Query Sampling Based High Dimensional Hybrid Index

Junqi Zhang  Xiangdong Zhou
Fudan University, Shanghai, China
{041021054, xdzhou}@fudan.edu.cn

ABSTRACT
Sparse and aggregate data exists in the same multimedia data set. Hence, the selection of an appropriate strategy to index such data is very difficult. To solve this problem, we propose a novel hybrid index to speed up processing of high-dimensional K-nearest neighbor (KNN) queries. In the first step the cluster analysis and cluster splitting methods are applied to construct a tree-based index, then the relationship between data distribution and index performance is derived by sampling. At last some tree branches with sparse data are extracted for linear scan, while the aggregate data remains in the tree. The complexity of the proposed sampling algorithm is only $\sqrt{N}$ (N is the size of data set). The proposed hybrid index improves the query efficiency by adaptively selecting different index strategies for the data with different distribution. Extensive experiments show that the proposed hybrid index structure performs better than iDistance, M-Tree and linear scan, and scales better with dimensions. The index is still faster than linear scan when the dimension reaches 336.

Keywords
K-nearest neighbor queries, sampling, high-dimensional index, outlier, cluster splitting

1. INTRODUCTION
Many emerging database applications such as image, medical imaging, time series and scientific databases, manipulate high-dimensional data. In these applications, one of the most frequently used and yet expensive operation is to find objects in the database that are similar to a given query object. Nearest neighbor search is a central requirement in such cases. There is a long stream of researches on solving the nearest neighbor search problem, and many multi-dimensional indexes have been proposed. Existing multi-dimensional indexes can be classified into tree-based and scan-based indexes. The tree-based indexes, such as R-tree[1] and its variations and M-tree[2], improve the pruning capability by data division. The scan-based indexes like VA-file[3], reduces the amount of data that must be read during similarity searches by scanning the approximation file.

The partitioning based index is meaningful only if the data aggregation is good enough[4]. The data distribution tends to be uniform and the tree based indexes fail when the data dimension is far higher than ten. In 1998 Webber et al.[3] prove that for any clustering and partitioning method there is a dimensionality d beyond which a simple sequential scan performs better and in practice this threshold d will be well below 610, which is called dimensionality curse. When the dimensionality is below 10 most of the existing tree-based indexes have proven their validity due to the uneven distribution and the fine data aggregation. In this case tree-based indexes perform better than VA-file because that VA-file brings the expensive computation of decompression and has low filter capacity in the uneven data distribution. Obviously it is appropriate to choose tree-based index for the low-dimensional data and the linear scan-based indexes for the high-dimensional data. However, it is not easy to judge which index strategy is better when the dimensionality is in the medium scale that is between 10 and 610. On the one hand, there is no direct corresponding relation between dimensionality and the data distribution. The aggregation in the data set with higher dimension is not necessarily worse. On the other hand, there exist two different types of data that are aggregate and sparse data in the same real data set. The aggregate data is suitable to be stored in the tree-based index and the sparse data is suitable for linear scan. Finally, there not exist direct functions for judging the data types and the derivation from the degree of the data aggregation to the index strategies.

How to choose suitable index strategies is challenging for the commonly seen multimedia data with the dimension between dozens and hundreds. Hence, the investigation shall be made into the relationship between data distribution and index strategies. Tree-based indexes filter aggregate data more efficiently and scan-based indexes perform better for sparse data. The degree of the data aggregation is determined by the proportion and distribution of aggregative and sparse data. The data aggregation is weakened and the data sparseness is enhanced with the dimension growth. When tree-based index is applied alone to the multimedia data with medium-sized dimension the sparse data inevitably joins in the aggregate clusters so that the clusters turn to be loose and the average cluster radiuses turn larger,
which brings the decreased filter capacity and worse performance to the tree-based indexes. When the scan-based index is applied alone, it is inappropriate to discard the tree-based index strategy in theory. Therfore, a single index strategy cannot distinguish between data with the medium-sized dimension, which limits the tree-based and the scan-based index strategies to play their respective strengths and inhibits the index from the scalability to the data dimension.

