Research on K Nearest Neighbor Skyline Query in Time Dependent Road Network

Song Lixiang 1*, Fei Ke 1
1 Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu, 210016, China
*Corresponding author’s e-mail: justlxsong@nuaa.edu.cn

Abstract. In recent years, with the increase for query preference, skyline query in road networks has gradually become a research hot topic. In order to help users get the desired query result, we propose a new algorithm. Firstly, we select landmark heuristically and establish corresponding shortest path tree (SPT). Then we construct a skyline matrix index called time dependent matrix index (TDMI), which help us associate the updated result set with dynamic k nearest neighbor (KNN) and simplify the non-space dominance relationship better. Finally, with SPT, TDMI, KNN and dynamic pruning conditions, an algorithm, the TDMI-KS algorithm, is proposed to rapidly answer the query result. Extensive experiments using Oldenburg road network datasets demonstrate the effectiveness and the efficiency of our proposed TDMI-KS algorithms.

1. Introduction
Since the 21st century, with the popularity of GPS and various mobile positioning devices, there are more and more data containing location information. For example, users can query more and more different kinds of object data points through mobile phones, such as hotels, restaurants, and cinemas. At the same time, skyline result set plays a very important role in many applications, which is an effective solution and important method for multi-objective optimization problems.

Skyline queries containing location information data include static attributes [1], Euclidean space [2] and road network [5]. In the road network, there is little research on the related query of the static road network, and the time-dependent road network for the dynamic road network. In a static road network, whenever a query is made, the result is the same. The time-dependent road network is different, and the influence of external factors on the side weight of the road network needs to be considered, which is more in line with the actual situation.

In addition to the direct value of each attribute, there are also attributes that cannot be compared with each other. Skyline query is to compare the values of each one-dimensional attribute, which requires the user's preference to be known in advance. For numerical attributes such as the price and score of restaurants, they can be compared directly. For classified attributes such as the business district where the restaurant is located, because each user has different preferences, they cannot be compared directly. Using existing methods to deal with this problem will lead to the problem of uncontrollable size of the result set [3], resulting in extremely low value of the query results. In order to improve the result quality, we must try to obtain the user preference information.

Therefore, we propose a KNN-based algorithm called TDMI-KS to answer KNN Skyline query in time dependent road networks (TD-KSQ). Firstly, we select landmark heuristically and establish corresponding SPT, so that the dynamic maintenance time depends on the shortest path in TDRN. After considering the actual needs of users, and uncontrollable result set problem, the object point attribute
dominance matrix is constructed through multiple interactions with users, and quickly index the preference attributes of the object points and associate the updated result set. We combine with K-nearest neighbor query method to prune result set and rapidly evaluate the query results on TDRN. A multi-dimensional comparative analysis was performed on the German Oldenburg road network dataset with existing algorithm. Under different parameters, the experimental results show that the TDMI-KS algorithm reduces storage overhead and greatly improves query efficiency.

2. K Nearest Neighbour-Skyline Query Based in Time Dependent Road Network

2.1. Shortest path tree on time dependent road networks
Depending on the time-dependent nature of the road network, the road distance of the object point can be changed, and the order of the object points can also be changed. The dynamic road network is updated with the shortest path tree on the time-dependent road network.

Updating the SPT will perform two operations according to the change of the side weight. One processing edge weight increases, and the other processing edge weight decreases. When the edge weight increases, the calculation of adding/removing elements and searching for the minimum min_d in the edge set Q_SPT can be simplified by maintaining the set M. When the edge weight is reduced, it is similar to the increase operation.

When the edge weight decreases and affects the original SPT, it is also necessary to perform the original SPT of the marker point. Because it is impossible to specify which nodes to update, the node set M is no longer used. Since the shortest distances of all nodes in des(j) are reduced, all nodes in des(j) will be updated. If the edge affects the original SPT, add Q_SPT. The terminal node T(e) selects the edge e’ with the smallest value-added min_d. If min_d is negative, the edge e and its min_d are added to the edge set Q_SPT. The created edge set Q_SPT contains all key edges.

2.2. Skyline matrix on time dependent road networks
The skyline matrix mainly solves two problems: how to select two attribute values for users to compare, and the attribute value pair should reduce the scale of skyline result set as much as possible; how to update skyline result set quickly by using the association relationship between multiple iterations after obtaining the user's preference relationship on some two attribute values, instead of updating skyline result set The entire object data set performs repeated skyline query calculations.

Firstly, the dominated condition table of object data points is established according to the original data set, which lays the foundation for subsequent interactive skyline query. As shown in Table 1, the initial SP result set is \{op_1, op_2, op_3, op_4, op_6, op_8, op_9\}, and the three object points are eliminated, mainly because the attribute value of the object point is weak and it is dominated by other object points. In the following table, “>” means stronger attribute value, “<” means weaker attribute value, “∧” means “AND”, and “∨” means “OR”. The average price of the original online ticket purchase price and the types of popcorn in the first two columns can be directly compared.

