Case Study: Distance-Based Image Retrieval in the MoBIoS DBMS

Rui Mao[1], Qasim Iqbal[2], Wenguo Liu[1], and Daniel P. Miranker[1]
[1] *Department of Computer Sciences*
Center for Computational Biology and Bioinformatics
University of Texas at Austin
1 University Station C0500, Austin, TX 78712-1188 USA
{rmao, liuwg, miranker}@cs.utexas.edu
[2] *LifeSize Communications, Austin, Texas 78746, USA*
qiqbal@lifesize.com

Abstract

Similarity search leveraging distance-based index structures is increasingly being used for complex data types. It has been shown that for high dimensional uniform vectors with similarity norms, any clustering and partitioning index method is outperformed by sequential scan. However, intrinsic clustering of real data usually leads to low intrinsic dimensionality. MoBIoS (the Molecular Biological Information System) is a next generation database management system comprising distance-based indices. Owing to its generality, we have built, evaluated and optimized a prototype of a distance-based image retrieval system. We show that under a metric distance function, image data is intrinsically low dimensional. We investigate the performance of three distance-based index structures (M-tree, RBT-index, and MVP-index), and, to optimize the construction of MVP-indexes, develop new heuristics that seek centers as pivots and partition the data according to its intrinsic clustering. Last, we show the SQL extension to embody distance-based image retrieval in MoBIoS.

1. Introduction

Today, a great challenge in databases is to manage various nontraditional types of data, such as spatial objects, video, image, voice, text and biological data types [4, 17, 19]. Management of images within a large database is an important and emerging area of research. Answering content-based queries typically requires representing data objects by their feature vectors and computing the relative distances between feature vectors, which is typically a very costly operation. The distance function often has the metric properties, i.e. non-negativity, symmetry and triangle inequality. Retrieval in large disk-resident datasets has to be supported by external data structures optimizing both disk I/O and the number of distance calculations.

This research was supported by the National Science Foundation contracts DBI-0241180, IIS-0325116, and EF 03-31453.

Multi-dimensional index methods, such as k-d trees [1] and R-trees [8], can be applied to images. In these methods, for each image a (usually high dimensional) feature vector is extracted so that the image can be represented by a spatial point or a spatial object in a high-dimensional vector space. Typical search operations are range query, i.e., find all objects that are within a given distance to the query point, and the nearest neighbor query, i.e., find the k closest points to the query point. Since k-nearest neighbor queries can be systematically implemented by range queries [4], we only consider range queries to simplify the discussion.

It is shown that for uniform high dimensional vector space, if the database size does not increase exponentially with respect to the dimension, any partitioning and clustering index method will be outperformed by a sequential scan [20].

However, real data is seldom uniform. Its intrinsic clustering usually leads to low intrinsic dimensionality although the feature vectors are in high dimension space, suggesting the application of distance-based indexing, or metric-space indexing. The value of a distance-based approach is that it is unnecessary to find a meaning for the data with respect to the axes of a coordinate system. The only way to save distance calculations is to take use of the triangle inequality [4].

In this paper, we discuss how distance-based image similarity retrieval is supported in MoBIoS [17]. MoBIoS' primary focus is life-science applications. In anticipation of bioinformatics efforts that correlate phenotypes derived from images (e.g. tumors) with gene and protein expression, and also to exercise the

generality of the system we have implemented an image retrieval system.

Many distance-based indexing methods have been developed [4]. We have investigated the performance of three distance-based methods, i.e. the Metric Tree (M-tree) [5, 6], the Radius-Based Tree (RBT), and the Multi-Vantage Point Tree (MVP-tree) [2] on images. Experimental results showed that the MVP-index performed the best.

Given the data, the initial construction of index structure, or bulkload, is critical to its query performance. The bulkload of MVP-index works in a top-down recursive style. In each recursion, there are two main steps: selection of pivots, or vantage-points, and data partition based on the distances to the pivots.

