时间依赖路网中反向 k 近邻查询

李佳佳 沈盼盼 夏秀峰 刘向宇
(沈阳航空航天大学计算机学院 沈阳 110136)

摘 要 在现存的反向 k 近邻查询方案中,比较高效的研究大多集中在欧氏空间或者静态路网,对时间依赖路网中的反向 k 近邻查询的研究相对较少。已有算法在兴趣点密度稀疏或者 k 值较大时,查询效率较低。对此,提出了基于子网划分的反向 k 近邻查询算法 mTD-SubG。首先,将整个路网划分为大小相同的子网,通过子网的边界节点向其他子网进行扩展,加快对路网中兴趣点的查找速度;其次,利用剪枝技术缩小路网的扩展范围;最后,利用已有时间依赖路网下的近邻查询算法,判定查找到的兴趣点是否为反向 k 近邻结果。实验中将 mTD-SubG 算法与已有算法 mTD-Eager 进行对比,结果表明 mTD-SubG 算法的响应时间比 mTD-Eager 算法减少了 85.05%,遍历节点个数比 mTD-Eager 算法减少了 51.40%。

关键词 时间依赖,路网,反向 k 近邻(RkNN),mTD-SubG 算法
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.01.036

Reverse k Nearest Neighbor Queries in Time-dependent Road Networks

LI Jia-jia SHEN Pan-pan XIA Xiu-feng LIU Xiang-yu
(School of Computer Science, Shenyang Aerospace University, Shenyang 110136, China)

Abstract Most existing efficient algorithms for reverse k nearest neighbor query focus on the Euclidean space or static networks, and few of them study the reverse k nearest neighbor query in time-dependent networks. However, the existing algorithm is inefficient if the density of interest points is sparse or the value of k is large. To address these problems, this paper proposed a sub net division based reverse k nearest neighbor query algorithm mTD-SubG. Firstly, the entire road network is divided into subnets with the same size, and they are expanded to other subnets through the border nodes to speed up the search process for interest points. Secondly, the pruning technology is utilized to narrow the expansion range of road network. Finally, the existing nearest neighbor query algorithm of time-dependent road networks is used for each searched interest points to determine whether it belongs to the reverse k nearest neighbor results. Extensive experiments were conducted to compare the proposed algorithm mTD-SubG with the existing algorithm mTD-Eager. The results show that the response time of mTD-SubG is 85.05% less than that of mTD-Eager, and mTD-SubG reduces the number of traversed nodes by 51.40% compared with mTD-Eager.

Keywords Time-dependent, Road networks, Reverse k nearest neighbor, mTD-SubG algorithm

1 引言

反向 k 近邻(Reverse k Nearest Neighbor, RkNN)查询是一种重要的位置相关查询[1]。目前在反向 k 近邻查询的研究领域,已经提出了很多高效的算法,但是这些研究大多集中在欧氏空间或者静态路网,对时间依赖路网(Time-Dependent Road Networks, TDN)中的 RkNN(Time-Dependent Reverse k Nearest Neighbore, TD-RkNN)查询的研究较少。而欧氏空间或者静态路网中的距离度量与时间依赖路网中的距离度量是不同的。在欧氏空间中,对象之间的距离是对象之间的相对位置;在静态路网中,对象之间的距离是对象之间的路径长度;而在时间依赖路网中,对象之间的距离是沿着路径的行驶时间。

如今人们对出行的时间成本更加关注,而车辆的增多导致道路行驶时间与距离不成正比,因此,TD-RkNN 查询算法中的度量单位由距离变成了行驶时间。一般情况下,行驶时间取决于交通环境和出发时间。例如,通过相同的路径,上下班高峰时段通常比非高峰时段需要花费更多的时间,因此,在时间依赖路网中的 kNN(Time Dependent k Nearest Neighbore, TD-kNN)查询[2-4]、RkNN 查询[5]、路径规划查询[6-9]、最短路径查询[10-14]越来越受到学者们的关注,因此研究 TD-RkNN 查询具有重要的实际意义。

到稿日期:2017-11-22 返修日期:2018-02-07 本文受国家自然科学基金(61502317),辽宁省自然科学基金(201602559,201602568)资助。
李佳佳(1987-),女,博士,讲师,CCF 会员,主要研究方向为时空数据管理、不确定数据管理;**沈盼盼**(1990-),男,硕士生,主要研究方向为时态数据库管理;**夏秀峰**(1964-),男,博士,教授,CCF 高级会员,主要研究方向为数据库理论与技术,E-mail:xiaxiufeng@163.com(通信作者);**刘向宇**(1981-),男,博士,讲师,主要研究方向为社交网络、隐私保护。

由于 TDN 中边的权值随时间而动态变化,且查询结果依赖于用户的出发时间,已有静态路网下的算法无法直接求解时间依赖路网下的各类路径相关的查询问题,因此 TD-RkNN 查询算法被提出。

