文章编号:1001 - 9081(2010)07 - 1947 - 03

基于 NNlists 的路网 k 路径近邻查询

王宝文,韩静静,陈子军,刘文远
(燕山大学 信息科学与工程学院,河北 秦皇岛 066004)
(zjchen@ysu.edu.cn; hanjingjing512@163.com)

摘 要:为满足 k 路径近邻查询的实时性要求,运用预计算思想提出了基于 NNlists 的 BNNL 算法,通过在用户当前位置和目的地节点进行双向 Dijkstra 扩展得到两点间的最短路径,再通过对最短路径上的路网节点预计算的 m 近邻进行优化处理,最终得到正确的 k 路径近邻。该方法提高了 k 路径近邻查询的查询速度,尤其适用于兴趣点密度较大、k 值较大的情况。

关键词:路网;NNlists;k 路径近邻;空间数据库
中图分类号: TP311 **文献标志码**:A

NNlists-based k-path nearest neighbor query in road networks

WANG Bao-wen, HAN Jing-jing, CHEN Zi-jun, LIU Wen-yuan
(*College of Information Science and Engineering, Yanshan University, Qinhuangdao Hebei* 066004, *China*)

Abstract: To satisfy the real-time requirement of k-path nearest neighbor (kPNN) query, the Based on NNlists (BNNL) algorithm based on the precomputed NNlists was proposed in this paper according to the pre-computation idea, using a bi-directional Dijkstra search scheme to acquire the current shortest path to destination, and then, the nodes in the shortest path were got. At last, these nodes' m nearest neighbors were optimized by a priority queue in order to get the correct kPNN. The performance of BNNL is more efficient in kPNN query speed, especially when the data objects are densely distributed or the number of k is large.

Key words: road network; NNlists; k-Path Nearest Neighbor (kPNN); spatial databases

0 引言

路网中的 k 近邻查询问题是空间数据库研究的一个重要的领域,主要解决找到距离给定的查询点 q 的 k 个最近邻对象的问题。在现实生活中路网中的查询点多数是向着某一目的点的方向移动,例如:出租车司机在前往目的地时,发现汽油已经不足,需要找到距离前往目的地的最短路径上最短的 k 个路径近邻。针对这样的问题,文献[1]提出了 k 路径近邻(k-Path Nearest Neighbor,kPNN)的定义,运用 BNE 三步算法来解决 kPNN 问题。对于 kPNN 查询对实时性要求较高的情况,本文提出了一种基于 NNlists 预计算的 kPNN 查询方法,对每个路网节点存储预计算的 m 近邻的信息,通过优先队列充分利用预计算信息,从而提高了查询效率。

1 相关工作

近邻查询是空间数据库研究的重要领域,即要求在给定的查询点 q 和对象点集 P,在 P 中找到距离 q 最近的 k 个对象。这类问题包括连续近邻查询(Continuous Nearest Neighbors, CNN)[2]、聚集近邻查询(Aggregate Nearest Neighbors, ANN)[3-4]、连续可视近邻查询(Continuous Visible Nearest Neighbor,CVNN)[5]等。随着空间数据库研究的不断深入,文献[6]提出了 IRNN 查询,已知用户从 s 出发到达 t,并由用户指定路线,需要查询离该路线最近的兴趣点(如加油站),IRNN 查询的缺点是有可能路线比较长,给定这样的路线不是很方便。因此,文献[1]提出 kPNN 查询,根据用户的当前位置 s 和目的地 t,为用户求得从 s 到 t 的最短路径,并求得离该最短路径最近的 k 个兴趣点。由于可以有多个用户同时提出 PNN 查询的请求,因此,对实时性的要求会比较高。本文在文献[1]的基础上,提出利用预计算的方法提高系统对查询的响应速度。

为满足用户的实时性要求,预计算方法被广泛应用到 k 近邻(kNN)查询中。文献[7]中提出了路网 Voronoi 图的思想,将整个路网划分成多个 Voronoi 多边形,预计算穿过细胞和每个细胞内部的路网距离,查找近邻时仅需扩展 Voronoi 多边形的边缘点。该方法同时节省了空间存储代价和计算时间,但在兴趣点密度较大的情况下执行效率降低。

