International Journal of Database Theory and Application
Vol.9, No.12 (2016), pp.9-22
http://dx.doi.org/10.14257/ijdta.2016.9.12.02

Applying Z-Curve Technique to Compute Skyline Set in Multi Criteria Decision Making System

T. Vijaya Saradhi[1], Kodukula Subrahmanyam[1], P. Venkateswara Rao[1] and Hye-jin Kim[2]

1Department of Computer Science and Engineering, K L University, Andhra Pradesh
{saradhi1440, smkodukula, pvrao}@kluniversity.in
2Sungshin Women's University, 2, Bomun-ro 34da-gil, Seongbuk-gu, Seoul, Korea
hyejinaa@daum.net
(Corresponding Author)

Abstract

The skyline queries are the best tools to be used in distributed multi criteria decision making of web based applications for user commendations. However, as the Data dimensions are increasing size of dominance set and skyline set is also increasing. Increasing dimensionality becomes the major problem with real word databases. In skyline computation major cost depends on finding dominance tests between high dimensional objects and the order in which they are accessing. Space filling Z-curve is the best suitable way to address the challenges in skyline computation. In this proposed work, we incorporated Z-curve with optimized skyline boundary detection algorithm to effective access and early pruning. In this paper efficient hybrid index structure was proposed which takes the advantage of sorting and partition approaches to improve the storage and search efficiency. Experimental results show that our propose approach is better than the previous static skyline computation techniques in terms of searching and finding skyline set.

***Keywords**: Skyline computation, Z-curve, probabilistic skyline computation, boundary detection algorithm*

1. Introduction

Decision making is the key aspect in everywhere. Now a day's Computational applications are highly distributed and generating huge high dimensional uncertain data. Users are interested in accuracy and posing queries on different dimensions. So there is need of efficient strategies. Skyline analysis is an approach which gained high significance attention as it is being used in the multiple criteria decision making applications. Our computation method should be designed in such a way that it can be extended from single dimensional to multiple dimensional subspaces. Given a set of T dimensional data set of real estate ventures containing ventures p1, p2, p3…..pn; we obtain a skyline set of T such a way that skyline venture objects are not overruled by any of the venture object in the given pool of objects. If we want to consider one venture point p1 dominates another venture point p2 if all dimensional values either equal or less than the corresponding dimensions of comparing object but at least one dimension value should be less than the corresponding dimension of other object.

Let us see Figure 1 which gives out skyline set for selecting best Real estate venture among other venture. Top venture is decided by the user based on two dimensions, price and distance to the beach. We can plot all venture objects in two dimensional space. X-dimension indicates distance from venture to beach, and y-axis represents venture price. Skyline retrieves venture objects with low Prices and short distances, *i.e.*, the dots p3, p2, p1, p5, p6 in Figure 1.

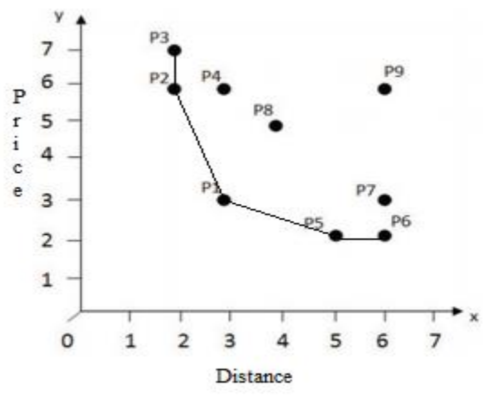


Figure 1. Skyline Example

In Real time applications user may interested in sub space skylines as well as full space. In our venture selection, different users may have different interests. So we need to compute the skylines in different subspaces such as (price; Space), (price; travel time to airport) and (price; travel time facilities). Normally, a skyline sets are changing with the change in the dimensions. To answer user queries efficiently, a skyline analysis system needs to maintain skyline of each and every subspace. If objects are having n dimensions then we need to find skyline sets in 2n-1 subspaces. The sky cube is a set of all skylines in all non-empty subspaces. Finding of sky cube will be a difficult task if number of dimensions is increasing. We observed that cost of skyline analysis impacted by 3 issues (1) Dominance Tests count, (2) Access Order of the objects and (3) Grouping and comparing techniques are important for the competence of dominance test.

