International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 22 (2017) pp. 12090-12095

Identification of Effective Retrieval Percentage for Ctree

Dr. S.Thabasu Kannan M.Tech., Ph.D., FIE.,
Principal, Pannai College of Engineering and Technology, Sivagangai – 630 561, Tamilnadu, India.

N.Mangalam M.Sc., M.Phil
Research Scholar & Lecturer, Department of Computer Science,
Raja Doraisingam Govt Arts College, Sivagangai – 630 561, Tamilnadu, India.

ORCID: 0000-0003-4728-4955

Abstract:

The main aim of the paper is to introduce a new indexing approach based on cluster CTree to find the nearest neighbor. This approach is used to represent clusters generated by a popular HARP algorithm, which forms the cluster in a bottom-up manner. Repeat the below two steps until there is only one remaining cluster in the pool; among all current clusters, pick the two clusters with the smallest distance and then replace the two clusters with a new one, formed by merging the two original clusters. In HARP, the accuracy and the scalability level is more by using relevance indexing and merge score. This method forms high similarity in the subspace formed by the relevant dimensions. It tries to find a number of disjoint clusters that optimize a certain evaluation function from all possible partitioning of objects and selections of relevant dimensions. In generally, a CTree is a hierarchy of clusters and sub-clusters, which incorporates the cluster representation into the index structure to achieve effective and efficient retrieval. This structure is highly adaptive to any kind of clusters. It is well accepted that most existing indexing techniques degrade rapidly when dimensionality goes higher.

Keywords: HARP Algorithm, Cluster CTree, Index Structure

INTRODUCTION

An index structure organizes the whole dataset to support efficient queries. Recently many applications require efficient access and manipulation of large-scale multi-dimensional datasets. The high dimensionality and enormous size of these datasets pose very challenging problems in indexing of the datasets for efficient querying. When the dimensionality increases and the dataset are very large, the efficiency of queries is a major consideration. To build an efficient index for a large dataset with higher dimensionality, the overall data distributions or patterns should be considered to reduce the affect of arbitrary inserting.

Our goal is to minimize the response time to a user's query. A CTree is a hierarchical representation of the clusters of a dataset. It organizes the data based on their cluster information from the coarse level to fine level, providing an efficient index structure on the data according to clustering.

The main contribution of this paper is:

- Introducing a new projected cluster based indexing techniques that do not rely on user parameters, which makes an automatic analysis of a large amount of data with little domain knowledge feasible.
- Providing a rich set of experimental results on synthetic and real datasets, including some comparison results between various tree-based indexing structures under many different situations.

This technique can be measured to be correct if its selected dimensions have high relevance values or it has a large number of selected dimensions or it contains a large number of objects. The reason for the first criterion is trivial, and the other two criteria ensure that the high relevance values of the selected dimensions are not due to random chance. In reality optimizing one property would usually sacrifice the other. Suppose a dimension is selected for a cluster if the average distance between the projected values

is below a certain threshold, then when the threshold is fixed, adding more objects to a cluster will probably decrease the number of relevant dimensions qualified for selection. In the same manner, if the members of a cluster are fixed, raising the threshold will probably reduce the number of dimensions qualified for selection. Again, a simple way to deal with the problem is to combine the criteria into a single score and let users decide the relative importance of each criterion. This solution may also affect the usability of the algorithms.

Construction of CTree

Algorithm CTree_build

1. Generate a root to represent all clusters

{

Create an entry Ei for each cluster Ci, add them into the root node; Push the root into stack;

}

2. If stack is not empty, current node = pop(Stack) and go to step 3; Else return;

3. For each entry Entryi in current node If Entryi.SN <= Page size then

{

a. Create a leaf node Childi for Entry i

b.Let the pointer SCiin Entryipoint to Childi c. Save data point into disk pages;

}

Else

{

a' Create a non leaf node Childi for Entryi

b' Let the pointer SCi in Entryi point to Childi c' generate k medoids;

generate sub-clusters to get k-subclusters for $Child_i$

d' create an entry for each subcluster and add them to $Child_i$*; push*$Child_i$ *into stack;*

}

Go to Step 2;

In the above algorithm, *page Size* is determined by the disk block size (M) and the size of a data point (S), where S=*dimensionality X sizeof(element of a data point)*. Where *pageSize= M/S*. The stack is used to storing the nodes which need to be processed. The child nodes will be created for each of the entries belonging to the popped node. If some of the child nodes need to be further split, then they will be pushed into the stack. When the stack is empty, which means that all of the nodes are processed, the procedure of creating the CTree is finished.

