International Journal of Database Theory and Application
Vol.9, No.3 (2016), pp.151-160
http://dx.doi.org/10.14257/ijdta.2016.9.3.16

An Effective K-Nearest Neighbor Track Retrieval Algorithm

Chen Wen

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

Abstract

Due to the mass track data accumulated day by day, new challenges are raised for traditional information retrieval. This paper studies the issue of k-nearest neighbor track retrieval facing moving object, and converts this issue into aggregate Top-k query issue of information retrieval field. A parallel TA algorithm in random access database is proposed, and it has effectively solved the issue of k-nearest neighbor track retrieval. Performance of this algorithm is verified through a large number of experiments.

Keywords*: Mobile object; k-nearest neighboring trajectories query; aggregate Top-k query; parallel algorithm*

1. Introduction

In numerous application fields covering intelligent transportation and location-based service, mass track data are accumulated day by day. In these applications, people hope to find some historical tracks that pass some certain locations from the historical tracks in most cases. For instance, users of social network sites want to find routes that pass some scenic spots as references for their travel plan from their friends' tracks; traffic management departments hope to find and analyze tracks that pass some important intersections from the historical tracks of taxi; biologists might be interested in tracks that run across some mountainous regions, lakes and forests in the migration tracks of migratory birds. Generally speaking, all the above applications need to efficiently retrieve tracks that pass some certain locations among the mass track data stored in the disk.

2. Relevant Work

Track-based query was proposed by Pfoser *et al.* [1] for the first time. It includes track topological query and track navigation query. Track topological query involves motion information of moving objects like "entrance", "leaving" and "passing", and its common query form is "to search all tracks of the moving objects that enter, leave and pass the query region within the query time interval". The common form of track navigation query is "to search all tracks of the moving objects that intersect with the query region within the query time interval and that intersect with another query region within another query time interval". Pfoser *et al.* processed track-based query by adopting STR-tree and TB-tree index structure. According to a large number of experimental results, track query processing via TB-tree structure is obviously better than similar query processing via STR-tree structure in efficiency and expandability. Zhu *et al.* [2] proposed the OP-tree index structure and evaluated performance of track query based on this structure.

Similar track query aims to seek similar tracks of the moving objects. Due to its importance of application in decision support and data mining fields, Scholars in the field of database also starts to study this issue during recent years. At present, research work of this aspect is mainly as follows. Vlachos *et al.* [3] compared the similarity among tracks by extracting the longest common subsequence in tracks. This method has avoided the continuity requirement of mapping and it is more flexible, but monotonic mapping between two tracks is required. Yanagisawa *et al.* [4] proposed a similar track query algorithm based on shape. This algorithm is based on Euclidean distance and only applicable to tracks with the same length or same time interval. In addition, they also proposed a distance measurement that allowed searching for similar tracks under translation, scaling and rotation transformation. Lin *et al.* [5] studied the similar space shape query issue of the moving object tracks. One-way distance (OWD) function is introduced to compare the space shapes among tracks. However, they only considered space information of the moving object tracks, but ignored time information. Chen *et al.* [6] introduced a real sequence editing distance function and proposed several pruning strategies to improve the similar query algorithm performance of moving objects. However, mapping between tracks in these algorithms is still continuous and monotonous. The above research work emphasizes space similarity among tracks but ignores the time information. Besides, they suppose that the tracks have the same length and sampling rate. In view of this, Frentzos *et al.* [7] studied the issue of the most similar track query based on structures similar to R-tree. In recent years, research scholars have considered uncertainty in studies on query processing for the moving object tracks. At present, research work in this aspect is as follows. Cheng *et al.* [8] proposed uncertain track query for the first time and designed effective probability time slice query, probability time slice nearest neighbor query and probability clustering query algorithms. However, these algorithms are only applicable to time slice query, and cannot be extended to time interval query easily. Therefore, Mokhtar *et al.* [9] discussed the issue of uncertain track query directing at time interval query.

All in all, domestic and overseas research scholars have made a certain progress in studies on track query, but these researches do not involve track query for some novel historical moving objects that meets the practical application demand. Therefore, under such research background, it is significant and necessary to study query processing technology of historical moving object tracks.