文献[5]解决了 TD-RkNN 查询存在的问题,其主要是对文献[15]提出的 Eager 算法进行了改进,使其适用于时间依赖路网。文献[5]利用改进的 Eager 算法对路网进行修剪,并对搜索到的每个兴趣点(Points of Interestes,POIs)进行验证。但当 POIs 密度稀疏或 k 值较大时,由于路网搜索范围过大,导致该算法存在查询时间增加以及遍历节点过多等问题。针对该不足,本文提出基于网格技术对路网进行子网划分,以访问子网的方式加快对 POIs 的查找,并对查找到的 POIs 进行验证。该方法通过子网的边界节点向其他子网进行扩展,并使用剪枝技术对路网进行修剪,以缩小查询范围。实验结果表明,当 POIs 密度稀疏和 k 值较大时,本文所提出的算法在查询时间以及遍历节点个数等方面优于文献[5]提出的算法。

2　相关工作

本节主要介绍静态路网中的 RkNN 查询和时间依赖路网中的 kNN 以及 RkNN 查询的国内外研究现状。

静态路网空间下的 RkNN 查询问题已经被广泛研究。Man 等[15]首次提出路网空间下的 RkNN 查询问题,在更新步骤中提出了 Eager 算法和 Lazy 算法,并将其用于识别可能的 RkNN 候选者;Samet 等[16]提出了使用基于 voronoi 图的修剪方法;针对移动兴趣点的连续 RkNN 查询问题,卢秉亮等[17]提出了 RNC-RkNN 算法,此算法可以扩展解决双色 RkNN 问题[18]。以上方法可为解决权值动态变化的时间依赖路网中的 RkNN 问题提供思路,但无法直接应用在时间依赖路网中的 RkNN 查询问题上。

对于 TD-kNN 查询,Demiryurek 等[2]首次使用时间扩展图来建模时间依赖路网;随后,Demiryurek 等[3]提出了基于紧密网络索引(TNI)和松散网络索引(LNI)的查询算法,该算法不仅减少了 k 近邻对象的候选数量,而且减少了 TDN 中最短路径的计算时间;Cruz 等[4]提出了基于增量网络扩展的 TD-NE-A* 算法,利用 TD-NE-A* 算法对路网中的兴趣点进行了搜索。本文利用 TD-NE-A* 算法对找到的兴趣点进行验证。

对于 TD-RkNN 查询,Borutta 等[5]提出了 mTD-Eager 算法并首次解决了 TD-RkNN 查询问题。基于 TDN 的上界图(即所有边的权值为最大值对应的路网),Borutta 等利用 mTD-Eager 算法对路网的扩展进行了修剪。例如 $p^*=\langle q, v_1,\cdots,v_i,\cdots,v_n\rangle$ 是当前一条扩展路径,对于 p^* 上的每个节点进行一次范围近邻(range Nearest Neighbor,rangNN)查询,即以节点 v_i 为中心,以节点 v_i 到查询点 q 的最小行驶时间为半径,查找该范围内所有的 POIs,并将其加入到集合 $Found$ 中,且 $|Found|<k$。如果节点 v_{i+1} 的范围近邻查询使得 $|Found\cup rangNN(v_{i+1})|\geqslant k$,则当前扩展路径在节点 v_i 终止,以此对查询搜索范围进行修剪,从而提高查询效率。

在路网修剪过程中,对于查找到的兴趣点,使用 TD-NE-A* 算法验证其 TD-kNN 查询结果是否包含查询点 q。若包含,则将该兴趣点加入到结果集,否则舍弃该兴趣点。

当 POIs 密度较大且 k 值较小时,mTD-Eager 算法完成得较快;但当 POIs 密度稀疏或 k 值较大时,由于搜索范围变大,使得该算法的查询时间增加,遍历节点数量增多。

为了解决以上问题,本文提出将路网划分为子网的方法,以加快对兴趣点的查找,并利用剪枝技术缩小查询范围。

3　问题定义

定义 1(时间依赖路网)　TDN 由集合 V,E 和 W 组成,记为 $G=(V,E,W)$,其中 $V=\{v_1,\cdots,v_n\}$ 是节点的集合,$E=\{(v_i,v_j)\,|\,v_i,v_j\in V\wedge i\neq j\}$ 是边的集合,$W=\{f_{v_iv_j}(t)\,|\,(v_i,v_j)\in E\}$ 是边的权值且 $f_{v_iv_j}(t)\rightarrow R+$ 。在 TDN 中,边的权值是一个时间函数,时刻 $t\in[0,T)$ 为正实数。假设边的权值函数具有周期性,T 为一个周期的长度,例如:一天中 $T=24$ h。

定义 2(出发时刻)　在 t 时刻开始通过某条边时,称 t 时刻为出发时刻。

定义 3(到达时刻)　在 t 时刻从 v_i 通过边 $(v_i,v_j)\in E$ 到达 v_j 的到达时刻被定义为 $AT(v_i,v_j,t)=(t+f_{v_iv_j}(t))\bmod T$。