This paper proposes a novel hybrid index to support the medium-sized dimensional multimedia data, which improves the query efficiency by adaptively choosing different index strategies for the data with different distribution. The combination of the tree-based and the scan-based strategies is a smooth transition to enhance the index scalability to dimensions. The tree-based index fades out and gradually transforms to the scan-based index with the dimension growth. Due to the difficulty of grasping the actual data distribution we propose a construction-and-extraction strategy. A tree-based index is constructed in the first step according to the actual data distribution. Then some tree branches with sparse data are extracted for linear scan based on the contribution of different data for the index performance. In order to obtain the actual data distribution the cluster analysis is applied, and then the clusters are further split into rings according to the internal data distribution of clusters. After the data partitioning we propose a sampling algorithm whose complexity is only $\sqrt{N}$ (N is the size of data set), by which the average probability of the data access in the tree-based index is obtained, and then the contribution of different data for the index performance is derived. With these preliminary knowledge the tree branches with sparse data are extracted for linear scan, while the aggregate data remains in the tree. Extensive experiments show that the proposed hybrid index structure is more efficient than iDistance, M-Tree and linear scan, and scales better with dimensions. The index performance is still faster than linear scan when the dimension reaches 336.

The rest of the paper is organized as follows: Section 2 is about the related work; Section 3 presents the hybrid index structure and the sampling algorithm; Section 4 elaborates on the query algorithm of the hybrid index; Experimental results are given in section 5; The final section concludes the work.

## 2. RELATED WORK

The tree-based indexes improve the query efficiency by partitioning the data. The scan-based indexes reduce the amount of data that must be read during similarity searches by scanning the approximation file. Tree-based indexes can filter the aggregate data efficiently and the scan-based indexes perform better for the sparse data. Some studies combine the tree-based and the scan-based strategies in a single index structure. These methods can be classified into two categories. One is based on the scan-based index and combines the tree-based strategy. Such as in 2002, Berchtold et al. propose the IQ-tree[5], which builds a tree-based index for VA-file to avoid to scan the whole approximation file. In 2002, Guang et al. propose the GC-tree[6], which indexes the approximation unit based on a density function that identifies dense and sparse regions in a data space. However, the criterion for judging whether the data is aggregate data needs to be predefined. The other type of the combination is based on the tree-based index and combines the scan strategy. Such as in 1996, Berchtold et al. propose the X-tree[5], which uses a split algorithm minimizing overlap and additionally utilizes the concept of supernodes for the R-tree based index structure. In 2000, Bohm et al. propose an algorithm[7] for the management of data pages with varying page-sizes in an kd-tree, which uses a flat directory whose entries consist of an MBR and uses a cost model to estimate whether the split operation is beneficial. In 2001, Yu Cui et al.[8][9] propose the iDistance, which clusters data first and find the optimal reference point for each cluster. Then the distance between an object and the reference point in the cluster to which the object belong can be indexed in a B+-tree. In the B+-tree, leaf nodes are linked to both the left and right siblings. This is to facilitate searching the neighboring nodes as local linear scan when the search region is gradually enlarged. In 2003, Bin Cui et al.[10] partition clusters into small regions to reduce the search space. In 2004, Edgar et al.[11] combine the concepts of spatial approximation and pivot-based algorithms, in which a sequential scan is applied to the overflow area, using pivots to prune the search space for free and additionally using the information obtained during the search into the tree. These hybrid indexing methods recognize that the data with different distribution shall be indexed with different strategies. However, a single index strategy cannot distinguish between data for the medium-sized dimensional real data, which limits the tree-based and the scan-based index strategies to play their respective strengths and reduce the adaption to the data dimension. The methods that are used to estimate the data distribution by cost model or data volume are not accurate and cannot adaptively reflect the real data distribution.

