Unified Framework for Fast Exact and Approximate Search in Dissimilarity Spaces

TOMÁŠ SKOPAL
Charles University in Prague

In multimedia systems we usually need to retrieve DB objects based on their similarity to a query object, while the similarity assessment is provided by a measure which defines a (dis)similarity score for every pair of DB objects. In most existing applications, the similarity measure is required to be a metric, where the triangle inequality is utilized to speedup the search for relevant objects by use of metric access methods (MAMs), e.g. the M-tree. A recent research has shown, however, that non-metric measures are more appropriate for similarity modeling due to their robustness and ease to model a made-to-measure similarity. Unfortunately, due to the lack of triangle inequality, the non-metric measures cannot be directly utilized by MAMs. From another point of view, some sophisticated similarity measures could be available in a black-box non-analytic form (e.g. as an algorithm or even a hardware device), where no information about their topological properties is provided, so we have to consider them as non-metric measures as well. From yet another point of view, the concept of similarity measuring itself is inherently imprecise and we often prefer fast but approximate retrieval over an exact but slower one.

To date, the mentioned aspects of similarity retrieval have been solved separately, i.e. *exact vs. approximate* search or *metric vs. non-metric* search. In this paper we introduce a similarity retrieval framework which incorporates both of the aspects into a single unified model. Based on the framework, we show that for any dissimilarity measure (either a metric or non-metric) we are able to change the "amount" of triangle inequality, and so to obtain an approximate or full metric which can be used for MAM-based retrieval. Due to the varying "amount" of triangle inequality, the measure is modified in a way suitable for either an exact but slower or an approximate but faster retrieval. Additionally, we introduce the *TriGen algorithm* aimed to construct the desired modification of any black-box distance automatically, using just a small fraction of the database.

Categories and Subject Descriptors: H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*indexing methods*

General Terms: design, algorithms

Additional Key Words and Phrases: similarity retrieval, approximate and exact search

1. INTRODUCTION

Since recent years the volume of available multimedia data grows rapidly, so the multimedia retrieval systems and multimedia databases are becoming more important than ever. As we see the progress in the fields of acquisition, storage, and dissemination of various multimedia formats, the application of effective and efficient multimedia management systems becomes indispensable in order to han-

Contact: Tomáš Skopal, tomas.skopal@mff.cuni.cz, http://siret.ms.mff.cuni.cz/skopal

dle all these formats. The application domains for multimedia retrieval include image/audio/video databases, CAD databases, but also molecular biology and medicine databases, geographical information systems, biometric databases and many others. In particular, more than 95% of web space is considered to store multimedia content, other multimedia data is stored in corporate and scientific databases, personal archives and digital libraries.

Due to the quick growth of multimedia data volumes, the *text-based* multimedia systems become useless, since the requirements on manual annotation exceed human possibilities and resources. The *metadata-based* search systems are of similar kind, we need an additional explicit information to effectively describe multimedia objects (e.g. structured semantic description, like class hierarchies or ontologies), which is not available in most cases.[1]

The only practicable way how to process and retrieve the vast volumes of raw multimedia data is the *content-based similarity search*, i.e. we consider the real content of each particular DB object. Because the multimedia objects have no universal syntactic and semantic structure (unlike traditional strong-typed rows in relational database tables or XML with a schema), the most general and feasible abstraction used in multimedia retrieval is the *query-by-example* concept, where the database objects are ranked according to similarity to a query object (the example). Only such database objects are retrieved, which have been ranked as sufficiently similar to the query object. The *similarity measure* returns a real-valued similarity score for any two multimedia objects on the input.

1.1 Preferences of Similarity Search

The retrieval *effectiveness* (i.e. the quality of the query result) is tightly bound to the similarity measure employed. To obtain just the expected results from the database, the similarity measure should follow the human perception (or judgment) of similarity. The measure, therefore, should not be limited by some topological properties (e.g. metric axioms) which restrict the richness of similarity modeling. Following this presumption, the most general similarity measuring can be achieved by a *non-metric* distance. However, despite the possibilities of non-metric measuring, the measures are often restricted to *metric* distances, since then the DB objects can be indexed by metric access methods, and then *efficiently* (i.e. quickly) retrieved when a similarity query is issued. We could call the choice between metric and non-metric measures as the *topological preference*.

From another point of view, unlike exact-match queries in traditional databases, the similarity measuring and retrieval in multimedia databases is inherently imprecise, subjective and changing over time. Thus, we might prefer faster but approximate methods which could retrieve some non-relevant objects (false hits) and miss some relevant ones (false dismissals). In many cases, the efficiency gain can be traded for an acceptable loss in effectiveness. Nevertheless, in some cases the similarity is precisely defined and then we require the search to be as exact as possible (e.g. biometric identification tasks). We could call the choice between exact and approximate search as the *precision preference*.

[1] The image search provided by Google is a successful example of text-based search engine, where the text annotation is extracted from web pages wherein the images are encapsulated.

Both the topological and precision preferences are orthogonal, we can distinguish metric or non-metric and exact or approximate preferences.

1.2 Paper Contributions

So far, the four preference combinations of similarity search have been handled separately. Most of the research has been carried out in the *exact metric search*, resulting in a large class of metric access methods. Also the *approximate metric search* has been extensively studied, while several proposals have been presented also in the area of exact as well as approximate *non-metric search*. However, to the best of our knowledge, there has not been proposed a unifying framework for all four aspects of similarity search. In this paper we propose such a framework based on modification of any dissimilarity measure (even a "black-box" one) to become applicable to any metric access method for either an exact or approximate retrieval.