3. Relevant Definition

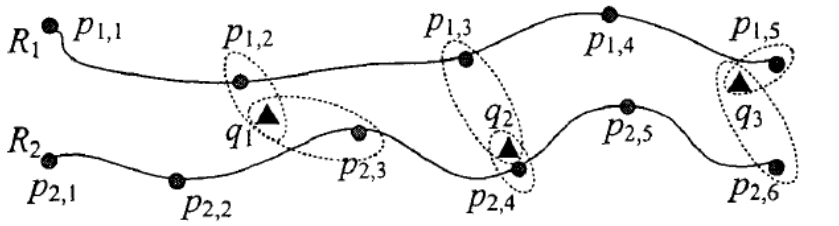
Definition 1: Distance between track and query point. A track R_i can represent a point sequence $R_i = \{p_{i,1}, p_{i,2}, \dots, p_{i,n}\}$, where $p_{i,j}$ is the point j of R_i. As for the given query point q, the matching pair between point j of the track $p_{i,j}$ and q is recorded as <$p_{i,j}, q$>. If $\mathrm{dis}(p_{i,j}) \le \mathrm{dis}(p_{i,k})$ for any $p_{i,k} \ne p_{i,j}$, then $p_{i,j}$ is called projection of q in R_i, recorded as $p_{i,j} = q \rightarrow |R_i$. The distance between track R_i and query q is defined as the distance between q and its projection in R_i. The distance between these two points can be calculated with Euclidean distance, great circle distance of earth surface or road network distance (shortest path).

Definition 2: Distance between track and query point set. Track $R_i = \{p_{i,1}, p_{i,2}, \dots, p_{i,n}\}$ and query point set $Q = \{q_1, q_2, \dots, q_m\}$ are given. Suppose that the distance aggregate function $t(x_1, x_2, \dots, x_m)$ is a m-dimensional strictly monotone increasing function, thus the distance between track R_i and query point set *Q* is defined as:

$$\begin{aligned}\mathrm{dis}(R_i, Q) &= t(dis(R_i, q_1), dis(R_i, q_2), \dots, dis(R_i, q_m)) \\ &= t(dist(q_1 \rightarrow |R_i, q_1), dist(q_2 \rightarrow |R_i, q_2), \dots, dist(q_m \rightarrow |R_i, q_m))\end{aligned}$$

In different applications, different distance aggregate functions might be used to describe the similarity degree of track and query point set. The above issue is illustrated with examples in literature [10]. Figure 1 presents two tracks R_1 and R_2 as well as the query point set $Q = \{q_1, q_2, q_3\}$. The projections of q_1, q_2 *and* q_3 in R_1 are $p_{1,2}, p_{1,3}$ *and* $p_{1,5}$ respectively, and their projections in R_2 are

$p_{2,3}$, $p_{2,4}$ *and* $p_{2,6}$ separately. In this example, Euclidean distance is used to calculate the distance between two points and summation function is treated as distance aggregate function, so as to obtain the distance between these projections and the query point as well as the distance between tracks R_1 & R_2 and Q. According to the results, though q_1 and q_3 are closer to R_1, the aggregate distance of R_2 is smaller than that of R_1. Therefore, R_2 is the track the closest to the query point set.



$dist(R_1, q_1) = dist(p_{1,2}, q_1) = 20$ m	$dist(R_2, q_1) = dist(p_{2,3}, q_1) = 30$ m
$dist(R_1, q_1) = dist(p_{1,3}, q_2) = 50$ m	$dist(R_2, q_1) = dist(p_{2,4}, q_2) = 5$ m
$dist(R_1, q_1) = dist(p_{1,5}, q_3) = 15$ m	$dist(R_2, q_1) = dist(p_{2,6}, q_3) = 40$ m
$dist(R_1, Q) = 85$ m	$dist(R_2, Q) = 75$ m

Figure 1. Distance Between Track and Query Point Set [10]

Definition 3: K-nearest neighbor track retrieval (kNNT). The track database D and query point set Q are given. K-nearest neighbor track retrieval aims to find out k tracks K={R_1, R_2, ..., R_k} to make any $R_i \in K$ and $R_j \notin K$ meet $dis(R_i, Q) \leq dis(R_j, Q)$.

