An Index Structure for Data Mining and Clustering*

Xiong Wang[1], Jason T. L. Wang**[2], King-Ip Lin[3]
Dennis Shasha[4], Bruce A. Shapiro[5], Kaizhong Zhang[6]

[1] Department of Computer and Information Science, New Jersey Institute of Technology, University Heights, Newark, NJ 07102 (xiong@cis.njit.edu)
[2] Department of Computer and Information Science, New Jersey Institute of Technology, University Heights, Newark, NJ 07102 (jason@cis.njit.edu)
[3] Department of Mathematical Sciences, The University of Memphis, Memphis, TN 38152 (linki@msci.memphis.edu)
[4] Courant Institute of Mathematical Sciences, New York University, New York, NY 10012 (shasha@cs.nyu.edu)
[5] Laboratory of Experimental and Computational Biology, National Cancer Institute, Frederick, MD 21702 (bshapiro@ncifcrf.gov)
[6] Department of Computer Science, The University of Western Ontario, London, Ontario, N6A 5B7, Canada (kzhang@csd.uwo.ca)

Abstract. In this paper we present an index structure, called *MetricMap*, that takes a set of objects and a distance metric and then maps those objects to a k-dimensional space in such a way that the distances among objects are approximately preserved. The index structure is a useful tool for clustering and visualization in data intensive applications, because it replaces expensive distance calculations by sum-of-square calculations. This can make clustering in large databases with expensive distance metrics practical. We compare the index structure with another data mining index structure, *FastMap*, recently proposed by Faloutsos and Lin, according to two criteria: relative error and clustering accuracy. For relative error, we show that (i) *FastMap* gives a lower relative error than *MetricMap* for Euclidean distances, (ii) *MetricMap* gives a lower relative error than *FastMap* for non-Euclidean distances (i.e., general distance metrics), and (iii) combining the two reduces the error yet further. A similar result is obtained when comparing the accuracy of clustering. These results hold for different data sizes. The main qualitative conclusion is that these two index structures capture complementary information about distance metrics and therefore can be used together

---

* This work was supported in part by the National Science Foundation under grant numbers IRI-9531548 and IRI-9531554, and by the Natural Sciences and Engineering Research Council of Canada under grant number OGP0046373. A preliminary version of this paper was presented in the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining held in San Diego, California in August 1999.

** Part of the work of this author was done while visiting Courant Institute of Mathematical Sciences, New York University.

to great benefit. The net effect is that multi-day computations can be done in minutes.

Keywords: biomedical applications, data engineering, distance metrics, knowledge discovery, visualization

1 Introduction

In [6], Faloutsos and Lin proposed an index structure, called *FastMap*, for knowledge discovery, visualization and clustering in data intensive applications. The index structure takes a set of objects and a distance metric and maps the objects to points in a k-dimensional target space in such a way that the distances between objects are approximately preserved. One can then perform data mining and clustering operations on the k-dimensional points in the target space. Empirical studies indicated that *FastMap* works well for Euclidean distances [5, 6]. In a later paper, the inventors showed that a modification to *FastMap* could also help detect patterns using the time-warping distance (which is not even a metric, i.e., it doesn't satisfy the triangle inequality) [27].

In this paper we present an index structure, called *MetricMap*, that works in a similar way as *FastMap*. We conduct experiments to compare the performance of *FastMap* and *MetricMap* based on both Euclidean distance and general distance metrics. A general distance metric is a function δ that takes pairs of objects into real numbers, satisfying the following properties: for any objects x, y, z, $\delta(x,x) = 0$ and $\delta(x,y) > 0, x \neq y$ (nonnegative definiteness); $\delta(x,y) = \delta(y,x)$ (symmetry); $\delta(x,y) \leq \delta(x,z) + \delta(z,y)$ (triangle inequality). Euclidean distance satisfies these properties. On the other hand, many general distance metrics of interest are not Euclidean, e.g. string edit distance as used in biology [17], document comparison [23] and the UNIX diff operator. Neither *FastMap* nor *MetricMap* (nor any other index structure that we know of) give guaranteed performance for general distance metrics. For this reason, an experimental analysis is worthwhile.

Section 2 surveys related work. Section 3 discusses the basic properties of *FastMap* and *MetricMap*. Section 4 compares the performance of the two index structures. Section 5 evaluates their performance in data clustering applications. Section 6 presents a further analysis of the index structures, addressing some tradeoff issues. Section 7 concludes the paper.

2 Related Work

Clustering is an important operation in data mining [1, 4, 22, 25]. Clustering algorithms can be broadly classified into two categories: *partitional* and *hierarchical* [11, 12]. A partitional algorithm partitions the objects into a collection of a user-specified number of clusters. A hierarchical algorithm is an iterative process, which either merges small clusters into larger ones, starting with atomic clusters containing single objects, or divides the set of objects into subunits, until some termination condition is met. These algorithms have been studied extensively by researchers in different communities, including statistics [7], pattern recognition [3, 11], machine learning [14], and databases. In particular, data intensive clustering algorithms include CLARANS [15], BIRCH [28], DBSCAN [4], STING [25], WaveCluster [21], CURE [10], CLIQUE [1], etc.

For example, the recently published CURE algorithm [10] utilizes multiple representatives for each cluster. The representatives are generated by selecting well scattered points from the cluster and then shrinking them toward the center of the cluster by a specified fraction. This enables the algorithm to adjust well to a geometry of clusters having non-spherical shapes and wide variances in size. CURE is designed to handle points (vectors) in k-d space only, not for general distance metric spaces, and therefore is considered as a *vector-based* clustering algorithm. It employs a combination of random sampling and partitioning to handle large datasets. The algorithm is a typical hierarchical one, which starts with each input point as a separate cluster, and at each successive step merges the closest pair of clusters.

By contrast, the popular K-means and K-medoid methods are partitional algorithms. The methods determine K cluster representatives and assign each object to the cluster with its representative closest to the object such that the sum of the distances squared between the objects and their representatives is minimized. The methods work for both Euclidean distance and general distance metrics, and therefore are considered as *distance-based* clustering algorithms. [12, 15, 28] presented extensions of the partitional methods for large and spatial databases, some of which are vector-based and some are distance-based.

In contrast to the above work, $FastMap$ and $MetricMap$ employ the approach of mapping objects to points in a k-dimensional (k-d) target space R^k and then cluster the points in R^k. The main benefit provided by this approach is that it saves time in distance computation. Calculating the actual distances among the objects is much more expensive than measuring the dissimilarities among the points in R^k.[7] This is particularly true for new, emerging applications in multimedia and scientific computing. As an example, comparing two RNA secondary structures may require a dynamic programming algorithm [19] or a genetic algorithm [18] that runs in seconds or minutes on current workstation. The presented mapping approach is useful not only for data mining and clustering, but also for visualization and retrieval in large datasets [5, 6].

3 $FastMap$ and $MetricMap$: a brief comparison

Consider a set of objects $\mathcal{D} = \{O_0, O_1, \ldots, O_{N-1}\}$ and a distance function d where for any two objects O_i, $O_j \in \mathcal{D}$, $d(O_i, O_j)$ (or $d_{i,j}$ for short) represents the distance between O_i and O_j. The function d can be Euclidean or a general distance metric. Both $FastMap$ and $MetricMap$ take the set of objects, some inter-object distances and embed the objects in a k-d space R^k (k is user-defined),

[7] We use "dissimilarity", rather than "distance", in the paper since there may be a negative dissimilarity value between two points in the target space.

such that the distances among the objects are approximately preserved. The k-d point P_i corresponding to the object O_i is called the *image* of O_i. The k-d space containing the *images* is called *target space*.

The differences between the two index structures lie in the algorithm they use for embedding and the target space they choose. *FastMap* embeds the objects in a Euclidean space, whereas *MetricMap* embeds them in a pseudo-Euclidean space [9, 13]. Below we describe the two index structures and their properties; detailed proofs can be found in [6, 26].

3.1 The *FastMap* Algorithm

The basic idea of this algorithm is to project objects on a line (O_a, O_b) in an n-dimensional (n-d) space R^n for some unknown n, $n \geq k$. The line is formed by two *pivot objects* O_a, O_b, chosen as follows. First, arbitrarily choose one object and let it be the second pivot object, O_b. Let O_a be the object that is farthest apart from O_b. Then update O_b to be the object that is farthest apart from O_a. The two resulting objects O_a, O_b are pivots.