定义 4(路径行驶时间)　给定一个 TDN,记为 $G=(V,E,W)$,其中存在路径 $p^*=\langle v_1,\cdots,v_i,v_{i+1},\cdots,v_n\rangle$。在 t 时刻发起查询,通过路径 p^* 所需时间被定义为 $TT(p^*,t)=\sum_{i=1}^{n-1}f_{v_iv_{i+1}}(t_i)$,其中 $t_1=t$,$t_{i+1}=AT(v_i,v_{i+1},t_i)$。

定义 5(时间依赖路网中 k 近邻查询)　给定一个 TDN,记为 $G=(V,E,W)$,在 TDN 中兴趣点集合为 $P\in V$,查询点为 q,出发时间为 t,近邻查询值为 $k\in N\backslash\{0\}$。TD-kNN 查询返回一个结果集 $R=\{v_1,v_2,\cdots,v_k\}\in P$,并且没有节点 $v\in P\backslash R$ 比任意节点 $v_i\in R$ 到达查询点 q 的时间代价小。TD-kNN 被定义为 $\{\forall v_i\in kNN(q,t):\exists v\in P\backslash kNN(q,t):cost(v,q)<cost(v_i,q)\}$,其中,$cost(v,q)$ 表示节点 v 到节点 q 的时间代价。

定义 6(时间依赖路网中反向 k 近邻查询)　给定一个 TDN,记为 $G=(V,E,W)$,在 TDN 中兴趣点集合为 $P\in V$,查询点为 q,出发时间为 t,近邻查询值为 $k\in N\backslash\{0\}$。TD-RkNN 查询结果返回一个结果集 $R=\{p\,|\,q\in\text{TD-}k\text{NN}(p,t)\}$,其中 $p\in P$。即在 TDN 中,当出发时刻为 t,兴趣点 p 的 TD-kNN 查询结果包含查询点 q。其被定义为 TD-RkNN$(q,t)=\{p\,|\,q\in\text{TD-}k\,\text{NN}(p,t)\}$。

本文做了以下假设:

(1)本文使用的时间依赖路网为无向图,即任意出发时刻,通过边的双向权值都是相同的(如图 1 中 $w_1=w_2$),但可以扩展至有向图。如图 1 所示,若从节点 A 向节点 B 行驶,则采用 w_1 权值函数,若从节点 B 向节点 A 行驶,则采用 w_2 权值函数。

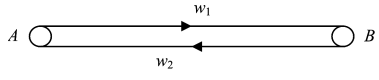


图 1　简单的有向时间依赖路网

Fig. 1　Simple direction time-dependent road map

(2)W 是分段线性函数,且满足先进先出(first-in first-out,FIFO)的特性,比如 $f_{v_iv_j}(t)<w+f_{v_iv_j}(t+w)$,其中 w 是一个正时间值。FIFO 的性质保证了时间依赖路网中最短路径的存在[4,19],否则时间依赖中最短路径查询将变成 NP(Non-deterministic Polynomial)难问题[20]。

(3)为了便于算法描述,假设兴趣点 p 或者查询点 q 都是位于边的端点上并且都是静止的,但可以扩展至兴趣点 p 或者查询点 q 在边上移动的情况[17-18]。

4 基于子网划分的 TD-RkNN 查询算法

本文基于网格技术对路网进行子网划分,提出了 mTD-SubG(monochromatic Timedependent Subgraph)算法,解决了 mTD-Eager 算法在兴趣点稀疏、k 值较大情况下查询效率低的问题。

mTD-SubG 算法分为两个阶段,分别是预处理阶段和查询阶段。在预处理阶段,为了加快对路网中兴趣点的查找,首先利用网格技术对时间依赖路网进行子网划分,然后将子网中的节点按照边界节点、兴趣点等类别进行分类处理并保存,以访问子网的方式加快对兴趣点的查找,而不再是以沿着路径扩展的方式对兴趣点进行查找。在查询阶段,查找路网中所有可能的兴趣点,并对其进行验证。首先确定查询点 q 所在的子网,查找该子网所包含的兴趣点,然后通过该子网的边界节点向其所连接的子网进行扩展,并基于定理 1 终止路网的扩展,在此期间利用 TD-NE-A* 算法对获得的每个兴趣点 p 进行一次 TD-kNN 查询,验证 p 的真伪性。

4.1 预处理阶段

在 POIs 密度稀疏或者 k 值较大的情况下,mTD-Eager 算法对每个扩展节点都调用一次 *rangeNN*,因此时间代价太大。为了解决以上问题,本文提出基于网格技术对时间依赖路网进行子网划分,以访问子网的方式加快对兴趣点的查找。