Most of the previous heuristics proposed to these two steps assume that the data is uniformly distributed. Basically, they suggest selecting the *corners* of the data as pivots and partitioning the data such that the index tree is balance. However, real data usually has high level of intrinsic clustering. Brin proposed that the index structure should reflect the intrinsic clustering of the data [3]. We proposed new heuristics for the two steps. The basic idea is to select the *centers* of the intrinsic clusters of the data as pivots, and partition the data according to the intrinsic clustering. Our heuristics usually lead to unbalanced index trees, however, empirical results show that they outperform the heuristics favoring of balance trees. Further, it is shown that the optimized MVP-index scales well for the image data. We have also tested our heuristics for biological data. Similar conclusion can be drawn [14].

Last but not least, we support image retrieval in the MoBIoS SQL (mSQL). Syntactically, mSQL is consistent with the SQL99 [7] object-relational extensions to SQL. Further extensions that support distance-based indices and similarity queries are aligned with standard spatial database extensions to SQL [18]. We first define the image data type, and then bulkload the data. Next, a distance-based index is created on the data, and finally range queries or nearest neighbor queries are executed through the index.

Our main contribution consists of investigating distance-based indexing of images to avoid the curse of dimensionality, comparing several distance-based methods and optimizing the MVP-index to show that the non-canonical unbalance index trees performs well, and the SQL support of image retrieval.

The rest of this paper is organized as follows. In Section 2, we show the low intrinsic dimension of images. Several index structures are outlined in Section 3, and the optimization of MVP-index is discussed in Section 4. We introduce the MoBIoS system and mSQL of image retrieval in Section 5, followed by conclusions and future work in Section 6.

2. Image intrinsic dimension

We start by a brief introduction of the metric distance function of images, which is not the focus of this paper, and interested readers are referred to [11].

Each image is represented by 3 sets of features reflecting the image properties in structure, texture, and color. Each set of features can be considered as a vector. Consequently, an image can be represented by three vectors, with length 3 for structure, 15 for color and 48 for texture. For texture and structure features, the distance functions are both L_2 norm. For color feature, the distance function is L_1 norm [11]. The final distance is a linear combination of the distances of each feature set, with coefficients 1/3 respectively. Since linear combination of metric functions is metric, the final distance also has the metric properties.

Due to the curse of dimensionality, the dimension of the data space dominates the efficiency of the search algorithms [20]. Since real data is seldom uniform, intrinsic dimension, instead of the dimension of the feature vectors, is a better way to quantify the characteristic of the data.

Table 1. Intrinsic dimension of images

Feature vector space	Definition 1	Definition 2
66	5.3	2.2

Two methods of measuring intrinsic dimensionality have been proposed. The intrinsic dimension of a metric space can be defined (**Definition 1**) as $\rho = \mu^2/2\sigma^2$, where μ and σ^2 are the mean and variance of the pair-wise distances among the data points [4]. Another way to estimate the intrinsic dimension is to measure how the number of points contained in a hyper-ball changes with respect to the radius. This measure agrees with the standard measure in d-dimensional Euclidean space with random and uniform data, since the ball with radius $c*r$ has c^d times the volume of the ball with radius r. To calculate a value of d in general, we can measure the number of results of a collection of range queries. By taking the average number for each radius we can perform regression (we use a linear regression on the log of the number and radius) on the measured radius-number pairs to get an estimate of d (**Definition 2**). We prefer definition 2 because it takes into account both the dataset and the queries.

The intrinsic dimensionalities of the images estimated by the two methods are listed in Table 1. We can see that both intrinsic dimensions are below 10, much less than the dimension of the feature vector

space, suggesting the applicability of distance-based methods and minor effect of curse of dimensionality.

3. Distance-based index methods

In this section, we give brief introduction to the three distance-based index methods we have studied. References are given for detail.

Answering similarity queries in partitioning and clustering methods consists of two steps. The first step is offline construction of an index structure, called bulkload. Bulkload actually hierarchically partitions and clusters the data, while each sub-tree can be defined by a predicate. The index keys of data objects are stored in the leaf nodes of the index tree.

The second step is online answering the similarity query. The answering process basically descends the index tree from the root to leaf nodes, in usually more than one path. At each internal node, computation is performed on the query and the predicates of the sub-trees to prune those that are impossible to contain any query results. Children that can not be pruned are further visited in the same way.