Consider an object O_i and the triangle formed by O_i, O_a and O_b (Figure 1). From the cosine law, one can get

$$d_{b,i}^2 = d_{a,i}^2 + d_{a,b}^2 - 2x_i d_{a,b} \tag{1}$$

Thus, the first coordinate x_i of object O_i with respect to the line (O_a, O_b) is

$$x_i = \frac{d_{a,i}^2 + d_{a,b}^2 - d_{b,i}^2}{2d_{a,b}} \tag{2}$$

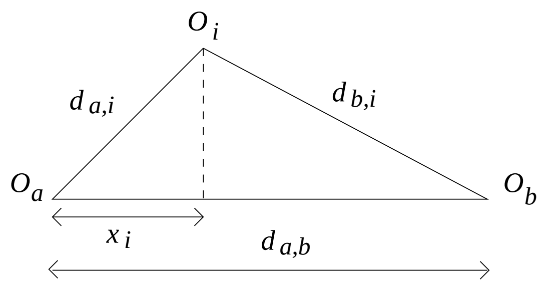

Fig. 1. Illustration of the projection method used in *FastMap*.

Now we can extend the above projection method to embed objects in the target space R^k as follows. Pretending that the given objects are indeed points in R^n, we consider an $(n-1)$-d hyper-plane $\mathcal{H}$ that is perpendicular to the line (O_a, O_b), where O_a and O_b are two pivot objects. We then project all the objects onto this hyper-plane. Let O_i, O_j be two objects and let $O_i^{'}$, $O_j^{'}$ be their projections on the hyper-plane $\mathcal{H}$. It can be shown that the dissimilarity d' between $O_i^{'}$, $O_j^{'}$ is

$$(d'(O_i^{'}, O_j^{'}))^2 = (d(O_i, O_j))^2 - (x_i - x_j)^2, \qquad i, j = 0, \ldots, N-1 \qquad (3)$$

Being able to compute d' allows one to project on a second line, lying on the hyper-plane $\mathcal{H}$, and therefore orthogonal to the first line (O_a, O_b). We repeat the steps recursively, k times, thus mapping all objects to points in R^k.

The discussion thus far assumes that the objects are indeed points in R^n. If the assumption doesn't hold, $(d(O_i, O_j))^2 - (x_i - x_j)^2$ may become negative. For this case, Equation (3) is modified as follows:

$$d'(O_i^{'}, O_j^{'}) = -\sqrt{(x_i - x_j)^2 - (d(O_i, O_j))^2} \qquad (4)$$

Let O_i, O_j be two objects in $\mathcal{D}$ and let $P_i = (x_i^1, \ldots, x_i^k)$, $P_j = (x_j^1, \ldots, x_j^k)$ be their images in the target space R^k. The dissimilarity between P_i and P_j, denoted $d_f(P_i, P_j)$, is calculated as

$$d_f(P_i, P_j) = \sqrt{\sum_{l=1}^{k} (x_i^l - x_j^l)^2} \qquad (5)$$

Note that if the objects are indeed points in R^n, $n \geq k$, and the distance function d is Euclidean, then from Equation (3), $FastMap$ guarantees a lower bound on inter-object distances. That is,

Proposition 3.1 $\quad d_f(P_i, P_j) \leq d(O_i, O_j)$.

Let $Cost_{fastmap}$ denote the total number of distance calculations required by $FastMap$. From Equations (2) and (3) and the way the pivot objects are chosen, we have

$$Cost_{fastmap} = 3Nk \qquad (6)$$

where N is the size of the dataset and k is the dimensionality of the target space.

3.2 The *MetricMap* Algorithm

The algorithm works by first choosing a small sample $\mathcal{A}$ of $2k$ objects from the dataset. In choosing the sample, one can either pick it up randomly, or use the $2k$ pivot objects found by $FastMap$. The algorithm calculates the pairwise distances among the sampling objects and uses these distances to establish the target space R^k. The algorithm then maps all objects in the dataset to points in R^k.

Specifically, assume, without loss of generality, that $\mathcal{A} = \{O_0, \ldots, O_{2k-1}\}$. We define a mapping α as follows: $\alpha : \mathcal{A} \rightarrow R^{2k-1}$ such that $\alpha(O_0) = a_0 = (0, \ldots, 0)$, $\alpha(O_i) = a_i = (0, \ldots, 1_{(i)}, \ldots, 0)$, $1 \leq i \leq 2k-1$ (see Figure 2(a)). Intuitively we map O_0 to the origin and map the other sampling objects to vectors (points)

$\{a_i\}_{1\le i\le 2k-1}$ in R^{2k-1} so that each of the objects corresponds to a base vector in R^{2k-1}.

Let

$$M(\psi_{<a>}) = (m_{i,j})_{1\le i,j\le 2k-1} \tag{7}$$

where

$$m_{i,j} = \frac{d_{i,0}^2 + d_{j,0}^2 - d_{i,j}^2}{2}, \qquad 1 \le i,j \le 2k-1 \tag{8}$$

Define the function ψ as follows: $\psi : R^{2k-1} \times R^{2k-1} \to R$ such that

$$\psi(x,y) = x^T M(\psi_{<a>}) y \tag{9}$$

where x^T is the transpose of vector x. Notice that $\psi(a_i, a_j) = m_{i,j}$, $1 \le i,j \le 2k-1$. The function ψ is called a *symmetric bilinear form* of R^{2k-1} [9]. $M(\psi_{<a>})$ is the matrix of ψ with respect to the basis $\{a_i\}_{1\le i\le 2k-1}$. The vector space R^{2k-1} equipped with the symmetric bilinear form ψ is called a pseudo-Euclidean space. For any two points (vectors) $x, y \in R^{2k-1}$, $\psi(x,y)$ is called the *inner product* of x and y. The *squared distance* between x and y, denoted $\|x-y\|^2$, is defined as

$$\|x-y\|^2 = \psi(x-y, x-y) \tag{10}$$

This squared distance is used to measure the dissimilarity of two points in the pseudo-Euclidean space.

Since the matrix $M(\psi_{<a>})$ is real symmetric, there is an orthogonal matrix $Q = (q_{i,j})_{1\le i,j\le 2k-1}$ and a diagonal matrix $D = diag(\lambda_i)_{1\le i\le 2k-1}$ such that

$$Q^T M(\psi_{<a>}) Q = D \tag{11}$$

where Q^T is the transpose of Q, $\lambda_i s$ are eigenvalues of $M(\psi_{<a>})$ arranged in some order, and columns of Q are the corresponding eigenvectors [8]. Note that if the matrix $M(\psi_{<a>})$ has negative eigenvalues, the squared distance between two points in the pseudo-Euclidean space may be negative. That's why we never say the "distance" between points in a pseudo-Euclidean space.

Now we find a ψ-*orthogonal* basis of R^{2k-1}, $\{e_i\}_{1\le i\le 2k-1}$, where

$$(e_1, \ldots, e_{2k-1}) = (a_1, \ldots, a_{2k-1})Q \tag{12}$$

or equivalently

$$(a_1, \ldots, a_{2k-1}) = (e_1, \ldots, e_{2k-1})Q^T \tag{13}$$

Each vector a_i, $1 \le i \le 2k-1$, can be represented as a vector in the space spanned by $\{e_i\}_{1\le i\le 2k-1}$ and the coordinate of a_j with respect to $\{e_i\}_{1\le i\le 2k-1}$ is the j^{th} row of Q (see Figure 2(b)). Each e_i corresponds to an eigenvector.

Suppose the eigenvalues are sorted in descending order by their absolute values, followed by the zero eigenvalues. The $MetricMap$ algorithm reduces the dimensionality of R^{2k-1} to obtain the subspace R^k by removing the $k-1$ dimensions along which the eigenvalues $\lambda_i s$ of $M(\psi_{<a>})$ are zero or their absolute values are smallest (see Figure 2(c)). Notice that among the remaining