文献[8]中预计算了每个路网节点 u 可能要到达的节点之间的最短路径,建立 u 的最短路径四叉树,查询时沿着 u 的最短路径四叉树进行扩展,直到找到要查找的对象点,减少了近邻扩展过程中的盲目扩展。文献[9]针对 CNN 的查询问题提出了基于 NNlists 近邻预计算的 UNICONS 算法,该方法尤其适用于兴趣点的密度较大的情况。本文利用文献[9]提出的 NNlists 的思想,提出了 BNNL 算法,预计算每个路口节点的 m 近邻(mNN),从而减少路径查询过程中的扩展次数。

2 BNNL 算法

BNNL 算法主要分为 3 个部分:

1)利用 Dijkstra 算法双向扩展,计算用户当前位置 s 与目

收稿日期:2010 - 01 - 21;**修回日期**:2010 - 03 - 09。 **基金项目**:国家自然科学基金资助项目(60970123)。

作者简介:王宝文(1956 -),男,内蒙古扎兰屯人,副教授,主要研究方向:智能计算、企业信息化; 韩静静(1984 -),女,河北秦皇岛人,硕士研究生,主要研究方向:移动对象数据库; 陈子军(1971 -),男,黑龙江齐齐哈尔人,副教授,博士,主要研究方向:移动对象数据库、传感器网络; 刘文远(1968 -),男,黑龙江牡丹江人,教授,博士生导师,主要研究方向:信息安全、电子商务、智能计算。

标点 t 之间的最短路径 $SP(s, t)$。分别从当前位置 s 和目标点 t 进行扩展，扩展相遇时得到 $SP(s, t)$。

2）初始化优先队列。在第1）部分得到 $SP(s, t)$ 的基础上，将 $SP(s, t)$ 上的每个路口节点 u 在 NNlists 中的第1近邻放入到优先队列。

3）利用优先队列得到 kPNN。从优先队列取出一个 PNN，设该 PNN 是从路口节点 u 得到的，则再从 NNlists 中取出 u 的下一个近邻，并放入优先队列中，如果 u 的下一个近邻不在 NNlists 中，则需要利用 Dijkstra 算法求得。然后再从优先队列取出一个 PNN，直到获得 kPNN。

BNNL 算法首先要计算当前位置 s 与目标点 t 之间的最短路径 $SP(s, t)$，该算法类似于 BNE 算法[1]的查找阶段，具体算法如下所示。

算法1：searchSP。

输入：当前位置 s；目标点 t；路网 $G(V, E)$；

输出：s 和 t 的之间最短路径 $SP(s, t)$。

```
1)  S = ∅;T = ∅;
    //S 和 T 分别为正向扩展的点集和反向扩展的点集
2)  Q_s = ∅;Q_t = ∅;
    //Q_s 和 Q_t 是优先队列,分别用于正向扩展和反向扩展
3)  L_f(s) = 0;L_r(t) = 0;
    //L_f(n) 和 L_r(n) 分别用于记录 s 到 n 和 t 到 n 的最短路径长度
4)  将 s 放入 Q_s;将 t 放入 Q_t;
5)  while Q_s 和 Q_t 都不为空{                    //进行正向扩展
6)    从 Q_s 中取出 u;
7)    S = S ∪ {u};                  //将 u 放入已扩展的点集 S 中
8)    if u 在 T 中{    //u 同时属于正向扩展和反向扩展的点集
9)    利用正向搜索树和反向搜索树得到 SP(s, t);
10)   return SP(s, t);}
      //得到最短路径 SP(s, t) 并报告
11)   for u 的每个邻接点 v{
        //对扩展得到的节点进一步扩展其相邻节点,得到扩展树
12)     if L_f(v) > L_f(u) + dist(u, v) {
13)       L_f(v) = L_f(u) + dist(u, v);
14)       将 v 放入 Q_s;
15)       π_f(v) = u;                          //u 是 v 的双亲
16)     }                                        //end if
17)   }                                        //end for
18)   反向扩展类似于正向扩展，只是用(T, Q_t, L_r(),π_r())
      代替(S, Q_s, L_f(),π_f());
19) }                                          //end while
```