1.2.1 *Differences from Previous work.* The framework is a generalization and extension of our previous contributions proposed in [Skopal 2006] and [Skopal et al. 2004]. In particular, the concept of distance modification is generalized, so that we not only consider so-called TG-modification of a semimetric distance (as in [Skopal 2006]), but we also propose the so-called TV-modification (proposed in a very restricted form in [Skopal et al. 2004]). The discussion on TG-/TV-modifiers (see Section 4) includes some new theoretical observations. The TriGen algorithm (proposed in [Skopal 2006]) has been generalized in order to support a new concept of T-bases – generators of both TG- and TV-modifiers. Additionally, a new algorithm for sampling anomalous distance triplets is proposed. Moreover, we introduce the ball-overlap factor (employed in the TriGen), which is an alternative to the intrinsic dimensionality. The content of experimental section is entirely new, the experiments have been performed on new datasets and using some new distance measures. The extended introduction to similarity search and related work sections are illustrated by many figures.

1.2.2 *Structure.* The rest of the paper is structured as follows. The preliminaries and dissimilarity measures are introduced in Section 2. In Section 3 we briefly survey the state-of-the-art approaches to all the mentioned aspects of similarity search. In Section 4 we present the key concept of our framework – similarity-preserving distance modifications. The Section 5 describes how to employ the modified distances in MAMs and in Section 6 we present the TriGen algorithm for automatic modification determination. The 7th section analyzes experimental results and in Section 8 we conclude the paper.

2. DISSIMILARITY SPACES

The similarity retrieval modeling is based on simplifying dissimilarity space abstraction. Let a multimedia object $\mathcal{O}$ be modeled by a *model object* $O \in \mathbb{U}$, where $\mathbb{U}$ is a model universe, which could be a cartesian product over attribute sets, a set of various structures (polygons, graphs, other sets, etc.), string closure, sequence closure, etc. A multimedia database $\mathcal{S}$ is then represented by a dataset $\mathbb{S} \subset \mathbb{U}$.

DEFINITION 1. (similarity & dissimilarity measure)
Let $s : \mathbb{U} \times \mathbb{U} \mapsto \mathbb{R}$ be a *similarity measure*, where $s(O_i, O_j)$ is considered as a

similarity score of multimedia objects $\mathcal{O}_i$ and $\mathcal{O}_j$. In many cases it is more suitable to use a *dissimilarity measure* $\delta : \mathbb{U} \times \mathbb{U} \mapsto \mathbb{R}$ equivalent to a similarity measure $s(\cdot,\cdot)$ as $s(Q,O_i) > s(Q,O_j) \Leftrightarrow \delta(Q,O_i) < \delta(Q,O_j)$. A dissimilarity measure (or *distance*) assigns a higher score to less similar objects, and vice versa. The pair $\mathcal{D} = (\mathbb{U}, \delta)$ is called a *dissimilarity space* – a kind of topological space. □

2.1 Metric distances

The distance measures often satisfy some of the metric properties ($\forall O_i, O_j, O_k \in \mathbb{U}$):

$$\delta(O_i,O_j) = 0 \quad \Leftrightarrow O_i = O_j \qquad \text{reflexivity (1)}$$
$$\delta(O_i,O_j) > 0 \quad \Leftrightarrow O_i \neq O_j \qquad \text{non-negativity (2)}$$
$$\delta(O_i,O_j) = \delta(O_j,O_i) \qquad \text{symmetry (3)}$$
$$\delta(O_i,O_j) + \delta(O_j,O_k) \geq \delta(O_i,O_k) \qquad \text{triangle inequality (4)}$$

The *reflexivity* (1) permits the zero distance just for identical objects. Both reflexivity and *non-negativity* (2) guarantee every two distinct objects are somehow positively dissimilar. If δ satisfies reflexivity, non-negativity and *symmetry* (3), we call δ a *semimetric*. Finally, if a semimetric δ satisfies also the *triangle inequality* (4), we call δ a *metric* (or metric distance). The triangle inequality is a kind of transitivity property; it says if O_i, O_j and O_j, O_k are similar, then also O_i, O_k are similar. If there is an upper bound d^+ such that $\delta : \mathbb{U} \times \mathbb{U} \mapsto \langle 0, d^+ \rangle$, we call δ a *bounded metric*. In such case $\mathcal{M} = (\mathbb{U}, \delta)$ is called a (bounded) *metric space*.

To complete the enumeration, we also distinguish *pseudometrics* (not satisfying the reflexivity), *quasimetrics* (not satisfying symmetry) and *ultrametrics* (a stronger type of metric, where the triangle inequality is restricted to ultrametric inequality – $max\{\delta(O_i,O_j), \delta(O_j,O_k)\} \geq \delta(O_i,O_k)$).

We also define the important concept of triangular triplet, which will serve us in the proposed framework in order to quantify the "amount" of triangle inequality fulfillment.

DEFINITION 2. (triangular triplet)
A tuple of real numbers (a,b,c), $a,b,c \geq 0$, $a+b \geq c$, $b+c \geq a$, $a+c \geq b$, is called a *triangular triplet*. Let (a,b,c) be ordered as $a \leq b \leq c$, then (a,b,c) is an *ordered triplet*. If $a \leq b \leq c$ and $a+b \geq c$, then (a,b,c) is an *ordered triangular triplet*. □

A metric δ *generates* just the (ordered) triangular triplets, i.e. $\forall O_i, O_j, O_k \in \mathbb{U}$, $(\delta(O_i,O_j),\ \delta(O_j,O_k),\ \delta(O_i,O_k))$ is triangular triplet. Conversely, if a measure generates just the triangular triplets, then it satisfies the triangle inequality. A triangular triplet can be perceived as a set of all L_2-drawn triangles of side lengths given by the triplet components (and vice versa). While some metrics generate every possible triangle (e.g. the Euclidean distance), some generate only a subset (e.g. an ultrametric distance generates just the isosceles triangles).