In order to tune and compare index structures, it is vital to have efficient cost prediction techniques for these structures. The query cost is generally estimated by two methods which are cost model and sampling technique. The cost models based on vector space or metric space can estimate the index performance, like M-tree[6], which is a paged, balanced, and dynamic secondary memory structure able to estimate data sets from generic metric spaces. M-tree adopts a filtering and refining scheme that checks the data in the area intersected with query region using the optimal KNN search algorithm[12]. Because of its strategy of splitting, the space efficiency of M-tree is low. Ciaccia et al. present a cost model[13][14] for M-tree based on the distance distribution, based on which the page size is tuned to optimize the index performance. Cost models can be classified into two set models according to the way the techniques model the data. The first set of approaches assumes uniformity of the data. The advantage of it is that they are simple, require only very few parameters (like the page capacity), and are relatively fast to compute. The disadvantage is that the uniformity assumption does not hold for real and high dimensional data and leads to bad prediction results in these cases. The second set models the data using global or local parameters. The advantage of these technique is that they can model data better than the simple uniformity assumption. The disadvantage is that they are not applicable on high dimensional real data sets.
Random sampling is a standard technique for query optimization in commercial databases because of its independence of the dimensionality and preservation of clusters which is important for representing skewed data. In 2001, Cristian et al. propose a general model for using sampling to predict the number of accessed index pages during a query execution under restricted memory assumptions in order to tune and compare index structures. In 2006, Jayendra et al. develop a reference-based index that reduces the number of costly edit distance computations in order to answer a query. In order to select references that represent all parts of database they develop a sampling strategy which finds a mapping of references to sequences that maximizes the pruning rate with a given probability. The advantage of sampling is that it preserves clusters and is simple and effective to the high-dimensional real data set. The disadvantage is that the accuracy of sampling is proportional to the query cost. Therefore, trade-off between accuracy and query cost of sampling is needed in large database. Furthermore, sampling can be guided by the combination with cost model to reduce the sampling cost and improve the index performance.

3. HYBRID INDEX STRUCTURE

3.1 Data Partitioning
To obtain the actual data distribution, the cluster analysis is applied in the first step, then clusters is further split into circles according to the internal data distribution of clusters. In the first step K-means clustering method is applied for the overall data analysis and the preliminary data distribution information is obtained. The intersection probability of cluster region and query area is proportional to the cluster volume, which is \( \frac{\pi r^2}{\sqrt{\pi d^2+1}} \). \( r \) is the cluster radius and \( d \) is the dimensionality of the data set. Hence the intersection probability of cluster region and query area is proportional to \( r^d \). Clearly the intersection probability of the marginal region in cluster and query area is higher than other cluster regions. Therefore clusters is further partitioned more finely to avoid searching the whole cluster whenever the cluster intersects the query region marginally. Therefore, for each data cluster, we estimate the density of the data points, and select an appropriate ratio, such as 1:1 for data points in each part to split the cluster into two sub-clusters: the core and the marginal sub-cluster. Suppose \( m \) clusters are obtained from the K-means clustering. We have \( 2m \) cluster rings after the splitting. Fig.1 shows the idea for cluster splitting.

![Figure 1: Cluster Partitioning](image)

3.2 Building the Hybrid Tree Index
Before building the hybrid index a B+-tree-based index is constructed by mapping the data points in the high-dimensional space to the value in the single dimensional space. Some technical details are given below: (a) Selection of the index reference point: in the index construction, we use principal component analysis (PCA) to find the reference point \[ p \]. (b) Cluster splitting is conducted. (c) The index key assignment formula for the B+-tree construction is defined as follows:

\[
y = i \times C + \text{dist}(p, O)
\]  

Let \( p(x_1, x_2, \ldots, x_d) \) be a random point vector, and \( 0 \leq x_j \leq 1, 0 \leq j \leq d \), \( d \) be the data dimensionality, \( i = 1, 2, \ldots, m \) be the number of the cluster rings, \( O \) be the index reference point. \( C \) is the hash factor, which is same as the iDistance, \( \text{dist}(p, O) \) denotes the distance between \( p \) and \( O \).

Fig.2 shows the proposed hybrid index, in which cluster 1 and cluster 2 are two clusters and are separately split into several rings. The rings \( C \times 1 \), \( C \times i \) are marginal rings derived by the sampling algorithm in the next section. After building a tree-based index all data in the marginal rings is stored in a sequential file, and the data in the rest rings still remains in the B+-tree. Query \( Q \) intersects with the rings \( C \times 3 \), \( C \times i \) and \( C \times (i + 1) \). Thus, the rings \( C \times 1 \) and \( C \times 2 \) containing unrelated data with query \( Q \) is filtered.