The challenges of efficient kNNT retrieval for traditional KNN retrieval are mainly reflected in the following aspects. (1) Sharp increase of data scale. Compared with traditional KNN aimed at point retrieval, track data have increased time dimensions, so the data size is far greater. (2) Distance calculation. The distance calculation of track and query point set is different from that of traditional KNN retrieval, and the result should be obtained via distance aggregate function. When different aggregate distance functions are applied, the retrieval results will also be different and the traditional preprocessing method is infeasible. In addition, if road network distance is adopted between track point and query point, the calculation cost will be increased, so online distance calculation becomes infeasible. (3) Non-uniform distribution. In many practical applications, data distribution is non-uniform. For instance, the taxi track might be dense in cities but sparse in the suburbs. Progressive increase retrieval method in traditional KNN does not perform well in such data.

4. kNNT is Modeled Into Aggregate Top-k Query

Definition 4: Suppose that there are n objects among which any object R_i has m attributes; a m-dimensional monotone aggregate function is given and the attribute list L_1, L_2, ..., L_m includes m sequence lists. Among them, any list stores the value of attribute j for all the n objects in 2-tuple form from large attribute value to small attribute value. Aggregate Top-k query aims to find k objects with the biggest aggregate value from the m lists.

According to definition 4, the issue of this study can be converted into the issue of aggregate Top-k query. The n objects are corresponding to all n tracks in the database, and each attribute list is corresponding to the reverse index by setting the index node where the query point is located as root node in track index (based on R-tree or similar structures). In practice, the attribute values can be normalized into the range of [0-1]. The distance from track to the query point is corresponding to aggregate value in Top-k. In this way, kNNT retrieval is actually a special example in Top-k query.

Efficient Top-k query plays a key role in multiple fields covering information retrieval [11-12], multimedia database [13-14], data mining [15] and network service [16-18]. Therefore, Top-k query have become the hot issue concerned by many researchers in recent years. For instance, in information retrieval field, users might enter multiple search terms, but the results about relevancy sorting directing at each term should be integrated before they are presented to users. In image retrieval, there will be a sorting result according to information like color histogram, edge histogram and texture by directing at images entered by users, and the final results need to be fed back to users through aggregate query.

Many researches have been done for Top-k query allowing random access. The earliest and most famous method is threshold algorithm (TA) [19]. Most TA-type algorithms suppose that the data support random access and sequential access at the same time, and the aggregate function is monotone function. Meanwhile, some works have studied query under non-monotone function situation [20-21]. In literature [22], the author proposed a unified framework of the current TA-type algorithms, and put forward a query algorithm of supporting random access database.

Under the situation where only sequential access is allowed, Guntzer *et al*. [13] proposed Stream-Combine algorithm. Meanwhile, Fagin [19] put forward no random access (NRA) algorithm. Stream-Combine algorithm only considers the upper limit of all objects and it can determine whether the object belongs to Top-k only under the situation where all attributes of one object are visited. In this sense, NRA algorithm is better than Stream-Combine algorithm. Theobald *et al*. [23] proposed a series of probability algorithms based on NRA algorithm to solve approximate Top-k. Mamoulis *et al*. [24] put forward a novel LARA algorithm based on “lattice” structure by studying algorithm behaviors. From the aspect of operation time cost, the query speed of LARA algorithm is obviously higher than that of NRA. However, it has no obvious advantages in visiting cost when compared with NRA. Gursky [25] proposed a group of 3-Phase NRA algorithms which improved the query time by utilizing heuristic optimization.

During aggregate Top-k query, some databases do not support random access or random access should be avoided as far as possible due to the high cost [19]. For instance, typical search engines cannot directly gain the score of relevancy between the search term and a certain text through random access when users enter the term. For another example, if the attribute list is the intermediate result gained via other operation or entered in data flow form, random access is unpractical. Under some situations, the cost of random access is greatly higher than that of sequential access. In literature [26], the author discussed why the cost of random access was higher than the cost of sequential access in many applications. By directing at kNNT, if random access is conducted for the distance from a certain query point to a given track, then the point sequence of the track should be searched in the database and the projection of this query point in the track must be determined by frequently calculating the distance between two points. The visiting cost of such operation in database based on R-tree index is too high. If the distance between two points is calculated by adopting road network distance, the calculation cost will be increased. Therefore, random access is almost impossible.

