International Journal of Scientific Engineering and Applied Science (IJSEAS) - Volume-1, Issue-6, September 2015
ISSN: 2395-3470
www.ijseas.com

EFFECTIVENESS OF SKYLINE COMPUTATION OVER SPATIAL DATA

T. GANESAN
PG Scholar, Department of CS,
Vels University
Pallavaram, Chennai – 600 117

S.PERUMAL
Professor/Head, Department of CS,
Vels University,
Pallavaram, Chennai – 600 117

ABSTRACT:

Skyline query processing has been investigated extensively in recent years, mostly for only one query reference point. An example of a single-source skyline query is to find hotels which are cheap and close to the beach (an absolute query), or close to a user-given location (a relatively query). A multi-source skyline query considers several query points at the same time In this paper, we consider the problem of efficient multi-source skyline query processing in road networks. It is not only the first effort to consider multi-source skyline query in road networks but also the first effort to process the relative skyline queries where the network distance between two locations needs to be computed on-the-fly. The main contributions of this paper are listed as follows: This paper presents a novel skyline algorithm SSPL on spatial data, which can utilize some small reconstructed data-structures to reduce I/O cost significantly.. The behavior analysis of SSPL is proposed to determine scan depth of the sorted positional index lists.

Keywords: skyline, positional index

1. INTRODUCTION

Skyline queries can be found in a wide spectrum of optimization applications. Without loss of generality, let us consider optimization as minimization here. Given a set of multidimensional points D, a *skyline query* finds the skyline points from D, such that every point on the skyline is not 'dominated' by any other point in D (i.e., if a point p is on the skyline, there exists no data point p′ in D such that p′ is pair-wise smaller than p for the values in all dimensions). The skyline queries can be either absolute or relative, where 'absolute' means that minimization is based on the static attribute values of data points in D, while 'relative' means that the difference between a data point in D and a user-given query point needs to be computed for minimization. Relative skyline query is also known as dynamic skyline query An important category of skyline query applications is related to Geographical Information Systems, to support decision making in the areas such as intelligent transport systems, urban planning, environmental applications, location based services, mobile workforce management, and military and utility deployment. Depending on different applications, the distance between two points a and b can be approximated using their Euclidean distance, or computed using the spatial network distance or the surface distance if the distance between a and b implies that an object needs to physically move from a to b along a spatial network (e.g., roads or water systems) or on/near

a terrain surface. The case of using the Euclidean distance is called constraint-free as the distance between any two points can be calculated by only using the coordinates of the two points. The case where an environment dataset (e.g., a road network) needs to be used for distance calculation is called constraint-based. These two types of queries use quite different query processing strategies. For a constraint-free query the key to efficient query processing is to reduce the number of data points to be accessed (i.e., to minimize the portion of D accessed when answering a query). For a constraint-based query, however, one additional and often more critical goal is to minimize the portion of the environment data to be accessed and the cost of network distance calculation. This is because the environment data is typically much larger and much more complex than D and network distance calculation is typically done by using a costly shortest path algorithm. While skyline query processing in a constraint-free space has been well studied, to the best of our knowledge, this paper is the first effort on relative skyline query processing in a constrained spaceSkyline is an important operation in many applications to return a set of interesting points from a potentially huge data space. A subset of attributes is designated as skyline criteria, on which the dominance relationship between tuples is defined. Given two tuples p and q in a table, p dominates q if, among skyline criteria, p is not larger than q in all attributes and strictly smaller than q in at least one attribute. Skyline finds all tuples that are not dominated by any other tuples. Recently, skyline has attracted extensive attention and many algorithms are proposed. A set of skyline algorithms, such as Bitmap, NN, BBS, SUBSKY, and ZBtree, utilize indexes to reduce the explored data space and return skyline results. However, because of the prohibitive pre-computation cost and space overhead to cover the attributes involved in skyline on big data, index-based algorithms have serious limitations and the used indexes can only be built on a small and selective set of attribute combinations.

The experimental results show that SSPL has significant advantage over the existing skyline algorithms.