定义 7(子网 SG_i) 给定一个 TDN,$G=(V,E,W)$,其中 V 表示路网中节点的集合,E 表示路网中边的集合,W 表示路网中边的权值集合。$SG_i(V_i,E_i,W_i)$是图 $G=(V,E,W)$的一个子图,其中 $V_i\in V$,$E_i\in E$,$W_i\in W$。如果一条边 $e_{jk}\in E_i$,那么 $v_j\in V_i$ 并且 $v_k\in V_i$。

定义 8(边界节点) 子网中与其他子网相连的节点被称为边界节点。

定义 9(内部节点) 子网中不与其他子网相连的节点被称为内部节点。

本文采用网格技术对整个路网进行均匀划分。子网划分的数量越少,单个子网的范围越大,则子网内包含的路网节点和 POIs 就越多,得到的候选集中对象就越多,因此在确认结果阶段花费的代价就越大;而子网划分的数量越多,利用剪枝算法对子网进行过滤所需的计算代价越大,找到兴趣点所需跨越的子网数量就越多,因此也会增加算法的整体代价。本文在实验中验证了这一结论,而如何根据实际路网中的节点数量和 POIs 数量来选择合适的子网划分粒度是本文未来的研究工作。

图 2 给出了无向时间依赖的路网,图中圆点代表路网中的普通节点,带有方框的点代表兴趣点,直线代表路网中的边。以图 2 为例,路网被划分为 4 个部分,其中 SG_a,SG_b,SG_c,SG_d 被称为路网的子网,节点 n_3 被称为边界节点,节点 n_1 被称为内部节点。

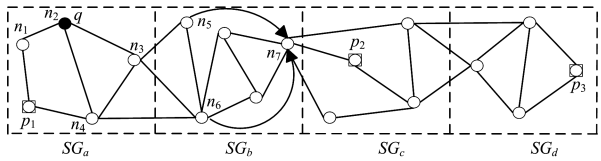


图 2 无向时间依赖路网

Fig. 2 Undirected time-depends road network

为了加快在线查询效率,本文采用多元组的形式对子网中所包含的节点信息进行存储,包括子网所包含节点的信息、节点的位置信息、边界节点的扩展信息等。

假设查询点 q 位于子网 SG_a 中节点 n_2 的位置。为了加快对兴趣点的查找,本文以 *SubInfo*〈子网 ID,所有节点 ID,边界节点 ID,兴趣点 ID,子网中节点的个数〉的形式,将子网的信息进行存储,即保存子网 SG_a 的信息:*SubInfo*〈*SGa*,$\{n_1,n_2,n_3,n_4,p_1\}$,$\{n_3,n_4\}$,$\{p_1\}$,5〉。此外,为了确定查询点 q 所在的子网,将所有节点的位置信息以 *WhereNode*〈节点 ID,节点所在子网 ID〉的形式进行存储,即保存子网 SG_a 中节点的位置信息 *WhereNode*〈$\{n_1,SG_a\}$,…,$\{n_4,SG_a\}$,$\{p_1,SG_a\}$〉。为了明确边界节点的扩展信息,本文将边界节点连接的子网以 *NextSub*〈边界节点 ID,边界节点连接的子网 ID〉的形式进行存储,即子网 SG_a 中边界节点的扩展信息 *NextSub*〈$\{n_3,\{SG_b\}\}$,$\{n_4,\{SG_b\}\}$〉,其中一个边界节点连接的子网可能有多个。

4.2 查询阶段

在查询阶段,为了加快对兴趣点的查找,主要以访问子网的方式,并通过子网的边界节点向其他子网进行扩展。在扩展的过程中,利用 TD-NE-A* 算法对子网中的每个 POIs 进行一次 TD-kNN 查询以验证查询点 q 是否包含在其中,若包含,则将该兴趣点加入到结果集中,否则舍弃该兴趣点。其中 TD-NE-A* 算法基于定义 4 计算两个节点之间的时间代价,并且使用剪枝技术终止路网扩展,以达到路网修剪的目的。

定理 1 若 q 为查询点,n 为图节点,$p_1,p_2,\cdots,p_k$ 为兴趣点,假设出发时刻为 t,满足 $cost(n,p_i)<cost(n,q)$,$p_i\in\{p_1,p_2,\cdots,p_k\}$。则对于任意兴趣点 $p(p\neq p_i)$,如果 p 到 q 最短行驶时间的路径经过节点 n,则有 $cost(p,p_i)<cost(p,q)$,即 $p\notin$ TD-RkNN(q)。

证明:$cost(p,p_i)=cost(p,n)+cost(n,p_i)<cost(p,n)+cost(n,q)=cost(p,q)$,故定理 1 成立。

当查询发起时,即给定起始查询条件(q,t,k),其中 q 为查询点,t 为出发时刻,k 为近邻查询。在图 2 中,查询点 q 位于节点 n_2 的位置。首先,访问 *WhereNode* 获得查询点 q 所在的子网 SG_a;其次访问子网 SG_a 的信息 *SubInfo*,获得子网 SG_a 中包含的兴趣点 $P=\{p_1\}$;然后利用 TD-NE-A* 算法对集合 P 中的每个兴趣点 p 进行一次 TD-kNN 查询,如果查询点 q 是兴趣点 p 的 TD-kNN 查询结果之一,则将该兴趣点 p

