文章编号:1671-4598(2016)05-0199-03 DOI:10.16526/j.cnki.11—4762/tp.2016.05.056 中图分类号:TP3 文献标识码:A

基于三次多项式拟合三角函数的地理空间距离计算算法

李雪佳[1]，封红旗[1]，梅 宇[1]，赵玉宁[2]
(1. 常州大学 信息科学与工程学院，江苏 常州 213000；2. 江苏省南京市鼓楼医院 信息科，南京 210000)

摘要：移动互联网技术的快速发展，现有的APP提供了非常人性化的操作，用户可以进行对商家的筛选，APP默认的选项“智能排序”，“离我最近”；这两个选择项，系统都会去计算当前用户的位置和各个商家之间的距离；这种大量计算距离的场景十分消耗资源，按照现在的统计数据100 W条数据时候，延迟将会达到140 ms；且随着数据的增长，服务器的性能堪忧；当前计算空间距离的方法是使用Lucene算法进行计算，但是开销比较大，不利于用户一眼，文章提出了一种基于三次多项式拟合三角函数的优化算法对空间距离进行计算，文章从距离计算的准确性和快速性两个方面与原有方法进行比较；最后总结出优化算法的优势之处。

关键词：余弦函数；空间距离；三次多项式；准确性

Geographical Space Distance Calculation Algorithm Based on Three Time Polynomial Fitting Trigonometric Function

Li Xuejia[1], Feng Hongqi[1], Mei Yu[1], Zhao Yuning[2]
(1. School of information science & Engineering, Changzhou 213000, China;
2. Nanjing Gulou Hospital, Nanjing 210000, China)

Abstract: With the rapid development of mobile Internet technology, the existing APP provides a very user-friendly operation, the user can make a selection of businesses, APP default option “smart sort”, “from me recently.” These two options, the system will calculate the current position of the user and the distance between the various businesses. This is a large number of computing distance of the scene is very consumption of resources, in accordance with the current statistical data 100 W data when the delay will reach 140 ms. And with the growth of data, the performance of the server is worrying. At present, the method of calculating the space distance is to use Lucene algorithm to calculate, but the cost is relatively large, and is not conducive to the user one eye, the paper presents a method based on the three polynomial fitting trigonometric function optimization algorithm to calculate the space distance, the accuracy and speed of the distance from the two aspects of the original method. Finally, the advantages of the optimization algorithm are summarized

Keywords: cosine function; space distance; three degree polynomial; accuracy

0 引言

互联网技术发展越来越快，以团购为代表OTO业务的移动终端受到用户的喜爱，OTO平台最大的特点是基于本地化的服务，在各大OTO平台中，都存在一个非常重要功能，商家的筛选，按距离、智能排序（智能排序中距离因素也是比较重要的因素）。其中对商家的筛选，主要依靠的是距离的因素。如果获取商家与用户之间的距离，是解决这一问题的关键所在。这里涉及到了地理空间距离计算的问题。在该算法中，其中的原理是比较简单的，最关键之处在于对算法的优化，在用户进行筛选时候系统可以在最快的时间给出最准确的答案。这样用户的体验性好。

本文提出的基于三次多项式拟合三角函数的地理空间距离计算算法，通过使用三次多项式来抵消原公式中三角函数的计算，使得算法在效率上提高很多。但是本文提出的算法是在同城范围内，主要是由于论文中约定了在同城范围内，经纬度可以看作是相互垂直的直线。最后文章给出了初始方法、基于坐标转化方法、以及文中所提出的方法之间相互比较。

1.1 原理

地理空间距离计算方法比较多，现在主要使用的是以下两种模型：一个是球面模型，该模型将地球看做一个标准的球体，球面上任意两点之间的距离表示了圆最大的弧长。这个方法使用的比较广泛。另一个是椭球模型，这个模型更加贴近地球的真实环境，但是椭圆的计算方法非常复杂，但是在实际的工程中对精度的要求没有那么高。

现在主要介绍基于球面模型的地理空间距离计算公式，

假设地球上有两个点A（ja，wa），B（jb，wb），其中ja，jb表示A、B两点的精度，wa，wb表示A、B两点的维度。所以A、B两点的距离就是AB之间的弧长，所以AB弧长$=R*$角AOB（注：角AOB是A跟B的夹角，O是地球的球心，R是地球半径，约为6367000米）。为了求解角AOB的角度，可以先对AOB角的最大边AB的长度进行求解，然后根据余弦定理算出其夹角。