2. Related Work

Database researchers are attracted by the applicability of skyline computation in various fields [4]. We have gone through the algorithm which works with different dimensions and with total or partial ordering domains [5]. SUBSKY, a dimensionality reduction algorithm it indexes the objects using B-Tree [5-6].The cost of the skyline computation on static data depends on dominance test. All these methods propose different heuristic ways to speed up the cost of dominance [9].

To efficiently answer the user queries sky cube is one solution. In this we need to find subspace skyline for every subspace. Second, it reduces the dominance tests by finding the skyline points by already derived skyline points [2, 10].Finding sky cube is very time consuming task when object dimensionality is high. In traditional pair wise dominance approach sky cube computation leads to dimensionality curse problem [7].

So many algorithms were proposed in database environment for static as well as probabilistic uncertain data [6] and data streams in centralized environment [9].These algorithms can be categorized based on two approaches, sorting and partitioning [14].

1. Sorting-based Algorithms: In this type of algorithms effective selection of cutoff points will be used for early pruning of non-skyline objects. So optimization of Pivot selection will be crucial in Sorting-based skyline algorithms. In these algorithms main focus is on optimizing of dominance. They didn’t consider incomparability and search space pruning [3-4].

2. Partition based algorithms: partitioning based algorithm divides the space into regions based on the commonality of data. So we can achieve high impact on cost of dominance by performing region level comparisons and incomparability of regions. Available partitioning based algorithms gets advantage by portioning but they dint considers the access order.

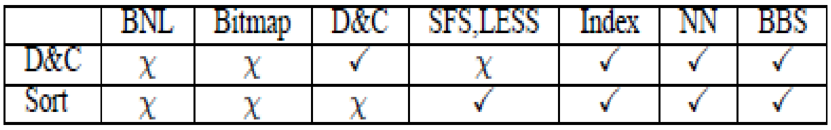
Table 1. Different Types of Techniques Used in Skyline Computation

Technique	Method
Block Nested Loop	Pair wise comparison
Divide and Conquer	Chop up the dataset into smaller enough ones that can fit into memory individually. Process each of them to get final results.
Index	Group tuples according to their minimum dimensions. Sort each group and process top tuples of all groups.
Nearest Neighbor	Use nearest neighbor search to find skyline points which further divide the space for recurring processes.
Streaming Skyline	Compute skyline points in a streaming database.

2.1. Skyline Query Processing

Different available Skyline computation algorithms, are Block-Nested Loop (BNL) [3], Divide-and-Conquer (D&C) [3], Bitmap [15], LESS [8], Index [15], Sort-Filter-Skyline (SFS) [7], Nearest Neighbor (NN) [10], and Branch and Bound Search (BBS) [13], all these can be categorized using the divide-and-conquer and/or sorting strategies. Below depicts the summary. BNL and Bitmap are not following any strategy so they are considered as brute force approach. Below is the categorization based on D&C and sorting techniques?

Table 2. Classification of Skyline Query Algorithms



	BNL	Bitmap	D&C	SFS,LESS	Index	NN	BBS
D&C	χ	χ	✓	χ	✓	✓	✓
Sort	χ	χ	χ	✓	✓	✓	✓

2.2. Divide and Conquer Algorithms

D&C partition a dataset into a few regions sufficiently little to be stacked into the memory for handling [3]. Every region finds its partial skyline set .By applying merge process on all the partial skylines it finds the final skyline set.

2.3. Sorting-based Algorithms

Sorting based algorithms needs presorted database. Data can be presorted based on any scoring function. If data is sorted then any object, with value less than the current object value, can't dominate. So we can easily find partial skyline. SFS and LESS are sorting based algorithms. Sorting based algorithms doesn't have search space pruning ability.

2.4. Hybrid Algorithms

NN and BBs are hybrid skyline algorithms. BBS is the available new efficient algorithm. This follows the traditional Iterative NN approach. Below Figure 2 demonstrates the BBS with 9 points in 2D space. In the whole space p1 is the nearest neighbor (nn) from origin. So definitely it will be the skyline. Point's falls under the rectangular dominance region spanned p1 and maximal corner will be definitely overruled by p1.Then repeatedly find next nn and prune any dominated points. In this next nn is p3.Next nn is p5.By finding p5 p7 will be eliminated. By finding p2 and p6 in similar way search will be terminated.