2. PROBLEM STATEMENT

Given a table TðA1; A2; . . .; AMÞ, 8t 2 T, let us denote by t½j_the jth attribute Aj of t. Without loss of generality, let a subset of attributes AS skyline ¼ fA1;A2; . . .;Amg be skyline criteria, and the dominance relationship between tuples is defined on AS skyline. 8t1; t2 2 T, t1 dominates t2 (denoted byt1 _ t2), if 8jð1 _ j _ mÞ, t1½j_ _ t2½j_, and 9jð1 _ j _ mÞ,t1½j_ < t2½j_, i.e., t1 _ t2 , ðTm j¼1ðt1½j_ _ t2½j_ÞÞ \ ðSmj¼1ðt1½j_ < t2½j_ÞÞ. For clarity, we assume that min condition only is used for skyline computation. However, the algorithm here can be extended to process any combination of conditions (min or max).Skyline query. Given a table T, skyline query returns a subset SKY ðTÞ of T, in which 8t1 2 SKY ðTÞ, 6 9t2 2 T, t2 _ t1.Given tuple number n in table T and size m of skyline criteria, the expected number s of skyline results under component independence is s ¼ Hm_1; n, here Hm;n is the mth order harmonic of n. For any n > 0,H0;n ¼ 1. For anym > 0, Hm;0 ¼ 0. For any n > 0 and m > 0, Hm;n is inductively defined as: Hm;n ¼ Pn i¼1Hm_1;I i , Hm;n can be approximated as:

According to the computation formula of Hm;n, it is found that the number of skyline results does not change significantly as the tuple number increases, while it is very sensitive to

the size of skyline criteria. For example, given ¼ 3, when n increases from 105 to 109, s changes from 66 to 214. Given n ¼ 109, when m increases from 2 to 5, changes from 20 to 7,684. Although the absolute number of skyline results is large, its proportion among all tuples is rather small. For example, given m=5 and n=109.

3. RELATED WORK

3.1 Index-Based Algorithms

Index-based skyline algorithms utilize the reconstructed data-structures to avoid scanning the entire data set. To make use of bitmap to compute skyline of actable TðA1; A2; . . .;AdÞ. Given a tuple x ¼ ðx1; x2; . . . ; xdÞ 2T, x is encoded as a b-bit bit-vector, b ¼ Pd i¼1 ki (ki is the cardinality of Ai). We assume that xi is the ðjiÞth smallest value in Ai, the ki-bit bit-vector representing xi is set as follows: bit 1 to bit ji _ 1 are set to 0, bit ji to bit ki are set to 1. The encoded table is stored as bit-transposed files let BSij represent the bit file corresponding to the jth bit in the ith attribute Ai. It is given that a tuple x ¼ ðx1; x2; . . . ; xdÞ 2T and xi is the ðjiÞth smallest value in Ai. Let A ¼BS1j1&BS2j2& . . . & BSdjd where & represents the bitwise and operation. And let B ¼ BS1ðj1_1ÞjBS2ðj2_1Þj . . . jBSdðjd_1Þ where j represents the bitwise or operation. If there is more than a single one-bit in C ¼ A&B, x is not a skyline tuple. Otherwise, x is a skyline tuple. To propose NN algorithm to process skyline query. NN utilizes the existing methods for nearest neighbour search to split data space recursively. By a reconstructed R-tree, NN first finds the nearest neighbour to the beginning of the axes. Certainly, the nearest neighbour is a skyline tuple. Next, the data space is partitioned by then earnest neighbour to several subspaces. The subspaces that are not dominated by the nearest neighbour are inserted into a to-do list. While the to-do list is not empty, NN removes one of the subspaces to perform the same process recursively. During the space partitioning, overlapping of the subspaces will incur duplicates, NN exploits the methods: Laisser-faire, Propagate, Merge and Fine-grained Partitioning, to eliminate duplicates. To develop a progressive algorithm BBS based on nearest neighbour search. BBS applies branch-and bound strategy. It begins with root node of R-tree, and inserts all its child-nodes in a min-heap according to their distances to the beginning of the axes. Next, the node with the minimum distance is selected to expand the heap, i.e., remove the node and insert all its child nodes. Each node needs to be checked for dominance twice: before it is inserted into the heap and before it is expanded. If the node is dominated by the current skyline results, it is discarded. If the node selected to expand is data point, it is a part of skyline results. BBS only performs a single access to the nodes of R-tree that may contain skyline results and does not retrieve duplicates.

To propose a technique SUBSKY to perform skyline in subspaces. SUBSKY converts each d-dimensional point p to a 1D value fðpÞ by a specified distance function: fðpÞ ¼ maxd i¼1ð1 _ p½i_ÞÞ (The coordinates of p are within[0, 1]). The points are accessed in descending order of their fðpÞ values by a B-tree. Meanwhile, SUBSKY maintains the current skyline set Ssky and the largest mind i¼1ð1 _ p½i_ÞÞ value U of the points in Ssky. SUBSKY terminates when U is larger than the fðpÞ value of the next point retrieved by B-tree. Besides, SUBSKY is

