Top-k Dominating Queries on Incremental Datasets*

Jimmy Ming-Tai Wu[1], Ke Wang[1], and Jerry Chun-Wei Lin[2]

[1] College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, China
wmt@wmt35.idv.tw, Wang_ke1998@126.com

[2] Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen, Norway
jerrylin@ieee.org

Abstract. Top-k dominance (TKD) query for incomplete datasets is a popular preference query for incomplete data, which analyzes the dominance relationships among objects in a dataset by a dominance method to reveal the top-k most valuable information in the dataset. At present, in-depth research has been conducted on this topic, and efficient query algorithms based on various pruning strategies have been proposed, as well as optimization algorithms based on a distributed computing framework for processing large-scale datasets. With the advent of the information age, data update iterations are accelerated, and in the face of dynamically updated data, the traditional TKD query algorithm based on static data can no longer meet our needs, and an efficient algorithm based on the dynamically updated data set environment is needed. In this paper, we conduct an in-depth study on the TKD query problem for dynamically updated incomplete datasets, and propose a dynamic update parallel algorithm based on MapReduce framework. The algorithm utilizes the query results of historical datasets, avoids the repeated analysis of the dominant relationships between historical objects, optimizes the computation process, reduces the space occupation, and proves through experiments that the dynamic update algorithm has more obvious advantages compared with the traditional algorithm.

Keywords: Incomplete datasets · Top-k dominance query · Static datasets · Dynamic update · Parallel compute.

1 Introduction

In the era of big data, data integrity among data quality issues has attracted a lot of attention. The data integrity problem may be the problem of missing data due to data hiding, transmission signal loss or other reasons. Incomplete datasets are common in real life, such as MovieLens, a movie evaluation system with several classic works, each movie can be represented by a tuple, and

* Supported by Shandong Provincial Natural Science Foundation (ZR201911150391).

the user's evaluation of it as an element in the tuple, because there is no guarantee that every user of the evaluation has seen all the movies in the system, so there must be some missing elements in the movie tuple. Due to the rise of incomplete datasets, the research on incomplete datasets has gradually become hot, including the research on incomplete relational database models [2, 8, 11]; The research on skyline queries based on incomplete data [7, 12, 15], where [12] firstly proposed the idea of skyline queries on incomplete datasets and designed a skyline query algorithm (ISkyline) for incomplete datasets; An in-depth study of the indexing problem on incomplete datasets [3, 17], in which two retrieval strategies for incomplete high-dimensional datasets are proposed and compared with exhaustive search, verifying that both retrieval strategies are efficient; And a study of the top-k query problem on incomplete datasets [9, 22], the top-k query problem on asynchronous incomplete data streams is studied in the literature [9] and an efficient query algorithm based on an object pruning strategy is proposed.

TKD query is an emerging preference query that is based on top-k query and skyline query. TKD query avoids the setting of scoring function compared to top-k query and can determine the number of results returned by the query compared to skyline query. TKD query was originally proposed by Papadias *et al.* [18] and proposed a branch-and-bound skyline (BBS) algorithm based on TKD queries to solve data mining problems on complete datasets. Next, Yiu *et al.* [27] proposed a TKD query algorithm for multidimensional complete datasets, in which a novel data structure aR-tree was designed for efficient data traversal, and the algorithm was experimentally proven to be effective. Tiakas *et al.* [23] investigated the use of an asymptotic approach to TKD query was investigated and various pruning strategies were devised and verified that the algorithm possesses better performance. With the rise of incomplete datasets, the study of TKD queries has been extended to incomplete datasets [1, 6, 16, 26], which includes the study of pruning strategies in performing TKD queries and how to implement TKD queries for large-scale incomplete datasets.