2.1.1 *Examples of Metric distances.* The most widely used metrics are the *Minkowski* (L_p) distances, defined on D-dimensional vector spaces as

$$L_p(x,y) = \left(\sum_{1 \leq i \leq D} |x_i - y_i|^p \right)^{1/p}, \ p \geq 1, \ x,y \in \mathbb{R}^D$$

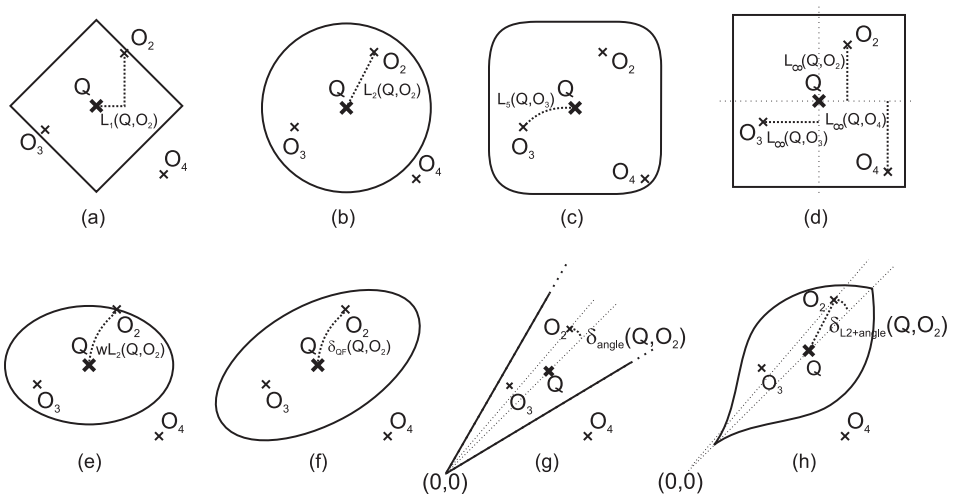

Fig. 1. Region balls of various metric distances.

The special cases are the *Manhattan* distance (or L_1 distance, see L_1-ball in Figure 1a – points at a fixed L_1-distance from a center point Q), the *Euclidean* distance (or L_2 distance, see Figure 1b), the *Chessboard* distance (or Chebyshev L_∞, see Figure 1d). For $p > 2$ the L_p ball looks more like a square with rounded corners (the higher p, the sharper corners, see L_5-ball in Figure 1c) and for $1 < p < 2$ the L_p ball looks like a diamond with knobbed edges.

A generalized case of the Euclidean distance is the *quadratic-form* distance (or Mahalanobis), defined as $\delta_{QF}(x, y) = \sqrt{(x-y)\Sigma(x-y)^T}$, where the Σ is a square positive-definite matrix, wherein some correlations between individual coordinates are specified. The quadratic-form distance allows to model similarity in such vector spaces where the features (dimensions) are supposed to be correlated (see Figure 1f). For example, in red-green-blue color space we suppose the green component is more correlated with the blue one than with the red. If Σ is just a diagonal matrix, we talk about *weighed Euclidean distance* (some dimensions are just said to be more or less important than the other ones, see Figure 1e), while unitary diagonal Σ turns the quadratic-form distance into an ordinary Euclidean. The Minkowski distances (especially L_2) have been widely used to measure dissimilarity [Rubner et al. 2001; Corboy et al. 2005]; some approaches prefer the L_1 distance [Bustos et al. 2005], some other the L_∞ distance [Li et al. 2006].

The angle between vectors is also a metric which can be viewed as an L_2-distance along the surface of origin-centered unitary L_2-ball (see Figure 1g). This *angle distance* is widely used in Information Retrieval [Baeza-Yates and Ribeiro-Neto 1999], however, in a form of non-metric *cosine measure* (see next subsection).

Since any *linear combination of metrics* is a metric as well, we can combine simple metrics into more complicated ones (see Figure 1h where the ball for L_2 + angle distances is depicted).

For non-vector data we can exploit a plenty of other distances. The *Levenshtein* (or edit) distance counts the minimum number of basic operations (character inser-

tion, deletion, replacement) in order to turn one string into another one. A similar edit distance can be defined on trees (the *tree edit distance*), where the basic operations are defined as node insertion, deletion and relabeling. The edit distance has been used in various areas, e.g. DNA sequence matching [Sankoff and Kruskal 1983], the tree edit distance is useful for e.g. XML structural similarity search [Nierman and Jagadish 2002].

For general finite sets we can use the *Jaccard distance* $\delta_{Jacc}(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|}$, which measures the (inverse of) normed overlap between sets A and B. Another set-based distance is the *Hausdorff distance* $\delta_{Haus}(A,B) = max\{h(A,B), h(B,A)\}$, where $h(A,B) = max_{a \in A}\{min_{b \in B}\{d(A,B)\}\}$. The function h measures the greatest *partial distance* d (which must be a metric) over the elements in A to their nearest neighbors in B. Since h is defined on general sets, the set elements can be of any type which can be passed to the partial metric distance d.

Finally, we could mention the *earth mover's distance*, defined as $\delta_{EMD}(A,B) = min_F \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} c_{ij} w^T f_{ij}$, where $F = \{f_{ij}\}$ is the flow matrix and each f_{ij} denotes mass to be moved from each region i of A to each region j of B (for details we refer to [Rubner et al. 2000]). The object could be an image divided into regions, e.g. a grid, but it can be generally anything divisible into (spatial) regions.

2.2 Non-metric distances

The metric properties have been argued against by some theories in psychology and computer vision as restrictive in similarity modeling [Santini and Jain 1999; Tversky 1977]. In particular, the reflexivity and non-negativity have been refuted by claiming that different objects could be differently self-similar [Krumhansl 1978; Tversky 1977]. For instance, in Figure 2a the image of a leaf on a trunk can be viewed as positively self-dissimilar if we consider a distance which measures the less similar parts of the objects (here the trunk and the leaf). Or alternatively, in Figure 2b the leaf-on-trunk and leaf could be treated as identical if we consider a distance which measures the most similar parts of the objects (the leaves). Nevertheless, the reflexivity and non-negativity are the less problematic properties.