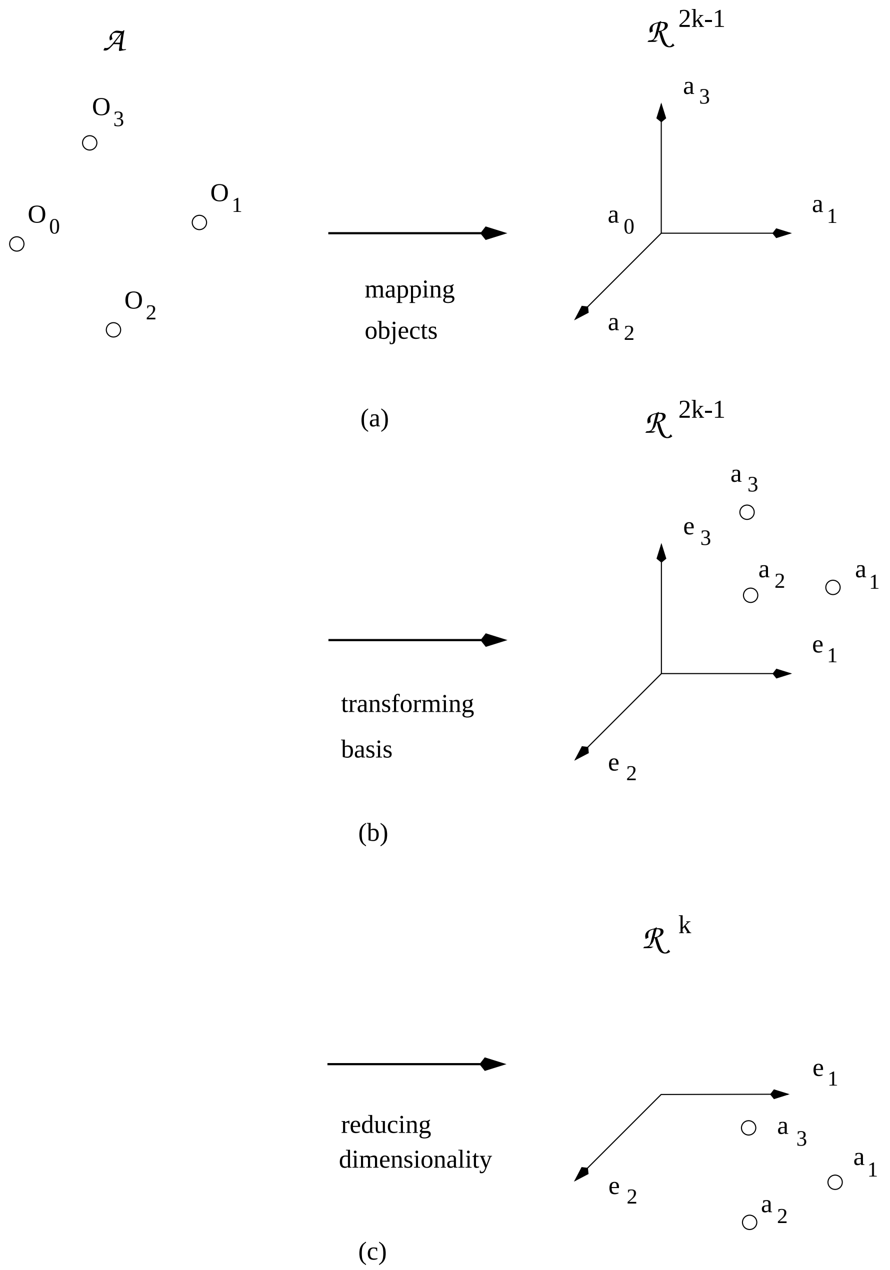

Fig. 2. Illustration of the *MetricMap* algorithm ($k = 2$).

k-dimensions, some may have negative eigenvalues. The algorithm then chooses $k+1$ objects, called the *reference objects*, that span R^k.

Once the target space R^k is established, the algorithm maps each object O_* in the dataset to a point (vector) P_* in the target space by comparing the object with the reference objects. The coordinate of P_* is calculated through matrix multiplication. Here is how.

Assume, without loss of generality, that the reference objects are O_0, O_1, ... , O_k. Let

$$b = (n_{*,j})_{1 \le j \le k} \tag{14}$$

where

$$n_{*,j} = \frac{d_{*,0}^2 + d_{j,0}^2 - d_{*,j}^2}{2}, \qquad 1 \le j \le k \tag{15}$$

Define

$$sign(\lambda_i) = \begin{cases} 1 & \text{if } \lambda_i > 0 \\ 0 & \text{if } \lambda_i = 0 \\ -1 & \text{if } \lambda_i < 0 \end{cases} \tag{16}$$

That is, $sign(\lambda_i)$ is the sign of the ith eigenvalue λ_i. Let $J = diag(sign(\lambda_i))_{1 \le i \le 2k-1}$ and $C = diag(c_i)_{1 \le i \le 2k-1}$ where

$$c_i = \begin{cases} |\lambda_i| & \text{if } \lambda_i \ne 0 \\ 1 & \text{otherwise} \end{cases} \tag{17}$$

Let $J_{[k]}$ be the k^{th} leading principal submatrix of the matrix J, i.e. $J_{[k]} = diag(sign(\lambda_i))_{1 \le i \le k}$. Let $C_{[k]}$ be the k^{th} leading principal submatrix of the matrix C, i.e. $C_{[k]} = diag(|\lambda_i|)_{1 \le i \le k}$. Let $Q_{[kk]}$ be the k^{th} leading principal submatrix of the orthogonal matrix Q, i.e. $Q_{[kk]} = (q_{i,j})_{1 \le i,j \le k}$. The coordinate of P_* in R^k, denoted $Coor(P_*)$, can be approximated as follows:

$$Coor(P_*) \approx J_{[k]} C_{[k]}^{-1/2} Q_{[kk]}^{-1} b \tag{18}$$

Let O_i, O_j be two objects in $\mathcal{D}$ and let $P_i = (x_i^1, \ldots, x_i^k)$, $P_j = (x_j^1, \ldots, x_j^k)$ be their images in R^k. Let

$$\Delta(P_i, P_j) = \sum_{l=1}^{k} sign(\lambda_l)(x_i^l - x_j^l)^2 \tag{19}$$

The dissimilarity between P_i and P_j, denoted $d_m(P_i, P_j)$, is approximated by

$$d_m(P_i, P_j) \approx \begin{cases} \sqrt{\Delta(P_i, P_j)} & \text{if } \Delta(P_i, P_j) \ge 0 \\ -\sqrt{-\Delta(P_i, P_j)} & \text{otherwise} \end{cases} \tag{20}$$

Note that if the objects are points in R^n, $n \ge k$, and the distance function d is Euclidean, then as in *FastMap*, *MetricMap* guarantees a lower bound on inter-object distances. That is,

Proposition 3.2 $\qquad d_m(P_i, P_j) \le d(O_i, O_j)$.

To see this, note that in the Euclidean spaces, the bilinear form ψ is positive definite, because for any non-zero vector x, $x^T M(\psi_{<a>})x$ is positive [16]. This implies that all the non-zero eigenvalues are positive. When projecting the points from R^n onto R^k, the images have fewer coordinates. From Equations (19) and (20), we conclude that the dissimilarity between two images is less than or equal to the distance between the corresponding objects.

Let $Cost_{metricmap}$ denote the total number of distance calculations required by $MetricMap$. From Equations (7), (8) and (11), we see that to calculate the eigenvalues of $M(\psi_{<a>})$, one needs to calculate the pairwise distances $d_{i,j}$, $0 \leq i, j \leq 2k-1$. This requires $(2k)^2 = 4k^2$ distance calculations. From Equations (14), (15) and (18), we see that to embed each object O_* in R^k, one needs to calculate the distances from O_* to the $k+1$ reference objects. Notice that if O_* is a sampling object, its distances to the reference objects need not be recalculated, since they are part of the distances $d_{i,j}$, $0 \leq i, j \leq 2k-1$ that are already computed. Totally there are N objects in the dataset, and therefore

$$Cost_{metricmap} = 4k^2 + (N - 2k)(k + 1) \tag{21}$$

Comparing Equations (6) and (21), since $N \geq k$, $Cost_{metricmap} \leq Cost_{fastmap}$.

4 Precision of Embedding

We conducted a series of experiments to evaluate the precision of embedding by calculating the errors induced by the index structures. The index structures were implemented in C and C++ under the UNIX operating system run on a SPARC 20. Four sets of distances were generated: synthetic Euclidean, synthetic non-Euclidean, protein and RNA. The last three were general distance metrics, so satisfied the triangle inequality, but were not Euclidean.

4.1 Data

In creating synthetic Euclidean distances, we generated N n-dimensional vectors. Each vector was generated by choosing n real numbers randomly and uniformly from the interval $[LowBound..HighBound]$. We then calculated the pairwise distances among the vectors. In creating synthetic non-Euclidean distances, we generated the pairwise distances among N objects randomly and uniformly in the interval $[MinDistance..MaxDistance]$, keeping only those objects that satisfied the triangle inequality as in [20]. Table 1 summarizes the parameters and base values used in the experiments.

In generating protein distances, we selected a set of 230 kinase sequences obtained from the protein database in the Cold Spring Harbor Laboratory. We used the string edit distance to measure the dissimilarity of two proteins [17]. The inter-protein distances were in the interval (1..2573).

Table 1: Parameters and base values used in the experiments for evaluating the precision of embedding.

Parameter	Value	Description
k	15	Dimensionality of the target space
N	3,000	Number of objects in the dataset
n	20	Dimensionality of synthetic vectors in Euclidean space
$LowBound$	0	Smallest possible value for each coordinate of the synthetic vectors
$HighBound$	100	Largest possible value for each coordinate of the synthetic vectors
$MinDistance$	1	Minimum distance between objects for the synthetic non-Euclidean data
$MaxDistance$	100	Maximum distance between objects for the synthetic non-Euclidean data

In generating RNA distances, we used 200 RNA secondary structures obtained from the virus database in the National Cancer Institute. The RNA secondary structures were created by first choosing two phylogenetically related mRNA sequences, rhino 14 and cox5, from GenBank [2] pertaining to the human rhinovirus and coxsackievirus. The 5' non-coding region of each sequence was folded and 100 secondary structures of that sequence were collected. The structures were then transformed into trees and their pairwise distances were calculated as described in [19, 24]. The trees had between 70 and 180 nodes. The distances for rhino 14's trees and cox5's trees were in the interval (1..75) and (1..60), respectively. The distances between rhino 14's trees and cox5's trees were in the interval (43..94). The secondary structures (trees) for each sequence roughly formed a cluster.