With the development of information technology, the update of data sets becomes frequent. How to efficiently perform data mining on dynamically updated data is a popular topic of much attention at present. The simplest way to solve this problem is to re-execute the traditional algorithm once for the updated dataset, but this solution not only does not make full use of the query results of the historical dataset, but also generates a large number of repeated calculation problems, which wastes a lot of time and space and causes untimely data query. Through in-depth research on this topic, a large number of related algorithms are proposed, including incremental association rule mining processing algorithms [4, 5, 10, 13, 20, 24, 25] and incremental sequence model mining algorithms [14, 19], which dynamically update the dataset mining algorithm utilizes the original query results, reduces the number of scans of the database, and improves query efficiency. Compared to dynamically updated complete datasets, the study of TKD queries for dynamically updated incomplete datasets is more difficult, be-

cause the problem of uncertain missing data in the dataset has to be considered on top of dealing with dynamically complete datasets.

Based on the consideration of the above problems, this paper proposes a TKD query algorithm for incremental update of incomplete data based on MapReduce architecture, which divides the analysis and calculation of incremental update dataset into two parts, the update of historical object dominant scores and the calculation of new object dominant scores, the method makes full use of the query results of historical dataset, avoids the repeated calculation of data, and achieves an efficient query processing. Moreover, the algorithm is based on MapReduce parallel architecture, which decomposes the complex analysis and calculation tasks into multiple subtasks assigned to different nodes for parallel calculation, and can realize the analysis and processing of large-scale incomplete datasets.

2 Literature review

This section describes top-k dominated queries for incomplete datasets and related work on dynamic incremental database data mining.

2.1 Top-k dominance query

Miao *et al.* [16] first started their research on the TKD query problem for incomplete data and proposed various algorithms (ESB, UBB, BIG, and IBIG) to solve the problem. ESB algorithm proposes the concept of bucket in order to apply the dominant transferability to TKD queries of incomplete data sets. It stores the objects in buckets according to the missing dimensions, and then prunes the objects in the buckets using the concept of k-skyband. UBB algorithm analyzes the dominance score of object in each dimension and uses the minimum dominance number as the dominance upper bound of the object and filters the objects in the dataset by this dominance upper bound first, avoiding a large number of unnecessary computational processes. BIG introduces the bitmap used for querying complete datasets to incomplete data sets, and uses the bitmap index to obtain the dominance score of objects by fast bit-by-bit calculation, which has a greater improvement in time consumption compared with the first two algorithms, but the algorithm will occupy a lot of memory and disk space due to the bitmap index and the storage of object sets. IBIG is an optimization of BIG algorithm, which effectively solves the problem of storage space occupation of BIG algorithm through bitmap compression technology and chunking strategy.

Ezatpoor *et al.* [6] found that most of the existing algorithms are effective for TKD queries on small incomplete datasets, but when the data size becomes large, the query task becomes difficult, and even traditional single machine query algorithms may fail the task due to lack of memory or disk space. Ezatpoor *et al.* performed the above problem in depth and proposed an algorithm based on MapReduce distributed framework (MRBIG). MRBIG is based on bitmap indexing, which decomposes some of the tasks originally executed on a single

machine into multiple subtasks and distributes them to different nodes for parallel computation, thus realizing the analytical computation of large-scale data sets.

Wu *et al.* [26] proposed two high-performance algorithms (EHBIG, IEHBIG) based on MapReduce architecture to improve the efficiency of TKD queries for large-scale incomplete datasets. EHBIG proposes the concept of maximum domination number of objects according to the domination relationship and designs an efficient pruning strategy to reduce the computational effort. However, EHBIG completes the query through MapReduce iterations, and when there are thousands of objects in the dataset, EHBIG may have to iterate through thousands of MapReduce tasks, which generates a large amount of resource consumption. IEHBIG algorithm computes the dominance scores of all objects in the dataset by one MapReduce task, avoiding the problem of resource wastage due to MapReduce iterations, but the algorithm is not designed with a pruning strategy and requires more memory space. The two algorithms proposed in the article have their own advantages and disadvantages, and need to be chosen in conjunction with reality.