The symmetry was questioned by showing that a prototypical object can be less similar to an indistinct one than vice versa [Rosch 1975; Rothkopf 1957]. In Figure 2c, the more prototypical "Great Britain and Ireland" image is more distant to the "Ireland alone" image than vice versa.

The triangle inequality is the most attacked property. Some theories point out the similarity has not to be transitive [Ashby and Perrin 1988; Tversky and Gati 1982]. Demonstrated by the well-known example, a man is similar to a centaur, the centaur is similar to a horse, but the man is completely dissimilar to the horse (see Figure 2d).

2.2.1 *Examples of Non-Metric distances.* The non-metric measures have been used mainly in the areas of multimedia databases and information retrieval. A common rationale for their usage is better *robustness* – a robust measure is resistant to outliers, i.e. to anomalous or "noisy" objects. In an "intra-object" meaning, a robust measure can neglect some portions of the measured objects which appear as the most dissimilar.

To mention some vector distances, the *fractional L_p distances* [Aggarwal et al.

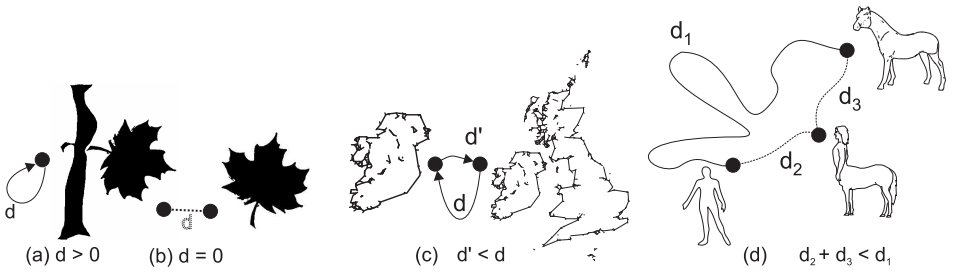

Fig. 2. Objections against metric properties in similarity measuring: (a) reflexivity (b) non-negativity (c) symmetry (d) triangle inequality

2001] extend the Minkowski metrics (see Section 2.1.1) for $0 < p < 1$, however, such L_p distances are only semimetrics. Unlike classic L_p metrics, the fractional L_p variants allow us to inhibit extreme differences in coordinate values – this can be viewed as a robust behavior (see Figure 3a). The fractional L_p distances have been suggested for robust image matching [Donahue et al. 1996] and retrieval [Howarth and Ruger 2005].

The *cosine measure*, defined as $s_{\cos}(u,v) = \frac{\sum_{i=1}^{D} u_i v_i}{\sqrt{\sum_{i=1}^{D} {u_i}^2 \cdot \sum_{i=1}^{D} {v_i}^2}}$, is widely used in the vector model of text retrieval [Baeza-Yates and Ribeiro-Neto 1999]. Although $s_{\cos}(u,v)$ is similarity measure, by $\delta_{\cos}(u,v) = 1 - s_{\cos}(u,v)$ we get an equivalent semimetric distance. The triangle inequality can be enforced by $arccos(s_{\cos})$, which, actually, turns $s_{\cos}$ into the angle distance (see Section 2.1.1).

The vectorial structure can also be used to store a distribution histogram (where each histogram bin is assigned to a vector coordinate), while distribution-based distances (both metric and non-metric) are used in many areas, e.g. in image retrieval [Rubner et al. 2001]. As an example, the non-metric *Jeffrey-divergence* is defined as $\delta_{JD}(x,y) = \sum_i x_i log\frac{2x_i}{x_i+y_i} + y_i log\frac{2y_i}{x_i+y_i}$.

Generally, we can obtain a non-metric distance by linear combination of other distances where at least one is a non-metric (see combination of L_2 distance and $1-$cosine measure in Figure 3b), or we can e.g. multiply (non)metric distances (see $L_{\frac{1}{3}} \cdot$ QF-ball in Figure 3c).

The robust behavior can be provided by various *k-median distances*; these measure the kth most similar portions of the compared objects. Generally, a k-median distance is of form

$$\delta(O_1, O_2) = k\text{–}med(d_1(O_1, O_2), d_2(O_1, O_2), \ldots, d_n(O_1, O_2))$$

where $d_i(O_1, O_2)$ is a partial distance between O_1 and O_2, considering the ith portions of the objects. Among the partial distance values $d_i(\cdot,\cdot)$, the *k–med* operator returns the kth smallest value. In Figure 3d see a region ball of 2-median distance consisting of quadratic-form distance, angle distance and weighed L_2.

As a special k-median distance derived from the Hausdorff metric, the *partial Hausdorff distance* (pHD) has been proposed for shape-based image retrieval [Huttenlocher et al. 1993]. Given two sets $\mathcal{S}_1, \mathcal{S}_2$ of points (e.g. two polygons or clouds of points), the partial Hausdorff distance uses $d_i(\mathcal{S}_1, \mathcal{S}_2) = dNP(\mathcal{S}_1^i, \mathcal{S}_2)$, where dNP is the Euclidean (L_2) distance of the ith point in $\mathcal{S}_1$ to the nearest point in $\mathcal{S}_2$. To

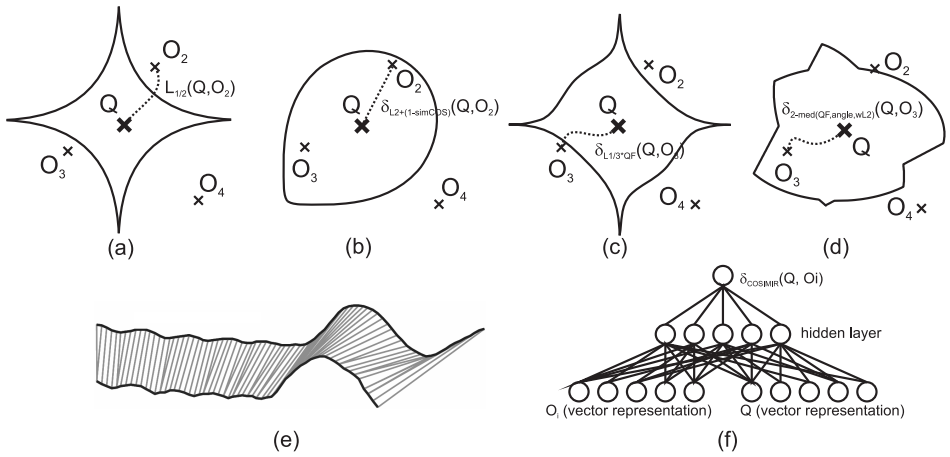

Fig. 3. (a-d) Region balls of some non-metric distances. (e) DTW distance. (f) COSIMIR model.

