下的在线化简方法的基础上用 C++实现了上述优化策略.实验的硬件平台为:Intel® Core™ i7-3630QM 2.4 GHz CPU, 16 G 内存和 750 GB 硬盘;软件环境为 Win7 操作系统和 VS2008 编译系统.实验数据通过 OpenStreetMap (OpenStreetMap. http://www. openstreetmap. org/)网站中的真实轨迹数据集提取获得.在实验中,化简率定义为原始轨迹的点数量与化简轨迹的点数量之比,化简误差定义为化简轨迹经由原始轨迹参考插值后的时态距离的平均值.

4.1 化简率提升验证实验

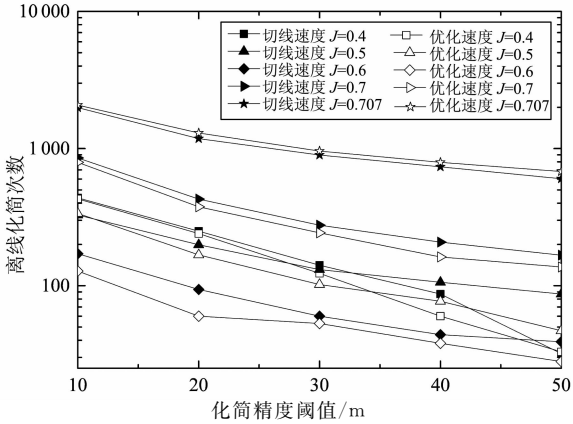
实验选取抖动系数分别为 $J = 0.4$, $J = 0.5$, $J = 0.6$, $J = 0.7$ 和 $J = 0.707$ 等 5 类抖动程度依次减弱的轨迹,在化简精度阈值为 10～50 m 变化过程中,分别使用切线速度和本文的优化速度时的化简率.

图 4(a)给出了不同化简精度阈值下 5 类抖动轨迹的化简率比较图,可以看出在抖动系数较小,即轨迹抖动非常剧烈时,本文的优化速度下在化简精度为 20～40 m 时与切线速度下相比具有一定的优势,但是随着抖动系数的增大,这种优势逐渐缩小至

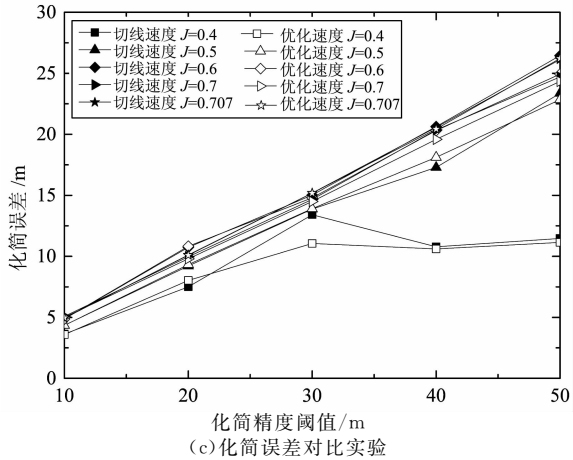
零.图 4(b)给出了不同化简精度阈值下 5 类抖动轨迹的离线化简次数情况,可以看出轨迹抖动系数较小时,本文的优化速度下的离线化简次数通常小于切线速度下的离线化简次数,说明本文的优化速度比切线速度更接近对象未来的速度.图 4(c)给出了与 4(a)相应条件下的化简误差,可以看出在保证化简误差小于化简精度阈值的前提下,本文的优化速度下的化简误差与切线速度下相比十分接近.