Ciaccia et al.'s M-tree effort stands out as the single investigation of a fully general external metric-space index structure [5, 6]. In M-tree, each sub-tree is associated with a center, C, and the maximum distance from C to points in the sub-tree, r, called covering radius. Thus, each sub-tree is defined by a predicate P(C, r), a bounding sphere. If a query point is too far from the center of a sub-tree, by virtue of the triangle inequality, the sub-tree may be pruned.

In previous work [15], we proposed a bi-directional bulkload algorithm for M-tree to save bulkload time and get better index structure. Further the M-tree's determination of the radius is refined by setting it as the maximum distance between the center and each data object in the sub-tree [15]. For clarity, we name the resulting data structure Radius-Based Tree (RBT).

Experimental results on images show that the RBT-index outperforms the M-tree (Section 5).

MVP-trees were also studied in the context of MoBIoS. Since the original MVP-tree resides in main-memory, we implemented a paged MVP-tree, MVP-index, which accounts for elements of an external data structure by organizing the leaves of the tree as disk-pages. The fanout of a node is determined by the disk page size and the occupancy constraints.

In one partition step of MVP-index bulkload, a number of pivots, or vantage points, are selected and data is partitioned into cells, each of which is bounded by a multi-dimensional interval of the form [P_i, $r_{min,\ i}$, $r_{max,\ i}$], indicating that for each pivot P_i, the distances from P_i to points in the cell are within the range [$r_{min,\ i}$, $r_{max,\ i}$]. A range query is transformed into a multi-dimensional interval based on the distances to each pivot and the search radius. The cells whose intervals do not intersect with that of the query are pruned. See Bozkaya and Ozsoyoglu for details [2].

We compared MVP-index and RBT on image workload, and the conclusion is that MVP-index is more suitable in the context of MoBIoS (section 5).

4. Optimizing the MVP-index

In this section, we introduce how we optimize the bulkload of MVP-index to improve its query performance.

From Section 3, we see that there are two critical parts in MVP-index bulkload, i.e. pivot selection and data partition, for which many heuristics have been proposed. For pivot selection, two dominant heuristics are random selection (random-pivot) and corner selection (corner-pivot). Random-pivot is the naïve way to select pivots. It is fast, O(1), and is often combined with some sampling technique. Corner-pivot advocates that the "corners", or the farthest points, of the dataset are the best candidates for vantage-points [21]. The farthest-first-traversal (FFT) [9] algorithm is proposed to find the farthest points. FFT is proposed as an approximation algorithm for the k-center problem, whose objective is to minimize the maximum cluster diameter. FFT gives 2-approximation to the optimum and had been proved to be the best possible [9]. Its time and space complexities are both O(n) [9].

Previously, two partition algorithms have been proposed. A cardinality-balanced algorithm balances the tree by balancing the cardinality of the partitions. A distance-balanced algorithm partitions the range of distances uniformly, usually creating a skewed tree.

For GNAT, an improvement of hyper-plane methods, Brin argued that distance-based indexing methods are effective due to the intrinsic hierarchical clustering of data [3]. Similarly, it is demonstrated that RBTs capture the hierarchical clustering of the data better than M-trees[15],.. We claim that the index tree structure should embody the intrinsic clustering of the data to gain good performance.

For pivot selection, we propose to select the centers of intrinsic clusters of the real data. In MVP-Indexes, we implement an algorithm derived from CLARA (Clustering LARge Applications) [12], which is a simple k-median algorithm based on sampling and iteration. Since the program usually converges rather rapidly, and the time to compute the objective function

is linear to the dataset size, the time complexity of the algorithm is approximately O(n).

```
ClusteringPartition (D: dataset, P: pivots, s: number
of partitions from each pivot) {
// each cluster is associated with a set of pivots,
   initially, D is associated with all the pivots
   while ( exists a cluster C with non-empty pivots set
         P){
      for ( each Pi in P) {
         compute the distances from all points to Pi;
         find s-1 split values by 1-d kmeans clustering;
         compute the sizes of sub-clusters;
         compute the variance of the sizes;
      }
      find the pivot P-mv resulting in least variance;
      split C based on the clustering resulted by P-mv;
      remove P-mv from P;
      copy P as the pivot set for each sub-clusters;
      remove C;
   }
   return all the clusters;
}
```