From Table 1, the governing conditions can still be reduced. For example, in the governing conditions of op_2, condition 2 (IMAX>Dolby ∧ Street JY>Street DJ) includes condition 3 (IMAX>Dolby), based on the principle of simplification, Condition 2 can be deleted. The algorithm can judge the attribute dominance relationship of the object point according to the following table to obtain the priority, thereby reducing the object point and directly eliminating the global calculation. But this still fails to meet the idea that was put forward at the beginning. It only gives the dominance relationship corresponding to each object point. It does not involve user interaction, nor does it defeat our original intention to further reduce the final SP result set, so we further generate its object points. The attribute dominating matrix, if there are m attribute values in the nth dimension, a matrix of \(m \times m\) scale is generated, and the i-th row j is a list \(attribute_{i,j} > attribute_{j,i}\), which can be directly reduced, as shown in Table 2.
Table 1. Object point dominated condition

| Object | Conditions |
|--------|------------|
| \( op_1 \) | \((\text{IMAX}>\text{Dolby} \land \text{Street JY}>\text{Street WD})\) |
| \( op_2 \) | \((\text{Street WD}>\text{Street DJ}) \lor (\text{IMAX}>\text{Dolby} \land \text{Street JY}>\text{Street WD}) \lor (\text{IMAX}>\text{Dolby} \land \text{Street JY}>\text{Street DJ}) \lor (\text{IMAX}>\text{Dolby}) \lor (\text{Leather}>\text{Dolby} \land \text{Street JY}>\text{Street DJ})\) |
| \( op_3 \) | \((\text{Dolby}>\text{Leather}) \lor (\text{IMAX}>\text{Leather} \land \text{Street JY}>\text{Street WD})\) |
| \( op_4 \) | \((\text{Leather}>\text{IMAX} \land \text{Street JY}>\text{Street DJ})\) |
| \( op_6 \) | \((\text{Dolby}>\text{Leather} \land \text{Street WD}>\text{Street DJ}) \lor (\text{Street WD}>\text{Street DJ}) \lor (\text{IMAX}>\text{Leather} \land \text{Street JY}>\text{Street WD}) \lor (\text{IMAX}>\text{Leather} \land \text{Street JY}>\text{Street DJ})\) |

Table 2. Matrix index

| Street JY | Street WD | Street DJ |
|----------|----------|----------|
| \( op_1, op_3 \) | \( op_2, op_4, op_8 \) | \( op_6, op_8 \) |

(a)

| IMAX | Dolby | Leather |
|------|-------|---------|
| \( op_1, op_3 \) | \( op_2, op_3 \) | \( op_6 \) |

(b)

The index of the object point dominated matrix can be generated according to the dominated attribute of the object point. In the specific implementation, the input of the object point attribute dominance matrix construction algorithm is the initial object point attribute dominance table \( T_{ab} \). The dominance matrix construction algorithm sequentially traverses the attribute dominance relationship for each object point in the attribute dominance table. Generate the initial matrix according to the number of user preference attribute values; then expand the search attribute dominance relationship table.

According to the attribute dominance matrix of the object point given above, the next will be given to the query user to compare and select the best attribute information. It can be deduced that the average number of reduced object points generated by the two values A and B in a certain preference attribute information dimension of the object point is recorded as \( \frac{N_{A>B} + N_{A<B} + P(A>B)N_{A>B} + P(A<B)N_{A<B}}{2} \), get

\[
N_{A,B} = \frac{N_{A>B} + N_{A<B}}{2}.
\]

The minimum number of reduced object points \( \text{Min}_{A,B} = \min(N_{A>B}, N_{A<B}) \) produced by the two values A and B in the dimension of the preference attribute information of the object point. The strategy for selecting attribute values to be judged by query users is as follows:

Event 1: If there is only one pair of preference attribute value \( \bar{N}_{A,B} \), it will be directly handed over to the query user to make a judgment result.

Event 2: The smallest number of reduced object points \( \text{Min}_{A,B} \) is calculated on the basis of the largest \( \bar{N}_{A,B} \), and then this set of preference attribute values is handed over to the querying user for judgment. If there are still multiple groups, they will be judged randomly.

After querying users to give and submit their own preference attribute information, the algorithm will iteratively update the original preference attribute information based on this. The input is the preference information \( \text{relation}_{x,y} \) fed back by the user, the reduced object point attribute dominance matrix index and the temporary Skyline set \( \text{Res}_{\text{temp}} \) are output as the updated set \( \text{Res}_{\text{temp}} \). First, according to the pros and cons relationship between x and y, find out the object point set \( op_1 \) that may be deleted and the set \( op_2 \) that cannot be deleted. If the object point p can be eliminated directly, all information related to \( op \) in \( T_s \) and \( \text{Ind}_m \) is deleted; if \( op \) is less than one condition from being reduced, it is updated Matrix index, delete \( \text{relation}_{x,y} \) in \( T_s \). For the object points that need to satisfy \( \text{relation}_{x,y} \) to
be eliminated, delete all values in the corresponding positions in $T_s$ and $Ind_m$; return the updated set $Res_temp$.