4.2 Experimental Results

Let O_i, O_j be two objects in $\mathcal{D}$ and let P_i, P_j be their images in R^k. The dissimilarity between P_i, P_j embedded by $FastMap$, denoted $d_f(P_i, P_j)$, was as in Equation (5). The dissimilarity between P_i, P_j embedded by $MetricMap$, denoted $d_m(P_i, P_j)$, was as in Equation (20). To understand whether the index structures might complement each other, we considered three combinations of the index structures: $AvgMap$, $MinMap$ and $MaxMap$, with the dissimilarities d_a, d_n, d_x defined as follows:

$$d_a(P_i, P_j) = \frac{d_f(P_i, P_j) + d_m(P_i, P_j)}{2} \tag{22}$$

$$d_n(P_i, P_j) = \min\{d_f(P_i, P_j), d_m(P_i, P_j)\} \tag{23}$$

$$d_x(P_i, P_j) = \max\{d_f(P_i, P_j), d_m(P_i, P_j)\} \tag{24}$$

We collectively refer to all these index structures as *mappers*. In building the mappers, we used random sampling objects for $MetricMap$ to establish the target space (cf. Section 3.2). Note here that the mappers have the same cost $O(Nk)$ asymptotically, cf. Equations (6) and (21).

The measure used for evaluating the precision of embedding was the *average relative error* (Err_r), defined as

$$Err_r = \frac{\sum_{O_i,O_j \in \mathcal{D}} |d(O_i,O_j) - |d_s(P_i,P_j)||}{\sum_{O_i,O_j \in \mathcal{D}} d(O_i,O_j)} \times 100\% \tag{25}$$

where $s = f, m, a, n, x$, respectively. One would like this percentage to be as low as possible. The lower Err_r is, the better performance the corresponding mapper has.

Figure 3 graphs Err_r as a function of the dimensionality of the target space, k, for the synthetic Euclidean data. The parameters have the values shown in Table 1. We see that for all the mappers, Err_r drops as k increases. Err_r approaches 0 when $k = 19$. $FastMap$ performs better than $MetricMap$, but $MaxMap$ dominates in all situations. From Proposition 3.1 and Proposition 3.2, both $FastMap$ and $MetricMap$ underestimate inter-object distances, so $MaxMap$ gives the lowest average relative error among all the mappers.

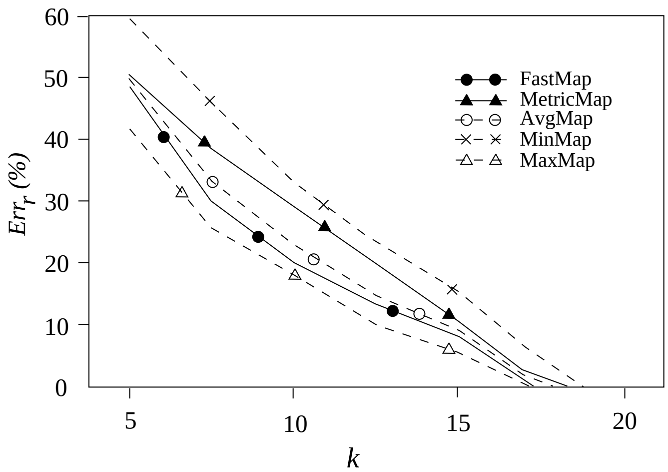

Fig. 3. Average relative errors of the mappers as a function of the dimensionality of the target space for synthetic Euclidean data.

We next examined the scalability of the results. Figure 4 compares $FastMap$, $MetricMap$ and $MaxMap$ for varying N, Figure 5 compares the three mappers for varying n, and Figure 6 plots Err_r as a function of $(HighBound/LowBound)$ for the three mappers. In each figure, only one parameter is tuned and the other parameters have the values shown in Table 1. The $LowBound$ in Figure 6 is fixed at 1. It can be seen that Err_r depends on the dimensionality of vectors n, but is independent of the dataset size N and coordinate ranges of the vectors. $MaxMap$ consistently beats the other two mappers in all these figures.

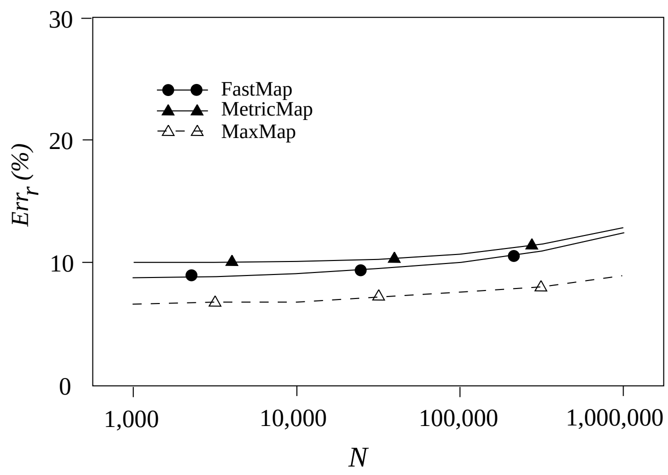

Fig. 4. Effect of dataset size for synthetic Euclidean data.

We then compared the relative performance of the mappers using the synthetic non-Euclidean data. Figure 7 graphs Err_r as a function of the dimensionality of the target space k. The parameters have the values shown in Table 1. The figure shows that $MetricMap$ outperforms $FastMap$ while $AvgMap$ is superior to both of them. As k increases, the performance of $MetricMap$ improves while the performance of $FastMap$ degrades. The larger the k, the more negative dissimilarity values $FastMap$ produces, cf. Equation (4). As a consequence, the more biased projections it creates. Note that $MetricMap$ also produces negative dissimilarity values during the projection. It has a better performance probably because the images' coordinates are calculated by matrix multiplication through a single projection, rather than through a series of projections as done in $FastMap$, and hence the effect incurred by these negative dissimilarity values is reduced.

The next two figures show the scalability of the results. Figure 8 compares the relative performance of $FastMap$, $MetricMap$ and $AvgMap$ for varying N and Figure 9 plots Err_r as a function of $\ln(MaxDistance/MinDistance)$ for the three mappers. The k value in both figures is fixed at 1000 and the $MinDistance$ in Figure 9 is fixed at 10. The other parameters have the values shown in Table 1. It can be seen that Err_r depends on the dataset size N, but is independent of the distance ranges. Clearly, $AvgMap$ is the best for all the non-Euclidean data. Both $FastMap$ and $MetricMap$ may overestimate or underestimate some inter-

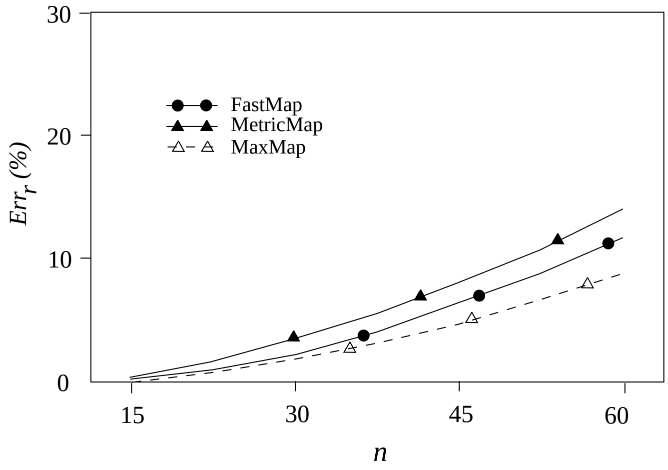

Fig. 5. Average relative errors of the mappers as a function of the dimensionality of vectors for synthetic Euclidean data.

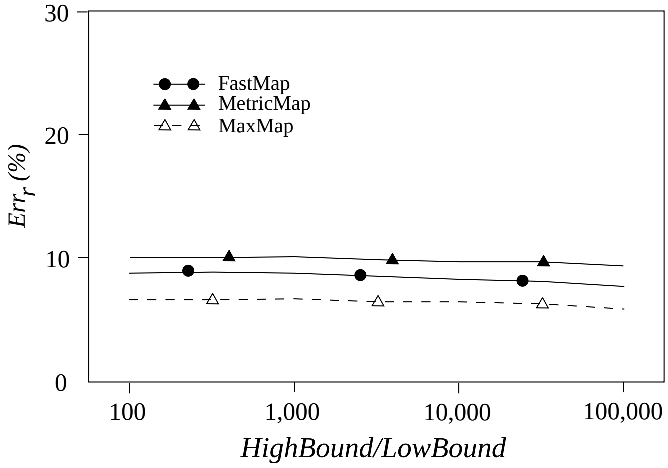

Fig. 6. Effect of coordinate ranges for synthetic Euclidean data.

object distances. The fact that $AvgMap$ outperforms either one individually is a good indication of the complementarity of the two index structures.