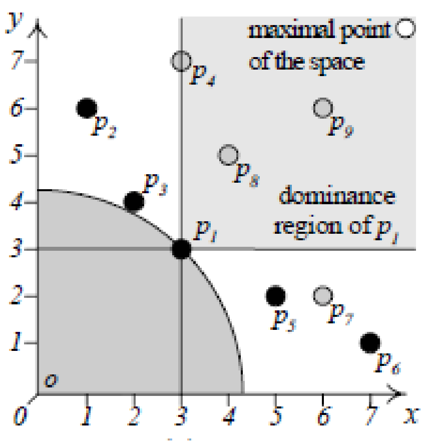


Figure 2. BBS Approach

3. Distributed Skyline Properties

Every efficient Distributed system has to satisfy three properties those are bandwidth minimization, progressiveness and effective pruning.

3.1. Minimization of Bandwidth Consumption

In distributed systems are constrained with high bandwidth consumption. Here bandwidth measured in terms of transmitted objects. In realistic scenario more bandwidth is required for message synchronization and for message routing information. But this will not considered when calculating bandwidth.

3.2. Progressiveness

Because of the complexity of skyline processing it may take more time than the range queries and top-k queries. The total query processing time may be increased drastically if we consider network delay in distributed skyline query processing. To overcome this it would be better if skyline algorithm follows progressiveness property. A progressive algorithm informs the resultant objects as soon as it founds. Later it produces left over results before query execution [10].This is the difficult objective to accomplish.

3.3. Effective Pruning

In distributed skyline computation dominance is costly effort. This can be eliminated by effective pruning. By effective pruning we can even eliminate distributed site, which can’t contribute to the final result, without participating into skyline process. Early pruning minimizes the computation cost drastically.

4. Architecture

Here, in Figure3 we are representing how the high dimensional data is handled. In this 4 stages are there. 1) Data access. 2) Dominance test and candidate confirmation. 3) Candidate Reexamination. 4) Update stage.

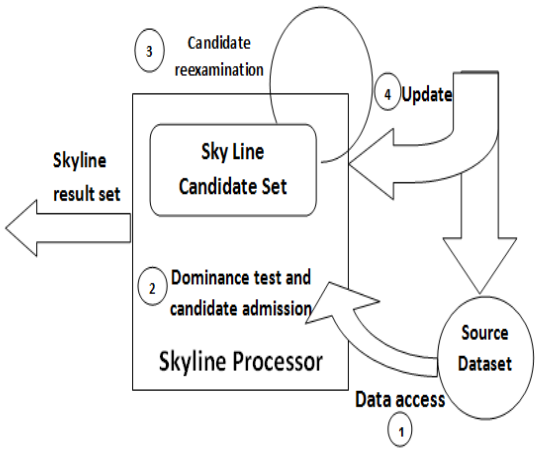


Figure 3. Distributed Site Skyline Architecture

In Distributed multi criteria decision making system query will be answered in all distributed sites. They will be arranged based on its local skyline probability and highest probability object will be sent to the central server. Thus server collects all local sites high probable objects (D_0) and again finds the skyline set of $D_{0.}$Highest probable skyline object will be sent to the every local site to prune their local skyline objects, which may not contribute to the final solution. It find the global skyline probability of the object to determine its eligibility to become member of final skyline object. To finds the local skyline point's skyline processor uses Z-order curve properties [19] and VS-Tree index structure to convert and organize skyline points. Below Figure4 shows the details of distributed skyline Framework.

5. Z-Order Curve

The ability of skyline preparation relies on conducting dominance test and the order of processing data points. Block based dominance instead of pair wise dominance reduces the cost. Order of accessing is essential because early identification of skyline object will be helpful to avoid many candidate examinations and reconsideration. The properties of Z-curve have well matched with the skyline processing strategies. Figure6 illustrate with 9 2D points example. Entire space is divided as 4 equivalent regions namely R1, R2, R3, and R4. Information points in RI are not overwhelmed by information focuses in the other three regions. Unexpectedly, all information focuses in R4 are ruled out by any point in R1. That is until R1 is nonempty every point located inside R4 can be disposed from examination. R2 and R3 are opposite to one another, and their data points don't dominate one another.

