International Journal of Database Theory and Application
Vol.9, No.4 (2016), pp. 151-160
http://dx.doi.org/10.14257/ijdta.2016.9.4.14

K-Aggregate Nearest Neighbor Query Method of Moving Objects in Road Networks

Chen Wen

School of Mathematics and Computer Science, Tongling College, Tongling, P. R. China
Email:tlxychenwen@163.com

Abstract

K-aggregate nearest neighbor query method of mobile objects in road networks is studied. Moving state model of object is introduced, and road network distance calculation formula is provided. Besides, this paper designs a kANN query algorithm that can find out the previous k target nodes with the smallest aggregate function value among multiple target nodes for various query points when query points and data points are under moving state in road networks. The candidate results are cut short via pruning method. Finally, performance of the algorithm is verified through simulation experiment, and results of the simulation experiment prove that this algorithm has high efficiency and accuracy.

Keywords*: road networks; k-aggregate nearest neighbor query; moving object; spatial databases*

1. Introduction

Location-based service includes location query, nearest neighbor query and range query. The basic issue of these services is k-nearest neighbor query (kNN) in moving database, and the idea is to find k nearest neighbor objects of a moving user among all moving objects. As a variety of kNN query, k-aggregate nearest neighbor query (kANN) can find out the previous k target nodes with the smallest aggregate function value among multiple target nodes for various query points. Query of this type will appear frequently in daily life. For instance, several friends in different positions (multiple query points) want to find a restaurant (target node) to dine together, or people in different positions (multiple query points) want to find a hotel (target node) to have a meeting. Different from conventional kNN query or other varieties, kANN query have multiple query points. In addition, results fed back by kANN query depend on a specific aggregate function. Generally speaking, aggregate function covers summation, maximum value and minimum value. For different aggregate functions, the query results and significances are different. For example, when many people dine together, summation function can minimize the total distance from all people to the restaurant, so as to guarantee the minimum cost. Maximum value function will minimize the time value of the person who spends the longest time (compared with other members) in arriving at the restaurant, so as to guarantee that they start to dine at the earliest time. Finally, minimum value function can minimize the arriving cost of one person (or make the person arrive at the earliest time), so as to order dishes as early as possible.

2. Relevant Work

In Euclidean space and road network environment, researchers have proposed many solutions by directing at the issue of kNN query and its varieties. At the early stage, Roussopoulos *et al*. [1] proposed kNN query of Euclidean space. Cheung *et al*. [2]

ISSN: 2005-4270 IJDTA

modified the pruning strategy in literature [1] to improve the query efficiency, and conducted certification in details. Hjaltason *et al*. [3] put forward increment method to solve kNN problems. Therefore, under road network environment, kNN has gained many research achievements. Jensen [4] *et al*. proposed data model of 1NN query in road networks as well as some abstract definitions for the first time, and they calculated the shortest distance via algorithms similar to Dijkstra algorithm. Papadias *et al*. [5] put forward a framework of integrating road networks and Euclidean space information; based on this framework, they obtained kNN query results by adopting method similar to that in literature [3].

For moving objects in road networks, calculation for the distance between data object and query point is absolutely different from that of Euclidean space, and distance in road networks is defined as the length of the shortest path between data object and query point (*i.e*. the sum of weights). Some scholars raised snapshot algorithm to handle continuous K-nearest neighbor query of static objects and moving query points in road networks [6-8]. Huang *et al*. [9] proposed a method based on periodic snapshot re-calculation to deal with continuous K-nearest neighbor query. In this method, data objects and query points move continuously in a road network, and the main issue of this method is to appropriately set the period of snapshot re-calculation. Long period will affect CKNN query results and short period will cause a high communication cost during update.

In terms of k-aggregate nearest neighbor query issue studied in this paper, Papadias *et al*. [10] proposed 3 methods to deal with kANN query in Euclidean space for the first time. However, the three methods are not suitable for road network environment. Yiu *et al*. [11] raised a method of solving kANN query in road networks, but it cannot be applied to dynamic road networks. Therefore, this paper proposes a kANN query method applicable to dynamic road networks (*i.e*. query points and query objects are in moving state).

3. Relevant Definition

Suppose that a road network is a graph which is composed of nodes, sides and objects moving on a side at a certain velocity.