The trends observed from protein and RNA data are similar to those from the synthetic data. We omit the results for protein and only present those for RNA secondary structures (Figure 10). In sum, $MaxMap$ is best for Euclidean data; its performance depends on the dimensionality of vectors n, but is independent of the size of datasets N. $AvgMap$ is best for non-Euclidean data; its perfor-

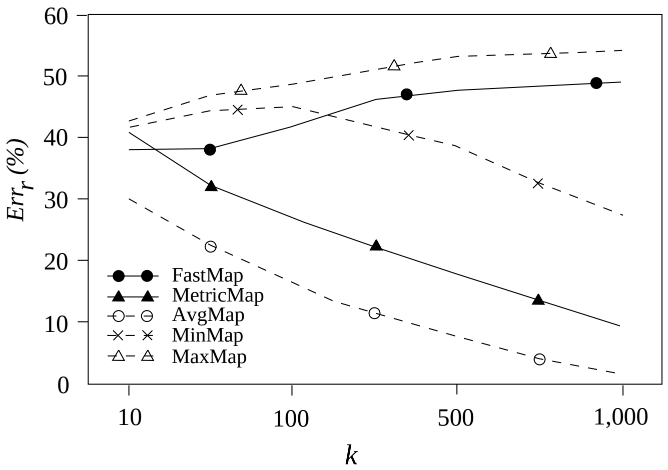

Fig. 7. Average relative errors of the mappers as a function of the dimensionality of the target space for synthetic non-Euclidean data.

mance depends on the dataset size. Both mappers' performance improves as the dimensionality of the target space k increases. For Euclidean data, $MaxMap$'s Err_r drops to 0 as k approaches n. For non-Euclidean data, $AvgMap$'s Err_r approaches 0 when $k = N/2$, i.e. when all the $2k = N$ data objects are used in the sample to establish the target space.

The last set of experiments examined the feasibility of retrieval with $MaxMap$ and $AvgMap$. Let d_s, $s = a, x$, represent the dissimilarity measures for the two mappers, cf. Equations (22) and (24). We randomly picked an object O_c and considered the sphere G_1 with O_c as the centroid and a properly chosen ϵ as the radius, i.e. G_1 contained all the objects O where $d(O, O_c) \le \epsilon$. Let P_c be the image of O_c. G_2 represented the sphere in the target space that contained all the images P where $|d_s(P, P_c)| \le \epsilon$. Let O_i be an object and let P_i be its image. We say O_i is a *false positive* if $P_i \in G_2$ whereas $O_i \notin G_1$. O_i is a *false negative* if $P_i \notin G_2$ but $O_i \in G_1$. The performance measure used was the *accuracy* (*Accu*), defined as

$$Accu = \frac{|G_1| + |G_2| - (N_p + N_n)}{|G_1| + |G_2|} \times 100\% \qquad (26)$$

where $|G_i|$, $i = 1, 2$, was the size of G_i, N_p was the number of false positives, and N_n was the number of false negatives. One would like this percentage to be as high as possible. The higher *Accu* is, the fewer false positives and negatives are, and therefore the better performance a mapper has.

Figure 11 illustrates $MaxMap$'s performance for the synthetic Euclidean data and Figure 12 illustrates $AvgMap$'s performance for the synthetic non-Euclidean data. The four curves represent four different dataset sizes (N = 1,000, 10,000, 100,000, 1,000,000, respectively, in Figure 11, and N = 2,000,

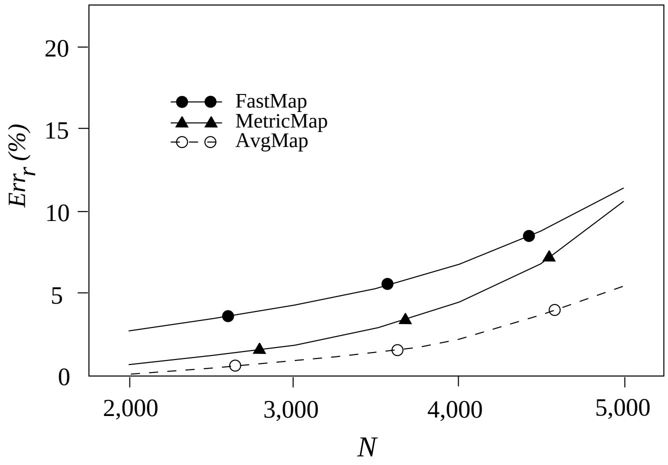

Fig. 8. Effect of dataset size for synthetic non-Euclidean data.

3,000, 4,000, 5,000, respectively, in Figure 12). The four points on each curve correspond to four different k values. The *Accu* plotted in the figures is the average value over all the N spheres where each sphere uses a different object as the centroid. The radius of a sphere is fixed at 50, i.e. $\epsilon = 50$. The X-axis shows the CPU time spent in embedding the objects. From the figures we see that as the dimensionality of the target space, k, increases, both the time and accuracy increase. For the Euclidean data with 20-dimensional vectors, *Accu* approaches 100% when $k = 18$. For the non-Euclidean data, *Accu* approaches 100% when $k = 1,000$. These results indicate that with the two best mappers, one can conduct the range search [5, 6] on the k-dimensional points by embedding the query object in the target space and then considering the sphere with the query object as the centroid in the target space. Embedding the data objects can be performed in the off-line stage, thus reducing the search time significantly.

5 Clustering

In this section we evaluate the accuracy of clustering in the presence of imprecise embedding. The purpose is twofold. First, this study shows the feasibility of clustering without performing expensive distance calculations. Second, through the study, one can understand how imprecision in the embedding may affect the accuracy of clustering.

5.1 Data

The data used in the experiments included the RNA secondary structures described in Section 4.1, because they roughly formed 2 clusters, each correspond-

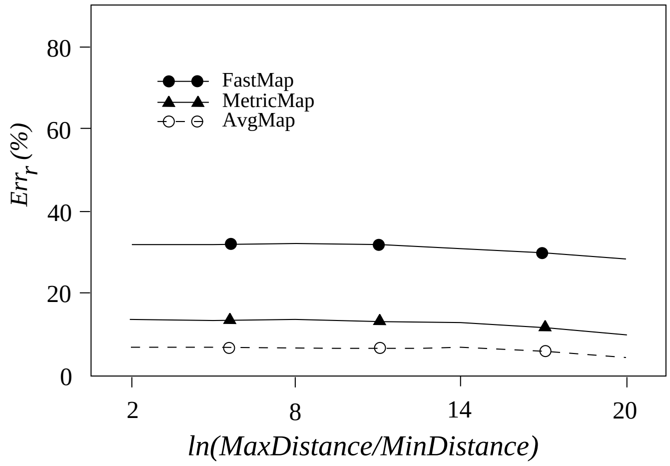

Fig. 9. Effect of distance ranges for synthetic non-Euclidean data.

ing to an mRNA sequence. RNA distance is non-Euclidean. In addition, we generated Euclidean clusters as follows: we built $p = q^2$ clusters as in [28]. Specifically, we generated q groups of n-dimensional vectors from an n-dimensional hypercube. The vectors were generated as described in Section 4.1. Each group had C vectors.

Initially the groups (clusters) might overlap. We considered all the q groups as sitting on the same line and moved them apart along the line by adding a constant $(i \times c)$, $1 \leq i \leq q$, to the first coordinate of all the vectors in the ith group; c was a tunable parameter. We used CURE [10] to adjust the clusters so that they were not too far apart. Specifically, c was chosen to be the minimum value, by which CURE can just separate the q clusters. In our case, $c = 1.15$. Once the first q clusters were generated, we moved to the second line, which was parallel to the first line, and generated another q clusters along the second line. This step was repeated until all the q lines were generated, each line comprising q clusters. Again we used CURE to adjust the distance between the lines so that they were not too far apart. Table 2 summarizes the parameters and base values used in the experiments.

Table 2: Parameters and base values used in the experiments for evaluating the accuracy of clustering Euclidean vectors.

Parameter	Value	Description
k	10	Dimensionality of the target space
p	4	Number of clusters
n	20	Dimensionality of synthetic vectors
C	100	Number of vectors in a cluster

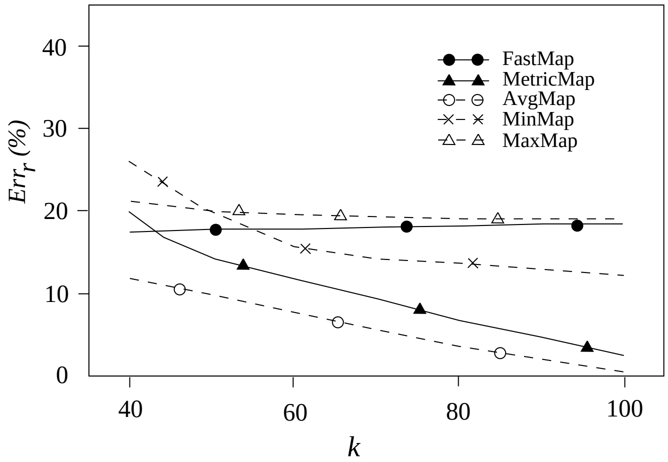

Fig. 10. Average relative errors of the mappers as a function of the dimensionality of the target space for RNA secondary structures.