keep the distance symmetric, pHD is the maximum of δ in both directions, i.e. $pHD(\mathcal{S}_1, \mathcal{S}_2) = max(\delta(\mathcal{S}_1, \mathcal{S}_2), \delta(\mathcal{S}_2, \mathcal{S}_1))$. Similar to pHD is another modification of Hausdorff metric, used for face detection [Jesorsky et al. 2001], where the average of dNP distances is considered, instead of k-median.

In the area of sequence and string matching, the idea of sequence (or string) alignment is utilized in many distance measures. In principle, the elements of two sequences are pair-wise aligned in a way that the sum of distances between aligned elements is minimized. Unlike edit distance, the popular *(dynamic) time warping distance* (DTW) is not a metric, while it has been used in time series retrieval [Yi et al. 1998], and even in shape retrieval [Bartolini et al. 2005]. The DTW distance is robust in sense that it is quite resistant to the sampling frequency or a time shift, i.e. two sequences sampled at (locally) different rates are ranked as similar, see Figure 3e. To eliminate some pathological alignments, there were also developed various constraints on DTW, e.g. the Sakoe-Chiba's band, Itakura's parallelogram, etc. [Keogh and Ratanamahatana 2005]. Another non-metric string measure is the *longest common subsequence* (LCSS), however, LCSS is a similarity measure (two strings with no common subsequence have LCSS = 0).

Besides distance measures based on simple description (like the L_p distances), some measures are very complex and, therefore, for them a "manual" enforcement of metric properties is nearly impossible. The COSIMIR model [Mandl 1998] consists of three-layer backpropagation network (see Figure 3f), which can be trained to model an arbitrary user-defined similarity measure (but hardly a metric one).

2.3 Learning & Dynamic distances

In addition to the topological and precision preferences, we can identify also a kind of *dynamic preference* declaring whether the similarity can or cannot evolve over the time (and also during the process of retrieval) [Guo et al. 2002; Brambilla et al. 1999]. The reason could be either learning (the similarity learns the human's cognition by e.g. relevance feedback) or just evolving due to the dynamic nature

of similarity (some objects appear more or less similar in different time periods; we can also consider user profiles which adjust the similarity for each user). By the way, the topological preference is not likely to be orthogonal to the dynamic preference, since learning measures hardly preserve metric properties.

When related to the process of retrieval, some approaches consider the query object as one of the factors which modifies the actual semantic of similarity in given query context [Ciaccia and Patella 2000]. In particular, in [Bustos et al. 2005] the authors suggest dynamic combinations of metrics for more effective 3D retrieval. We can observe that such "multi-metric" approach improves the flexibility of similarity measuring, however, in a different way than the rich but static non-metric measuring.

Unlike the "static" similarity measures, the topological properties of learning and dynamic distances can vary over the time. We suppose the (topological properties of) measures are static, so the dynamic preference is out of scope of this paper.

2.4 Black-box & Hardware-supported distances

Besides the analytical descriptions of various measures (even the very complex ones like the COSIMIR), we can design a similarity which can be described just by an algorithm written in context-free language – as a black box returning a real-value output from a two-object input. The topological properties (i.e. the metric axioms) of an algorithmically described similarity measure are generally undecidable, so we have to treat such a measure as a non-metric.

To improve the performance of similarity measuring, there were designed specialized hardware ASIC (Application-specific integrated circuit) co-processors, following the VLSI (Very-large-scale integration) paradigm. The ASICs offer high performance for specialized tasks (e.g. a particular similarity measuring), however, a disadvantage of such specialized devices is their limited usage when the similarity measure is to be re-designed. In [Mukherjee 1989] the author proposes HW algorithms for string matching (the LCSS, in particular). A recent trend in VLSI is the reconfigurable computing, where a general-purpose FPGA (field-programmable gate array) is physically configured to act as a specialized processor (instead of a brand new ASIC design). Unlike CPUs where just the control flow is driven by software, the FPGAs allow to change the "native" data flow throughout the circuit – in simple words, to configure a task-specific hardware design. Recently, the FPGA-based implementation of Euclidean distance and cosine measure was proposed in [Freeman 2006]. An unknown FPGA device implementing a similarity measure has to be considered also as a non-metric black box.

2.5 Computational Complexity of Dissimilarity Measures

The time complexity of algorithms implementing the similarity measures can vary from linear to exponential (when related to the object sizes n_1, n_2).

The Minkowski metrics are quite cheap – they are computed in $O(n)$ time. The Jaccard distance could be generally computed in $O((n_1 + n_2)log(n_1 + n_2))$ time, where a sort operation must be performed in order to compute union/intersection of the sets. The (k-median) Hausdorff distance is generally computable in $O(n_1 n_2) \cdot O(d)$ time, where $O(d)$ is time complexity of the partial distance function d. The quadratic-form distance is of complexity $O(n^2)$ (due to the vector-and-matrix mul-

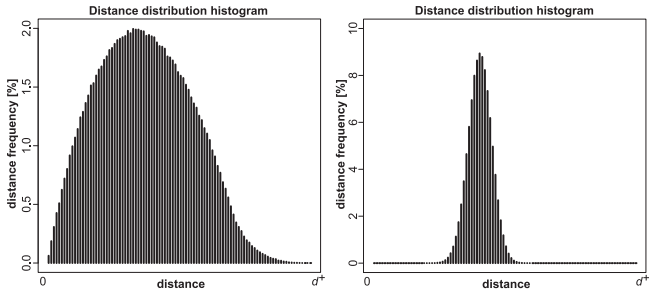

Fig. 4. Distance distribution histograms indicating (a) low and (b) high intrinsic dimensionality.

tiplication). The sequence alignment measures (like edit distance, DTW, LCSS, etc.) are also of quadratic complexity (when computed by dynamic programming). An example of a very expensive distance measure is the earth mover's distance, which is of exponential complexity (considering linear programming computation).