![Figure 2: Cluster Partitioning Based Hybrid Index](image)

3.3 Analysis of the Marginal Data in the Tree-Based Index
This paper analyzes the relationship between data distribution and index performance on a cluster splitting based image retrieval system (Fig.3). The index of the system is built on the a 58 dimensional data set using the B+-tree. The number of clusters is 100 and these clusters are split into 600 rings. The columns in the upper left corner of the system interface are used to elaborate our cluster splitting method, and show how it can improve the query efficiency. In Fig.4, We use the columns to show that how four random queries are sped up by the index. Each column represents a cluster and is sorted in ascending order according to the distance from the query. In each column the rings are displayed from the bottom up in ascending order according to the distance from the cluster center. Red represents that the ring is visited and green represents the ring is not visited during the query process. The following phenomena can be observed from the system: First, the more rings are visited the poorer the index performance is. Second, the cluster splitting method avoids searching the whole cluster whenever the cluster intersects the query region marginally. Third, the marginal rings in the clusters are frequently visited almost by all queries, which weakens the index perfor-
Function that rings are visited by queries. \( N_i \) is the data size of cluster ring \( i \). Let \( b \) be the maximum capacity of a node and \( u \) be the node fanout (the average number of entries in a node, typically \( u \approx 69\% b \)). \( H \) is the height of the intermediate nodes on the tree. The linear scan cost for cluster ring \( i \) is \( \frac{n_i}{u} \), while the query cost for it on the \( B^+ \)-tree is \( P(i)(H + \frac{N_i}{u}) \). We define the difference of the both cost as the index capability (IC) over cluster ring \( i \). When \( IC \) is positive, the linear scan cost for the data in the cluster ring is greater than the query cost on the \( B^+ \)-tree. When \( IC \) is negative, the linear scan strategy is more appropriate for the data in the cluster ring. To get the relationship between the data distribution and the index strategies, these thresholds are essential to the judgement upon which the tree and the high probability of being visited by queries. The marginal rings shall be extracted from the tree and be linear scanned due to the more expensive positioning cost on the tree and the high probability of being visited by queries. However, the relation between the data distribution and the index strategies is essential to the judgement upon which circle are marginal circles.

### 3.4 Sampling Algorithm

The average probability that data is visited on the tree is the key to the judgement whether the data is suitable to be stored in the tree-based index. However, it is very difficult to estimate the accurate average probability of the data to be visited by queries based on the cost model. Hence, we treat the cluster rings as the basic units. When the average probabilities that rings are visited on the tree reach some thresholds, the linear scan-based strategy is more suitable for these rings. To get the relationship between the data distribution and the index strategies, these thresholds of probabilities shall be estimated accurately.

Let \( c_i \) be the \( i-th \) cluster ring. \( P(i) \) is the probability

**Definition** 1: The index capability (IC) of cluster ring \( i \) is defined as: \( IC_i = \frac{N_i}{u} - P(i)(H + \frac{N_i}{u}) \)