1）由经纬度，地球半径R，将A、B两点的经纬度转化

收稿日期:2015-11-09； **修回日期**:2015-12-15。
基金项目:国家自然科学基金资助项目(61103172)。
作者简介:李雪佳(1990-)，湖北荆州人，硕士研究生，主要从事数据挖掘、地理信息系统。
封红旗(1976-)，江苏常州人，博士，副教授，主要从事数据挖掘、地理信息系统方向的研究。

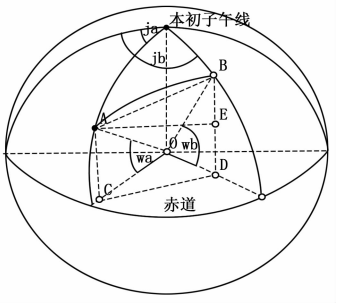

图 1 球面模式示意图

成球体的三维坐标：

$$A=\begin{pmatrix} Xa=Rcos(wa)cos(ja)\\ Ya=Rcos(wa)sin(ja)\\ Za=Rsin(wa)\end{pmatrix}$$

$$B=\begin{pmatrix} Xb=Rcos(wb)cos(jb)\\ Yb=Rcos(wb)sin(jb)\\ Zb=Rsin(wb)\end{pmatrix}$$

2）求出 AB 之间的长度：

$$AB^2=(Xa-Xb)^2+(Ya-Yb)^2+(Za-Zb)^2=\ldots=2R^2(1-\cos(wa)\cos(wb)\cos(jb-ja)-\sin(wa)\sin(wb))$$

3）求出角 AOB：

$$AB^2=AO^2+BO^2-2AO\times BOCos(\Delta AOB)$$

$$\cos(\Delta AOB)=\frac{AB^2-AO^2-BO^2}{-2AO*BO}=1-\frac{AB^2}{2R^2}$$

4）AB 的弧长：

$$AB=Rarc\cos(\cos(wa)\cos(wb)-\cos(ja-jb)+\sin(wa)\sin(wb))$$

以上便是空间距离计算的原理。在这里只是简单对如果进行论述。

1.2 使用 Lucene 进行距离的计算

现有的 APP 后台基本使用 Lucene 来对商家进行筛选，Lucene 使用了 spatial4j 工具包进行对距离的计算，在 spatial4j 工具包中同时提供了基于多种球面的地理空间距离公式，上面描述的就是其中的一种。在这个工具包中有一中常用的工具 Haversine 公式，这也是 spatial4j 工具包中默认的公式。引入这个公式的好处：前面的余弦公式中 cos（$jb-ja$），当系统的浮点运算精度比较低，在计算 AB 两点距离比较近的时候产生的误差比较大，Haversine 方法中通过某种变化的方式对 cos（$jb-ja$）进行消除。这样解决了近距离误差较大的问题。

（1）Haversine 公式调用：

```
public static double distHaversineRAD (double lat1, double lon1, double lat2, double lon2) {
    double hsinX = Math. sin ( (lon1 - lon2) * 0.5);
    double hsinY = Math. sin ( (lat1 - lat2) * 0.5)
    double h = hsinY * hsinY +
    (Math. cos (lat1) * Math. cos (lat2) * hsinX * hsinX);
    return 2 * Math. atan2 (Math. sqrt (h), Math. sqrt (1 - h)) * 6367000;
    }
```

（2）Haversine 公式性能：

本文的测试环境处理器：2.9 GHz Intel Core i7，内存为 8 GB 1 600 MHz DDR3，操作系统为 OSX10.8.3，实验在单线程环境下运行。测试的数据等级 5 w，10 w，100 w。测试结果如下图，在下图中可以发现当数据只有 5 w 个时候，计算一遍距离 7 ms，但当数据增加到 100 W 的时候，系统只是进行计算距离一项就需要 144 ms，还不包括筛选过程中其他的运行项目，性能十分堪忧。

表 1 Lucene 性能分析

数量级别	时间/ms
5w	7
10w	14
100w	144

2 优化方案

为了提高计算的效率，提供更好的用户体验，在实验过程中进行了抓栈实验，在实现过程中发现了消耗 cpu 较多线程的地方大多数存在计算执行距离公式中的三角函数。正是这个给我们提出了一些优化方案，去消除或者消减三角函数。