(a)化简率对比实验



(b)离线化简次数实验



(c)化简误差对比实验

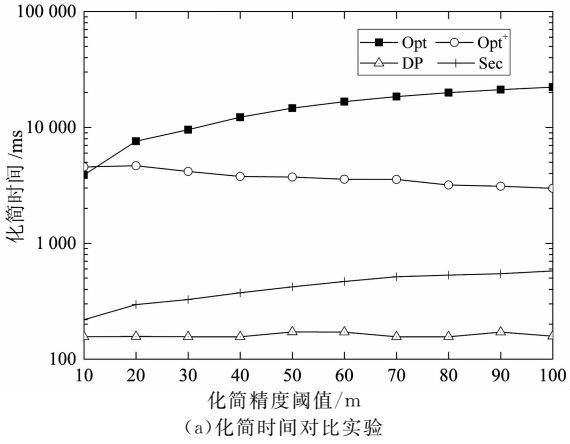
图 4　不同抖动程度轨迹的化简率、离线化简次数和化简误差对比实验

Fig. 4　Contrast of reduction rates, times of offline simplification and simplification error for different jittering trajectory

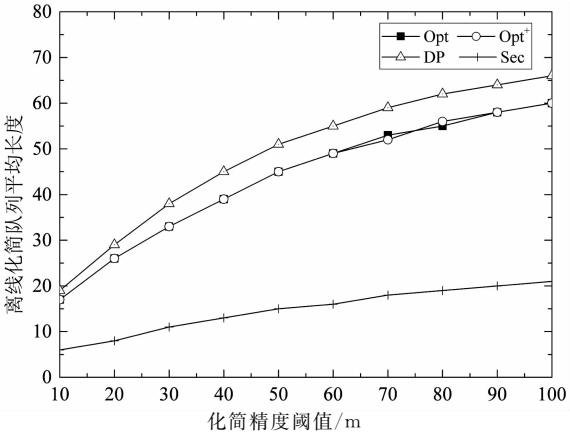
4.2　化简时间性能提升验证实验

实验选取轨迹点数量为 100 万的轨迹,在化简精度阈值由 10～100 m 的变化过程中,比较回溯化简框架下 Opt$^+$算法与 Opt[6]算法的化简时间性能,同时将 DP[8]算法和 Sec[7]算法作为参照.

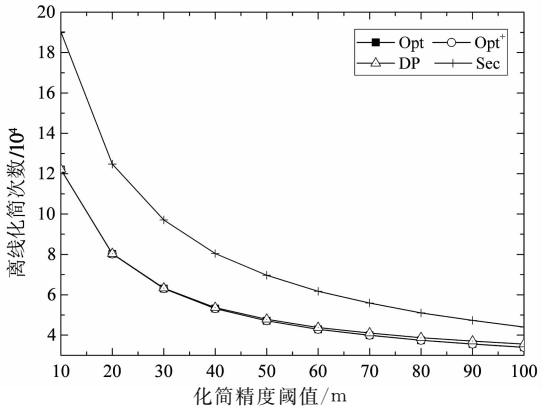
图 5(a)给出了 100 万轨迹下化简精度阈值变化过程中化简所消耗时间的情况.4 种算法中,Opt 算法的化简时间随着化简精度阈值的变大而显著增长,Opt$^+$算法的化简时间随着化简精度阈值的变大而略有减少,最终趋于平稳,DP 算法受化简精度阈值变化的影响较小,Sec 算法的化简时间随着化简精度阈值的变大而略有增长.化简精度阈值超过 10 m 时,Opt$^+$算法的化简时间性能优于 Opt 算法.在化简精度阈值为 20～100 m 时,Opt$^+$算法所消耗的时间约为 Opt 算法的 10%～70%,这是由于化简精度阈值的增大导致化简队列平均长度变大,如图 5(b)所示,使得 Opt 算法中构建有向无环图所消耗的时间显著增加,同时总体的化简次数减少,如图 5(c)所示,使得变长段 Opt$^+$算法消耗的时间降低到一定范围.