Figure 1. Clustering Partition

From the discussion of the online query step, we can see that the average query performance is dominated by average number of branches the queries have to descend in the internal nodes. Assuming the queries have the same distribution as the database, it is important to reduce the proportion of points that are close to the cluster boundaries. Based on this observation, we propose the *clustering partition* algorithm. *Clustering partition* tries to find the intrinsic clusters in the data, and put each cluster into a partition. It does not consider the number of objects in each cluster. Thus, without refinement, the cluster partition algorithm may lead to unbalanced trees. We ameliorate this effect by reordering the pivots within a node from best to worst. The quality of a pivot is greedily determined by the balance of the size of the subsets when considering only that pivot. In particular, for each pivot, we partition the dataset by computing the variance of the sizes of the partitions. The pivot resulting in the smallest variance is the best and is considered first within the node. The clustering partition algorithm is detailed in Figure 1.

Each step of the algorithm takes constant or linear time except the k-means algorithm. Each iteration of k-means takes linear time. Since k-means normally converges quickly, and our experiments also show this, we can assume that the number of iterations of k-means is bounded by a constant, thus k-means takes linear time. Again, we consider the number of pivots and the number of partition from each pivot are constant, therefore the time complexity of the clustering partition algorithm is O(n).

Empirical results show that our heuristics outperform other heuristics in most of the cases.

5. Empirical results

In this section, we present the empirical results to compare the query performance of the three distance-based methods, the MVP-index bulkload heuristics, and the scalability of our optimized MVP-index.

We used the M-tree open source release v0.91, which is written in C++ [13]. Our own implementation of the RBT and the MVP-index are written in Java. Therefore, we report on language independent measurements such as the number of distance computations and the number of page accesses rather than execution time. For the RBT and the MVP-index, since the node sizes are equal to disk page size, we measure the number of nodes visited and present that as the number of disk I/O operations.

The image dataset [10] consists of 10221 images. A number of data points are randomly selected as queries. For each index structure, range queries are run with multiple radii. For each radius the average number of distance calculations and nodes visited are computed as the performance measurements.

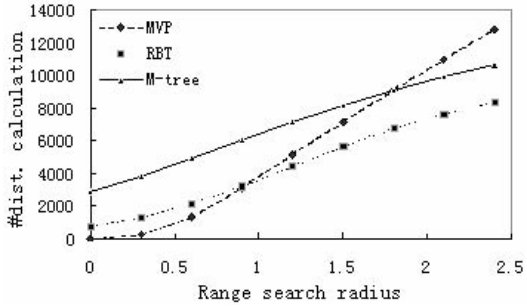

Figure 2. Num. of dist. calc. vs. radius

The query performance of the M-tree, the RBT and the MVP-index are first compared. The average number of distance calculations of all range queries with different radii is shown in Figures 2. From the figures we can see that the RBT always outperforms the M-tree. For small radii, the MVP-index yields the best performance, while for large radii, MVP-index performs the worst. The average number of I/O operations demonstrates similar relationship and is therefore not shown in this paper.

We focus our discussion on radii that are neither too small nor too large. If the radius is too small, only a very small fraction of the database is qualified. This is makes the query like an exact match but not a range query. For a large radius, a major part of the database is returned, which makes the search degrade to a linear scan. For example, Figure 2 shows that the cross over

of RBT and MVP-index is at radius 0.8, where the distance calculation number is about 3000, 29.4% (3000/10221) of the total number of images. Therefore, this radius is too large to be of interest. Consequently, MVP-index is the best.

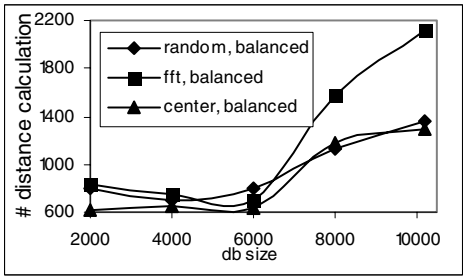

Figure 3. Comp. of VP selection heuristics, radius: 0.1

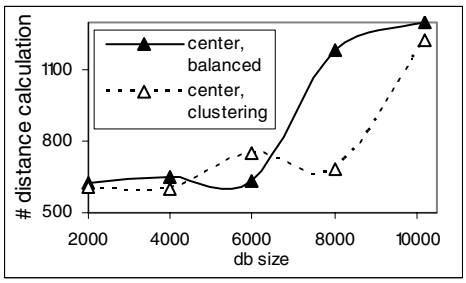

Figure 4. Comp. of partition heuristics, radius: 0.1

Next, we compare the MVP-index bulkload heuristics. Since the figures of number of distance calculations and number I/O operations show are similar, to save space, we only show the number of distance calculation for typical radii.