5. Parallel TA Algorithm

In order to solve the mass data of historical tracks in kNNT, this paper proposes a parallel TA algorithm on the basis of TA algorithm by fully utilizing the parallelism of data processing.

In previous studies on centralized multimedia database, researchers often transfer middleware realization issue into top-k query issue and complete middleware system design by studying top-k query algorithm. In such top-k query issue transferred from middleware system, the system supports sequential access and random access at the same time. A famous centralized multimedia database is Garlic system of IBM Company. Fagin *et al*. designed a middleware module for Galic system on the basis of threshold algorithm (TA), and verified the correctness and case optimality of TA algorithm.

In order to process top-k query of mass data by utilizing cluster system, the problem of data partitioning should be solved at first. For a data space DB composed of m n-tuple lists, there are two partitioning methods: list partitioning and ID partitioning.

1. List partitioning. The whole data space is partitioned by setting list as the unit. The list space is partitioned into several subspaces that do not intersect with each other. Each subspace contains a list subset.

2. ID partitioning. The whole data space is partitioned by setting tuple ID as the unit. The object ID set is partitioned into several subsets that do not intersect with each other. Each subspace contains all attributes of ID subset.

The above two partitioning methods have advantages and disadvantages in partitioning cost and partitioning effect. In terms of list partitioning, attributes of the object are saved according to list. Therefore, we just need to move all lists in the global space into the subspace and the partitioning cost is low. On the other hand, number of lists processed by top-k query is limited in practical application. The partitioning quantity is small, so the degree of parallelism is low. Meanwhile, list partitioning will separate the connection among different attributes of the object, and different attributes of the same object will be inevitably in different subspaces after partitioning. When parallel algorithm maps the subspace to the actual processor set, these attributes will be scattered in the storage space of various processors. A large amount of cluster communication should be introduced when the aggregate score of the object is calculated. As a result, the speed-up ratio of the calculation process is low and the system efficiency is not high. Therefore, top-k query research aimed at vertical partitioning is often based on coarse-grained task level parallelization, and the existing distributed algorithms can be used as references.

For ID partitioning, ID set of the object should be decomposed at first and then all lists should be scanned according to subsets after ID decomposition. Thus the cost is high. However, after decomposition is completed, number of objects n is often higher than number of lists by several orders of magnitudes. The higher the n value is, the finer the partitioning of the global space will be. It is beneficial to follow-up data mapping and the data after partitioning possess good parallelism. Meanwhile, ID partitioning will not separate attributes of the object and all attributes of the same object are located in the same subspace. When parallel algorithm maps the subspace to the actual processor set, these attributes will be allocated to the same processor storage space (stored in different lists). Only local calculation is required when the aggregate score of the object is calculated. There is no need to introduce cluster communication and the algorithm supports asynchronous processing, so the communication cost is low. Therefore, ID partitioning mode of data is beneficial to parallel algorithm design based on message passing model.

In view of the advantages and disadvantages of the above two partitioning methods, data processing of list partitioning is applicable to coarse-grained parallel development and data processing of ID partitioning is suitable for fine-grained parallel development. There have already been relatively mature distributed algorithms aimed at list partitioning, so our study is mainly conducted by directing at parallel processing of data after ID partitioning.

ID partitioning method divides big data space into a series of data subspaces that do not intersect with each other. The global data space is recorded as DB. If the cluster system has p processors P_1, P_2, …, P_p and the subspace composed of data allocated to the processor Pi is recorded as $\mathrm{Sub}_i\,\mathrm{DB}$, then DB $=\bigcup_{1\le i\le p}\mathrm{Sub}_i\,\mathrm{DB}$ and $\mathrm{Sub}_i\,\mathrm{DB}\cap\mathrm{Sub}_j\,\mathrm{DB}=\emptyset$, ($i\neq j$).