加入到结果集 R 中,否则舍弃该兴趣点 p。

通过访问子网 SG_a 的信息 $SubInfo$,同样能够获得子网 SG_a 中包含的边界节点 $BV=\{n_3,n_4\}$。然后根据 BV 中每个边界节点 b 访问 $NextSub$,则可以获得边界节点 b 所连接的子网 ID,即通过边界节点 b 可以进行扩展的子网,例如通过边界节点 n_3 可以向子网 SG_b 进行扩展,最后将获得的子网 ID 加入到优先队列 PQ 中。

以上是访问查询点 q 所在子网的过程,其基本思想可以用启动子网算法(StartSG 算法)的伪代码表示,如算法 1 所示。

算法 1　StartSG(q,t,k)

```
输入:q,t,k
输出:PQ
1. R=null,PQ=null;
2. sg←WhereNode(q);
3. P←FindPOIinSG(q);
4. BV←FindBorderinSG(q);
5.   For (∀p∈P)
6.     If (q∈TD-kNN(p,t,k))
7.       R←R∪p;
8.   For (∀b∈BV)
9.     PQ←connectSG(b);
10. Return PQ;
```

路网的扩展过程如下。

假设从优先队列 PQ 中取出的子网为 SG_b,由于子网 SG_b 可能由子网 SG_a 的边界节点 n_3 或者 n_4 扩展而来,即可能从优先队列 PQ 中多次取出子网 SG_b,因此为了防止子网 SG_b 被多次访问,将其标记为已访问。接着访问子网 SG_b 的信息 $SubInfo$,找到其包含的兴趣点,此时子网 SG_b 中没有任何的兴趣点,即 $P=\emptyset$,则通过子网 SG_b 的每个边界节点向其连接的子网进行扩展。图 2 中,由节点 n_7 向子网 SG_c 进行扩展,此时实现了子网 SG_b 的跳跃过程。

当从优先队列 PQ 中取出子网 SG_c 时,访问 SG_c 的子网信息 $SubInfo$,找到其包含的兴趣点 $P=\{p_2\}$,利用 TD-NE-A* 算法对兴趣点 p_2 进行一次 TD-kNN 查询。如果兴趣点 p_2 的 TD-kNN 查询结果包含查询点 q,则将兴趣点 p_2 加入到结果集 R 中,并且通过子网 SG_c 的每个边界节点向其连接的子网进行扩展;否则舍弃兴趣点 p_2,此时子网 SG_c 中不存在任何兴趣点是查询点 q 的 TD-RkNN 查询结果,那么利用 TD-NE-A* 算法对子网 SG_c 中的每个边界节点进行一次 TD-kNN 查询,验证其查询结果是否包含查询点 q。如果为真,则对该边界节点进行扩展;否则,基于定理 1 可知,通过该边界节点扩展查找到的兴趣点不可能成为查询点 q 的 TD-RkNN 结果,因此终止该边界节点的扩展。当子网 SG_c 中的每个边界节点都不再向其他子网进行扩展时,子网 SG_c 即停止了路网的扩展过程,实现了对路网的剪枝。

值得注意的是路网中的剪枝策略,当子网中所有兴趣点的 TD-kNN 查询结果都不包含查询点 q 时,验证子网中的每个边界节点,判断其边界是否能够进行扩展。

路网的扩展过程可以用 mTD-SubG 查询算法的伪代码表示,如算法 2 所示。

算法 2　mTD-SubG(q,t,k)

```
输入:q,t,k
输出:R
1. R=null,PQ=null;
2. StartSG(q,t,k);
3. While (PQ≠null)
4.     SGi←PQ.deQueue();
5.     If (SGi is not visited)
6.         mark SGi as vsitied;
7.         flag=0;
8.         If (P==null)
9.           flag=1;
10.        Else
11.          For (p∀∈P)
12.            If (q∈TD-kNN(p,t,k))
13.              R←R∪p;
14.              flag=1;
15.        If (flag==1)
16.          For (∀b∈BV)
17.            PQ←connectSG(b);
18.        else
19.          For (∀b∈BV)
20.            If (q∈TD-kNN(b,t,k))
21.                PQ←connectSG(b);
22. Return R;
```