Definition 1: Road network model. A road network can be modeled into an undirected weighted graph, G (V, E). In this graph, V is composed of all network nodes and E is the set of all sides, *i.e*. E∈V×V.

Definition 2: distance determinate. Suppose that there are two different sides, ei and ej; for any two objects oi and oj, they move on ei and ej. Regardless of the position where object oi is located on side ei and the position where object oj is located on side ej, if the shortest path from object oi to oj is the sole path that passes the road network, then the distance of ei and ej is certain.

As for a pair of sides, ei and ej in a road network, the following method can be used to examine whether ei and ej have a certain distance. Suppose that the side ei has two nodes ei.snode and e*i.e*node at both ends and the side ej also has two nodes ej.snode and ej.enode at both ends, thus calculation for network distance between any two nodes among the four nodes can be expressed as d(ei.snode, ej.snode), d(ei.snode, ej.enode), d(e*i.e*node, ej.snode) and d(e*i.e*node, ej.enode), signified as d1, d2, d3 and d4 for short. When and only when the following equation is tenable, the distance of ei and ej is certain.

$$d2=d1+ej.w$$

$$d3=d1+ei.w$$

$$d4=d1+ ei.w+ej.w$$

An example in literature [16] is introduced to illustrate the concept of distance determinate. As shown in Figure 1, the moving object oi and query point q are marked with dots, and the arrow means the moving direction of the moving object. Positive and

negative definitions of the moving direction are as follows. If an object moves from the starting node on the side to the terminal node, then its direction and velocity are positive values; otherwise, its direction and velocity are negative values. The moving velocity of the object is indicated in the small brackets after the mark of object oi. For each side ei, the length and speed limit are also indicated in corresponding small brackets. Suppose that the query point q needs to process 2NN; according to the data structure of side and moving object, their specific information is displayed in Table 1, Table 2 and Table 3.

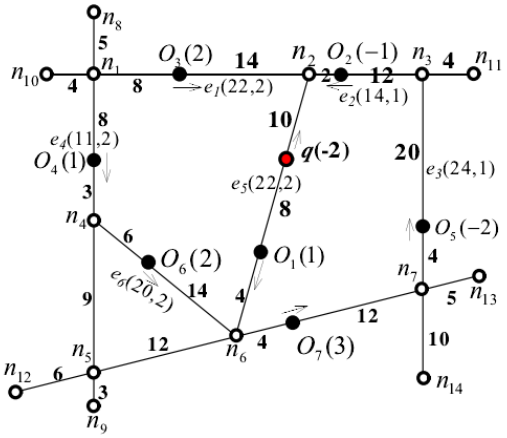

Figure1. Road Networks Graph [16]

Table 1. Side Information

id	w	includeobj	snode	enode	maxv
e1	22	{o3}	n1	n2	2
e2	14	{o2}	n2	n3	1
e3	24	{o5}	n3	n7	2
e4	11	{o4}	n1	n4	1
e5	22	{o1,q}	n2	n6	2
e6	20	{o6}	n4	n6	2

Table 2. Moving Object Information

id	e	Dist	v
o1	e5	18	1
o2	e2	2	-1
o3	e1	8	2

Table 3. Mobile Status Information

qi	oi	moving_state
q	o1	away
q	o2	closer

The process is illustrated by taking side e1 and side e3 in Table 1 as examples. As for side e1 (n1, n2), e1.w=22; in terms of side e3 (n3, n7), e3.w=24. The four distances among the four pairs of nodes composed of nodes n1 and n2 as well as nodes n3 and n7 are d(n1, n3), d(n1, n7), d(n2, n3) and d(n2, n7).

d(n1, n3) = d(n2, n3) + e1.w

d(n2, n7) = d(n2, n3) + e3.w

d(n1, n7) = d(n2, n3) + e1.w + e3.w

According to the above calculation, the following conclusion can be gained: the distance between side e1 and side e3 is certain.

3.1 Moving State Model of Object [12-13]

In order to conveniently calculate the distance from various query points to query objects in dynamic road networks, moving state model of object (MSO) in literature [12-13] is introduced.