定理1 算法 searchSP 是可终止的并且能正确给出当前位置 s 和目标点 t 之间的最短路径 $SP(s,t)$。

证明 ①可终止性。由于路网是连通图，一定存在 s 和 t 之间的最短路径，当正向扩展和反向扩展的节点相遇时，即找到最短路径，在第10）行返回，故算法是可终止的。②正确性。由文献[10]可知在两点 s，t 之间进行双向 Dijkstra 算法可以得到这两点之间的最短路径。 证毕。

算法2：getNNlists。利用 NNlists 求路口节点的第 i 近邻。

输入：节点 u；近邻序号 i；节点预计算的近邻个数 m；

输出：u 的第 i 近邻。

```
1)  if (i < = m) {从 NNlists[u] 中取出 u 的第 i 近邻元素 e;
2)    return e;}
3)  else{ if 通过 Dijkstra 算法得到 u 的 i 近邻 p'{
4)          设 dist' 为 u 与 p' 的最短路径长度,path' 为 u 与 p' 的
            最短路径;
5)          return ( p', dist', path');}
6)    else return NULL;
7)  }
```

从 $NNlists[u]$ 中取出 u 的第 i 近邻元素的数据结构：第 i 近邻 p，u 与 p 的最短路径长度 $dist$，u 与 p 的最短路径 $path$。

算法3：BNNL。利用优先队列求出 kPNN。

输入：当前位置 s；目标点 t；路径近邻数 k；路网 $G(V,E)$；

输出：kPNN。

```
1)  kPNN = ∅;
2)  构造空的优先队列 Q;
3)  SP = searchSP(s, t, G);
4)  for SP 中每个路口节点 u{
5)    e = getNNlists(u, 1);
6)    将( u,e.p,e.dist,e.path,1) 放入 Q;
7)  }                                          //优先队列初始化
8)  count = 0;
9)  while( count < k){
10)   if Q 非空 从 Q 中取出 e';
11)   else return NULL;
12)   if kPNN 不存在元素 e'',使得 e''.p = = e'.p{
      //判断 kPNN 中是否已经存在相同的兴趣点
13)   kPNN = kPNN ∪{( e'.p,e'.dist,e(.path)};
14)   count ++;}
15)   e = getNNlists(e'.u, e'.order + 1);
16)   if e ≠ NULL 将(u,e.p,e.dist,e.path,e'.order +1) 放入 Q;
17) }                                          //end while
18) return kPNN;
```

优先队列中元素 e 的数据结构为：路口节点 u，u 的近邻 p，路口节点 u 与兴趣点 p 的最短路径长度 $dist$，u 与兴趣点 p 的最短路路径 $path$，p 在 u 的近邻中的序号 $order$。

kPNN 的数据结构为：u 的近邻 p，路口节点 u 与兴趣点 p 的最短路径长度 $dist$，u 与兴趣点 p 的最短路径 $path$。

定理2 如果兴趣点个数大于等于 k，算法 BNNL 是可终止的，并且能得到 kPNN。

证明 ①可终止性。算法3的第4）~7）行对优先队列初始化，第9）~17）行每次循环会在第15）行得到一个路口节点一个兴趣点近邻，由于兴趣点是有限的，因此，在第18）行返回 kPNN，或者不存在 kPNN 在第11）行返回空，算法是可终止的。②正确性。利用归纳法。由于 NNlists 中存储每个路口节点的 m 近邻，因此，可以利用 NNlists 对优先队列 Q 初始化。此时所有路口节点 u 的第1近邻都在 Q 中，因此，1PNN 一定在 Q 的队头元素中。假设已经求得 iPNN，设第 iPNN 是 u 的第 j 近邻，当第 iPNN 离开 Q 时，第(i+1)PNN 一定会从当前的队头元素和 u 的第 $j+1$ 近邻中产生，因此，在第16）行将 u 的第 $j+1$ 近邻放入 Q 中，此时，第(i+1)PNN 一定在 Q 的队头元素中。因此，如果兴趣点个数大于等于 k，则算法3一定能够得到 kPNN。 证毕。