Ak is set as top-k set in the global data space DB and $\mathrm{Sub}_i\,\mathrm{Ak}$ is recorded as local top-k set in the data subspace $\mathrm{Sub}_i\,\mathrm{DB}$, thus $\mathrm{Ak}\subseteq\bigcup_{1\le i\le p}\mathrm{Sub}_i\,\mathrm{Ak}$ and $\mathrm{Sub}_i\,\mathrm{Ak}\cap\mathrm{Sub}_j\,\mathrm{Ak}=\emptyset$, ($i\neq j$).

In order to solve top-k set of the global space, division method can be used to concurrently solve local top-k set in each subspace via each processor. Then the global top-k set can be solved by integrating local top-k sets of subspaces.

We have designed a top-k parallel TA algorithm in distributed storage structure according to the above steps. The execution flow of parallel TA algorithm is as follows. For p processors, each processor carries out sequential access for m lists in the subspace by way of round-robin. Once the new object o is read, all attribute sores pi(o) ($1\le i\le m$) of this tuple will be gained through m−1 random accesses, and the aggregate value F(o) = F(p1(o),…,pm(o)) of object o should be calculated. Each processor maintains its own candidate set $\mathrm{Sub}_i A_k$, so as to record k objects with maximum aggregate value in the subspace. The minimum aggregate value $\mathrm{Sub}_i M_k$ of candidate set in the subspace and threshold value $\mathrm{Sub}_i T$ in the subspace are calculated. When the minimum aggregate value $\mathrm{Sub}_i M_k$ of candidate set in the subspace is greater than the threshold value $\mathrm{Sub}_i T$ of list in the subspace, all processors will send local Top-k set $\mathrm{Sub}_i A_k$ in the subspace to P1 processor. P1 processor will output k objects with the maximum aggregate value among p∗k objects collected according to aggregate value sorting as Top-k set of the whole space.

Formal description of the algorithm is given in the following.

Input: DB composed of m sorted lists, and p processors

Output: Top-k set in DB

1. Allocate DB to the storage space of p processors via ID division method.

2. For (all Pi, where $1\le i\le p$)do

Conduct sequential access for each list in the subspace.

Calculate the aggregate value F(o) of each new object o.

Record the existing k objects with the maximum aggregate value in the subspace with candidate set $\mathrm{Sub}_i A_k$; if number of objects in the candidate set is smaller than k, return to step 1.

Calculate the minimum lower limit $\mathrm{Sub}_i M_k$ of candidate set $\mathrm{Sub}_i A_k$, $\mathrm{Sub}_i M_k = \min\{F(t) : t\in \mathrm{Sub}_i A_k\}$.

Calculate the threshold value $\mathrm{Sub}_i T$ in the list.

Compare the minimum lower limit $\mathrm{Sub}_i M_k$ and threshold value $\mathrm{Sub}_i T$; if $\mathrm{Sub}_i M_k < \mathrm{Sub}_i T$, return to step 1.

3. Root processor collect and sort the p ∗ m objects, and output k objects with maximum aggregate score.

6. Experimental Analysis

Experimental environment adopted by algorithm of this paper is IBM HPC calculation platform including 2 IBM X3650M3 management nodes, 28 two-way 8-core IBM BladeCenter HS22 blade servers, and 4 IBM X3550M3 rack servers that constitute 32 calculation nodes. All calculation nodes and management nodes are connected through a set of 40GB QDR Infiniband network switch, to operate Red Hat Enterprise Linux 5.5 and Apache Hadoop-0.21.

The algorithm is realized through C++ and contrast experiment is made in real data set. The data set comes from data of restaurants on Dianping.com (captured from Dianping.com). Dianping.com is the largest consumption guidance and comment website in China at present. Data set used in this experiment includes evaluation data related to the restaurants. Such data include 80,000 restaurants and cover 50 Chinese cities. Each restaurant has the following 6 attributes: star level (based on users' comments), price, number of users' comments, taste, environment and service. The data formula $(xi - Min)/(Max - Min)$ is normalized.

(1) Multiple common aggregate functions are tested in the experiment: average number (ave), weighted average (wav), sum of squares (squ), and exponential sum (exp). Figure 2(a) tests the visiting cost of the four aggregate functions in test data set. When k value increases gradually, visiting cost of the algorithm rises. Generally speaking, algorithm of this paper has relatively small visiting cost for all aggregate functions.