1. The object o and query point q are on the same side

The query point q and object o are on the same side e; the side starts from the node ns and ends in node ne. |q.v| and |o.v| represent the absolute values of the velocity respectively. According to the moving velocity and direction of query point q and object o, their moving state moving_state can be set as follows:

(1) If q.v>0, then qo.moving_state= closer

(2) If q.v>0∧o.v<0, then qo.moving_state = closer

(3) If q.v>0∧o.v>0∧q.v<o.v and q.v<0∧o.v<0∧|q.v|>|o.v|, then qo.moving_state= away

(4) If q.v<0∧o.v>0 , then qo.moving_state= away

2. The object o and query point q are on two different sides

Two endpoints that connect the shortest path between query point q and object o are expressed as path(ei, ej).inode and path(ei, ej).jnode.

(1) If the query point q is close to the node path(ei, ej).inode and the object o is close to the node path(ei, ej).jnode, then qo.moving_state = closer.

(2) If the query point q is close to the node path(ei, ej).inode, the object o is away from the node path(ei, ej).jnode, and |q.v|>|o.v|, then qo.moving_state = closer. Otherwise, qo.moving_state = away.

(3) If the query point q is away from the node path(ei, ej).inode , the object o is close to the node path(ei, ej).jnode, and |q.v|>|o.v|, then qo.moving_state = away. Otherwise, qo.moving_state = closer.

(4) If the query point q is away from the node path(ei, ej).inode , the object o is away from the node path(ei, ej).jnode, then qo.moving_state = away.

3.2 Road Network Distance Calculation

1. If the distance of side ei and side ej is certain (the sole path of passing side ei and side ej is certain, and it passes nodes ni and nj), then the network distance between query point q and object o at time t can be expressed as $D_{q,o}(t)$; the calculation formula is as follows:

$$D_{q,o}(t)=| q.dist+q.v\times(t-t0)|+DN[i,j]+|o.dist+o.v\times(t- t0)|$$

Here q.dist (o.dist) is the distance from the query point q (object o) to the starting node of the side where it is located; q.v (o.v) means the moving velocity; t0 refers to the starting time. The shortest network distance between each pair of nodes ni and nj is DN[i,j].

2. If the distance of side ei and side ej is uncertain, then the network distance Dq,o(t) between query point q and object o at time t is composed of three parts:

(1) Distance between query point q and node ni (or node ni');

(2) The shortest network distance between node ni (or node ni') and nj (or node nj');

(3) Distance between object o and nj (or node nj').

Here the network distance between query point q and object o has four situations. Firstly, nodes ni and nj are considered, and the network distance between query point q and object o is expressed as $D^1_{q,o}(t)$. Secondly, nodes ni and nj' are considered, and the network distance between query point q and object o is expressed as $D^2_{q,o}(t)$. Thirdly, nodes ni' and nj are considered, and the network distance between query point q and

object o is expressed as $D^3_{q,o}(t)$. Fourthly, nodes ni' and nj' are considered, and the network distance between query point q and object o is expressed as $D^4_{q,o}(t)$. The shortest distance among the four distances is selected as Dq,o(t), and the specific calculation is as follows:

$$D^1_{q,o}(t) = | q.dist+q.v\times(t-t0)|+DN[i,j]+|o.dist+o.v\times(t- t0)|,$$

$$D^2_{q,o}(t) = |q.dist+q.v\times(t- t0)|+ DN[i,j]+|o.e.w-o.dist+o.v\times(t- t0)|,$$

$$D^3_{q,o}(t) = |q.e.w-q.dist+q.v\times(t- t0)|+ DN[i,j]+|o.dist+o.v\times(t- t0) |,$$

$$D^4_{q,o}(t) = | q.e.w -q.dist+q.v\times(t- t0)|+ DN[i,j]+|o.e.w-o.dist+o.v\times(t- t0)|.$$

$$D_{q,o}(t) = \min (D^1_{q,o}(t), D^2_{q,o}(t), D^3_{q,o}(t), D^4_{q,o}(t))$$

4. kANN Query Algorithm of Moving Objects

The most intuitional method of solving kANN query is to extend the whole road network from each query point. In this way, the aggregate distance from query points to each target node can be gained. Then these aggregate distances are sorted, and the previous k target nodes are the query results. However, such method is quite time consuming. The biggest drawback of this method is that we might do a lot of useless work. In another word, the aggregate distance from query points to some target nodes that cannot be the kANN query results is also calculated, and the cost of extending the road network is very high. In addition, under dynamic road networks, both query points and query objects are in a moving state, so static kNN query cannot dynamically monitor the changes of road network. Continuous KNN query method can reflect real-time changes of dynamic road networks. In order to handle continuous kANN issue of moving objects in road networks more effectively, this paper proposes a candidate object processing algorithm on the basis of MSO model. Candidate objects are cut short at the pruning stage in this algorithm.