5.2 Experimental Results

The clustering algorithm used in our experiments was the well known average-group method [12], which works as follows. Initially, every object is a cluster. The algorithm merges two nearest clusters to form a new cluster, until there are only K clusters left where K is p for the Euclidean clusters and 2 for the RNA data. The distance between two clusters C_1 and C_2 is given as

$$\frac{1}{|C_1||C_2|} \sum_{O_p \in C_1, O_q \in C_2} |d(O_p, O_q)| \tag{27}$$

where $|C_i|$, $i = 1, 2$, is the size of cluster C_i. The algorithm requires $O(N^2)$ distance calculations where N is the total number of objects in the dataset.

An object O is said to be *mis-clustered* if O is in a cluster C created by the average-group method, but its image is not in C's corresponding cluster, which is also created by the average-group method, in the target space. The performance measure we used was the *mis-clustering rate* (Err_c), defined as

$$Err_c = \frac{N_c}{N} \times 100\% \tag{28}$$

where N_c was the number of mis-clustered objects.

Figure 13 graphs Err_c as a function the dimensionality of the target space, k, for the Euclidean clusters and Figure 14 shows the results for the RNA data. The parameters have the values shown in Table 2. For the Euclidean data, the average-group method successfully found the 4 clusters in the dataset. For the RNA data, the average-group method missed 5 objects in the dataset (i.e., the

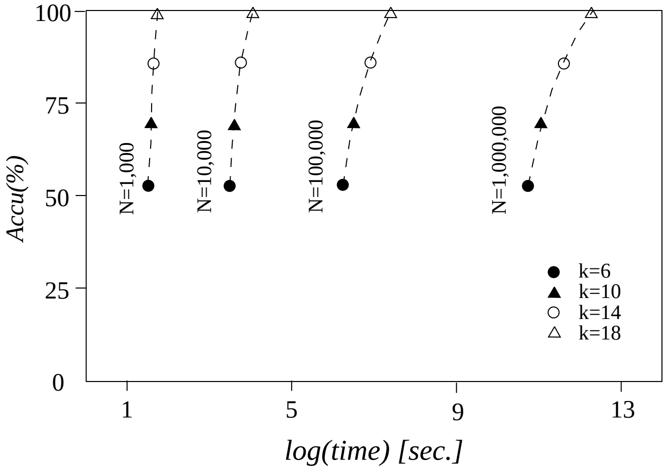

Fig. 11. Accuracy of $MaxMap$ for synthetic Euclidean data.

5 RNA secondary structures were not detected to belong to their corresponding sequence's cluster). The images of these 5 objects were also missed in the target space; they were excluded when calculating Err_c.

As in Section 4.2, the clustering performance improves as the dimensionality of the target space increases, because the embedding becomes more precise. Figure 13 shows that the Err_cs of all the mappers approach 0 when $k = 9$. Figure 14 shows that $MetricMap$ outperforms $FastMap$; its Err_c approaches 0 when $k = 80$. Overall, $MaxMap$ is best for the Euclidean data and $AvgMap$ is best for the non-Euclidean RNA data. The results indicate that with the two best mappers, one can perform clustering on the k-dimensional points. Embedding the data objects can be performed in the off-line stage, thus reducing the clustering time significantly.

It is worth pointing out that one may achieve an accurate clustering even with an imprecise embedding. For example, in Figure 13, the clustering accuracy is over 90% when $k = 2$, though the relative errors for the $k = 2$ case are over 50% (cf. Figure 3). This happens because after the embedding is performed, those objects that are close to each other in the original space remain close in the target space, though the distances are underestimated significantly.

We next examined the scalability of the results using Euclidean clusters. Figure 15 compares $FastMap$, $MetricMap$ and $MaxMap$ for varying numbers of clusters, Figure 16 compares them for varying sizes of clusters, and Figure 17 compares the mappers for varying dimensionalities of the vectors in each cluster. With higher dimensional vectors (e.g. 60-dimensions) and more clusters, the average-group method missed several objects in the dataset. However, $MaxMap$ consistently gives the lowest mis-clustering rate in all the figures.

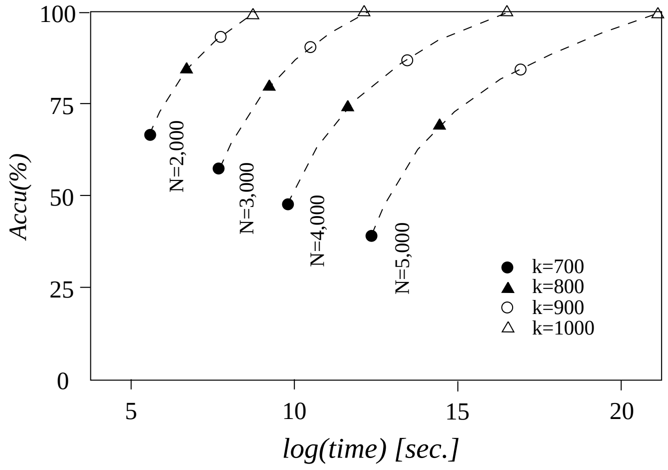

Fig. 12. Accuracy of $AvgMap$ for synthetic non-Euclidean data.

To see how different clustering techniques might affect the performance, we have also conducted experiments using some other clustering algorithms, e.g. the single-linkage and complete-linkage methods [12]. The two methods work in a similar way as the average-group method. The differences lie in the way they calculate the distance between two clusters. In the single-linkage algorithm, the distance between two clusters C_1 and C_2 is given as

$$\min_{O_p \in C_1, O_q \in C_2} |d(O_p, O_q)| \tag{29}$$

In the complete-linkage algorithm, the distance is given as

$$\max_{O_p \in C_1, O_q \in C_2} |d(O_p, O_q)| \tag{30}$$

The results were slightly worse. The reason is that these two methods use the distance between a specific pair of objects, as opposed to the average distance between the objects in the two clusters. The errors incurred from measuring the distance between the specific pair of objects may affect the clustering accuracy seriously.

Finally we conducted experiments by replacing the random sampling objects used by $MetricMap$ with the $2k$ pivot objects found by $FastMap$. The performance of $MetricMap$ improves for the Euclidean data, but degrades for the non-Euclidean data. $MaxMap$ and $AvgMap$ remain the best as in the random sampling case.

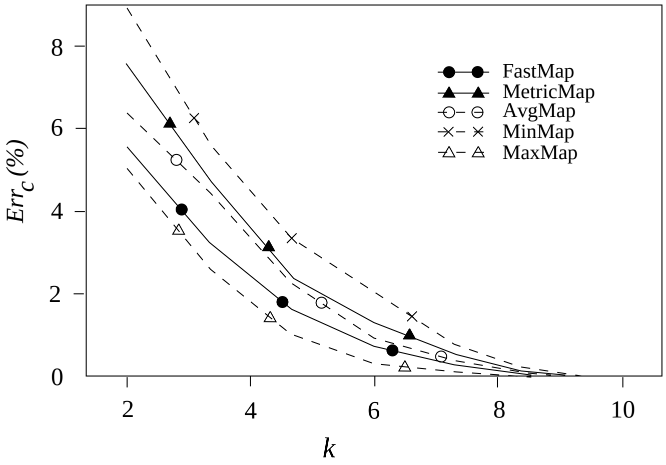

Fig. 13. Mis-clustering rates of the mappers as a function of the dimensionality of the target space for synthetic Euclidean data.

6 Discussion

Since $MaxMap$ and $AvgMap$ are educed from both $FastMap$ and $MetricMap$, their cost is approximately the sum of the costs of $FastMap$ and $MetricMap$. Figure 3 shows that when the dimensionality of the target space, k, increases, the relative errors of the mappers decrease. On the other hand, increasing k also increases the embedding cost (cf. Figure 11). One may wonder whether using $MaxMap$ and $AvgMap$ with a smaller k is better than using $FastMap$ and $MetricMap$ with a bigger k when they all have approximately the same cost. We have conducted experiments to answer this question.

Figures 18 and 19 depict the running times of the mappers as a function of k for synthetic Euclidean and non-Euclidean data, respectively. The dataset size N was 5000 for the Euclidean data and 3000 for the non-Euclidean data. It can be seen from the figures that the costs of the mappers are proportional to the dimensionality of the target space k. The cost of $MaxMap$ and $AvgMap$ with a k-dimensional target space is approximately the same as the cost of $FastMap$ with a $2k$-dimensional target space. Comparing with Figure 3, we see that using $FastMap$ or $MetricMap$ with a $2k$-dimensional target space yields a smaller relative error than using $MaxMap$ with a k-dimensional target space for the synthetic Euclidean data. On the other hand, comparing with Figure 7, we see that using $AvgMap$ with a k-dimensional target space achieves a more precise embedding than using $FastMap$ or $MetricMap$ with a $2k$-dimensional target space for the synthetic non-Euclidean data.