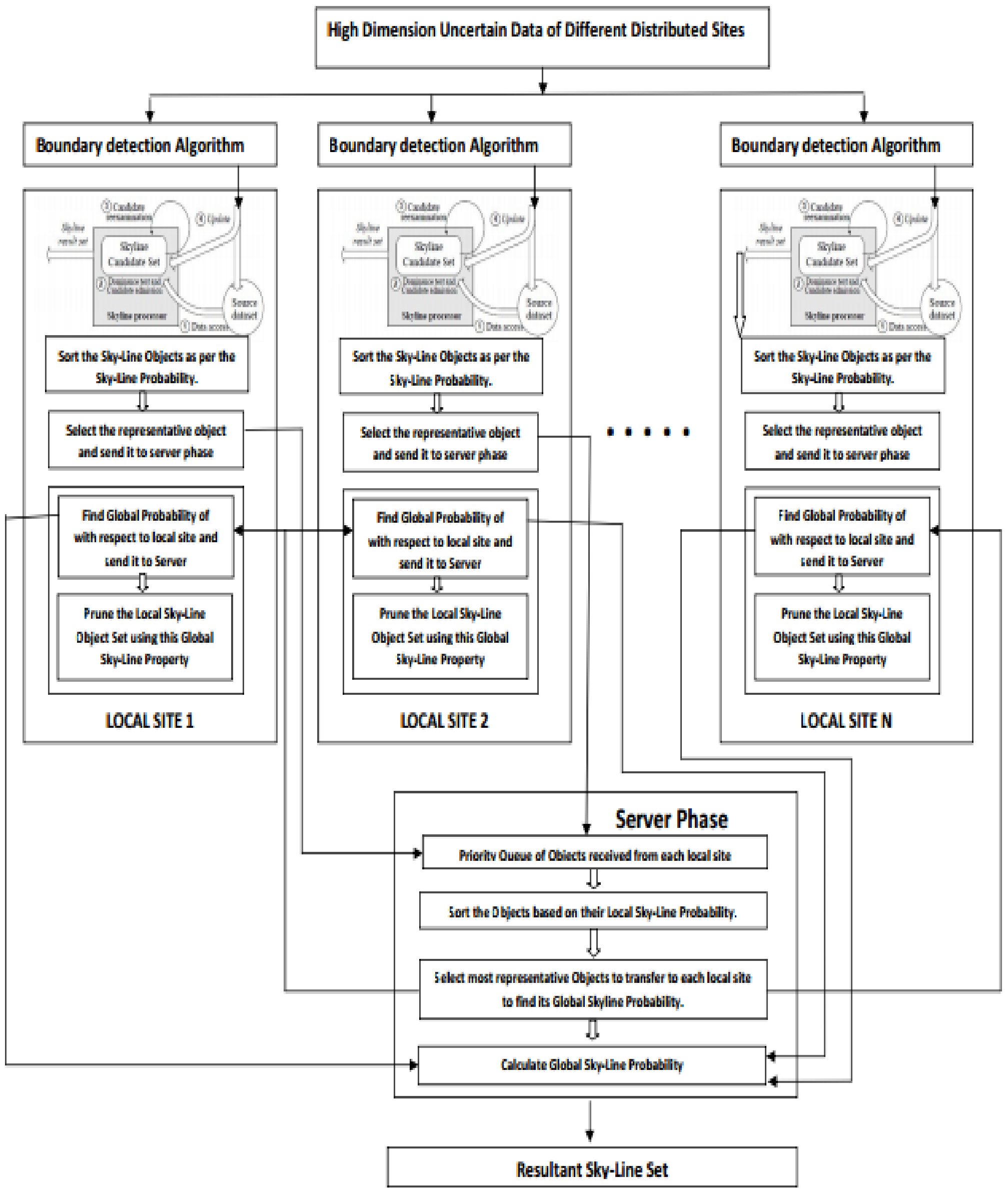

High Dimension Uncertain Data of Different Distributed Sites
Boundary detection Algorithm
Boundary detection Algorithm
Boundary detection Algorithm
Skyline Candidate Set
Skyline processor
Source dataset
Sort the Sky-Line Objects as per the Sky-Line Probability.
Select the representative object and send it to server phase
Find Global Probability of with respect to local site and send it to Server
Prune the Local Sky-Line Object Set using this Global Sky-Line Property
LOCAL SITE 1
LOCAL SITE 2
LOCAL SITE N
Server Phase
Priority Queue of Objects received from each local site
Sort the Objects based on their Local Sky-Line Probability.
Select most representative Objects to transfer to each local site to find its Global Skyline Probability.
Calculate Global Sky-Line Probability
Resultant Sky-Line Set

Figure 4. Distributed Skyline Framework Architecture

Dominance tests between them can be kept away from. But some of the data points of R2 and R3 may be dominated by R1 .These perceptions will be used to speed up the dominance tests in region level

These facts help to follow the access order which will be exactly fit for skyline process. According to this we access Region I Region II Region III and finally Region IV. Same logic will be applicable to sub regions also. This access order exactly looks like rotated Z. This Z-order space filling curve can start access from origin.