The objects continuously move in the road network within the time interval [t0, tn]. Therefore, the time interval is divided into several subintervals at first and the direction of moving object is certain in these time subintervals. At the starting time t0, as for the previous k moving objects with the smallest value of aggregate function of query points, we will calculate the time required by each object and query point to move to a cross node in road network at a certain moving velocity. After various time points are gained, the fastest (smallest) time point will be selected among them, recorded as t1; thus the time interval [t0, tn] is divided into two subintervals: [t0, t1] and [t1, tn]. Next, the subinterval [t1, tn] is divided into [t1, t2] and [t2, tn] with the same method. This process will be repeated till the smallest time point is greater than time tn. In this way, the time interval time interval can be divided into [t0,t1], [t1,t2],…, [ti-1, ti],…, [tn-1,tn].

In terms of the time subinterval [ti-1, ti], suppose that the previous k moving objects with the smallest value of aggregate function of the query point q at time ti-1 are o1, o2,…,ok. These k objects are sorted in an ascending order of distance to the query point q. By detecting the moving state moving_state of these objects and query point q, we will find the object whose moving_state is away, and this object should be the last object whose moving_state is away in the orderly results, recorded as oi. As for those objects whose moving_state is closer before oi, they cannot be further away from the query point q when compared with object oi. The set of these objects is expressed as OSn. In the process of calculating the monitoring distance, there is no need to work out the distance between them and query points, so the calculation cost is reduced.

For each object oj in the set {o1, o2,…,ok}-OSn within the time interval [ti-1, ti], the distance Dq,oj(ti) between object oj and query point q is calculated as follows according to different moving states qo.moving_state of the object and query point.

(1) If qo.moving_state= closer, then $D_{q,o}(ti) = |D_{q,o}(ti-1)-|q.v-o.v|\times(ti-ti-1)|$.

(2) If qo.moving_state= away, then $D_{q,o}(ti)= D_{q,o}(ti-1)+|q.v-o.v|\times(ti-ti-1)$.

In conclusion, pruning method has two advantages, as shown in the following:

(1) The calculation for pruning distance involves a lower cost;

(2) The pruning distance is smaller, so fewer candidate objects and better system performance can be brought about.

After the pruning stage, we can cut off objects that cannot be the query results and obtain a candidate object set within the given time interval [t0, tn]. The objects are continuous and moving, so dynamic update is required for the query results. In another word, the above process should be repeated dynamically to gain the target data point set. The description of the algorithm is given below.

Input: previous k moving objects O with the smallest value of aggregate function of the query point q, a number K, and a time interval [t0, tn]

Output: A set of objects O that are candidates for the kANN query

```
ti=t0;
while ti≤tn
  ti=min{the time t when o and q reach a node along their direction};
Time interval [t0, tn] is divided into [t0,t1], [t1,t2],…[ti-1,ti],….[tn-1,tn];
for each object o∈O and each time subintervals [ti-1,ti] do
  if (o.e= =q.e) // q and o are in same edge
   if (o.v>0 and q.v>0) or( o.v>0 and q.v>0) //directions are same
    if (q.v>o.v and |q.v|<|o.v|)
      qo.moving_state=closer; insert o to closer_object set;
    else if q.v<o.v and |q.v|>|o.v|
        qo.moving_state=away; insert o to away_object set;
       end if
    end if
  else   //directions are different
   if (q.v>0∧o.v<0)
     qo.moving_state=closer; insert o to closer_object set;
   end if
   if (q.v<0∧o.v>0)
     qo.moving_state=away; insert o to away_object set;
   end if
  end if
 else // q and o are in two different edges
   if (q.e and o.e are not distance determinate)
    qo.moving_state=uncertain_away;
   else
     o.e.spvianode= o.e.snode or o.e.enode;
```



```
q. e.spvianode= q.e.snode or q.e.enode;
if (o is getting closer to o.e.spvianode) ∧(q is getting closer to q. e.spvianode)
   qo.moving_state=closer; insert o to closer_object set;
else if (o is getting closer to o.e.spvianode) ∧(q is moving away from q. e.spvianode)
    if (|q.v|>|o.v|)
      qo.moving_state=away; insert o to away_object set ;
    else
      qo.moving_state=closer; insert o to closer_object set;
    end if
else if (o is moving away from o.e.spvianode) ∧(q is getting closer to q. e.spvianode)
    if (|q.v|>|o.v|)
     qo.moving_state=closer; insert o to closer_object set ;
    else
     qo.moving_state=away; insert o to away_object set ;
    end if
else
  qo.moving_state=away; insert o to away_object set ;
 end if
end if
end if // determining of moving_state value is finished
end for
```