In general, there is a tradeoff between the embedding cost and the embedding precision. Recall that the asymptotic cost of all the mappers is $O(Nk)$ where N is the size of the dataset. In the case of Euclidean data, k is independent of

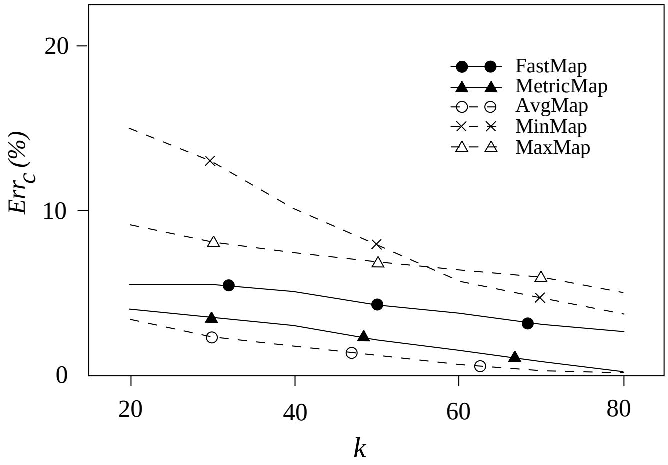

Fig. 14. Mis-clustering rates of the mappers as a function of the dimensionality of the target space for RNA data.

N. One can achieve a very precise embedding when k approaches the original dimensionality n of the vectors. Thus for a very large dataset of N Euclidean vectors, we can build a precise mapper (e.g. $MaxMap$) with a relatively low, asymptotically $O(N)$, cost. On the other hand, for the non-Euclidean data, the precision of the embedding depends on N. In order to build a precise mapper (e.g. $AvgMap$), k should be close to $N/2$, which leads to an $O(N^2)$ cost asymptotically.

We have experimented with different distance functions in the paper. Our approach can also be applied to nominal values when a proper metric is defined for these values. Nominal values are identified by their names and do not have numeric values. The colors of eyes [12] are an example. Colors are represented by hexadecimal numbers in the SRGB (Standard Red Green Blue) model as used for Web pages. SRGB is a default color space for the Internet proposed by Hewlett-Packard and Microsoft, and accepted by the W3 organization as a standard. Each color corresponds to six hexadecimal digits, which are decomposed to three pairs. Each pair corresponds to a primary color. One can define the distance between two different colors as the sum of the differences between the corresponding components. Specifically, let $c_1 = x_{11}\ x_{12}\ x_{13}$ and $c_2 = x_{21}\ x_{22}\ x_{23}$ be two colors, where x_{ij}, $i = 1, 2$, $j = 1, 2, 3$, denotes two hexadecimal digits. We define the distance between c_1 and c_2, denoted $d(c_1, c_2)$, as

$$d(c_1, c_2) = \sum_{j=1}^{3} |x_{1j} - x_{2j}| \tag{31}$$

For example, suppose the color "blue" corresponds to 00 00 FF, the color "black" corresponds to 00 00 00, and the color "green" corresponds to 00 80 00. The

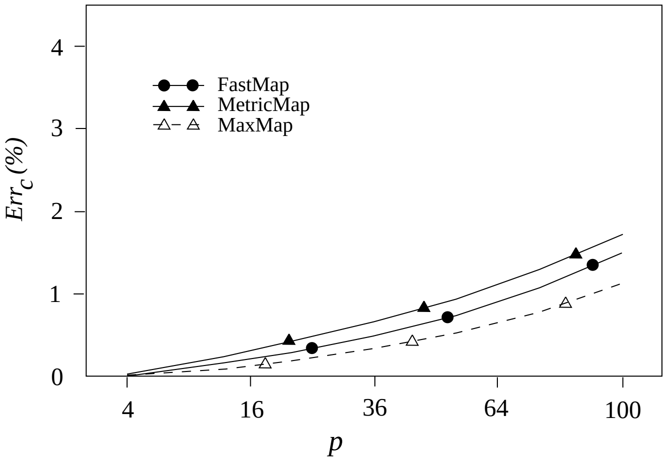

Fig. 15. Impact of the number of clusters.

distance between black eyes and blue eyes is $|00-00|+|00-00|+|FF-00| = FF$ in hexadecimal number or 255 in decimal number. Similarly, the distance between green eyes and blue eyes is $|00-00|+|00-80|+|FF-00| = 01\ 7F$ in hexadecimal number or 383 in decimal number. Clearly, for any three colors c_1, c_2 and c_3, we have $d(c_1, c_2) > 0$, $c_1 \neq c_2$ and $d(c_1, c_1) = 0$, $d(c_1, c_2) = d(c_2, c_1)$ and $d(c_1, c_2) \leq d(c_1, c_3) + d(c_3, c_2)$. Thus d is a metric and our approach is applicable.

7 Conclusion

In this paper we have presented an index structure, $MetricMap$, and compared it with the previously published index structure $FastMap$ [6]. The two index structures take a set of N objects, a distance metric d and embed those objects in a target space R^k, $k \leq N$, in such a way that the distances among objects are approximately preserved. $FastMap$ considers R^k to be Euclidean; $MetricMap$ considers R^k to be pseudo-Euclidean. Both index structures perform the embedding at an asymptotic cost $O(Nk)$.

We have conducted experiments to evaluate the accuracy of the embedding and the accuracy of clustering for the two index structures. The experiments were based on synthetic data as well as protein and virus datasets obtained from the Cold Spring Harbor Laboratory and National Cancer Institute. Our results showed that $MetricMap$ complements $FastMap$. In every case, combining the two index structures performs better than using either one alone. Specifically, $FastMap$ is more accurate than $MetricMap$ for Euclidean distances, but taking the maximum of the distances (we use the term dissimilarities because some of these values can be negative) gives the best accuracy of all. $MetricMap$ is more

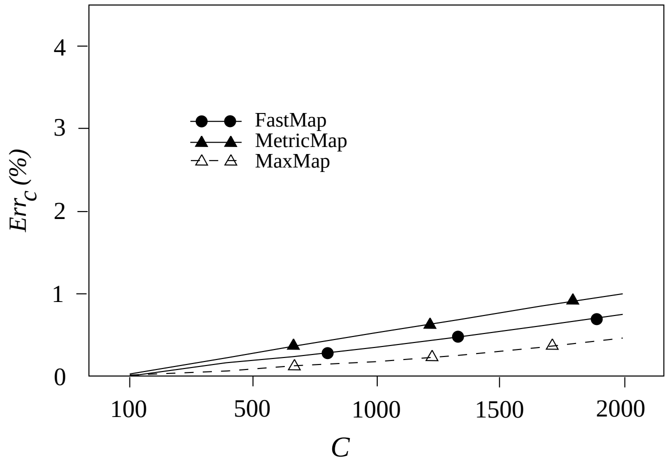

Fig. 16. Impact of the size of clusters.

accurate than $FastMap$ for non-Euclidean distances, but the average of the dissimilarities is best of all.

Besides the 4 datasets mentioned here, we have confirmed these results on 3 other datasets taken from dictionary words and other protein sequences. The practical significance of this work is that the proper use of these index structures can reduce the computation time substantially, thus achieving high efficiency for data mining and clustering applications.

Acknowledgments

We thank the anonymous reviewers and the executive editor, Dr. Xindong Wu, for their thoughtful comments and suggestions that helped to improve the paper. We also thank Dr. Tom Marr and Wojcieok Kasprzak for providing the protein and RNA data used in the experiments.

References

1. R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, Seattle, WA, 1998, pp. 94–105.
2. C. Burks, M. Cassidy, M. J. Cinkosky, K. E. Cumella, P. Gilna, J. E.-D. Hayden, G. M. Keen, T. A. Kelley, M. Kelly, D. Kristofferson, J. Ryals. GenBank, *Nucleic Acids Research* **19**, 2221–2225, 1991.
3. R. O. Duda, P. E. Hart. *Pattern Classification and Scene Analysis*, John Wiley & Sons: New York, 1973.

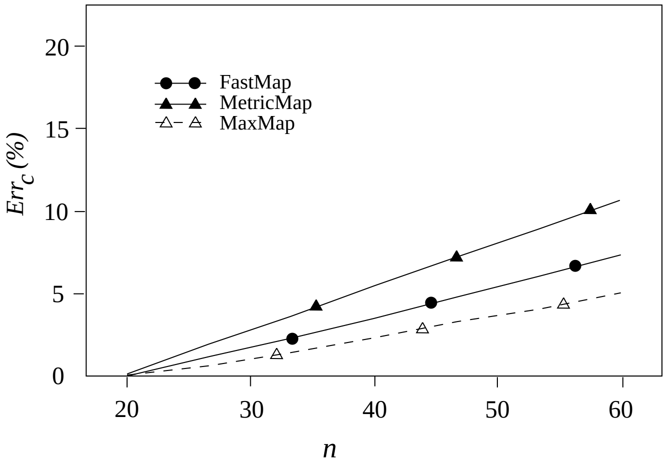

Fig. 17. Effect of the dimensionality of vectors in a cluster.