**Definition** 2: The cluster rings with negative IC are marginal rings, the data in which is marginal data.

**Definition** 3: The threshold probability of cluster ring \( i \) is defined as: \( \frac{uN_i}{H + \frac{u}{b}N_i} \)

After the tree-based index built a query set \( Q \) is predefined by random sampling. Queries of \( Q \) is executed on the \( B^+ \)-tree index, during which the statistical probabilities that rings are visited are obtained. The higher the probabilities, the worse the index pruning capability on such rings. The rings are inappropriate stored in the tree and are extracted from the \( B^+ \)-tree into the sequential marginal data file when the average probabilities that rings are visited by queries are greater than their thresholds of probabilities. Hence, the sampling algorithm can catch the more accurate relationship between data and index strategies, which avoids the limitation of the estimation in traditional cost model and assumption on the data distribution. But for large databases, excessive queries in the query set \( Q \) shall introduce too much sampling cost. It is necessary to trade off between the sampling quality and cost. The accuracy of the sampling algorithm is guessed at by making use of the Central Limit Theorem (CLT), which implies that errors over estimated sums is the number of sampling. There exists a \( t \)-distribution: \( \frac{P - \mu}{\sqrt{\sigma}} \sim t_\xi (n-1) \). We terminate sampling when \( \frac{S}{\sqrt{\sigma}} (n-1) \) is less than \( \xi \) and we can be assured that the relative error is less than with \( (1-\alpha)\% \) probability, which is defined as the termination condition of the sampling algorithm. It is a binomial distribution about whether the \( i-th \) cluster ring is visited and \( S_i = P_i(1-P_i) \). Hence, \( S_i \leq 0.25 \). The smaller \( \xi \), the smaller the sampling error.
To trade off between the sampling accuracy and efficiency we let the number of sampling where $N$ is the data size and $m$ is a constant. Let the sampling error $\xi = \sqrt{\frac{2m}{\sqrt{N}}\frac{t}{2}}(n-1)$. Thus the sampling algorithm terminates by all means when the number of sampling $n \leq m\sqrt{N}$. In the experiment we prove that the sampling algorithm gains the same sampling accuracy at $m1(n \leq \sqrt{N})$ as the that when $m = 10$.

### 3.4.2 Speeding Up the Termination of the Sampling under the Same Confidence

The sampling algorithm can be traded off between the accuracy and the efficiency by $m$ of the error threshold and the confidence $(1-\alpha)\%$. When $\sqrt{\frac{2m}{\sqrt{N}}\frac{t}{2}}(n-1)$ is less than $\xi$ we can be assured that the relative error is less than with $(1-\alpha)\%$ probability. In practice the purpose of the sampling algorithm is to estimate the sign of the index capability $(IC)$ of each ring. Hence we can determine the sampling earlier when the sign of $IC$ no longer changes again under the same confidence. We use the following theory to further reduce the sampling amount under the same confidence.

**Theory1:** The sampling algorithm can be terminated earlier under the same confidence when the following conditions are satisfied: $P(i) < \frac{\sum_{i}^{N}n}{H_{b}+B_{n+1}} - \frac{\sum_{i}^{N}n}{\sqrt{N}}t(\frac{n-1}{2})$ or $P(i) > \frac{\sum_{i}^{N}n}{H_{b}+B_{n+1}} + \frac{\sum_{i}^{N}n}{\sqrt{N}}t(\frac{n-1}{2})$.

**Prove:** When $IC$ is equal to zero, $\frac{\sum_{i}^{N}n}{H_{b}+B_{n+1}} = P(i)(H + \frac{N}{2}) \Rightarrow P(i) = \frac{\sum_{i}^{N}n}{H_{b}+B_{n+1}}$, the value of $P(i)$ is the threshold of probability about whether the ring is suitable to be stored in the tree index. Obviously, the sign of IC no longer changes with the $(1-\alpha)\%$ probability when the current average probability that the $i$-th ring is visited by queries is in the following ranges: $P(i) < \frac{\sum_{i}^{N}n}{H_{b}+B_{n+1}} - \frac{\sum_{i}^{N}n}{\sqrt{N}}t(\frac{n-1}{2})$ or $P(i) > \frac{\sum_{i}^{N}n}{H_{b}+B_{n+1}} + \frac{\sum_{i}^{N}n}{\sqrt{N}}t(\frac{n-1}{2})$.