2.2 Dynamic update of data mining

Cheung *et al.* [4] firstly studied the maintenance of association rules in incremental datasets and proposed the association rule mining algorithm (FUP) for incremental datasets, which is based on the logic of the idea of Apriori algorithm to determine whether to rescan the historical datasets by mining the added datasets. FUP algorithm considers the association rule maintenance problem of incremental datasets, but does not consider the existence of deletion of old things in the update of datasets in the actual environment, so the FUP_2 algorithm [5] is proposed to deal with the association rule maintenance problem of dynamic datasets in the case of deletion of old transactions.

Hong *et al.* [10] first proposed the concept of pre-large itemsets, which are filtered by two given thresholds, similar to the concept of buffers. When the number of new transactions is within the calculated threshold, only the support numbers of frequent and pre-large itemsets need to be updated, reducing the amount of computation, and when the cumulative number of new transactions exceeds the calculated maximum threshold, the original database is rescanned again. This algorithm can effectively handle the situation where there are fewer transactions in the historical data set and more transactions in the new data set, which is more suitable for handling real-life cases.

Saleti *et al.* [21] studied sequence pattern mining in incremental update datasets and proposed a parallel algorithm based on MapReduce architecture to achieve efficient sequence pattern mining in large-scale incremental datasets in response to the trend of big data. Wu*et al.* [25] applied the concept of pre-large to find high average-utility patterns in incremental update data sets, and proposed APHAUI algorithm, which achieves efficient querying of high average-utility patterns by setting two upper bounds of pub and lead-pub.

3 Query base preparation

In this section, the dominance relation, object score, and top-k dominance query for incomplete datasets are introduced. First, Table 1 gives a sample movie rating dataset consisting of five movies by four users, and the rating of movie m_1 is shown by the tuple (3,4,-,2), where the rating value of the third dimension is missing.

Table 1. Recommendation System Dataset Example.

ID	Movie Name	Audience ratings			
		a_1	a_2	a_3	a_4
m_1	The Lion King (1994)	3	4	-	2
m_2	Forrest Gump (1994)	4	-	-	3
m_3	The Blind Side (2009)	-	-	2	-
m_4	The Martian (2015)	-	4	3	2
m_5	Zootopia (2016)	3	-	2	3

Definition 1. ***A dominance relation between incomplete objects***

Objects p and q in the incomplete dataset S, comparable dimension IDs are recorded in the set C_{pq} and C_{pq} != $\emptyset$. p dominates q (record as $p \prec q$) if it satisfies $\forall\ d_i \in C_{pq}$, $p[d_i] \leq q[d_i]$ and $\exists\ d_j \in C_{pq}$, $p[d_j] < q[d_j]$.

Definition 2. ***Object score***

For an object p in the incomplete dataset S, the number of objects in the dataset S that p dominates is recorded as the score of p, denoted as Score(p), according to the dominance relationship in Definition 1.

Definition 3. ***Top-k domination query***

Calculate the score of each object in the dataset S and return the top-k objects with the highest scores, denoted as S_k.

4 Algorithm description

This section presents the incremental update TKD query algorithm for the incomplete dataset proposed in the paper. The definition of object dominance relationship shows that when two objects are analyzed for dominance relationship, the objects are compared in terms of values by dimension. So the algorithm first processes the storage format of the added data and stores the multidimensional incomplete data in HBase, a distributed file storage database, by dimension to facilitate subsequent comparisons. After the preprocessing of the storage format for the new data set, the dimension values saved in the historical data set in

Table 2. Recommendation System New Data Example.

ID	Movie Name	Audience ratings			
		a_1	a_2	a_3	a_4
m_6	La La Land (2016)	4	4	2	3
m_7	Uncle Drew (2018)	4	3	-	2

HBase are updated. For example, when the historical dataset shown in Table 1 welcomes the addition of new data, as shown in Table 2, the dataset in Table 2 is first preprocessed and then the information saved in HBase is updated, and the data in HBase after the update is shown in Table 3 (the missing data are represented by 0).

Table 3. HBase Storage Data Example.