整个过程是以访问子网的方式加快对兴趣点的查找过程,并通过子网的边界节点进行路网的扩展。mTD-SubG 算法的第 3－16 行表示从优先队列 PQ 中取出一个子网 SG_i,利用 TD-NE-A* 算法对子网中的每个兴趣点 p 进行一次 TD-kNN 查询操作,如果查询结果包含查询点 q,则将该兴趣点 p 加入到结果集 R。mTD-SubG 算法的第 17－23 行表示若子网中没有兴趣点或者至少存在一个兴趣点 p 使其 TD-kNN 查询结果包含查询点 q,那么将该子网中每个边界节点所连接的子网 ID 加入到优先队列 PQ 中,否则对每个边界节点进行验证。若为真,则对该边界节点进行扩展;否则终止该边界节点的扩展。

5　实验结果及分析

5.1　实验环境及实验数据

为了验证本文所提 mTD-SubG 算法的性能,实验中将其与 Borutta 等提出的 mTD-Eager 算法进行比较。算法使用 Java 语言实现,实验环境设置为 Intel(R) Core(TM) i5-3470 CPU @3.20 GHz,4 GB 内存,1 TB 硬盘,64 位 Windows 操作系统。

实验采用德国奥尔登堡的路网图进行仿真,该地图含有 6105 个节点,7035 条边。实验中的兴趣点随机产生且均匀分布。本文将全天分为 5 个时段:[22:00,7:00),[7:00,9:00),[9:00,17:00),[17:00,19:00),[19:00,22:00),每隔 5 min 对边赋予一次权值,得到 288 个边的权值,根据这些边的权值拟合分段线性函数,将该函数作为路网中边的权值函数。

本文实验参数设置如表 1 所列,其中加粗数字代表实验

参数的默认值，查询时间默认值设置为 240 min。

表 1 实验参数设置

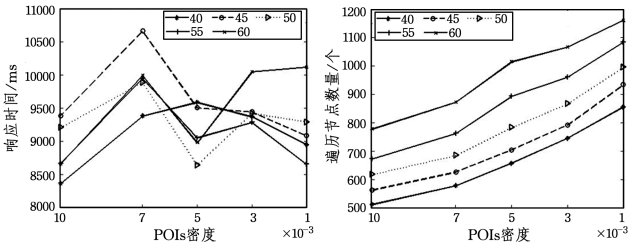
Table 1 Setting of experimental parameters

k 值	网格大小	POIs 密度/%	查询时间/min
4	40 * 40	0.1	
5	45 * 45	0.3	
6	50 * 50	0.5	1～1440
7	55 * 55	0.7	
8	60 * 60	1	

5.2 实验结果及分析

实验根据不同 POIs 密度与 k 值，分别对两种算法的响应时间和遍历节点个数进行了对比。

(1)在不同 POIs 密度下，令 $k=6$，$t=240$ min，随机进行 50 次查询。图 3(a)显示了随着 POIs 密度的减小，不同大小子网响应时间的变化趋势。从图 3(b)可以看出，不同大小子网的遍历节点个数会随着 POIs 密度的减小而增加。



(a)POIs 密度对不同大小子网的查询时间的影响 (b)POIs 密度对不同大小子网的遍历节点数量的影响

图 3 POIs 密度对不同大小子网的影响

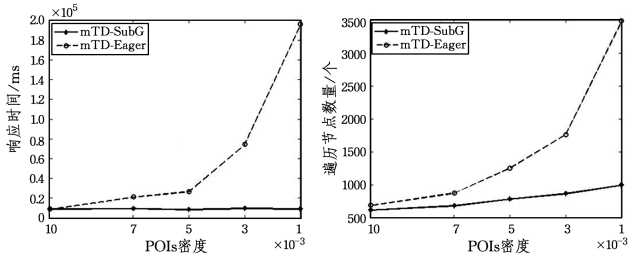
Fig. 3 Impact of POIs density on different sizes of subgraph

随着 POIs 密度的减小，mTD-SubG 算法中子网扩展的数量随之增加，当子网扩展的数量起主导因素时，会使 mTD-SubG 算法的响应时间延长；当 POIs 密度减小起主导因素时，会使 mTD-SubG 算法的响应时间缩短。因此，随着 POIs 密度的减小，mTD-SubG 算法的响应时间会产生波动变化。当 POIs 密度为 0.7%时，子网划分为 40 * 40 最优，而在 POIs 密度为 0.5%时，子网划分为 50 * 50 最优，说明划分粒度对查询时间的影响与 POIs 密度相关。

当 POIs 密度一定时，子网个数越多，即子网的范围越小，mTD-SubG 算法需要遍历的节点就越多。因为 POIs 密度一定时，总的搜索范围相当，但是子网的范围越小，mTD-SubG 算法需要访问的子网就越多，所以遍历的子网边界节点就越多。但是随着 POIs 密度的减小，在子网大小一定时，mTD-SubG 算法需要遍历的节点的数量随之增加。因为随着 POIs 密度的减小，扩展的子网数量随之增加，导致遍历子网边界节点的数量也随之增加。虽然每个子网中的 POIs 个数有所减少，但是相对于增加的子网边界点而言数量较少，因此在子网大小一定时，mTD-SubG 算法遍历节点的数量随着 POIs 密度的减小而增加。