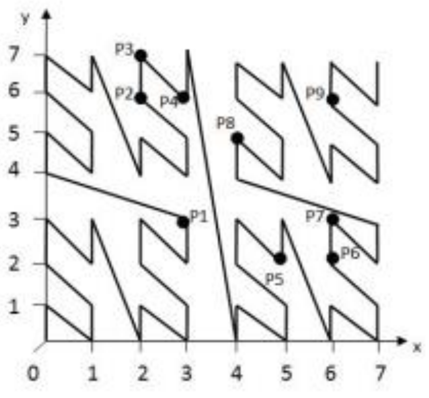


Figure 5. Example Z-order Curve

Z-curve is proposed to map the data from high dimensional to single dimensional. Each point will be represented by one number. That number is called Z-number. It is a sequence of bits. This bit sequence is formed by picking the bits in all dimensions alternatively. Take a k-dimensional object space with coordinate domain range ([0,2^n - 1]).We can covert that point as single dimensional "kn" bits. This can be viewed as n k bit groups. The jth bit of any Z-number is formed by the (j/k)th bit of the (j%k)th dimension. To compute the Z-number first convert the coordinate values into binary format. Then by interleaving the bits of all dimensions we can form the Z-number. In above example P7 Z-number is 101101 and p4 Z-number is 011111. In our example first k bits partitions the space into 4 subspaces. 2^{nd} k-bit group divides the subspace into 4 subspaces. Points with similar bits will share the same subspace. Example p2 and p4 share same subspace because they have same first bit group (*i.e.*, 01).

5.1. Monotonic Ordering

Non decreasing arrangement of Z-number follows the property that "dominating point placed before dominated points". So that we can reduce computation cost.In Figure 5, according to Z-order curve, p1 is accessed first before p8 and p9. Before p4, point's p2, p3 will be accessed and p5 is accessed before p7. Reexamination will be avoided by this access order.

5.2. Clustering

Non decreasing Z-numbers follow clustering property. Because of this, data points of the same region will have similar address. For example, p2, p3 and p4 are available in the same region and they share common prefix 01. Similarly, p5, p6, and p7 are sharing same subspace because of similar bit group. This kind of making Grouping data can give chance to block-based dominance tests. We can eliminate dominance tests when two groups are incomparable. In addition to this as soon as finding data point in dominating region we can avoid the dominance test between points in the dominating region and dominated region. For example, p8 and p9 are in region IV. So can be safely discarding them once if p1 is identified in dominating region.

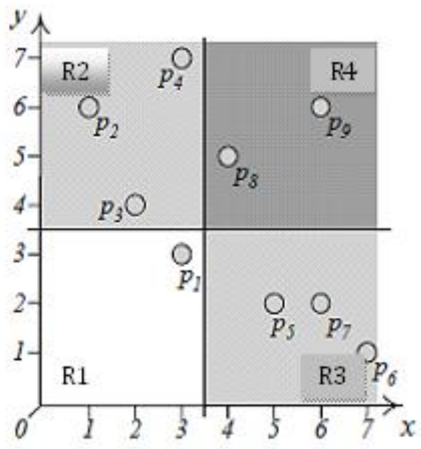


Figure 6. 2D Data Space

6. VS Tree Index Structure

Z-order curve supports two key properties in skyline set calculation *i.e*, monotonic ordering and clustering. To make our process efficient we need to design one efficient index structure which incorporates these key properties. We need to convert high dimensional data points onto one dimensional address. For all these we will create a new index approach with Z-order curve and B+ tree. Existing Index structures like R-Tree, UB-tree are having their own limitation so our goals is to i) processing data in Z-order fashion and ii) Maintain data points in blocks to support efficient pruning. Thus, this work proposes VS Tree, a new variant of B+-tree. VS tree divide a Z-curve into disjoint segments. Each segment is a region. So that clustering property is attained. In VS tree, data will be maintained in leaf nodes and non-leaf nodes represents objects range in the form of Intervals [a, b].The space segment enclosed by a Z-order curve segment is called Z-region. For example, the curve starting at point p8 and ending at point 9 is Z-region..There is no restriction on Z-region size and form as shown in fig-4. We bound a Z-region with a ZR-region. A ZR region is defined as below.