After this algorithm applied $O(\log N)$ times (N - the size of the whole dataset) the no of data points in each leaf node becomes equal to or less than the constant *pageSize*. Thus the maximum height of the CTree is $O(\log N)$. During the growth of the CTree, we need to scan the whole dataset $O(k)$times for each level of the CTree. The height of the CTree in the worst case is $O(\log N)$. Thus, the time complexity to construct the CTree is $O(k.N.\log N)$. In the average case, the time complexity to construct the CTree is

$O(k.N.\log_k N)$, since the height of the CTree, is $O(\log_k N)$.

DBIN is based on a probabilistic approach, and also achieves very high accuracy. The fraction of the data points that are searched in our approach is much lower than that of DBIN because the CTree decomposes a cluster into several sub-clusters. Instead of searching the entire cluster, it searches the related sub-clusters. DBIN only performs queries at the cluster level.

PERFORMANCE EVALUATION

The performance evaluation of the nearest neighbor search in CTree by the accuracy of retrieval results and its percentage is evaluated here. When the size of datasets grows, the dimensionality goes higher or the number of the nearest neighbors required increases. We need to show the efficiency of the CTree by comparing with the optimal search algorithm which can achieve the best performance.

Our approach can accurately perform the nearest neighbor search. For a given query, it only searches within the most related clusters for the nearest neighbors. From the above diagrams, the CTree achieved 100% accuracy on searching the nearest neighbors.

Figure 3 shows the detailed comparison between the CTree and SR-Tree on 10-dimensional datasets with different sizes. The synthetic datasets used are Datasets 1 to 10. From the figure 3, the retrieval percentage for 5-NN search decreases from 12.5% to 3%, and the retrieval percentage for 100-NN decreases from 13% to 6% as the size increases. When the size of the datasets is small, the cluster is very sparse.

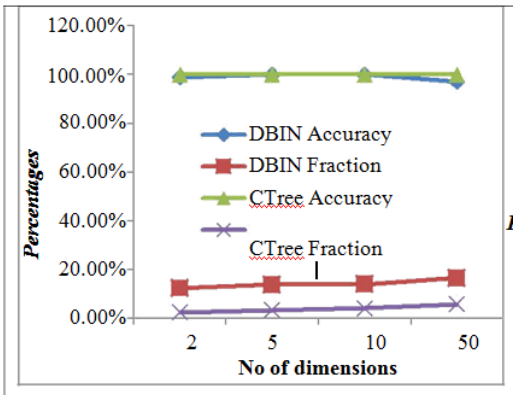

Figure 1: Accuracy and fraction for12-dimensional dataset with 10,00,000 data points.

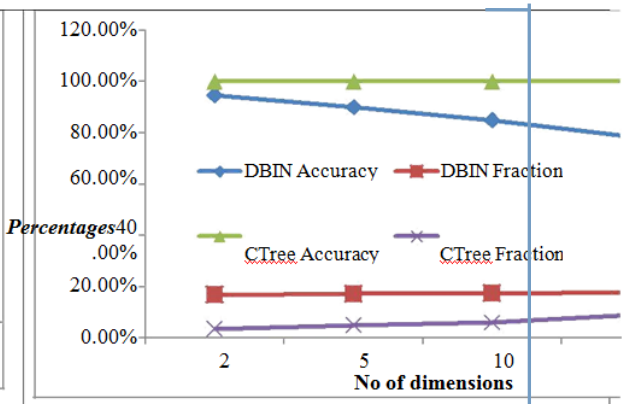

Figure 2: Accuracy and fraction searched for 30-dimensional dataset with 1,00,000 data points.