extended to process clustered data by multiple anchors to achieve a better performance. Lee et al. [24] develop a suite of skyline algorithms which scale well to dimensionality and cardinality. Z-curve is adopted here to map data points in a multidimensional space to a 1D space, and each point is represented by-address. A new index ZBtree is proposed to organize data points in accordance with monotonic Z-address. ZBtree divides Z-order curve into disjoint segments with each of which representing a region. ZSearch is presented to process skyline efficiently based on ZBtree. It traverses the ZBtree to visit the regions and potential skyline points in a depth-first order. And a stack is used to keep the unexplored paths. At each round, the region of a node popped from the stack is examined against the current skyline results. If the region is not dominated, the node is further explored. The data points of the leaf node which is not dominated are compared with the current skyline results. ZSearch terminates when the stack is empty. To sum up, because of the prohibitive pre computation cost and space overhead, index-based algorithms have serious limitations. It is much expensive for bitmap algorithm to perform preconstruction and computation of the skyline results. The bit-vector length of each encoded tuple in bitmap algorithm equals the sum of cardinalities of all attributes. If some attributes have high cardinalities, the space overhead for storage is large. Besides, for checking whether each tuple in table is a part of skyline, bitmap algorithm has to retrieve the corresponding bit-transposed files involving all tuples. For tree-based algorithms, although the size of skyline criteria is typically small, the combination of the attributes over which the queries are posed can be quite large. Given a table with M attributes and skyline criteria involves not more than m attributes, tree-based algorithms have to build Pmi¼2ðMi Þ indexes to cover the attributes used in queries. That is, the number of indexes has an exponential relationship with the number of attributes. Therefore, due to prohibitive computation cost and space overhead, index-based algorithms are not suitable for processing skyline on big data.

3.2 GENERIC ALGORITHM

Generic skyline algorithms do not require pre-processing steps or data-structures; they are performed on the table directly. The computation of skyline results is also called maximal vector problem. Kung et all. propose a basic divide-and conquer algorithm DD&C to process maximal vector problem, together with its theoretical analysis. DD&C splits the input into two partitions P1 and P2 by the median along certain attribute dp. 8t1 2 P1; 8t2 2 P2, we have t1:dp _ t2:dp.DD&C is recursively executed on P1 and P2 along the remaining attributes to get maximal vector results S1 andS2, respectively. Then DD&C merges S1 and S2 to return the results by discarding the dominated vectors in S2. Sheng and Tao propose a novel divide-and-conquer external skyline algorithm with a lower I/O complexity. The algorithm is on the basis of the processing on 3-d space as below. The algorithm assumes that the input is arranged in ascending order of x-coordinate. At first, the algorithm splits the input by the disjoint intervals in x-coordinate into k ¼ _ðM=BÞ partitions P1; P2; . . . ; Pk with roughly the same size such that each point in PðiÞ has a smaller x-coordinate than all points in PðjÞ for any 1 _ i < j _ k (M is the capacity of main memory and B is the capacity of block). The memory-resident skyline algorithms are invoked to obtain skyline results _ð1Þ;_ð2Þ; . . . ;_ðkÞ of

P1; P2; . . . ; Pk, respectively, if they fit into memory. Otherwise, if Pið1 _ i _ kÞ does not fit in memory, it is further divided into smaller partitions, each of which is processed recursively. Before _ðiÞð1 _ i _ kÞ is outputted to disk, the algorithm guarantees that _ðiÞð1 _ i _ kÞ is arranged in ascending order of y-coordinate. Then, according to y-coordinate, the algorithm performs a k-way merging on _ðiÞð1 _ i _ kÞ. During the k-way merging, the algorithm maintains by _ðiÞ the minimum z-coordinate of all the tuples seen already from _ðiÞ.

3.3 THE SSPL ALGORITHM

This section first introduces the data-structures required by SSPL , then describes the overview of the SSPL algorithm, next shows how toper form pruning, followed that presents the implementation and analysis of SSPL, and finally introduces how to extend SSPL to cover other cases.

3.4 Sorted Positional Index List

Definition 4.1 (Positional Index). Given a table T, the Positional index ðPIÞ of t 2 T is i if t is the ith tuple in T.We denote by TðiÞ the tuple in T with its PI ¼ i, and by TðiÞ½j the jth attribute of TðiÞ. The execution of SSPL requires sorted positional index lists. Given a table TðA1;A2; . . .;AMÞ, we maintain a sorted positional index list Lj for each attribute Aj ð1 j MÞ. Lj keeps the positional index information in T and is arranged in ascending order of Aj, that is 8i1; i2ð1 i1 < i2 nÞ, TðLjði1ÞÞ½jTðLjði2ÞÞ½j.

3.5 SSPL consists of two phases:

- **Phase 1:** SSPL retrieves the sorted positional index lists corresponding to AS skyline to obtain candidate positional index set SETcand.
- **Phase 2:** By SETcand, SSPL performs a selective and sequential scan on the table to compute skyline results.