2.1 三次多项式拟合三角函数的地理空间距离计算算法

1）思路：

业务场景大多是仅限于在同一个城市进行距离的计算，也就是说两点之间的直线距离一般不会超过 300 KM，针对于范围比较小，则可以近似的认为经纬度是垂直的，要求 A（116.8，39.78）和 B（116.9，39.68）两点的距离，我们可以先求出南北方向距离 AM，然后求出东西方向距离 BM，最后求矩形对角线距离，即 sqrt（AMAM + BMBM）。

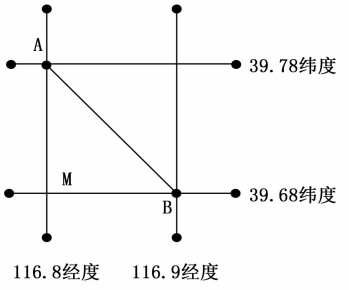

图 2 近距离下经纬度示意图

南北方向 AM = R 纬度差 Math. PI/180.0；

东西方向 BM = R 经度差 Cos＜当地纬度数 * Math. PI/180.0＞

这种方式仅仅需要计算一次 cos 函数。

```
public static double distanceSimplify (double lat1, double lng1, double lat2, double lng2, double[] a) {
    double dx = lng1 - lng2; // 经度差值
    double dy = lat1 - lat2; // 纬度差值
    double b = (lat1 + lat2) / 2.0; // 平均纬度
    double Lx = toRadians(dx) * 6367000.0 * Math. cos(toRadians (b)); // 东西距离
    double Ly = 6367000.0 * toRadians(dy); // 南北距离
    return Math. sqrt(Lx * Lx + Ly * Ly); // 用平面的矩形对角距离公式计算总距离
```

```
}
}
```

2）有效性验证：

我们首先检验这种简化是否能满足我们应用的精度，如果精度较差将不能用于实际生产环境。

我们的方法叫 distanceSimplify，lucene 的方法叫 distHaversineRAD。下表是在不同尺度下两个方法的相差情况。

可以看到两者在百米、千米尺度上几乎没有差别，在万米尺度上也仅有分米的差别，此外由于我们的业务是在一个城市范围内进行筛选排序，所以我们选择了北京左下角和右上角两点进行比较，两点相距有 260 多千米，两个方法差别 17 m。从精度上看该优化方法能满足我们应用需求。

表 2　改进算法的有效性

测试点对	distanceSimplify（米）	distHaversineRAD（米）	差别（米）
(39.941,116.45) (39.94，116.451)	140.0285167225230	140.02851671981400	0.0
(39.96，116.45) (39.94，116.40)	4804.421262839180	4804.421153907680	0.0
(39.96，116.45) (39.94，117.30)	72444.81551882200	72444.54071519510	0.3
(39.26，115.25) (41.04，117.30)	263525.6167839860	263508.55921886700	17.1

3）性能测试：

表 3　改进性能分析

数量级别	时间/ms
5w	0.5
10w	1.1
100w	11

2.2　进一步优化方案

在上面的过程中，可以看出进行了一次的 cos 三角函数的运算，如何将三角函数完全的进行消除，这样便于进一步提高优化的效率。

在这里主要使用利用多项式来拟合三角函数，这是高等代数里面高斯不等式的相关知识。在一定的范围内，当然等式的次数越高，获得到的准确率将越来越精确。在这里选择了三次多项式进行拟合。

这里对原有算法进行改进，中国的纬度范围在 10～60 之间，即我们将此区间离散成 Length 份作为我们的训练集。

```
public static double[] trainPolyFit(int degree, int Length){
PolynomialCurveFitter polynomialCurveFitter = PolynomialCurveFitter.create(degree);
double minLat = 10.0; //中国最低纬度
double maxLat = 60.0; //中国最高纬度
double interv = (maxLat - minLat) / (double)Length;
List<WeightedObservedPoint> weightedObservedPoints = new ArrayList<WeightedObservedPoint>();
for(int i = 0; i < Length; i++) {
WeightedObservedPoint weightedObservedPoint = new WeightedObservedPoint(1, minLat + (double)i * interv, Math.cos(toRadians(x[i])));
weightedObservedPoints.add(weightedObservedPoint);
}
return polynomialCurveFitter.fit(weightedObservedPoints);
}
public static double distanceSimplifyMore(double lat1, double lng1, double lat2, double lng2, double[] a) {
//1）计算 3 个参数
double dx = lng1 - lng2; // 经度差值
double dy = lat1 - lat2; // 纬度差值
double b = (lat1 + lat2) / 2.0; // 平均纬度
//2）计算东西方向距离和南北方向距离（单位：米），东西距离采用三阶多项式
double Lx = (a[3] * b * b * b + a[2] * b * b + a[1] * b + a[0]) * toRadians(dx) * 6367000.0; // 东西距离
double Ly = 6367000.0 * toRadians(dy); // 南北距离
//3）用平面的矩形对角距离公式计算总距离
return Math.sqrt(Lx * Lx + Ly * Ly);
}
```