Row Key	Column Family:Qualifier	Column Value
d_1	info:1	3, 4, 0, 0, 3 + 4, 4
d_2	info:2	4, 0, 0, 4, 0 + 4, 3
d_3	info:3	0, 0, 2, 3, 2 + 2, 0
d_4	info:4	2, 3, 0, 2, 3 + 3, 2

After updating the information in HBase, the next step is to use the MapReduce architecture to calculate the object domination number, which is divided into two parts, namely, the update of the domination score of historical objects and the calculation of the domination score of new objects, which can be subdivided into the domination relationship of historical objects to new objects, the domination relationship of new objects to historical objects, and the domination relationship between new objects. The three parts are shown in Fig. 1. Compared with the traditional algorithm for static data sets, the incremental update algorithm updates the dominance scores of historical objects using the query results of historical data sets, which avoids the repeated calculation of dominance relationships among historical objects and improves the query performance.

The next step is to introduce the incremental update algorithm in detail by example. Map stage reads the column with RowKey 1 in HBase, and analyzes the dominance of the first dimension by this column, firstly, the evaluation value of the historical object is compared with the evaluation value of the new object in turn, and the bit vectors $[Q_1]$ and $[T_1]$ are obtained. When the evaluation value of the new object is missing or not better than the evaluation value of the historical object, it is recorded as 1, otherwise it is recorded as 0, and the comparison result is recorded by the bit vector $[Q_1]$; When the evaluation value of the new object is missing or equal to the evaluation value of the historical

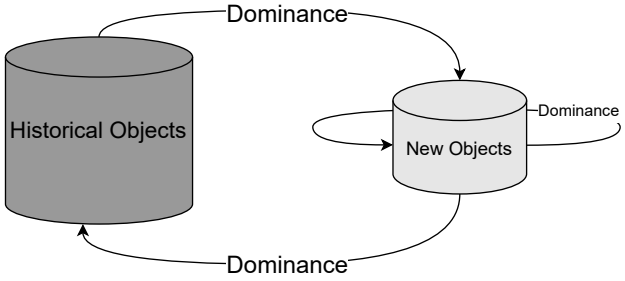

Fig. 1. Incremental update TKD query algorithm calculation logic for incomplete datasets.

object, it is recorded as 1, otherwise it is recorded as 0, and it is expressed by the bit vector $[T_1]$. Then, the evaluation value of the new object is compared with all the evaluation values in the dimension in turn, and the bit vectors $[Q_1]$ and $[T_1]$ of the new object are calculated by the above-mentioned comparison method to obtain the dominance of the new object over all the objects in the dimension. Next, the remaining dimensions are compared in turn. Reduce stage takes the intersection of $[Q_i]$ and $[T_i]$ of each object to obtain the bit vectors $[Q]$ and $[T]$ of the objects, and then obtains the set Q and T of the objects. From the above analysis, it is known that the set Q of object p contains objects that are dominated by p, objects that cannot be compared with p, and objects that have no dominance relationship with p, while the set T of object p contains objects that cannot be compared with p and objects that do not have a dominant relationship with p. Therefore, we can obtain $Score(p) = \mid Q$ - $T \mid$. For example, the bit vector $[Q_1] = 11$, $[Q_2] = 10$, $[Q_4] = 11$, $[T_1] = 00$, $[T_2] = 10$, and $[T_4] = 01$ for object M_1, and the bit vector $[Q] = 10$, $[T] = 00$, set $Q = [M_6]$, set $T = \emptyset$, so the dominance of M_1 over the new object is M_1 dominates M_6, and finally the incremental update algorithm obtains $Score(M_1) = Score_{History} + Score_{New} = 3$.