2.6 Indexability Indicators

To efficiently search in a multimedia database, the topological properties of employed dissimilarity measure are just a partial information needed for design of a successful access method.

2.6.1 *Intrinsic Dimensionality.* The *distance distribution* inside the dataset $\mathbb{S}$ is another important factor for similarity indexing. The distance distribution can reveal a structure inside the dataset, i.e. a fact whether there exist clusters of objects and how tight they might be. Given a dataset $\mathbb{S}$ and a distance[2] δ, the efficiency limits of any access method are indicated by the *intrinsic dimensionality* (IDIM)[3] of $\mathbb{S}$, defined as

$$\rho(\mathbb{S}, \delta) = \frac{\mu^2}{2\sigma^2}$$

where μ and σ^2 are the mean and the variance of the distance distribution in $\mathbb{S}$ (proposed in [Chávez et al. 2001]). In Figure 4 see an example of *distance distribution histograms* (DDHs) indicating low ($\rho = 3.61$) and high ($\rho = 42.35$) intrinsic dimensionalities.

The intrinsic dimensionality is low if there exist tight clusters of objects – some objects are close to each other and far from the other ones. If all the indexed objects are almost equally distant, then intrinsic dimensionality is high, which means the dataset is poorly intrinsically structured. A high ρ value says that many (even all) of partitions created on $\mathbb{S}$ are likely to be overlapped by every possible query, so the query processing deteriorates to sequential search in all the partitions. The problem of high intrinsic dimensionality is, in fact, a generalization of the well-known *curse of dimensionality* [Weber et al. 1998; Chávez et al. 2001] into metric spaces.

[2]The δ was originally assumed to be a metric distance, but we consider also the generalized dissimilarity case.

[3]Actually, there exist other interpretations of intrinsic dimensionality, e.g. the fractal dimensionality [Faloutsos and Kamel 1994] or mapping dimensionality [Kao et al. 1997], however, we consider the one presented in this paper as the most appropriate to similarity search purposes.

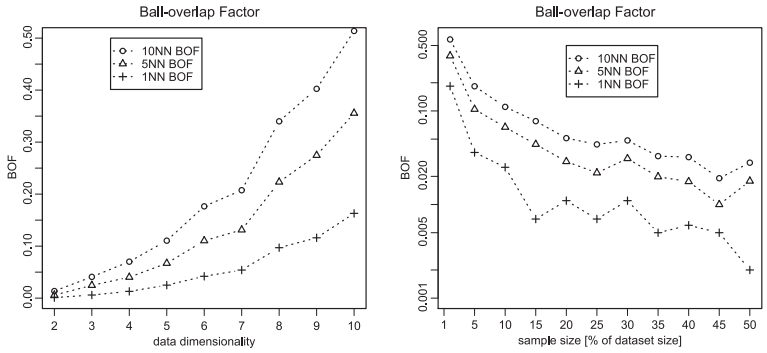

Fig. 5. Ball-overlap factor.

2.6.2 *Ball-Overlap Factor.* Given a dataset and a dissimilarity measure, the intrinsic dimensionality gives us an indirect prediction (or cost model) about the indexing efficiency. However, instead of simple statistical properties like mean and variance of distances, we would like to capture rather an information about real relationships between data clusters described by some regions in the dissimilarity space. The regions should be of a shape/form used by metric access methods (see next section).

In particular, the *fat* and *bloat factors* [Traina Jr. et al. 2000] have been proposed to classify the likelihood of efficient search in a dataset indexed by M-tree. However, these factors are related to an existing M-tree-based index, while we would like to classify the dataset $\mathbb{S}$ itself, regardless of a specific access method and a particular index built for the dataset. Therefore, we propose the *ball-overlap factor* (BOF), which is defined as

$$\mathrm{BOF}_k(\mathbb{S},\delta) = \frac{2}{|\mathbb{S}^*| * (|\mathbb{S}^*|-1)} \sum_{\forall O_i,O_j \in \mathbb{S}^*, i>j} sgn(|(O_i,\delta(O_i,kNN(O_i)))\bar{\cap} \bar{\cap}(O_j,\delta(O_j,kNN(O_j)))|)$$

where $\delta(O_i,kNN(O_i))$ is the distance to O_i's k-th nearest neighbor in $\mathbb{S}^*$ and $(O_i,\delta(O_i,kNN(O_i)))$ is thus the ball in metric space centered in O_i of radius $\delta(O_i,kNN(O_i))$. The statement $sgn((\cdot,\cdot)\bar{\cap}(\cdot,\cdot))$ returns 1 if the two balls overlap (in geometric-based, not data-based, meaning) and 0 if they do not. The ball overlap condition is defined as $\delta(O_i,kNN(O_i)) + \delta(O_j,kNN(O_j)) \geq \delta(O_i,O_j)$.[4]

In simple words, the BOF_k calculates the ratio of overlaps between ball regions, where each region is made of an object (from the dataset sample) and of such a covering radius which guarantees (at least) $k+1$ data objects are located inside the ball. In such a way the balls can be regarded as indexing regions. The overlap ratio then predicts the likelihood that two arbitrary ball-shaped regions will overlap or not. The BOF factor can thus serve us as a more appropriate MAM-independent

[4]This is actually correct just in metric spaces, however, we have no correct possibility to perform an exact geometric-based overlap in non-metric spaces. Nevertheless, we suppose BOF will work also in non-metric spaces when used to compare two non-metric distances (as proved in experiments).

efficiency indicator for metric access methods based on ball partitioning, e.g. the M-tree (see next section). In Figure 5a see the BOF_1, BOF_5 and BOF_{10} factors for vector datasets of increasing dimensionality. The Figure 5b shows how the size of data sample $\mathbb{S}^*$ affects the values of BOF.