5. Experimental Analysis

In this paper, experiments are made for sum aggregate function and max aggregate function respectively. The comparison algorithm is IER algorithm [11]. In literature [11], when the weight of sides in road network is the Euclidean distance between two endpoints, IER algorithm possesses the best performance when extending the road network. The dataset adopted is real road network: road network in California of America [14]. In the experiment, target data points are generated on the side at random, and the default density is 4%. The query points distribute in the road network randomly and the default density is 4%. K means the number of target data points, and the default value is 10. Moving objects are generated by using generator mentioned in literature [15]. The moving object moves at a random velocity between 0 mph and 70 mph. In road network, when an object o reaches node n along its moving direction, the next side toward which the object o moves will be selected randomly from sides of connecting node n. In each experiment, the experimental result is the average effect among 20 times. The contrasting algorithm is noted as algorithm b and the algorithm of this paper is noted as a. The algorithm of this paper is realized with C++, and it runs in Windows XP with E3400 2.59GHz processor and 2 G internal storage.

(1)Figure 2(a) shows performance change of the algorithm under density fluctuation of target data points. Experiments are made for sum aggregate function and max aggregate

function respectively. The comparison algorithm needs to extend more road network information, and the cpu time consumed increases substantially. This algorithm does not need to extend the road network through nodes and sides of the road network, and unnecessary road network distance calculation at the pruning stage is reduced. Therefore, the algorithm possesses high efficiency. In addition, both algorithms present a good performance when the density of target data points rises. With the increase of target data points, the scope to be selected decreases and the cpu time also reduces naturally.

(2)Figure 2(b) presents performance change of the algorithm under the changes of parameter k. Experiments are made for sum aggregate function and max aggregate function respectively. Generally speaking, the value of K does not have a great influence on the algorithm. As for the major reason, ANN result k+1 is often located near result k. Therefore, when k ANN results are gained, all query points can extend to ANN result k+1 after extension for several times.

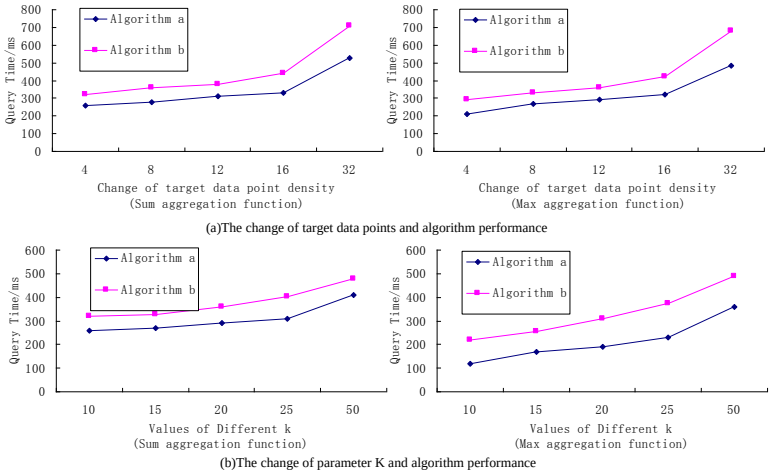

Figure 2. Experimental Results

6. Conclusion

As a variety of kNN query, kANN query based on road network has an extensive application prospect. By directing at the actual application background in which both query points and data points move in road networks, this paper introduces a moving state model of object and gives a formula of calculating road network distance. Besides, the candidate results are reduced via pruning method and an effective kANN query algorithm of moving object is designed. In the part of experiment, parameters that might affect the performance are tested. According to the results, method of this paper possesses high efficiency in various situations.