5 Experiment and Analysis

Through the introduction of the algorithm in Section 4, it can be analyzed that compared with the traditional algorithm based on static data, this algorithm reduces a lot of calculation process and improves the query efficiency. In this section, experiments are conducted in a Hadoop cluster built by five PCs and one switch to compare the query performance of the algorithm. The operating system of the PC is Ubuntu 18.10 and the Hadoop Framework version is 2.8.5. The experimental results are shown in fig 2 below. Through the experimental results, it can be found that the query performance of the algorithm proposed in this paper is significantly better than the traditional algorithm on the TKD query of dynamic incremental incomplete data sets.

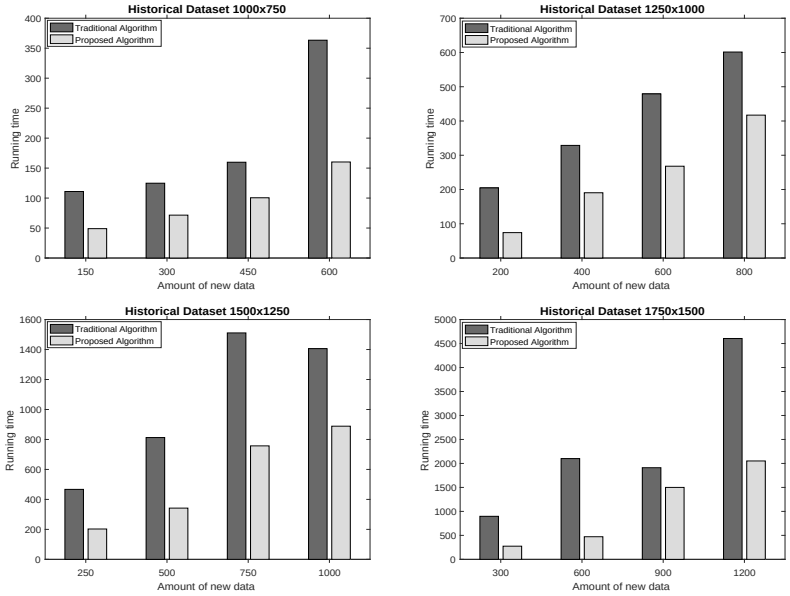

Fig. 2. Runtime under Different Database Sizes.

6 Conclusion

In this paper, we propose an incremental update algorithm based on incomplete datasets, which makes full use of the query results of previous historical datasets when calculating the dominance relationships between historical objects and new objects, avoiding the repeated analysis of dominance relationships between historical objects and reducing a large number of complex calculations. The algorithm is implemented by MapReduce, which decomposes complex analysis and computation tasks into multiple subtasks distributed to different nodes in the cluster for parallel execution, and demonstrates through experimental results that the algorithm achieves efficient query processing of dynamically updated incomplete datasets.

In addition, in the actual production activities, the operations on the database are not only the addition of data, but also the deletion and modification operations are very common and important. So, we will explore these research directions more deeply next.

References

1. Amagata, D., Sasaki, Y., Hara, T., and Nishio, S.: Efficient processing of top-k dominating queries in distributed environments. *World Wide Web*, 19(4):545–577, 2016.