3 实验及结果分析

实验环境为：Inter(R) Pentium(R) 4 CPU 2.93 GHz，1 GB 内存，Window XP 操作环境。用 California 的路网数据作为实验数据（http://www.cs.fsu.edu/lifeifei/SpatialDataset.htm），该路网中有 21 048 节点，21 693 条边。

图1所示为用户当前位置和目的地之间最短路径长度 l 对查询时间的影响。实验中路径长度 l 是指该路径上路口节点数，m 的取值为12，兴趣点密度为8%（兴趣点密度 = 兴趣点个数/路网中节点数），要查找的路径近邻个数 k 为20。两点间的路径长度主要用双向 Dijkstra 扩展的方法得到，随着路径长度的增加，双向扩展要扩展的数据点数也增加，所以得

到用户当前位置和目的地之间最短路径的时间也增加。在实验中我们取路径的长度 l 为 200。

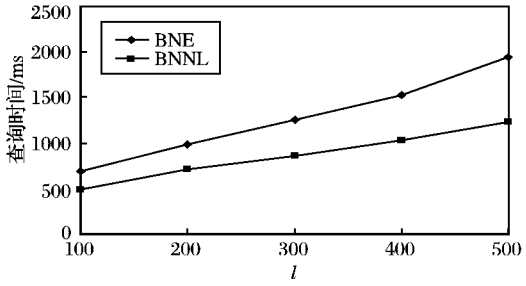

图 1　l 值对查询时间的影响

图 2 所示为预计算的近邻个数 m 对查询时间的影响。预计算路网节点的 m 个近邻,如果 m 取值太大,会有更大的空间开销,太小对查询速度会产生影响,因此,通过实验分析了 m 的取值。路径长度 l 为 200,兴趣点密度为 8%,要查找的路径近邻个数 k 为 20。当 m 值增大时查询时间减少,因为在 m 值增大时预计算的信息增加,因此,要求 Dijkstra 扩展下一近邻的次数减少,总的查询时间也随之较少。但当 m 值增大到一定值时,查询时间减少的程度不明显,因此考虑时间与空间代价的关系,实验中 m 的取值为 12。

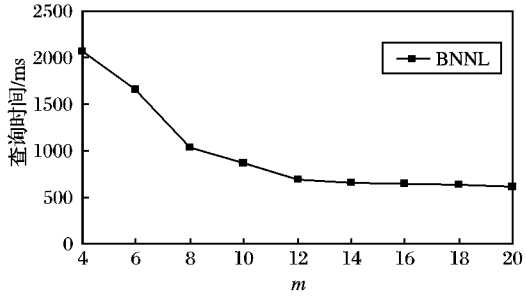

图 2　m 值对查询时间的影响

图 3 所示为兴趣点密度对查询时间的影响。实验中要查找的路径近邻个数 k 为 20,路径长度 l 为 200,预计算近邻个数 m 为 12,由于当兴趣点密度增大时,BNE 算法中候选集中的兴趣点向最短路径上的路口节点进行扩展的节点数减少,所以查询时间减少。BNNL 算法中路口节点的预计算的 m 近邻成为 kPNN 查询最终结果的可能性增大,出入优先队列的次数减少,因此,查询时间减少。由上述分析可知,BNNL 算法的优越性在兴趣点密度较大的情况下比较明显。