1）有效性验证：

我们的优化方法叫 distanceSimplifyMore，lucene 的方法叫 distHaversineRAD，下表是在不同尺度下两个方法的相差情况。

表 4　最终优化的有效性

测试点对	distanceSimplifyMore（米）	distHaversineRAD（米）	差别（米）
(39.941,116.45) (39.94，116.451)	140.0242769266660	140.02851671981400	0.0
(39.96，116.45) (39.94，116.40)	4804.113098854450	4804.421153907680	0.3
(39.96，116.45) (39.94，117.30)	72438.90919479560	72444.54071519510	5.6
(39.26，115.25) (41.04，117.30)	263516.676171262	263508.55921886700	8.1

可以看到在百米尺度上两者几乎未有差别，在千米尺度上仅有分米的区别，在更高尺度上如 72 千米仅有 5.6 m 米别，在 264 千米也仅有 8.1 米区别，因此该优化方法的精度能满足我们的应用需求。

2）性能验证：

表 5　最终改进算法性能分析

数量级别	时间(ms)
5w	0.1
10w	0.4
100w	4

3　三种方案的对比

在总结前，还将简单论述了一种比较简单的转化方法，最后将文章中所提出的 3 种方法进行比较详细的总结：

我们的计算场景是计算用户位置与所有筛选出来的商户的距离，这里会涉及到大量三角函数计算。一种优化思路是商户数据不保存经纬度而保存球面模型下的三维坐标（x，y，z），映射方法如下：

（下转第 206 页）

4 结束语

针对云资源调度的动态性、实时性等特点，提出了一种基于模拟退火思想的改进遗传算法 SAIGA。该算法比 GA、SA 和 RR 在任务平均完成时间上分别缩短了 11%，25%和 56%，在算法的收敛精度和速度上都有显著的提高，收敛速度上提高了 40 代左右，能够有效避免超常个体误导种群的进化方向和算法陷入局部最优，同时改进算法能够较均衡地分配任务到各个节点上，提高了资源的利用率。因此，改进算法更加适合云环境下的资源调度，具有较好的实用性。

参考文献：

[1] 刘 鹏. 云计算 [M]. 北京：电子工业出版社，2011.

[2] Armbrust M，Fox A，Griffith R，et al. A View of Cloud Computing [J]. Communications of the ACM，2010，53 (4)：50－58.

[3] Caballer M，Blanquer I，Moltó G，et al. Dynamic Management of Virtual Infrastructures [J]. Journal of Grid Computing，2015，13 (1)：53－70.

[4] 林伟伟，齐德昱. 云计算资源调度研究综述 [J]. 计算机科学，2012，39 (10)：1－6.

[5] 刘 愉，赵志文，李小兰，等. 云计算环境中优化遗传算法的资源调度策略 [J]. 北京师范大学学报 (自然科学版)，2012，48 (4)：378－384.

[6] 袁 浩，李昌兵. 基于社会力群智能优化算法的云计算资源调度 [J]. 计算机科学，2015，42 (04)：206－208.

[7] 徐文忠，彭志平，左敬龙. 基于遗传算法的云计算资源调度策略研究 [J]. 计算机测量与控制，2015，23 (5)：1653－1656.

[8] 李建锋，彭 舰. 云计算环境下基于改进遗传算法的任务调度算法 [J]. 计算机应用，2011，31 (1)：184－186.

[9] 薛 玉. 云计算环境下的资源调度优化模型研究 [J]. 计算机仿真，2013，30 (5)：362－365.

[10] 邬海艳. 基于云计算环境下资源调度算法研究 [D]：赣州：江西理工大学，2012.

[11] Gao Y，Guan H，Qi Z，et al. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing [J]. Journal of Computer and System Sciences，2013，79 (8)：1230－1242.

[12] Zhan Z H，Zhang G Y，Ying L，et al. Load Balance Aware Genetic Algorithm for Task Scheduling in Cloud Computing [A] . In：Dick G，Browne W，Whigham P，Zhang M，Bui L，Ishibuchi H，Jin Y，Li X，Shi Y，Singh P，Tan K，Tang K，editors. Simulated Evolution and Learning [C] . Springer International Publishing，2014：644－655.