ZR-reign: A ZR-region is a small square spanning a ZR region covered by interval [A,B] A and B are minimum and maximum z-numbers.

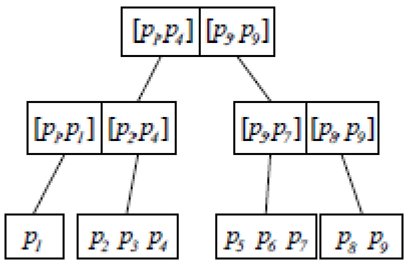

$[p_1, p_4]$ $[p_5, p_9]$
$[p_1, p_1]$ $[p_2, p_4]$
$[p_5, p_7]$ $[p_8, p_9]$
p_1
p_2 p_3 p_4
p_5 p_6 p_7
p_8 p_9

Figure 6. VS-tree

The above tree index structure takes more memory than required but it increases the search efficiency. In this tree structure leaf nodes used to hold the data and non-leaf nodes were used to represent the range or interval of descendent leaf nodes of that node. This process will be recursively continued in bottom up fashion to construct tree.

6.1. Algorithm for Dominance Test

```
Boolean Test_Dominate (T, Min, Max)
         Begin
                    Queue Q, NODE N;
              If (T==NULL)
               {
                 printf ("Tree Null");
                 Return (false);
               }
                 Enqueue (Q, T);
       While (! empty (Q))
        Begin
                 N=Dequeue (Q)
           If N not a leaf node then
           Begin
            For all children nodes K of N do
           If K's RZ-region's Max Dominates
              Min then return (True)
            Else
                 K's RZ-region's Min Dominates
                 Max then   Enqueue (Q, K);
          End
       Else
     Begin
      For all children nodes K of N do
        If K dominates Min then Return (True);
      End
   End
     Return (false);
End
```

6.2. Algorithm for Search

```
VSTree Search (VSTree T)
Begin
         VSTree SL, Stack S, NODE N;
  If (T== NULL)
{
  Printf (“No data source data”);
 Return;
}
Push(S, T);
  While (! Empty(S))
  Begin
       Xyz:   N=pop(S);
    if (Test_dominate (SL,Min (N),Max(N)) then
       Goto  xyz;
    if N is a non-leaf node then
      For all children nodes K of N do
     Push(S, K);
else
For all children nodes K of N do
if (! Test_Dominate (SL, Min (K), Max (K)) then
  Insert (SL, K);
End
Return (SL);
End.
```

7. Results

Here we evaluate our framework in terms of bandwidth consumption, computation time and resource utilization. In these experiments we have taken default values for some attributes of distributed system. Those are total database cardinality N as 2000K objects, object dimensionality d varying from 3 to 10, probabilistic threshold q varying from 0.3 to 0.9. and total number of distributed sites m equal to 50.These experiments were conducted on both uniform and anti-correlated data distribution on an average of 20 queries. It is evident that efficiency of proposed framework is superior to DUSD. All the below results were taken under the same parameter settings with an average of 20 queries.

7.1. Data Transmissions Vs. Dimensionality

We conducted experiment to evaluate the impact of distributed skyline query execution performance with dimensionality varying. These results were observed on two distributions, namely uniform and anti-correlated distributions with the default values such as local distributed data base cardinality N/m, number of local distributed sites(m) equals to 20,object dimensionality varying from 4 to 10. And threshold probability 0.4.With these default setting we observe that if dimensionality is increasing then domination gets decreasing so size of the local skyline is increasing thus the communication bandwidth expense will also increasing in both the distributions. Our proposed framework takes less bandwidth. Improved global probability threshold and early pruning makes our framework outstanding.