(2) Figure 2(b) tests the influences of the four different aggregate functions on the operation time under different k values. Parallel algorithm and random access proposed in this paper have very obvious advantages, and the consumption of cpu time can meet the requirements of practical application.

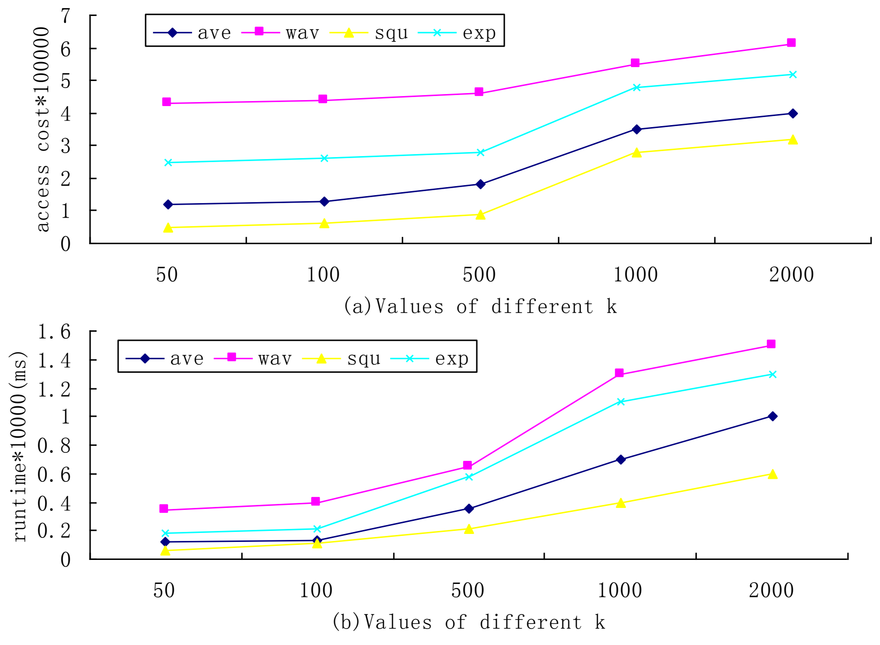


Figure 2. The Influence of Different Aggregate Functions on Access Cost and Running Time

7. Conclusion

In recent years, with the wide application of positioning devices in mobile terminals and extensive development of location-based service and mobile social networks, mass track data are accumulated day by day. Thus a great challenge is raised for management and utilization of mass track data. On the premise of organizing the current situations about track query studies, by combining with the existing literature thoughts, this paper converts the issue of nearest neighbor track retrieval into the issue of aggregate Top-k query. An efficient algorithm of parallel TA aggregate Top-k query is proposed, and high efficiency and validity of this algorithm are verified in real data set.