3. SIMILARITY SEARCH – RELATED WORK

In this section we survey the main approaches to similarity search. In particular, we summarize main access methods for metric search[5] – we overview both exact and approximate search methods. To the best of our knowledge, we offer for the first time overview of the state-of-the-art non-metric approaches to similarity search.

3.1 Similarity Queries

First of all, in the following we consider the *query-by-example* concept; we look for objects similar to a query object $Q \in \mathbb{U}$ (Q is derived from an example multimedia object). Necessary to the query-by-example retrieval is a notion of *similarity ordering*, where the objects $O_i \in \mathbb{S}$ are ordered according to the distances to Q. For a particular type of query, there is specified a portion of the ordering returned as the query result. The *range query* and the *k nearest neighbors (kNN) query* are the most popular ones[6]. A range query (Q, r_Q) selects all objects from the similarity ordering for which $\delta(Q, O_i) \leq r_Q$, where $r_Q \geq 0$ is a distance threshold (or query radius). A kNN query (Q, k) selects the k most similar objects (first k objects in the ordering).

Each particular range query region is represented by a *ball* in the metric space, centered in Q and of radius r_Q. In a kNN query the r_Q radius is not known in advance, so it must be incrementally refined during the kNN query processing. The simplest implementation of similarity query evaluation is the *sequential search* over the entire dataset. The query object is compared against every object in the dataset, resulting in a similarity ordering which is used for the query evaluation. The sequential search often provides a baseline for other access methods.

3.2 Metric Access Methods And Exact Search

The metric access methods (MAMs) provide data structures and algorithms by use of which the objects relevant to a similarity query can be retrieved efficiently (i.e. quickly). The MAMs build a persistent auxiliary data structure, called *metric index*, so we also talk about metric indexing. The main principle behind all MAMs is a utilization of the triangle inequality property (satisfied by every metric), due to which MAMs can organize/index the objects of $\mathbb{S}$ within distinct classes. When a query is processed, only the candidate classes are searched (such classes which overlap the query), so the searching becomes more efficient (see Figure 6).

The efficiency of a MAM depends not only on *I/O costs* (as in case of spatial access methods, e.g. R-tree), the second important (and often the dominant) component are the *computation costs* – the number of distance computations needed to answer a query.

[5]For a comprehensive survey on metric access methods we refer to monograph [Zezula et al. 2005].
[6]There are more types of similarity queries – reverse kNN query, (k-)closest pairs, similarity join – however, the range and kNN queries are primitives used to compose more complex query types.

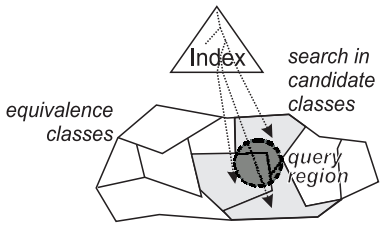

Fig. 6. Classes of similar objects indexed by a metric access method.

There were developed many MAMs for different scenarios (e.g. designed to either secondary storage or main memory management). Besides others we name the *M-tree* [Ciaccia et al. 1997], *vp-tree* [Yianilos 1993], *(m)vp-tree* [Bozkaya and Özsoyoglu 1999], *gh-tree* [Uhlmann 1991], *GNAT* [Brin 1995], *LAESA* [Micó et al. 1992], or more recent ones, *D-index* [Dohnal et al. 2003] and *PM-tree* [Skopal et al. 2005].

The MAM-based similarity search is accomplished by applying metric properties to quickly prune the search space. Basically, the MAM classes are represented by data regions in the metric space which are described either by *ball regions* (or their compositions, e.g. rings), which is the most used representation (M-tree family, (m)vp-tree, D-index) or by *hyper-plane partitioning* (gh-tree, GNAT). During query processing, a candidate data region is checked for an overlap by the query ball. In case of overlap, the respective region has to be search – this means either filtering of data objects (if the region contains already the data objects, e.g. tree leaf) or filtering of nested regions (when considering hierarchical MAMs, e.g. trees or D-index). In Figure 7 see several examples of MAMs, in particular M-tree (there are many extensions of M-tree, e.g. the PM-tree), GNAT, mvp-tree, D-index.

3.2.1 *Mapping Methods.* An indirect way to accomplish metric search is mapping the dataset into a low-dimensional vector space. There have been proposed various *mapping (or embedding) methods*, e.g. MDS, FastMap, MetricMap, SparseMap, to name a few [Faloutsos and Lin 1995; Hjaltason and Samet 2003]. The dataset $\mathbb{S}$ is embedded into a vector space $(\mathbb{R}^k, \delta_V)$ by a mapping $F : \mathbb{S} \mapsto \mathbb{R}^k$, where the distance $\delta(\cdot,\cdot)$ is (approximately) preserved by a cheap vector metric δ_V (often the L_2 distance).

In many cases the mapping F is *contractive*, i.e. $\delta_V(F(O_i), F(O_j)) \leq \delta(O_i, O_j)$, which allows to filter out some non-relevant objects using δ_V, but some other non-relevant objects, called *false hits*, must be re-filtered by δ (see e.g. [Filho et al. 2001]). The mapped vectors can be indexed/searched by any MAM, however, since the data objects are mapped into a vector space, we can utilize also *spatial/point access methods*, like R-tree, X-tree or VA-file [Böhm et al. 2001].