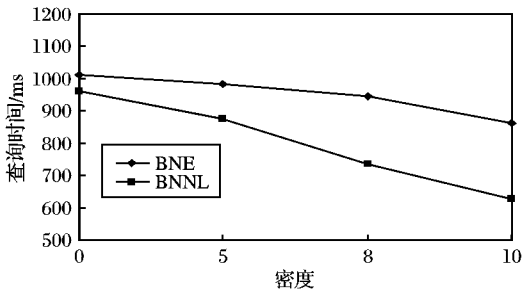

图 3　密度对查询时间的影响

图 4 所示为 k 值增大对查询时间的影响。实验中兴趣点密度为 8%,路径长度 l 为 200,预计算近邻个数 m 为 12。当 k 值增大时,BNE 算法候选集中的节点要扩展的节点个数增加,因此,查询时间增加。当 k 值增大时,BNNL 算法中循环次数增加,查询时间也随之增加,由图 4 可知,在 k 值较大的情况下,BNNL 算法的优越性较明显。

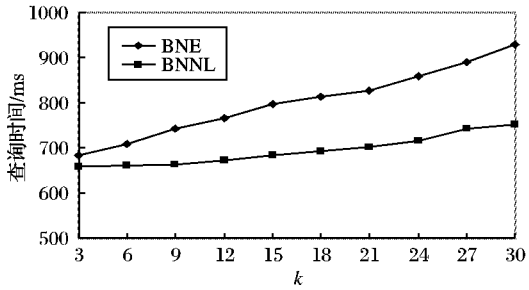

图 4　k 值对查询时间的影响

4　结语

预计算在解决路网近邻查询过程中有其优越性,当前的预计算处理主要应用于 k 近邻的查询中。针对路径近邻查询对实时性要求高的情况,本文提出了基于 NNlists 的 BNNL 算法,通过预计算每个路口节点的 m 近邻,减少了查询过程中的扩展节点次数。实验结果表明,BNNL 算法在 kPNN 查询中的优越性,尤其适用于兴趣点密度较大和 k 值较大的情况。

参考文献:

[1] CHEN ZAIBEN, HENG TAO, SHEN XIAOFANG, *et al*. Monitoring path nearest neighbor in road networks[C]// Proceedings of the 35th SIGMOD International Conference on Management of Data. New York: ACM, 2009: 591 - 602.

[2] MOURATIDIS K, PAPADIAS D, HADJIELEFTHERIOU M. Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring[C]// Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data. New York: ACM, 2005: 634 - 645.

[3] PAPADIAS D, SHEN Q, TAO Y, *et al*. Group nearest neighbor queries[C]// Proceedings of the 20th International Conference on Data Engineering. Washington, DC: IEEE Computer Society, 2004: 301 - 311.

[4] YIU M L, MAMOULIS N, PAPADIAS D. Aggregatenearest neighbor queries in road networks[J]. IEEETransactions on Knowledge and Data Engineering, 2005, 17(6): 820 - 833.

[5] GAO YUNJUN, ZHENG BAIHUA, CHEN GENCAI, *et al*. Continuous visible nearest neighbor queries[C]// Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology. New York: ACM, 2009: 144 - 155.

[6] SHEKHAR S, YOO J. Processing in-route nearest neighbor queries: A comparison of alternative approaches[C]// Proceedings of the 11th ACM International Symposium on Advances in Geographic Information Systems. New York: ACM, 2003: 9 - 16.

[7] KOLAHDOUZAN M, SHAHABI C. Voronoi-based k nearest neighbor search for spatial network databases[C]// Proceedings of the Thirtieth International Conference on Very Large Data Bases. New York: ACM, 2004: 840 - 851.

[8] SAMET H, SANKARANARAYANAN J, ALBORZI H. Scalable network distance browsing in spatial databases[C]// Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. New York: ACM, 2008: 43 - 54.

[9] CHO H J, CHUNG C W. An effcient and scalable approach to CNN queries in a road network[C]// Proceedings of the 31st International Conference on Very Large Data Bases. New York: ACM, 2005: 865 - 876.

[10] NICICJOLSON T A J. Finding the shortest route between two points in a network[J]. Computer Journal, 1966, 9(3): 275 - 280.