(2)在不同的 POIs 密度下，令 $k=6$，$t=240$ min，子网大小为 50 * 50，随机进行 50 次查询。从图 4(a)中可以看出，随着 POIs 密度的减小，mTD-Eager 算法的响应时间有所延长，而 mTD-SubG 算法的响应时间虽有起伏，但相比于 mTD-Eager 算法，其响应时间更短。从图 4(b)可以看出，两种查询算法遍历节点的个数随着 POIs 密度的减小而增加，并且 mTD-SubG 算法遍历节点的数量更少。

当近邻查询 k 值一定时，POIs 密度减小，mTD-SubG 算法和 mTD-Eager 算法都需要扩大对 POIs 的搜索范围，因此遍历节点的个数随之增加。由于 mTD-SubG 算法只对查询范围内子网的边界节点和 POIs 进行遍历，不遍历其他节点，而 mTD-Eager 算法对查询范围内的每个节点都进行遍历，因此当 POIs 密度相同时，mTD-SubG 算法比 mTD-Eager 算法遍历节点的个数少，mTD-SubG 算法更高效。并且随着 POIs 密度的减小，mTD-SubG 算法遍历的节点个数的增长速度比 mTD-Eager 算法更缓慢。



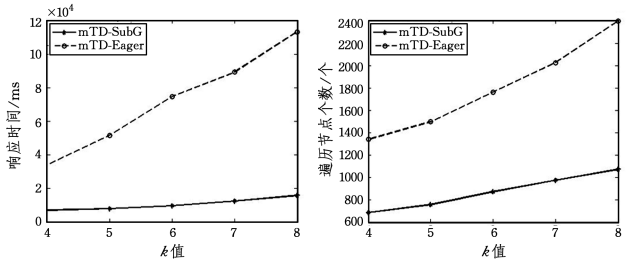
(a)POIs 密度对响应时间的影响 (b)POIs 密度对遍历节点数量的影响

图 4 POIs 密度对查询的影响

Fig. 4 Impact of POIs density on queries

由于 mTD-Eager 算法的响应时间由遍历节点的个数决定，且与遍历节点个数随着 POIs 密度变化的趋势相同，因此，由遍历节点数量变化规律可知，mTD-Eager 算法的响应时间也同样随着 POIs 密度的减小而延长。而随着 POIs 密度的减小，mTD-SubG 算法子网扩展的数量随之增加，若子网扩展的数量起主导因素，则将导致 mTD-SubG 算法的响应时间延长；若 POIs 密度减小起主导因素，则将导致 mTD-SubG 算法的响应时间缩短。因此随着 POIs 密度减小，mTD-SubG 算法的响应时间会产生波动变化。

(3)令 POIs 密度为 0.3%，$t=240$ min，子网大小为 50 * 50，对于不同的 k 值，随机进行 50 次查询。图 5(a)给出了随 k 值的增加，两种查询算法响应时间的对比曲线。图 5(b)给出了随 k 值的增加，两种查询算法遍历节点数量的对比曲线。由图 5 可以看出，两种查询算法的响应时间和遍历节点数量都随着 k 值的增加而增加。



(a)k 值对响应时间的影响 (b)k 值对遍历节点数量的影响

图 5 k 值对查询的影响

Fig. 5 Impact of k value on queries

相比于 mTD-Eager 算法，mTD-SubG 算法遍历节点的数量减少了 51.40%，因为随着 k 值的增大，POIs 的 TD-kNN 查询结果包含兴趣点 q 的概率将提高，使得 mTD-SubG 算法和 mTD-Eager 算法的查询范围扩大，mTD-SubG 算法只是对

查询范围内子网的边界节点和 POIs 进行遍历,而 mTD-Eager 算法对查询范围内的每个节点都进行遍历,因此 mTD-SubG 算法遍历节点的个数更少,算法更高效。mTD-SubG 算法的响应时间比 mTD-Eager 算法的缩短了 85.05%,因为响应时间主要由遍历节点的数量决定,所以响应时间随 k 值的变化趋势与遍历节点个数随 k 值的变化趋势相同。

由以上分析结果可知,相比于 mTD-Eager 算法,本文提出的 mTD-SubG 算法在响应时间和遍历节点个数上均有优势;并且在密度或者 k 值相同的情况下,mTD-SubG 算法的响应时间和遍历节点个数的值均更小。