(a)化简时间对比实验



(b)离线化简队列平均长度对比实验



(c)离线化简次数对比实验

图 5 100 万轨迹下化简时间、离线化简队列平均长度、离线化简次数对比实验

Fig. 5 Contrast of time, average length of offline simplification queue and times of offline simplification on one million trajectory points

5 结 论

本文在基于推测定位的时序轨迹回溯化简框架下,分别对化简过程中的速度预测模型和离线化简算法进行优化.实验结果表明,对于抖动剧烈的轨迹化简,优化后的速度下与原有切线速度下相比在化简率上有一定的优势,而优化后的离线化简算法与原方法相比在时间性能上有较大的提升.当前的工作主要有两点不足,一是仍然在欧氏空间中讨论移动对象时序轨迹的化简问题,而目前一部分移动对象的轨迹数据都是在道路网络下产生的,本研究没有考虑路网约束对化简的影响;二是化简没有将轨迹点的空间维度与时间维度相结合,仍然只是对空间维度进行化简,而把时间维仅作为空间维的一个附加维度.因此未来的工作将重点考察道路网络约束下的轨迹特征,着眼于基于道路网络的移动对象时序轨迹的时空化简方法研究.

参考文献

[1] 吴乙万, 黄智. 基于动态虚拟障碍物的智能车辆局部路径规划方法[J]. 湖南大学学报:自然科学版, 2013,40(1):33-37.
WU Yiwan, HUANG Zhi. Dynamic virtual obstacle based local path planning for intelligent vehicle[J]. Journal of Hunan University: Natural Sciences, 2013, 40(1): 33-37. (In Chinese)

[2] DAI J, YANG B, GUO C, *et al*. Personalized route recommendation using big trajectory data[C]// International Conference on Data Engineering. Seoul: IEEE Computer Society, 2015:543-554.

[3] 许佳捷, 郑凯, 池明旻,等. 轨迹大数据:数据、应用与技术现状[J]. 通信学报, 2015, 36(12):97-105.
XU Jiajie, ZHENG Kai, CHI Mingmin, *et al*. Trajectory big data: data, applications and techniques[J]. Journal on Communications, 2015, 36(12):97-105. (In Chinese)

[4] BAIER P, DÜRR F, ROTHERMEL K. Opportunistic position update protocols for mobile devices[C]// Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing. New York: ACM SIGMOD Record, 2013:787-796.

[5] LANGE R, DÜRR F, ROTHERMEL K. Efficient real-time trajectory tracking[J]. The International Journal on Very Large Data Bases, 2011, 20(5):671-694.

[6] KATSIKOULI P, SARKAR R, GAO J. Persistence based online signal and trajectory simplification for mobile devices [C]// Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. New York: ACM SIGMOD Record, 2014:371-380.

[7] MERATNIA N, ROLF A. Spatio-temporal compression techniques for moving point objects[C]// Proceedings of the International Conference on Extending Database Technology. Greece: Berlin Springer, 2004: 765-782.

[8] 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 and Geovisualization, 1973, 10(2):112-122.

[9] 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.
WANG Xinran, YANG Zhiying. Real-time trajectory simplification algorithm of moving objects based on minimum bounding sector[J]. Journal of Computer Applications, 2014, 34(8): 2409-2414. (In Chinese)

[10] 王欣然, 杨智应. 基于 PLAZA 的移动对象轨迹实时化简方法[J]. 计算机应用研究, 2014, 31(5): 1315-1319.
WANG Xinran, YANG Zhiying. Method of real-time trajectory simplification of moving object based on PLAZA[J]. Application Research of Computer, 2014, 31(5):1315-1319. (In Chinese)

[11] 李文海, 程志光, 文卫东, 等. 基于自适应安全区域的轨迹实时化简方法[J]. 计算机学报, 2014, 37(9):1923-1935.
LI Wenhai, CHENG Zhiguang, WEN Weidong, *et al*. A safe-region based adaptive method for real-time trajectory simplification[J]. Chinese Journal of Computer, 2014, 37(9):1923-1935. (In Chinese)

[12] MUCKELL J, HWANG J H, PATIL V, *et al*. SQUISH: an online approach for GPS trajectory compression[C]// Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications. Washington DC: ACM SIGMOD Record, 2011, 13:1-8.

[13] MUCKELL J, OLSEN P W, HWANG J H, *et al*. Compression of trajectory data: a comprehensive evaluation and new approach[J]. An International Journal on Advances of Computer Science for Geographic Information Systems, 2013, 18(3):435-460.