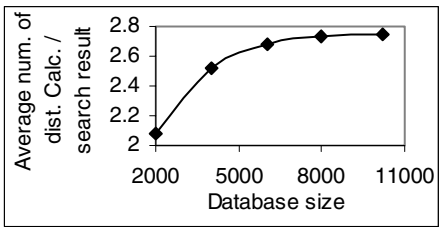

Figure 5. Scalability: average num. of dist. Calc. per query result vs db. size

Figure 3 compares the pivot selection heuristics, and the cardinality-balanced partition is used for all experiments. From the figure, we can see that center-pivot always outperforms corner-pivot by FFT algorithm. Also, center-pivot and random-pivot have similar performance. Note that the query radius, 0.1, is fairly small. An advantage of random selection is that it usually leads to relatively balance clustering of the dataset compared to other heuristics. For very small range query radii, the result of a range query is a very small fraction of the whole dataset, and at each level of the index tree, the search usually descends to only one child. In this case, like the B+-tree for traditional exact match query, balanced trees lead to better performance. Therefore, we expect that random selection leads to better performance for very small radii.

The data comparing the partition algorithms is shown in Figure 4. Since we consider the clustering partition as a derivation of the distance-balanced partition, for simplicity we only show the data of the cardinality-balanced partition and the clustering partition. From the figure, we can clearly see that the clustering partition outperforms balance clustering for most of the cases.

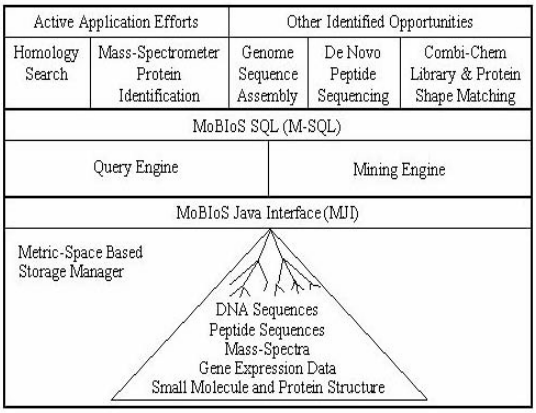

Figure 6. MoBIoS architecture

In the following, we present the scalability of the MVP-index in Figure 5 by showing the average number of distance calculations for each range query result with search radius 0.3 as database size increases. From the figure we can see that as database size increases from 2000 to 10221, nearly 400%, the average number of distance calculations per query result only increases from 2.1 to 2.7, just 29%, indicating good scalability.

6. MoBIoS and mSQL of image retrieval

MoBIoS [17] is a project that aims at inventing a new generation database management system (DBMS) targeting life-science data. MoBIoS is built on top of Mckoi [16], an open source Java RDBMS. MoBIoS contains built-in biological data types entailing the semantics of biological dogma, general-purpose metric-space indices, storage and retrieval of sequences and spectra based on similarity metrics, and a query language (mSQL) (Figure 6).

The storage manager is the foundation of MoBIoS. It uses a B+-tree as the primary index. At this time, metric-space indexes are only used as secondary dense indexes. The storage manager is built by integrating our metric-space index library with Mckoi [16]. The

metric-space index library is written in Java below an interface, called the MoBIoS Java Interface (MJI). With MJI, the library can easily be integrated with other Java DBMSs.

```
CREATE TYPE imagekey_type
AS ( IMAGE_ID INTEGER,
   STRUCTURE          DOUBLE . 3],
   COLOR              DOUBLE . 48],
   TEXTURE            DOUBLE . 15])
INSTANTIABLE
REF ( IMAGE_ID ) ;

CREATE TABLE          image
(  FILENAME   VARCHAR(256),
   KEY          REF(imagekey_type) );
```