Phase 1: Candidate positional index set

Initially SSPL computes scan depth. The lists are retrieved in a round-robin fashion.Phase1 ends when there is a candidate positional index seen in all of the involved lists.

Phase 2: Retrieving the Skyline Results

SSPL exploits the obtained candidate positional indexes to get skyline results by,

- Selective
- Sequential scan

SSPL has a significant advantage over the existing skyline algorithms.

4. PROJECT DESCRIPTION

4.1 Probabilistic top-k queries

Same queries are said to be Top k queries Different queries are said to be continuous queries.

- Top k queries
- Data set
- skyline

4.2 Candidate position module

Candidate position is being found out by latitude and longitude position of the user.

- User

- Admin

4.3 System setup module

Network environment is used to create many nodes. For each node query is being generated by the intermediator.

- Receiver
- Query

4.4 Pruning

Although the ratio rc and of the candidate tuples to all tuple is small, the absolute number of the candidate tuples retrieved in phase 2 is still large.

- Early pruning (EP)
- Late pruning (LP)

EP performs pruning on each candidate positional index seen in phase 1.SSPL splits coordinate space into 2m regions by the computed scan depth.

LP part introduces late pruning which is executed on the candidate positional indexes at the end of phase1.

CONCLUSION

In this paper, we consider the problem of processing skyline query on big data. It is analyzed that the current skyline algorithms cannot perform skyline on big data efficiently. This paper proposes a novel skyline algorithm SSPL, which utilizes sorted positional index lists of low space overhead, to reduce the I/O cost significantly. SSPL consists of two phases. In phase 1, it retrieves the sorted positional index lists specified by skyline criteria in a round-robin fashion until there is a candidate positional index seen in all of the involved lists. In phase 2, SSPL performs a sequential and selective scan on the table by the candidate positional indexes obtained in phase 1 to compute skyline results. Early pruning and late pruning are presented in the paper to discard the unsatisfied candidate positional indexes in phase 1. The experimental results on synthetic and real data sets show that SSPL has a significant advantage over the existing skyline algorithms.

REFERENCES:

1. I. Bartolini, P. Ciaccia, and M. Patella, “Efficient Sort-Based Skyline Evaluation,” ACM Trans. Database Systems, vol. 33, no. 4, pp. 31:1-31:49, 2008.

2. I. Bartolini, Z. Zhang, and D. Papadias, “Collaborative Filtering with Personalized Skylines,” IEEE Trans. Knowledge Data Eng., vol. 23, no. 2, pp. 190-203, Feb. 2011.

3. J.L. Bentley, H.T. Kung, M. Schkolnick, and C.D. Thompson, “On the Average Number of Maxima in a Set of Vectors and Applications,” J. ACM, vol. 25, no. 4, pp. 536-543, 1978.

[4] J.L. Bentley, K.L. Clarkson, and D.B. Levine, “Fast Linear Expected-Time Algorithms for Computing Maxima and Convex Hulls,” Proc. First Ann. ACM-SIAM Symp. Discrete Algorithms (SODA ’90), pp. 179-187, 1990.

[5] M. Berger, Geometry I, vol. 1, Springer, 1987.

[6] B.H. Bloom, “Space/Time Trade-Offs in Hash Coding with Allowable Errors,” Comm. ACM, vol. 13, no. 7, pp. 422-426, 1970.

[7] S. Bo¨ rzso¨ nyi, D. Kossmann, and K. Stocker, “The Skyline Operator,” Proc. 17th Int’l Conf. Data Eng. (ICDE ’01), pp. 421- 430, 2001.

[8]R.E. Bryan, “Data-Intensive Supercomputing: The Case for Disc,” Technical Report CMU-CS-07-128, School of Computer Science, Carnegie Mellon Univ., 2007.

[9] C.-Y. Chan, H.V. Jagadish, K.-L. Tan, .K.H. Tung, and Z. Zhang, “Finding K-Dominant Skylines in High Dimensional Space,” Proc. ACM SIGMOD Int’l Conf. Management of Data (SIGMOD ’06), pp. 503-514, 2006.

[10] L. Chen and X. Lian, “Efficient Processing of Metric Skyline Queries,” IEEE Trans. Knowledge Data Eng., vol. 21, no. 3, pp. 351-365, Mar. 2009.

[11] L. Chen, B. Cui, and H. Lu, “Constrained Skyline Query Processing against Distributed Data Sites,” IEEE Trans. Knowledge Data Eng., vol. 23, no. 2, pp. 204-217, Feb.2011.

[12] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, “Skyline with Presorting,” Proc. 19th Int’l Conf. Data Eng. (ICDE ’03), pp. 717-719, 2003.