The search range can include most of the points. When the size of dataset increases, the clusters become dense, the neighboring data points are clustered into the same cluster or subcluster. Therefore, the search region can be reduced.The search methods used are Minimum Distance based search (MDS); Average Distance based Search (ADS); Optimized Distance based Search (ODS);

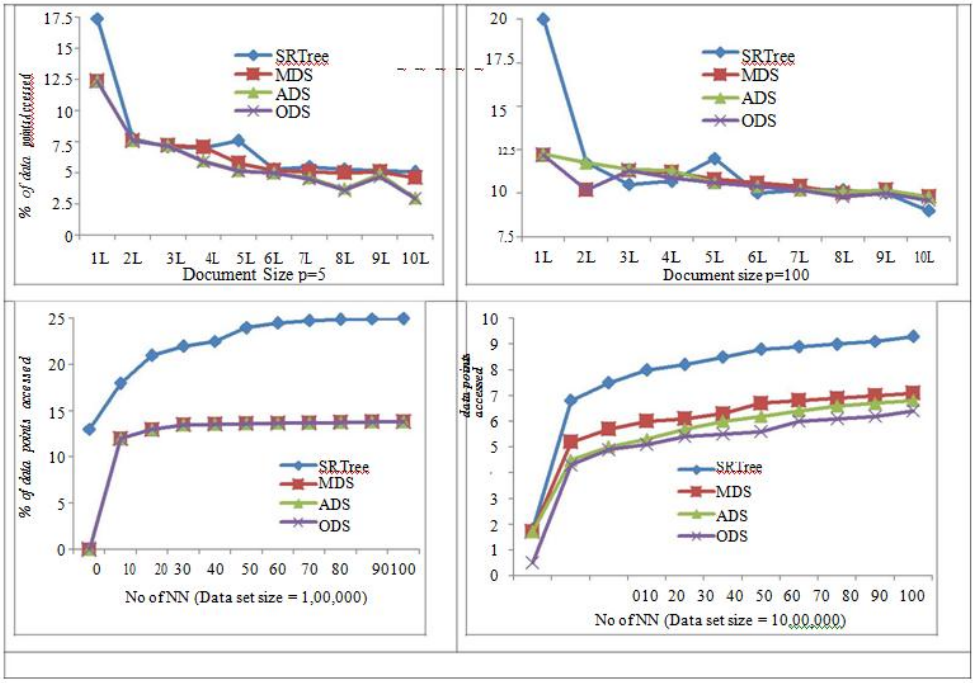

Figure 3: Retrieval percentage of CTrees and SR-Tree using datasets with the same dimensionality

For *small datasets*, we can afford to search a large part of the dataset, and the response time can still be very low. But for the *large datasets*, the retrieval percentage is crucial to the performance. The experiments showed that the retrieval percentage decreases when the size of the datasets becomes larger. This can reduce the cost of the CPU time and the number of disk accesses. Figure 3-a shows that ODS has uniformly lower retrieval percentage than MDS, ADS and the SR-Tree. The SR-Tree has relatively high retrieval percentage when the size is 1,00,000, but its retrieval percentage decreases very fast and outperforms MDS and ADS slightly when the size is 2,00,000 -4,00,000. MDS and ADS perform better than the SR-Tree when the size of the datasets is bigger than 4,00,000. Figure 3-b shows all the three search algorithms (ADS, MDS&ODS) have better performance than the SR-Tree. In figure 3-c, the retrieval

percentage for the three search strategies, ADS, MDS&ODS are very close. In figure 3-d, the CTree outperforms the SR-Tree for any given value of *p* in both datasets.