Acknowledgement

This work was supported by funds from Universities Key Fund of Anhui Province for Young Talents of China under Grant 2013SQRL082ZD, Natural Science Research Universities Key Project of Anhui Province of China under Grant KJ2014A256 and 2016 Anhui province colleges and universities outstanding young talent support program key projects of China under Grant gxyqZD2016319.

References

[1] N. Roussopoulos, S. Kelley and F. Vincent, "Nearest neighbor queries[C]//ACM sigmod record", ACM, vol. 24, no. 2, **(1995)**, pp. 71-79.
[2] K. L. Cheung and A. W. C. Fu, "Enhanced nearest neighbour search on the R-tree", ACM SIGMOD Record, vol. 27, no. 3, **(1998)**, pp. 16-21.
[3] G. R. Hjaltason and H. Samet, "Distance browsing in spatial databases", ACM Transactions on Database Systems (TODS), vol. 24, no. 2, **(1999)**, pp. 265-318.
[4] C. S. Jensen, J. Kolářvr and T. B. Pedersen, "Nearest neighbor queries in road networks", //Proceedings of the 11th ACM international symposium on Advances in geographic information systems. ACM, **(2003)**, pp. 1-8.
[5] D. Papadias, J. Zhang and N. Mamoulis, "Query processing in spatial network databases", //Proceedings of the 29th international conference on Very large data bases-Volume 29. VLDB Endowment, **(2003)**, pp. 802-813.
[6] M. R. Kolahdouzan and C. Shahabi, "Continuous K-Nearest Neighbor Queries in Spatial Network Databases", //STDBM, **(2004)**, pp. 33-40.
[7] H. J. Cho and C. W. Chung, "An efficient and scalable approach to CNN queries in a road network", //Proceedings of the 31st international conference on Very large data bases. VLDB Endowment, **(2005)**, pp. 865-876.
[8] K. Mouratidis, M. L. Yiu and D. Papadias, "Continuous nearest neighbor monitoring in road networks", //Proceedings of the 32nd international conference on Very large data bases. VLDB Endowment, **(2006)**. pp. 43-54.
[9] Y. K. Huang, Z. W. Chen and C. Lee, "Continuous k-nearest neighbor query over moving objects in road networks", //Advances in Data and Web Management. Springer Berlin Heidelberg, **(2009)**, pp. 27-38.
[10] D. Papadias, Y. Tao and K. Mouratidis, "Aggregate nearest neighbor queries in spatial databases", ACM Transactions on Database Systems (TODS), vol. 30, no. 2, **(2005)**, pp. 529-576.
[11] M. L. Yiu, N. Mamoulis and D. Papadias, "Aggregate nearest neighbor queries in road networks", Knowledge and Data Engineering, IEEE Transactions on, vol. 17, no. 6, **(2005)**, pp. 820-833.
[12] P. Fan, G. Li and L. Yuan, "Continuous K-Nearest Neighbor processing based on speed and direction of moving objects in a road network", Telecommunication systems, vol. 55, no. 3, **(2014)**, pp. 403-419.
[13] P. Fan, G. Li and L. Yuan, "Vague continuous K-nearest neighbor queries over moving objects with uncertain velocity in road networks", Information Systems, vol. 37, no. 1, **(2012)**, pp. 13-32.
[14] Real Datasets for Spatial Databases: Road Networks and Points of Interest[EB/OL].http://www.cs.fsu.edu/~lifeifei/SpatialDataset.htm
[15] T. Brinkhoff, "A framework for generating network-based moving objects", GeoInformatica, vol. 6, no. 2, **(2002)**, pp. 153-180.
[16] F. Ping, "The Algorithm Research on Continuous K Nearest Neighbor Query Considered Moving State in Road Networks", Wuhan, Huazhong University of Science and Technology, **(2012)**.

Author

Chen Wen, is an Associate Professor in the School of Mathematics and Computer Science, Tongling College, Tongling, P. R. China. He holds a master degree in Computer Science and Technology from the Anhui University, Anhui, P. R. China. His previous research areas include privacy preserving.