[13] Gall J，Rosenhahn B，Seidel H P. An Introduction to Interacting Simulated Annealing [A] . In：Rosenhahn B，Klette R，Metaxas D，editors. Human Motion [C] . Springer Netherlands，2008：319－345.

[14] S K，P VM. Optimization by simmulated annealing [J]. science，1983，220 (4598).

(上接第 201 页)

x = Math. cos(lat) Math. cos(lon)；

y = Math. cos(lat) Math. sin(lon)；

z = Math. sin(lat)；

那么当我们求夹角 AOB 时，只需要做一次点乘操作。比如求 (lon1，lat1) 和 (lon2，lat2) 的夹角，只需要计算 $x1x2 + y1y2 + z1 * z2$，这样避免了大量三角函数的计算。

在得到夹角之后，还需要执行 arccos 函数，将其转换成角度，AB 弧长＝角 $AOB * R$ (R 是地球半径)。上面的这种方法成为坐标转换方法。

坐标转换方法和基于三次多项式拟合三角函数方法性能都非常高，相比 lucene 使用的 Haversine 算法大大提高了计算效率，然而坐标转换方法存在一些缺点：

1) 坐标转换后的数据不能被直接用于空间索引。lucene 可以直接对经纬度进行 geohash 空间索引，而通过空间转换变成三维数据后不能直接使用。我们的应用有附近范围筛选功能 (例如附近 5 km 的团购单子)，通过 geohash 空间索引可以提高范围筛选的效率；

2) 坐标转换方法增大内存开销。我们会将坐标写入倒排索引中，之前坐标是 2 列 (经度和纬度)，现在变成 3 列 (x，y，z)，在使用中我们往往会将这数据放入到 cache 中，因此会增大内存开销；

3) 坐标转换方法增大建索引开销。此方法本质上是将计算从查询阶段放至到索引阶段，因此提高了建索引的开销。

所以，文章中所提出的基于三次多项式拟合三角函数方法计算地球空间距离，在同城范围内的优势非常明显，在效率和有效性方面想比较与以前的方法都有比较大的提升。这是今后研究的一个方向。

4 结束语

文中通过利用三次多项式拟合三角函数的方案改进了原有的地理空间距离计算的算法，从实验结果证明了，改进算法所具备了搜索目标点的快速性，同时也保证了相应的准确率。当然这个实验的结果是基于同城范围的基础上，完全符合 OTO 业务对于搜索场景的要求。同时方案中也存在有待改进的地方，在方案中是用三次多项式去拟合三角函数，根据数学知识可知，多项式的次数越高拟合的三角函数越精确，但是随之的计算的复杂度也在增加。所以如何在计算的复杂度和多项式拟合三角函数的精确度之间做出比较好的选择，可以进一步提高搜索的准确率，这个方向是今后研究的重点方向。

参考文献：

[1] Hua M C，Lou D C，Chang M C. Dual-Wrapped digital watermarking scheme for image copyright protection [J]. Computers & Security，2007，16 (1)：1－12.

[2] 司少林，关 永. 三角函数曲线拟合最佳次数的确定 [J]. 计算机工程与设计，2006. 24 (1)：30－35.

[3] Xiao Y Q，Ji Q. A robust content-based digital image watermarking scheme [J]. Signal Processing，2007，7：1264－1280.

[4] 陈 娱，徐 君. 考虑地理距离的复杂网络社区挖据算法 [J]. 地球信息科学，2013 (3)：58－65.

[5] 李耀彬，曾祥斌，沈钺武. 基于抛物线插值的正弦波拟合算法 [J]. 计算机工程与设计，2009 (11)：100－105.

[6] 薛国新，孙玉强. 正弦曲线三点拟合问题的一种新方法 [J]. 计算机仿真，2006 (2)：99－105.

[7] 罗成汉，刘小山. 曲线拟合法的 Matlab 实现 [J]. 现代电子技术，2003 (20)：54－59.

[8] 田 峥，徐 成，米 超，等. 基于消 失点和主方向估计的道路分割算法 [J]. 计算机研究与发展，2014 (4)：99－104.

[9] 谭方勇，于复生，吴建平. 基于消失点的坐标校准算法 [J]. 计算机应用，2011 (1)：101－105.

[10] 罗小松，房 斌，杨维斌. 采用韦伯局部特征的道路消失点检测 [J]. 计算机应用，2014 (S1)：219－222.