4. M. Ester, H.-P. Kriegel, J. Sander, X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, Portland, Oregon, 1996, pp. 226–231.
5. C. Faloutsos. *Searching Multimedia Databases by Content*, Kluwer Academic Publishers: Norwell, MA, 1996.
6. C. Faloutsos, K.-I. Lin. *FastMap*: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, San Jose, CA, 1995, pp. 163–174.
7. K. Fukunaga. *Introduction to Statistical Pattern Recognition*, Academic Press: San Diego, CA, 1990.
8. G. H. Golub, C. F. Van Loan. *Matrix Computations*, The Johns Hopkins University Press: Baltimore, MD, 1996.
9. W. Greub. *Linear Algebra*, Springer-Verlag: New York, NY, 1975.
10. S. Guha, R. Rastogi, K. Shim. CURE: An efficient clustering algorithm for large databases. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, Seattle, WA, 1998, pp. 73–84.
11. A. K. Jain, R. C. Dubes. *Algorithms for Clustering Data*, Prentice Hall: Englewood Cliffs, NJ, 1988.
12. L. Kaufman, P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons: New York, NY, 1990.
13. P. D. Lax. *Linear Algebra*, John Wiley & Sons: New York, NY, 1997.
14. R. S. Michalski, R. E. Stepp. Learning from observation: Conceptual clustering. In: R. S. Michalski, J. G. Carbonell, T. M. Mitchell (eds.), *Machine Learning: An Artificial Intelligence Approach*, volume I, Morgan Kaufmann: San Francisco, CA, 1983, pp. 331–363.

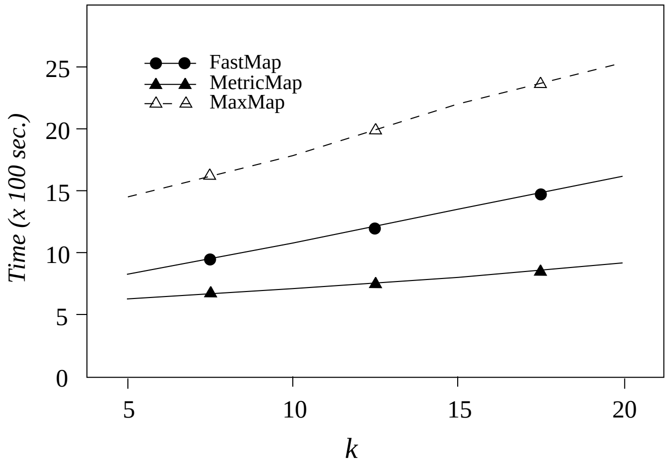

Fig. 18. Running times of the mappers as a function of the dimensionality of the target space for synthetic Euclidean data.

15. R. T. Ng, J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases*, Santiago, Chile, 1994, pp. 144–155.
16. J. M. Ortega. *Matrix Theory*, Plenum Press: New York, 1987.
17. D. Sankoff, J. B. Kruskal (eds.). *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley: Reading, MA, 1983.
18. B. A. Shapiro, J. Navetta. A massively parallel genetic algorithm for RNA secondary structure prediction, *Journal of Supercomputing* **8**, 195–207, 1994.
19. B. A. Shapiro, K. Zhang. Comparing multiple RNA secondary structures using tree comparisons, *Computer Applications in the Biosciences* **6**(4), 309–318, 1990.
20. D. Shasha, T. L. Wang. New techniques for best-match retrieval, *ACM Transactions on Information Systems* **8**(2), 140–158, 1990.
21. G. Sheikholeslami, S. Chatterjee, A. Zhang. WaveCluster: A multi-resolution clustering approach for very large spatial databases. In *Proceedings of the 24th International Conference on Very Large Data Bases*, New York, NY, 1998, pp. 428–439.
22. J. T. L. Wang, B. A. Shapiro, D. Shasha (eds.). *Pattern Discovery in Biomolecular Data: Tools, Techniques and Applications*, Oxford University Press: New York, NY, 1999.
23. J. T. L. Wang, D. Shasha, G. Chang, L. Relihan, K. Zhang, G. Patel. Structural matching and discovery in document databases. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, Tucson, AZ, 1997, pp. 560–563.
24. J. T. L. Wang, K. Zhang, K. Jeong, D. Shasha. A system for approximate tree matching, *IEEE Transactions on Knowledge and Data Engineering* **6**(4), 559–571, 1994.
25. W. Wang, J. Yang, R. Muntz. STING: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd International Conference on Very*

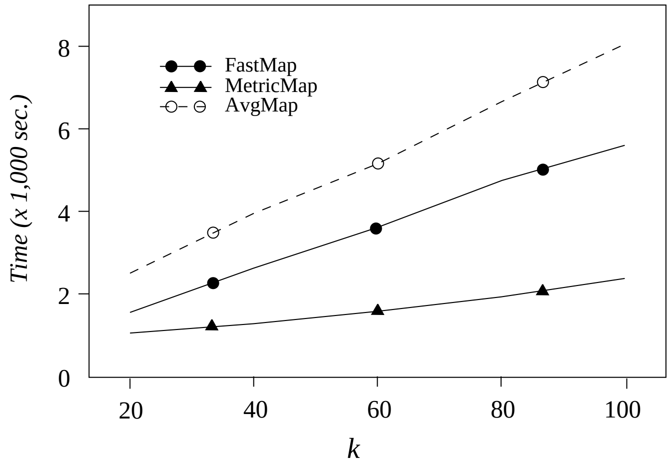

Fig. 19. Running times of the mappers as a function of the dimensionality of the target space for synthetic non-Euclidean data.

Large Data Bases, Athens, Greece, 1997, pp. 186–195.

26. Y. Yang, K. Zhang, X. Wang, J. T. L. Wang, D. Shasha. An approximate oracle for distance in metric spaces. In: M. Farach-Colton (ed.), *Combinatorial Pattern Matching*, Lecture Notes in Computer Science 1448, Springer-Verlag: Berlin, 1998, pp. 104–117.

27. B.-K. Yi, H. V. Jagadish, C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Proceedings of the International Conference on Data Engineering*, Orlando, Florida, 1998, pp. 201–208.

28. T. Zhang, R. Ramakrishnan, M. Livny. BIRCH: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, Montreal, Canada, 1996, pp. 103–114.

Xiong Wang received his M.S. degree in computer science from Fudan University, Shanghai, People's Republic of China, and the Ph.D. degree in computer science from the New Jersey Institute of Technology. He is currently a special lecturer in the Computer and Information Science Department at the New Jersey Institute of Technology. His research interests include scientific databases, knowledge bases, information retrieval and mining in high dimensional databases. He is a member of ACM and IEEE.

Jason T. L. Wang received the Ph.D. degree in computer science from the Courant Institute of Mathematical Sciences, New York University. He is currently an associate professor at NJIT's Computer and Information Science Department. His research interests include data mining and databases, pattern recognition, and computational biology. He has published 80 technical papers and is an editor and author of the book *Pattern Discovery in Biomolecular Data* (Oxford University Press, 1999).

King-Ip (David) Lin received the Ph.D. degree in computer science from the University of Maryland, College Park, USA. He is currently an assistant professor in the Department of Mathematical Sciences, University of Memphis, Tennessee. His research interests include databases, index structures, and data mining.

Dennis Shasha received degrees from Yale (BS), Syracuse (Master's), and Harvard (Ph.D). He is currently a professor of Computer Science at New York University's Courant Institute of Mathematical Sciences. His three principle research projects concern computational biology, navigation of unfamiliar databases, and data mining. He also writes books and a column about mathematical puzzles.

Bruce A. Shapiro received the B.S. degree from Brooklyn College and the Ph.D. degree from the University of Maryland, College Park, USA. He is a principal investigator for the Laboratory of Experimental and Computational Biology, National Cancer Institute, Frederick, Maryland. He does research in nucleic structure, developing algorithms and computational systems for determining structure/function relationships of nucleic acids. He is an editor and author of the book *Pattern Discovery in Biomolecular Data* (Oxford University Press, 1999).

Kaizhong Zhang received the M.S. degree in mathematics from Peking University, Beijing, People's Republic of China, in 1981, and the M.S. and Ph.D. degrees in computer science from the Courant Institute of Mathematical Sciences, New York University, New York, NY, USA, in 1986 and 1989, respectively. Currently, he is an associate professor in the Department of Computer Science, University of Western Ontario, London, ON, Canada. His research interests include pattern recognition, computational biology, sequential and parallel algorithms.

This article was processed using the LaTeX macro package with LLNCS style