Table 4. Object Transmission vs. Dimensionality

Object Dimensionality(d)	Total Number of Objects Transmitted in SKYPLAN Approach	Total Number of Objects Transmitted in DUSD	Total Number of Objects Transmitted in Proposed VS-Framework
3	166	55	36
4	194	80	66
5	228	113	86
6	252	130	104
7	282	158	125
8	340	210	160
9	386	250	190
10	434	296	240

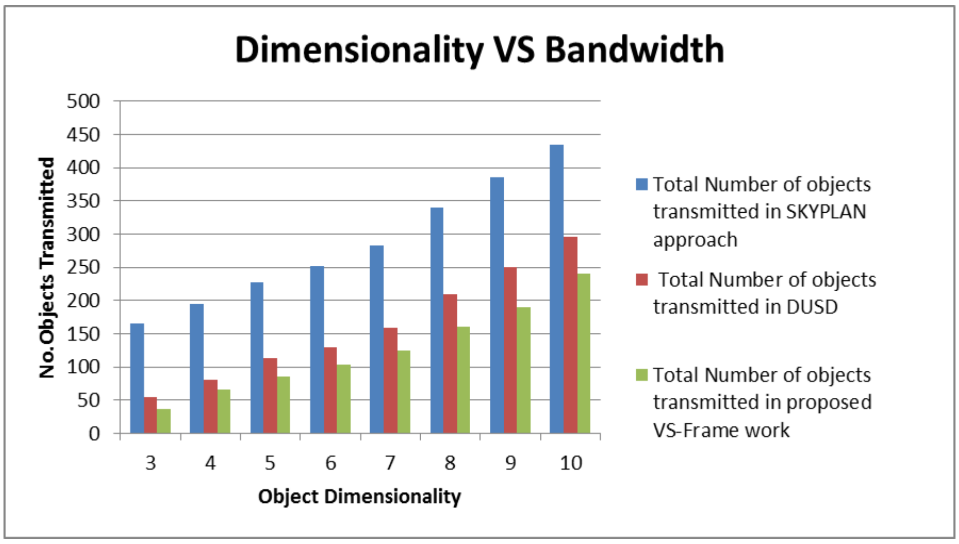


Figure 7. Dimensionality vs. Bandwidth

Table 5. Local Sites vs. Data Transmission

Total Number of Local Sites in Distributed System(m)	Total Number of Objects Transmitted in Skyplan Approach	Total Number of Objects Transmitted in DUSD.	Total Number of Objects Transmitted using Proposed VS-Framework.
4	225	124	80
5	248	140	98
6	264	166	138
7	292	188	168
8	326	227	187
9	382	242	209
10	434	296	240

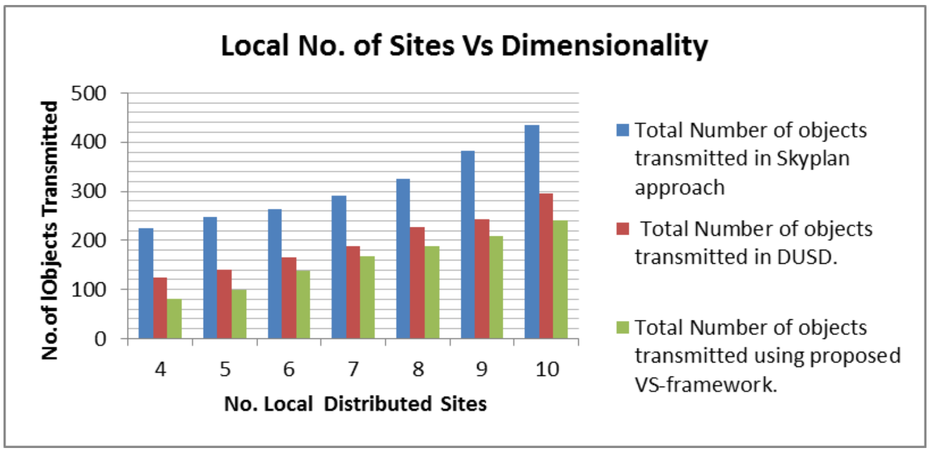


Figure 8. Local Site vs. Data Transmission

8. Conclusion

In this work, we proposed a novel method for organizing and retrieving of high dimensional data to enable skyline computation using Z-order curve. This will be useful in finding skyline in many applications handling high dimensional objects. Previous methods are having dimensionality curse. Their performance degrades with dimensionality increase. Here we study the skyline problems and identified the organization and grouping property of skyline process will be improved when we follow Z-order curve properties. With the help of B+tree and Z-order we propose to use new Index tree, VS-Tree, as a primary organizational mechanism to address efficient skyline processing issues like dominance and search. The Developed Search algorithm is best with respect to both dimensionality and cardinality, and firmly overcomes DUSD and BBS, the best search algorithms. We examined our approach with existing best algorithms with respect to different variants like k-dominance skyline queries. The result shows that our method is best.