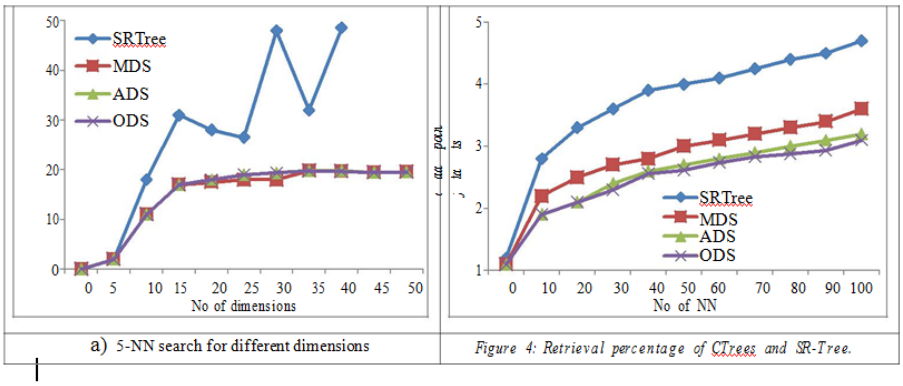

Figure 4: a shows the retrieval percentage of the 5-NN search when the dimension is 5, 10, ... 50 and the size of the datasets is 1,00,000.

Table 1: Retrieval percentage of CTree and SR-Tree for different p-value.

p	**Dimensions = 10**				**Dimensions = 40**			
	SRTree	**MDS**	**ADS**	**ODS**	**SRTree**	**MDS**	**ADS**	**ODS**
2	16.10%	10.61	10.35	10.20	48.40	17.60	19.08	17.59
5	17%	11.61	11.48	11.30	48.40	18.25	19.24	18.25
10	18%	12.10	12.10	11.87	48.50	18.27	19.26	18.26
15	18.5%	12.36	12.42	12.15	48.50	18.77	19.27	18.77
20	18.80	12.52	12.60	12.34	48.50	18.86	19.28	18.86
25	19.00	12.64	12.74	12.45	48.50	19.25	19.28	19.25
30	19.20	12.74	12.86	12.57	48.60	19.27	19.29	19.27
35	19.30	12.83	12.98	12.68	48.60	19.28	19.29	19.28
40	19.50	12.90	13.09	12.76	48.60	19.29	19.29	19.29
45	19.60	12.96	13.15	12.82	48.60	19.29	19.29	19.29
50	19.60	13.02	13.22	12.88	48.60	19.29	19.29	19.29
60	19.70	13/09	13.32	12.96	48.60	19.29	19.30	19.30
70	19.70	13.16	13.41	13/03	48.60	19.30	19.30	19.30
80	19.80	13.22	13.47	13.10	48.70	17.59	19.30	19.30
90	19.80	13.26	13.52	13.14	48.70	19.30	19.30	19.30
100	20.00	13.30	13.57	13.18	48.70	19.30	19.30	19.30

The retrieval percentage of the SR-Tree reaches 50% when the dimension is 40, while the CTree still keeps the retrieval percentage lower than 20%. The MDS performs slightly better than ADS, and ODS achieves the best performance among these three search methods. Figure 4-b shows the retrieval percentage of *p*-NN search in a 5-dimensional dataset with p = 1, 2, ... 100. Table 1 lists the retrieval percentages for dimensions 10 and 40. When the dataset is 5-dimensional, ADS performs better than ODS (shown in Figure 4-b), while MDS shows better performance when the dataset is 10-dimensional (shown in Table 1). In both datasets, ODS has lower retrieval percentage than MDS and ADS. ODS takes advantage of the data distribution, therefore, it achieves better performance than ADS and MDS. As Table 1 shows, when the dataset is 40 dimensional, the SR-Tree searches half of the dataset to get the nearest neighbors. This is because of the distribution of the distances between the data points within the dataset show that the minimum distance between any two data points grows drastically as dimensionality increases, and the ratio between the minimum distance and maximum distance is more than 60% when the dimensionality is 40. That means that the variation of distance decreases when the dimensionality increases. This implies that the ratio between *dmin*and *davr*is close to 1. Therefore, MDS, ADS and ODS have similar performance when dimension is 40, as shown in Table 1. With the introduction of clusters and sub-clusters, the retrieval percentage of the CTree is 30% lower than that of the SR-Tree.

CONCLUSION

This paper clearly shows that the CTree has very low retrieval percentage, therefore the time spent on reading the data points into main memory from disk and checking whether the data points are the nearest neighbors is relatively low when compared with SR-Tree.

This technique is used to simultaneous determination of both cluster members and relevant dimensions. Cluster members are determined by calculating object distances in the subspace formed by the relevant dimensions, while the relevant dimensions are determined by measuring the projected distances of the cluster members along different dimensions.