### 4. NEAREST NEIGHBOR QUERIES

Tree-based strategy filters the aggregate data more efficiently and the scan-based strategy performs better for the sparse data. After the tree-based index built some tree branches with sparse data are extracted from $B^+$-tree and stored into a marginal sequential file for linear scan, while the aggregate data remains in the tree. So the $KNN$ search algorithm[12] first scans the marginal data on the sequential file and then uses the optimal $KNN$ search algorithm to search the rest data on the $B^+$-tree, finally the accurate result is obtained. Thus, the initial query radius of the optimal $KNN$ search algorithm converges to a relative smaller value after scanning on the marginal sequential file, so as to accelerate the search on the $B^+$-tree. In the $KNN$ search algorithm the queue $PQ$ contains ascending distances which are the distances between query $Q$ and the outer circles of rings $C$ with center $O$, where $d$ is the distance metric, $r$ is the outer radius of ring $C$, $d(Q,O)$ is the distance between query $Q$ and the ring center $O$, and $d(Q,C) = \max(0, d(Q,O) - r)$. The cluster rings in queue $PQ$ are judged in sequence by whether intersected with the query region. If the current ring $C$ intersects with the query area, all distances between the data in ring $C$ and query $Q$ shall be evaluated. The query radius shrinks dynamically during the query process and reaches a convergence in the end. When the current ring does not intersect with the query region, the search ends. When the data volume is huge and the data dimension is high, VA-file like scan-based index strategies can be applied to the marginal sequential file. The $KNN$ search algorithm is given in Algorithm 1.

**Algorithm 1** algorithm of K-NN search

| Input: $Q$ is the query point, $K$ is a integer of K-NN search, $M$ is the number of cluster rings, $nnQ,K$ is the distance from $Q$ to the $k$th nearest neighbor, $SD$ is sequence data file where the marginal ring is stored, $P$ is the reference point. |
| Output: the result set $RL$ |

**Initialization:** $RL[i] = +\infty$, $i = 1, \ldots, K$, $nnQ,K = +\infty$ for all $O_j \in SD$ do if $d(Q,O_j) \geq nnQ,K$ then $RL$ is updated by $[O_j, d(Q,O_j)]$ $nnQ,K = d(Q,O_j)$ end if end for all cluster circles are sorted by $d(Q,C_i)$ in ascending order in the queue $PQ$, $[C_i, d(Q,C_i)]$ is the element in $PQ$, $i = 1, \ldots, M$ while $PQ \neq \phi$ do $C_i$ is the first cluster circle in $PQ$ and is picked off if $d(Q,C_i) \geq nnQ,K$ then exit else for all $O_j \in C_i$ do if $|d(Q,P) - d(O_i, P)| < nnQ,K$ then compute $d(O_j, Q)$ if $d(O_j, Q) < nnQ,K$ then $RL$ is updated by $[O_j, d(Q,O_j)]$ $nnQ,K = d(Q,O_j)$ end if end if end for end if end while

### 5. EXPERIMENTS

We test our method on the following three aspects: (a) The query time comparison between the proposed hybrid index and other existing indexes. (b) The performance of sampling. In all experiments the index performance is obtained by 10,000 random queries. Our test system is running on a PC with one P4 2.6GHz CPU and 768MB memory. Three data sets are used: D1: A collection of 32 dimensional color histogram feature vectors obtained from http://kdd.ics.uci.edu/databases/CorelFeatures/CorelFeatures.data.html. It contains 68,040 images, and is
is lower than linear scan on Due to the strategy of node splitting, the efficiency of M-tree scales better with dimensions than other methods.

The cluster analysis and cluster splitting methods are applied to construct the tree-based index in the first step, and then the relationship between data distribution and index performance is derived by sampling. At last some tree branches with sparse data are extracted for linear scan, while the aggregate data remains in the tree. The experiments show that the sampling algorithm based hybrid index is more faster than iDistance, M-Tree and linear scan, and scales better with dimensions than other methods.

6. CONCLUSIONS
In this paper, we propose a novel hybrid index to speed up processing of high-dimensional K-nearest neighbor (KNN) queries. The cluster analysis and cluster splitting methods are applied to construct the tree-based index in the first step, and then the relationship between data distribution and index performance is derived by sampling. At last some tree branches with sparse data are extracted for linear scan, while the aggregate data remains in the tree. The experiments show that the sampling algorithm based hybrid index is more faster than iDistance, M-Tree and linear scan, and scales better with dimensions than other methods.

7. ACKNOWLEDGMENTS
This work is partially supported by NSF of China grants No.60403018, by NSF of Shanghai grants No.04ZR14011.

8. REFERENCES