2.3. TDMI-KS algorithm

The main concern is to maintain the road distance sequence of these k skyline object points. First obtain the initial skyline object point set according to the POI, and generate the corresponding object point attribute dominating matrix index; then judge the number K and the index is not empty, and then the relation $x, y$ will be handed over to the query user for comparison, and the Skyline result set will be updated continuously.

When the extended node queries, the more likely it is to be the nearest neighbor of Q, the higher the priority of POI, and the one with higher priority will be extended the earliest. Based on the triangle inequality and the dynamic shortest path tree with marked nodes, the following heuristic function is constructed to guide the search, so that the path planning of each vertex always tends to the nearest poi:

$$f(n) = \max_{l \in L} \{d(l, t) - d(l, n), d(n, l) - d(t, l)\}$$ (1)

$f(n)$ denotes the estimate of node n to t, the landmark $L \in L$ is obtained by the difference between the two sides of the triangle is less than the third side $d(n, t) \geq d(l, t) - d(l, n)$ and $d(n, t) \geq d(n, l) - d(t, l)$, after traversing all marked nodes, $f(n)$ is taken as the maximum value.

In the subsequent refinement and maintenance phase, the k-th skyline object point is used as the boundary point to process two different event processing procedures, which are the skyline collection update and the K nearest neighbor sequence update. The KNN sequence is used for maintenance and combined with SPT. The minimum common ancestor and conditions are pruned to further optimize the update process, thereby completing the maintenance of the K nearest neighbor skyline result set of the entire event segment.

3. Experimental results and analysis

All the experiments were implemented by C++, running on with Windows 10 operating system, Intel Core i5-3470 CPU @ 3.20Ghz processor, 8GB memory. The compiler is GNU compiler collection.

| Table 3. Experimental parameters |
|----------------------------------|
| **Parameter** | **Default** | **Range** |
| Dataset cardinality (k) | 3.5 | 1, 1.5, 2, 2.5, 3, 3.5 |
| POI set cardinality (k) | 20 | 5, 10, 15, 20, 25, 30 |
| Non spatial dimension | 2 | 1, 2, 3, 4, 5, 6 |
| K | 5 | 1, 5, 10, 15, 20 |

We use Brinkhoff’s moving object generator [4] to create a comprehensive data set, and select Oldenburg road network in the example for simulation experiment. The weights are given to the edges every 5 minutes. The continuous K-nearest neighbor skyline query of moving objects researched by Huang [5] is the closest research to this paper so far. The CKNN-SQ+ algorithm is based on grid index, and at the same time, a given query path is required for query. In order to minimize the difference caused by randomness, 30 tests are carried out in the same environment, and the average of the remaining 28 tests after removing the maximum and minimum values is taken as the reference index of algorithm performance.
Figure 1. Comparative experiment results
Figure (a) and (b) compares the CPU time and I/O overhead of the two algorithms when the base of the data set increases. Both CPU time and I/O overhead are positively correlated with the base of the data set. Figure (c) and (d) shows the impact of POI base increase on the CPU time and I/O of the two algorithms. The CPU time of the TDMI-KS algorithm is always less than that of the CKNN-SQ+ algorithm.

Figure (e) and (f) shows the impact of the increase of the non-spatial dimension of the data set on the CPU time and I/O overhead of the two algorithms. The CPU time of the TDMI-KS algorithm is always less than that of the CKNN-SQ+ algorithm. Figure (g) compares the CPU time of the two algorithms when the value of k in the experimental environment increases. Figure (h) shows the comparison of the changes in I/O overhead as the value of k increases. As the value of k changes, the CPU time and I/O cost of all algorithms are positively correlated with the value of k. The number of nodes traversed by the TDMI-KS algorithm increases slowly as the value of k increases, and its performance is far better than that of the CKNN-SQ+ algorithm.

4. Conclusion
In this paper, we try to evaluate the TDRN-KSQ in dynamic road networks. We propose TDMI-KS algorithm to solve the problems of uncontrollable demand and result set of skyline query on time-dependent road network and underutilization of user interaction attribute dominance. To achieve this goal, TDRN-SPT and TDRN-MI are utilized to keep information of time dependent road networks.

In future, we would like to improve the algorithm, adapt to the cloud environment, and perform parallel optimization.

Acknowledgments
This work was supported by the Science and Technology Project (SGTYHT/17-JS-199).

References
[1] Papadias D, Tao Y, Fu G, et al. Progressive Skyline computation in database systems[J]. Acm Transactions on Database Systems, 2005, 30(1):41-82.
[2] Safar M, El-Amin D, Taniar D. Optimized Skyline queries on road networks using nearest neighbors[J]. Personal & Ubiquitous Computing, 2011, 15(8):845-856.
[3] Wong C W, Pei J, Fu W C, et al. Online Skyline Analysis with Dynamic Preferences on Nominal Attributes[J]. IEEE Transactions on Knowledge & Data Engineering, 2008, 21(1):35-49.
[4] Brinkhoff T. A Framework for Generating Network-Based Moving Objects[J]. Geoinformatica, 2002, 6(2):153-180.
[5] Huang Y K, Chang C H, Lee C. Continuous distance-based skyline queries in road networks[J]. Information Systems, 2012, 37(7): 611-633.