References

[1] B. W. Tilo, U. Güntzer and J. X. Zheng, "Efficient distributed skylining for web information systems", Advances in Database Technology-EDBT 2004, Springer Berlin Heidelberg, **(2004)**, pp. 256-273.
[2] B. Kevin, J. Goldstein, R. Ramakrishnan and U. Shaft, "When is "nearest neighbor" meaningful", Database Theory-ICDT'99, Springer Berlin Heidelberg, **(1999)**, pp. 217-235.
[3] S. Borzsony, D. Kossmann and K. Stocker, "The skyline operator", Proceedings of 17th IEEE International Conference on Data Engineering, **(2001)**, pp. 421-430.
[4] B. Christian, "On the average number of maxima in a set of vectors", Information Processing Letters, vol. 33, no. 2, **(1989)**, pp. 63-65.
[5] C. C. Yong, H. V. Jagadish, K. L. Tan, A. K. H. Tung and Z. Zhang, "Finding k-dominant skylines in high dimensional space", Proceedings of the 2006 ACM SIGMOD international conference on Management of data, ACM, **(2006)**, pp. 503-514.
[6] S. Kaushik, S. Chaudhuri and N. N. Dalvi, "Robust cardinality and cost estimation for skyline operator", U.S. Patent 7,707,207, **(2010)**.
[7] J. Chomicki, "P. Godfrey. J. Gryz and D. Liang. Skyline with presorting", Proceedings of 19th Int., vol. 1, **(2003)**, pp. 717-719.
[8] G. Parke, R. Shipley and J. Gryz, "Maximal vector computation in large data sets", In Proceedings of the 31st international conference on Very large data bases, VLDB Endowment, **(2005)**, pp. 229-240.
[9] H. Zhiyong, C. S. Jensen, H. Lu and B. C. Ooi, "Skyline queries against mobile lightweight devices in MANETs", Proceedings of the 22nd International Conference on Data Engineering, 2006. ICDE'06, IEEE, **(2006)**, pp. 66-66.
[10] K. Donald, F. Ramsak and S. Rost, "Shooting stars in the sky: An online algorithm for skyline queries", In Proceedings of the 28th international conference on Very Large Data Bases, VLDB Endowment, **(2002)**, pp. 275-286.

[11] L. Xuemin, Y. Yuan, W. Wang and H. Lu, "Stabbing the sky: Efficient skyline computation over sliding windows", In Proceedings of 21st International Conference on Data Engineering, 2005. ICDE, **(2005)**, pp. 502-513.
[12] A. O. Jack and T. H. Merrett, "A class of data structures for associative searching", In Proceedings of the 3rd ACM SIGACT-SIGMOD symposium on Principles of database systems, ACM, **(1984)**, pp. 181-190.
[13] P. Dimitris, Y. Tao, G. Fu and B. Seeger, "Progressive skyline computation in database systems", ACM Transactions on Database Systems (TODS), vol. 30, no. 1, **(2005)**, pp. 41-82.
[14] R. Frank, V. Markl, R. Fenk, M. Zirkel, K. Elhardt and R. Bayer, "Integrating the UB-Tree into a Database System Kernel", In VLDB, vol. 2000, pp. 263-272.
[15] T. K. Lee, P. K. Eng and B. C. Ooi, "Efficient progressive skyline computation", In VLDB, vol. 1, **(2001)**, pp. 301-310.
[16] Y. Tao and D. Papadias, "Maintaining sliding window skylines on data streams", IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 3, **(2006)**, pp. 377-391.
[17] W. Ping, D. Agrawal, O. Egecioglu and A. E. Abbadi, "Deltasky: Optimal maintenance of skyline deletions without exclusive dominance region generation", In IEEE 23rd International Conference on Data Engineering, 2007. ICDE, **(2007)**, pp. 486-495.
[18] P. Wu, C. Zhang, Y. Feng, B. Y. Zhao, D. Agrawal and A. E. Abbadi, "Parallelizing skyline queries for scalable distribution", In Advances in Database Technology-EDBT 2006, Springer Berlin Heidelberg, **(2006)**, pp. 112-130.
[19] C. K. L. Ken, W. C. Lee, B. Zheng, H. Li and Y. Tian, "Z-SKY: an efficient skyline query processing framework based on Z-order", The VLDB Journal, vol. 19, no. 3, **(2010)**, pp. 333-362.