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] D. Pfoser, C. S. Jensen and Y. Theodoridis, “Novel approaches in query processing for moving object trajectories”, In Proceeding the 26th International Conference on Very Large Data Bases, Cairo, Egypt, **(2000)**, pp. 395-406.
[2] H. Zhu, J. Su and O. H. Ibarra, “Trajectory queries and octagons in moving object databases”, Proceedings of the eleventh international conference on Information and knowledge management. ACM, **(2002)**, pp. 413-421.
[3] M. Vlachos, G. Kollios and D. Gunopulos, “Discovering similar multidimensional trajectories”, Data Engineering, 2002. Proceedings. 18th International Conference on. IEEE, **(2002)**, pp. 673-684.
[4] Y. Yanagisawa, J. Akahani and T. Satoh, “T. Shape-based similarity query for trajectory of mobile objects”, Mobile data management. Springer Berlin Heidelberg, **(2003)**, pp. 63-77.
[5] B. Lin and J. Su, “Shapes based trajectory queries for moving objects”, Proceedings of the 13th annual ACM international workshop on Geographic information systems. ACM, **(2005)**, pp. 21-30.
[6] L. Chen, M. T. Özsu and V. Oria, “Robust and fast similarity search for moving object trajectories”, Proceedings of the 2005 ACM SIGMOD international conference on Management of data. ACM, **(2005)**, pp. 491-502.
[7] E. Frentzos, K. Gratsias and Y. Theodoridis, “Index-based most similar trajectory search”, Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on. IEEE, **(2007)**, pp. 816-825.
[8] R. Cheng, D. V. Kalashnikov and S. Prabhakar, “Querying imprecise data in moving object environments”, Knowledge and Data Engineering, IEEE Transactions on, vol. 16, no. 9, **(2004)**, pp. 1112-1127.
[9] H. M. O. Mokhtar and J. Su, “Universal trajectory queries for moving object databases”, Mobile Data Management, 2004. Proceedings. 2004 IEEE International Conference on. IEEE, **(2004)**, pp. 133-144.
[10] J. Yuan., “Querying, mining with applications on large-scale trajectory data”, Hefei, University of science and technology of China, **(2012)**.
[11] G. Salton, “Automatic text processing: The transformation, analysis, and retrieval”, Reading: Addison-Wesley, **(1989)**.
[12] X. Long and T. Suel, “Three-level caching for efficient query processing in large web search engines”, World Wide Web, vol. 9, no. 4, **(2006)**, pp. 369-395.
[13] U. Güntzer, W. T. Balke and W. Kießling, “Towards efficient multi-feature queries in heterogeneous environments”, Information Technology: Coding and Computing, 2001. Proceedings. International Conference on. IEEE, **(2001)**, pp. 622-628.
[14] S. Nepal and M. V. Ramakrishna, “Query processing issues in image (multimedia) databases”, Data Engineering, 1999. Proceedings, 15th International Conference on. IEEE, **(1999)**, pp. 22-29.
[15] L. Getoor and C. P. Diehl, “Link mining: a survey”, ACM SIGKDD Explorations Newsletter, vol. 7, no. 2, **(2005)**, pp. 3-12.
[16] M. Zhu, S. Shi and M. Li, “Effective Top-k computation in retrieving structured documents with term-proximity support”, Proceedings of the sixteenth ACM conference on Conference on information and knowledge management. ACM, **(2007)**, pp. 771-780.
[17] M. S. Scheuer, C. Li and Y. Mass, “Best-effort Top-k query processing under budgetary constraints”, Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on. IEEE, **(2009)**, pp. 928-939.
[18] W. T. Balke, U. Güntzer and W. Kießling, “On real-time top k querying for mobile services”, On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE. Springer Berlin Heidelberg, **(2002)**, pp. 125-143.
[19] R. Fagin, A. Lotem and M. Naor, “Optimal aggregation algorithms for middleware”, Journal of Computer and System Sciences, vol. 66, no. 4, **(2003)**, pp. 614-656.
[20] D. Xin, J. Han and K. C. Chang, “Progressive and selective merge: computing top-k with ad-hoc ranking functions”, Proceedings of the 2007 ACM SIGMOD international conference on Management of data. ACM, **(2007)**, pp. 103-114.
[21] Y. Luo, X. Lin and W. Wang, “Spark: top-k keyword query in relational databases”, Proceedings of the 2007 ACM SIGMOD international conference on Management of data. ACM, **(2007)**, pp. 115-126.
[22] S. Hwang and K. C. Chang, “Optimizing top-k queries for middleware access: A unified cost-based approach”, ACM Transactions on Database Systems (TODS), vol. 32, no. 1, **(2007)**, pp. 5.
[23] M. Theobald, G. Weikum and R. Schenkel, “Top-k query evaluation with probabilistic guarantees”, Proceedings of the Thirtieth international conference on Very large data bases. VLDB Endowment, vol. 30, **(2004)**, pp. 648-659.
[24] N. Mamoulis, K. H. Cheng and M. L. Yiu, “Efficient aggregation of ranked inputs”, Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on. IEEE, **(2006)**, pp. 72-72.

[25] P. Gurský and P. Vojtáš, “Speeding up the nra algorithm”, Scalable Uncertainty Management. Springer Berlin Heidelberg, **(2008)**, pp. 243-255.
[26] E. L. Wimmers, L. M. Haas and M. T. Roth, “Using Fagin's algorithm for merging ranked results in multimedia middleware”, Cooperative Information Systems, 1999. CoopIS'99. Proceedings. 1999 IFCIS International Conference on. IEEE, **(1999)**, pp. 267-278.

Author

Chen Wen, He 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.