(a) Create image index key type, and table of images

```
CREATE TYPE image_distance UNDER METRIC
INSTANCE   METHOD
   getDistance     ( FIRST imagekey_type,
                     SECOND imagekey_type)
      RETURNS DOUBLE;
```

(b) Declare the image distance function

```
CREATE INDEX image_index on image(KEY)
   USING image_distance;
```

(c) Create metric-space index on the image index key

```
SELECT * FROM image
WHERE image_distance.getDistance(KEY,
         imagekey_type("query.jpg" )  <= 5;
SELECT NN("query.jpg", 10) FROM image;
```

(d) Execute an image range query

Figure 7. mSQL statements for image similarity query

mSQL is an extension to the standard SQL language and is under development for MoBIoS. It embodies the semantics of genomics and proteomics and allows for concise expression of Bioinformatics studies. Syntactically mSQL is consistent with the SQL99 object-relational extensions to SQL. Its syntax design also refers to standard spatial database extensions to SQL [7]. mSQL extends standard SQL in three aspects: metric-space indices, built-in biological data types, and biologically related functions and operators.

Since the choice of metric is sometimes a parameter, mSQL departs from the standards on occasion. To specify that an index is a metric-space index, we introduce the reserved word "using" as an argument to create index. The name of a built-in or user-defined metric distance function on which the index is built follows "using" (see Figure 7. (c)).

In mSQL, any data type with a metric distance can be used to create metric-space indices to support similarity queries. The following four steps detail the mSQL statements for image retrieval.

(a) Define a new (nontraditional) object, and use it to define an attribute of a table. (Figure 7 (a))

As in SQL99, mSQL supports user-defined types.

(b) Define the metric distance function. (Figure 7 (b))

There is a built-in type named *METRIC*, which is the super type of any metric distance function. The user can define his or her own distance function as a sub-type of *METRIC*.

(c) Create a metric-space index on the table using the user's metric-distance function (Figure 7. (c)).

The index is created by adding a keyword, *USING,* to the traditional SQL statements to create index. The corresponding distance function is provided in the statement in the form of the *USING* clause.

(d) Execute the range query or nearest neighbor query (Figure 7. (d)).

mSQL also consists of some built-in biological data types such as k-mer, sequence, and mass spectra. The semantics of these data types include subsequence operators and the concept of local alignment. Some functions and operators are defined or extended based on these types. Query optimization rules are given based on these operator extensions. Some biological application procedures are available, such as homology search and conserved primer pair discovery [19].

7. Conclusion and future work

In this paper, we explore the application of MoBIoS to support distance-based image similarity retrieval. Curse of dimensionality has prohibited the application for multi-dimensional partitioning and clustering index methods to high dimensional uniform vector space. However, real data is seldom uniform, and its intrinsic dimension can be far less than the dimension of its feature vector. By two methods, we show that our image data has very low intrinsic dimension. Then, we apply distance-based indexing, or metric-space indexing, methods to images. The merit of distance-based indexing is that the distance function is treated as a black box and the interpretation to coordinate system is not necessary.

Many distance-based index methods have been proposed, but their performance on real image data are not compared enough. We applied three methods, i.e. M-tree, RBT-index and MVP-index to images, and the results show that MVP-index is the one of choice.

More importantly, we further propose new heuristics for pivot selection and data partition of the MVP-index bulkload, aiming at reflecting the intrinsic clustering of real data in the index tree. In the research of database indexing, a prevailing idea is that balance index trees

result in better performance than unbalance trees. However, although our heuristics usually lead to unbalance trees, empirical results show that they outperform heuristics that in favor of balance trees.

As discussed above, because of curse of dimensionality, indexing methods are usually outperformed by sequential scan. In out case, since the data is intrinsically low dimensional, with distance-based indexing and heuristics designed for real clustered data, our MVP-index show good scalability.

Last of all, under the framework of MoBIoS, we have extended the SQL language to support image similarity retrieval syntactically. Just a few lines of mSQL statements are enough to express the whole process from creating an image data type, creating a metric-space index to executing a similarity query.

In addition to image retrieval, we have also applied MoBIoS to other applications successfully, such as scalable protein sequence retrieval, indexing mass-spectra data and finding conserved primer pairs [19]. Similar results are reported.