结束语　与静态路网相比,时间依赖路网在位置广告、智能导航、紧急救援等领域更具现实意义。本文在预处理阶段基于网格技术对路网进行子网划分,对子网中的节点进行分类处理并保存。在查询阶段,以访问子网的方式,查找每个子网包含的 POIs,并对这些 POIs 进行验证,通过子网中的边界节点向其他子网进行路网扩展。若子网中的所有兴趣点都不是查询点 q 的 TD-RkNN 查询结果,则验证子网的边界节点,判断该节点能否向其他子网进行扩展,以达到路网扩展以及修剪的目的。该方法解决了 mTD-Eager 算法在兴趣点稀疏、k 值较大的情况下查询效率较低的问题。仿真实验表明,相比于 mTD-Eager 算法,mTD-SubG 算法的响应时间缩短了 85.05%,遍历节点的个数减少了 51.40%。如何根据实际路网中的节点数量和 POIs 数量来选择合适的子网划分粒度是我们未来的研究工作。

参 考 文 献

[1] LI Y H,LI G H,DU X K. Study on continuous bichromatic reverse k nearse neighbor query processing in road networks[J]. Computer Science,2012,39(11):131-136. (in Chinese)
李艳红,李国徽,杜小坤. 路网中双色数据集上连续反向 k 近邻查询处理的研究[J]. 计算机科学,2012,39(11):131-136.

[2] DEMIRYUREK U,BANAEIKASHANI F,SHAHABI C. Towards K-Nearest Neighbor Search in Time-Dependent Spatial Network Databases[C]// International Conference on Databases in Networked Information Systems. Springer-Verlag, 2010: 296-310.

[3] DEMIRYUREK U,BANAEI-KASHANI F,SHAHABI C. Efficient K-Nearest Neighbor Search in Time-Dependent Spatial Networks[M] // Database and Expert Systems Applications. Springer Berlin Heidelberg,2010:432-449.

[4] CRUZ L A,NASCIMENTO M A,MACÊDO J A F D,et al. K-nearest neighbors queries in time-dependent road networks[J]. Journal of Information & Data Management,2012,3(3):211-226.

[5] BORUTTA F,NASCIMENTO M A,NIEDERMAYER J. Monochromatic RkNN queries in time-dependent road networks[C]// ACM Sigspatial International Workshop on Mobile Geographic Information Systems. ACM,2014:26-33.

[6] LI L,HUA W,DU X,et al. Minimal on-road time route scheduling on time-dependent graphs[J]. Proceedings of the Vldb Endowment,2017,10(11):1274-1285.

[7] YANG Y,GAO H,YU J X,et al. Finding the cost-optimal path with time constraint over time-dependent graphs[J]. Proceedings of the Vldb Endowment,2014,7(9):673-684.

[8] WANG S,LIN W,YANG Y,et al. Efficient Route Planning on Public Transportation Networks: A Labelling Approach[C]// Acm Sigmod International Conference on Management of Data. ACM,2015:967-982.

[9] ZHENG B,SU H,HUA W,et al. Efficient Clue-Based Route Search on Road Networks[J]. IEEE Transactions on Knowledge & Data Engineering,2017,29(9):1846-1859.

[10] DEMIRYUREK U,SHAHABI C. Hierarchical and exact fastest path computation in time-dependent spatial networks: US, US8660789[P]. 2014-02-25.

[11] FOSCHINI L,HERSHBERGER J,SURI S. On the Complexity of Time-Dependent Shortest Paths.[C]// Acm-siam Symposium on Discrete Algorithms. Springer US,2011:327-341.

[12] DELLING D. Time-Dependent SHARC-Routing[J]. Algorithmica,2011,60(1):60-94.

[13] LI L,ZHOU X,ZHENG K. Finding Least On-Road Travel Time on Road Network[C] // Australasian Database Conference. Springer International Publishing,2016:137-149.

[14] LIN W,LIN W,LIN W,et al. Effective indexing for approximate constrained shortest path queries on large road networks [J]. Proceedings of the Vldb Endowment,2016,10(2):61-72.

[15] MAN L Y,PAPADIAS D,MAMOULIS N,et al. Reverse nearest neighbors in large graphs[J]. IEEE Transactions on Knowledge & Data Engineering,2006,18(4):540-553.

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

[17] LU B L,CUI X Y,LIU N. Continuous k-nearest neighbor query processing in network[J]. Computer Engineering and Design, 2014(7):2395-2401. (in Chinese)
卢秉亮,崔晓玉,刘娜. 路网中连续反向 k 近邻查询处理[J]. 计算机工程与设计,2014(7):2395-2401.

[18] LU B L,CUI X Y,LIU N. Bichromatic k-nearest neighbor query processing in network[J]. Journal of Chinese Computer Systems,2015,36(2):266-270. (in Chinese)
卢秉亮,崔晓玉,刘娜. 路网中双色反向 k 近邻查询处理[J]. 小型微型计算机系统,2015,36(2):266-270.

[19] MAN L Y,MAMOULIS N. Reverse Nearest Neighbors Search in Ad-hoc Subspaces[C]// International Conference on Data Engineering. IEEE,2006:76-76.

[20] PAPADIAS D,ZHANG J,MAMOULIS N,et al. Query Processing in Spatial Network Databases[C]// Morgan-Kaufmann. 2003:802-813.