The *second challenge* of this technique is the evaluation of right quality, which is in turn related to the determination of the dimensionality of each cluster. Traditionally, the quality of a cluster is measured by some objective functions. If a correct clustering model is chosen, a better objective score implies a larger chance that the clusters formed are correct.

One possible solution is to get the dimensionalities of the clusters by some other means, and then optimize the objective function subject to the dimensionality requirements. The simplest way to obtain the cluster dimensionalities is to set them as algorithm parameters and request users to supply the values. While this solution has been adopted in some of the projected clustering algorithms, it has a usability implication.

REFERENCES

[1] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman and A.Y. Wu: *"An OptimalAlgorithm for Approximate Nearest Neighbor Searching Fixed Dimensions"*,Journal of the ACM (JACM), Vol.45, No.6, pp.891-923, 2013.

[2] S. Berchtold, C. Böhm, D.A. Keim and H.P. Kriegel: "A Cost Model for NearestNeighbor Search in High-Dimensional Data Space", *Proceedings 18th ACMPODS Symposium (PODS'97)*, pp.78-86, Tucson, AZ, 2009.

[3] S. Berchtold, D.A. Keim and H.P. Kriegel: "The X-tree: An Index Structure forHigh- Dimensional Data", *Proceedings 22nd VLDB Conference*, pp.28-39,Bombay, India, 2016.

[4] N. Beckmann, H.P. Kriegel, R. Schneider and B. Seeger: "The R*-tree: AnEfficient and Robust Access Method for Points and Rectangles", *Proceedings1990 ACM SIGMOD Conference*, pp.322-331, Atlantic City, NJ, 2015.

[5] A. Corral, M. Vassilakopoulos and Y. Manolopoulos: "Algorithms for Joining RTreesand Linear Region Quadtrees", *Proceedings 6th of Advances in SpatialDatabases (SSD'99)*, pp.251-269, Hong Kong, China, 1999.

[6] V. Gaede and O. Günther: "Multidimensional Access Methods", *ACM ComputingSurveys*, Vol.30, No.2, pp.170-231, 1998.

[7] E.G. Gilbert, D.W. Johnson and S.S. Keerthi: "A Fast Procedure for Computing the Distance Between Complex Objects in Three-dimensional Space", *IEEE Journal of Robotics and Automation*, Vol.4, No.2, pp.193-203, 1988.

[8] G.R. Hjaltason and H. Samet: "Incremental Distance Join Algorithms for SpatialDatabases", *Proceedings 1998 ACM SIGMOD Conference*, pp.237-248, Seattle,WA, 1998.

[9] N. Katayama and S. Satoh: "The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries", *Proceedings 1997 ACM SIGMODConference*, pp.369-380, Tucson, AZ, 1997.

[10] S.T. Leutenegger, J.M. Edgington and M.A. Lopez: "STR: A Simple and EfficientAlgorithm

for R-Tree Packing", *Proceedings of the 13th International Conferenceon Data Engineering*, pp.497-506, Birmingham, United Kingdom, 1997.

[11] M.C. Lin and J.F. Canny: "A Fast Algorithm for Incremental DistanceCalculation", *Proceedings of IEEE International Conference on Robotics andAutomation*, pp.1008-1014, Sacramento, CA, 1991.

[12] Y. Manolopoulos, Y. Theodoridis and V. Tsotras: "Advanced DatabaseIndexing", *Kluwer Academic Publishers*, 1999.

[13] A.N. Papadopoulos and Y. Manolopoulos: "Performance of Nearest NeighborQueries in RTrees",*Proceedings 6th International Conference on DatabaseTheory (ICDT'97)*, pp.394408,Delphi, Greece, 1997.

[14] H. Shin, B. Moon and S. Lee: "Adaptive Multi-Stage Distance Join Processing",*Proceedings 2000 ACM SIGMOD Conference*, pp.343-354, Dallas, TX, 2000.

[15] T. Sellis, N. Roussopoulos and C. Faloutsos: "The R+-tree: A Dynamic Index forMultiDimensionalObjects", *Proceedings 13th VLDB Conference*, pp.507518,Brighton, UK, 1987.