2. Antova, L., Koch, C., and Olteanu, D.: From complete to incomplete information and back. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 713–724, 2007.
3. Canahuate, G., Gibas, M., and Ferhatosmanoglu, H.:Indexing incomplete databases. In *International Conference on Extending Database Technology*, pages 884–901. Springer, 2006.
4. Cheung, D.W., Han, D.-W., Ng, V.T., and Wong, C.Y.: Maintenance of discovered association rules in large databases: An incremental updating technique. In *Proceedings of the twelfth international conference on data engineering*, pages 106–114. IEEE, 1996.
5. Cheung, D.W., Lee, S.D., and Kao, B.: A general incremental technique for maintaining discovered association rules. In *Database Systems For Advanced Applications' 97*, pages 185–194. World Scientific, 1997.
6. Ezatpoor, P., Zhan, J., Wu, J.M.-T., and Chiu, C.: Finding top-k dominance on incomplete big data using mapreduceframework. *IEEE Access*, 6:7872–7887, 2018.
7. Gao, Y., Miao, X., Cui, H., Chen, G., and Li, Q.: Processing k-skyband, constrained skyline, and group-by skyline queries on incomplete data. *Expert Systems with Applications*, 41(10):4959–4974, 2014.
8. Green, T.J., Tannen, V.: Models for incomplete and probabilistic information. In *International Conference on Extending Database Technology*, pages 278–296. Springer, 2006.
9. Haghani, P., Michel, S., and Aberer, K.: Evaluating top-k queries over incomplete data streams. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 877–886, 2009.
10. Hong, T.-P., Wang, C.-Y., and Tao, Y.-H.: A new incremental data mining algorithm using pre-large itemsets. *Intelligent Data Analysis*, 5(2):111–129, 2001.
11. Imieliński, T., Jr, W.L.: Incomplete information in relational databases.In *Readings in Artificial Intelligence and Databases*, pages 342–360. Elsevier, 1989.
12. Khalefa, M.E., Mokbel, M.F., and Levandoski, J.J.: Skyline query processing for incomplete data. In *2008 IEEE 24th International Conference on Data Engineering*, pages 556–565. IEEE, 2008.
13. Lee, C.-H., Lin, C.-R., and Chen, M.-S.: Sliding-window filtering: an efficient algorithm for incremental mining. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 263–270, 2001.
14. Lin, M.-Y., Lee, S.-Y.: Incremental update on sequential patterns in large databases. In *Proceedings Tenth IEEE International Conference on Tools with Artificial Intelligence (Cat. No. 98CH36294)*, pages 24–31. IEEE, 1998.
15. Lofi, C., Maarry, K.E., and Balke, W.T.: Skyline queries in crowd-enabled databases. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 465–476, 2013.
16. Miao, X., Gao, Y., Zheng, B., Chen, G., and Cui, H.: Top-k dominating queries on incomplete data. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):252–266, 2015.
17. Ooi, B.C., Goh, C.H., and Tan, K.L.: Fast high-dimensional data search in incomplete databases. In *VLDB*, pages 357–367, 1998.
18. Papadias, D., Tao, Y., Fu, G., and Seeger, B.: Progressive skyline computation in database systems. *ACM Transactions on Database Systems (TODS)*, 30(1):41–82,2005.
19. Parthasarathy, S., Zaki, M.J., Ogihara, M., and Dwarkadas, S.: Incremental and interactive sequence mining. In *Proceedings of the eighth international conference on Information and knowledge management*, pages 251–258, 1999.

20. Pudi, V., Haritsa, J.R.: Quantifying the utility of the past in mining large databases. *Information Systems*, 25(5):323–343, 2000.
21. Saleti, S., Subramanyam, R.: A mapreduce solution for incremental mining of sequential patterns from big data. *Expert Systems With Applications*, 133:109–125, 2019.
22. Soliman, M.A., Ilyas, I.F., and Ben-David, S.: Supporting ranking queries on uncertain and incomplete data. *The VLDB Journal*, 19(4):477–501, 2010.
23. Tiakas, E., Papadopoulos, A.N., and Manolopoulos, Y., Progressive processing of subspace dominating queries. *The VLDB Journal*, 20(6):921–948, 2011.
24. Wang, K.: Discovering patterns from large and dynamic sequential data. *Journal of Intelligent Information Systems*, 9(1):33–56, 1997.
25. Wu, J.M.-T., Teng, Q., Lin, J.C.-W., and Cheng, C.-F.: Incrementally updating the discovered high average-utility patterns with the pre-large concept. *IEEE Access*, 8:66788–66798, 2020.
26. Wu, J.M.-T., Wei, M., Wu, M.-E., and Tayeb, S.: Top-k dominating queries on incomplete large dataset. *The Journal of Supercomputing*, pages 1–22, 2021.
27. Yiu, M.L., Mamoulis, N.: Efficient processing of top-k dominating queries on multi-dimensional data. In *VLDB*, volume 7, pages 483–494, 2007.