In ongoing work we are continue to explore index structures and clustering algorithms in hopes of finding a single generally applicable solution. There is considerable flexibility in the choice of the predicate of subsets, pivot selection and data partition heuristics

References

[1] Bentley, J. L., "Multidimensional binary search trees used for associative searching", *Communications of the ACM*, September (1975), 18(9):509-517.

[2] Bozkaya, T. and Ozsoyoglu, M., "Indexing Large Metric Spaces for Similarity Search Queries", *ACM Transactions on Database System*, (1999) 1-34.

[3] Brin, S., "Near Neighbor Search in Large Metric Spaces", In *Proc. 21st. Int. Conf. Very Large Data Bases (VLDB)*, (1995), 574-584.

[4] Chavez, E., Navarro, G., Baeza-Yatesand R., Marroquin, J. L., "Searching in metric spaces", *ACM Computing Surveys*, (2001), 33(3): 273-321.

[5] Ciaccia, P. and Patella, M., "Bulk loading the M-tree", In *Proceedings of the 9th Australasian Database Conference (ADC'98)*, Perth, Australia, February (1998), 15--26.

[6] Ciaccia, P., Patella, M., and Zezula, P., "M-tree: an efficient access method for similarity search in metric spaces", *Proc. 23rd Int. Conf. Very Large Databases (VLDB)*, (1997).

[7] Eisenberg, A., Melton, J, "SQL: 1999, formerly known as SQL 3", *SIGMOD Record*, 1999, 28(1): 131-138.

[8] Guttman, A., "R-trees: A Dynamic Index Structure for Spatial Searching". *Proc. of SIGMOD*, (1984).

[9] Hochbaum, D.S. and Shmoys, D.B., "A best possible heuristic for the k-center problem", *Mathematics of Operational Research,* (1985), Vol. 10(2), pp.180-184.

[10] Image dataset: http://mobios.csres.utexas.edu/~rmao/imagedata

[11] Iqbal, Q. and Aggarwal, J. K., "Image Retrieval via Isotropic and Anisotropic Mappings", *Pattern Recognition Journal,* (2002), Vol. 35, no. 12, 2673-2686.

[12] Kaufman, L. and Peter, J.R., *Finding groups in data: An introduction to cluster analysis*. John Wiley & Sons (1990)

[13] M-tree project home page. http://www-db.deis.unibo.it/Mtree/index.html

[14] Mao R., Xu W., Ramakrishnan S., Nuckolls G., Miranker D. P., "On Optimizing Distance-Based Similarity Search for Biological Databases". In the *Proceedings of the 2005 IEEE Computational Systems Bioinformatics Conference*, Stanford University, California, USA, August 8-11, (2005).

[15] Mao, R., Xu, W., Singh, N. and Miranker, D. P., "An Assessment of a Metric Space Database Index to Support Sequence Homology", In the *proceeding of the 3rd IEEE Symposium on Bioinformatics and Bioengineering*, Washington D.C, March 10-12 (2003)

[16] Mckoi project: http://mckoi.com/database/

[17] Miranker, D. P., Xu, W., and Mao, R.,, "Architecture and Application of MoBIoS, a Metric-Space DBMS to Support Biological Discovery", The *15th International Conference on Scientific and Statistical Database Management*, (2003)

[18] OGIS, *Open GIS consortium: Open GIS simple features specification for SQL (Revision 1.1)* (1999). At URL: *htt://www.opengis.org/techno/specs.htm*

[19] Xu, W., Briggs, W. J., Padolina, J., Liu, W., Linder, C. R. & Miranker, D.P., "Using MoBIoS' Scalable Genome Joins to Find Conserved Primer Pair Candidates Between Two Genomes", *ISMB04*, Glasgow, Scottish (2004).

[20] Weber R., Schek H.-J., and Blott S., "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces", In *Proceedings of the Int. Conf. on Very Large Data Bases*, New York City, New York, August 1998, pages 194-205.

[21] Yianilos, P., "Data structures and algorithms for nearest neighbor search in general metric spaces", In *Proc. 4th ACM-SIAM. Symposium on Discrete Algorithms (SODA'93)*, (1993)311-321.