A particular method based on mapping is LAESA, where a contractive mapping of the metric space to $(\mathbb{R}^k, L_\infty)$ is constructed using k *pivots* $P_i \in \mathbb{S}$. The mapping function F turns an object O_i to a vector $(\delta(P_1, O_i), \delta(P_2, O_i), \ldots, \delta(P_k, O_i))$. When searching, a range query (Q, r_Q) is mapped to the target space as $(F(Q), r_Q)$, see Figure 8 (kNN queries are processed in a similar way). The retrieved candidate objects (here O_1, O_4) have to be refiltered to eliminate possible false hits (here O_4).

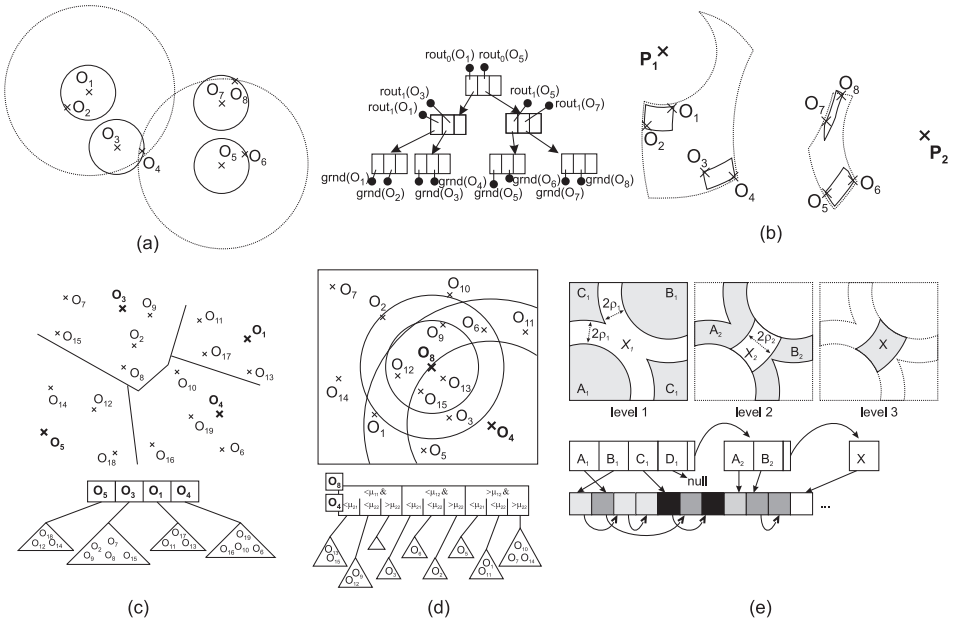

Fig. 7. Several MAMs: (a) M-tree (b) PM-tree (c) GNAT (d) mvp-tree (e) D-index

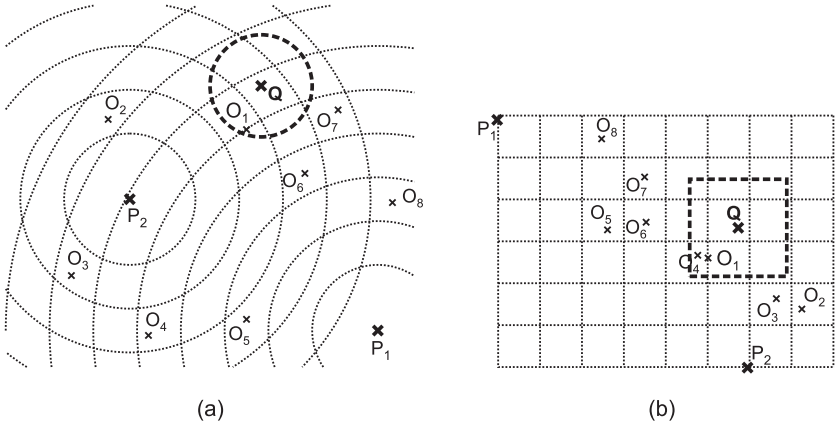

Fig. 8. Mapping from (a) source metric space to (b) target vector space

To say the drawbacks, the mapping methods are expensive, while the distances are preserved only approximately, which leads to *false dismissals* (relevant objects being not retrieved). The contractive methods eliminate the false dismissals but suffer from a great number of false hits (especially in case of a low k), which leads to lower retrieval efficiency. In most cases the methods need to process the dataset in batch, so they are suitable for static databases and MAMs only.

3.2.2 *M-tree Family.* The M-tree (and its variants) is a popular MAM designed for database environments. The M-tree is based on B^+-tree; it is a paged, dynamic and balanced index structure (see Figure 7a). Its inner nodes contain *routing entries* which describe ball-shaped metric regions which bound the underlying data objects in leaves. The leaf nodes consist of *ground entries* – the indexed data objects themselves.

In recent years, the M-tree has been modified or improved either to achieve better performance, or to extend the query model. The former case modifications are the *Slim-tree* [Traina Jr. et al. 2000] (cheaper splitting of nodes and redistribution of ground entries resulting in more compact regions), the *M^+-tree* [Zhou et al. 2003] (employs twin-nodes to better partition the Euclidean space) and the *PM-tree* [Skopal et al. 2005] (the region balls are further pruned by ring regions, see Figure 7b). The latter case includes the *QIC-M-tree* [Ciaccia and Patella 2002] (allowing to query by a user-defined distance metric), *M^2-tree* [Ciaccia and Patella 2000] and *M^3-tree* [Bustos and Skopal 2006] (allowing to query by a combination of metrics where the weight of each particular metric is specified at query time).

3.3 Approximate Search in Metric Spaces

Nowadays, an efficient search in (intrinsically) high-dimensional datasets is feasible solely by usage of approximate methods. Fortunately, since the similarity measuring and search is inherently imprecise, a retrieval which is approximate to some extent can be satisfactory in many cases. Among many approaches to the approximate search, in this section we outline several representative methods.

3.3.1 *Approximately Correct Search.* In [Zezula et al. 1998] the authors propose three techniques of approximate search in M-tree, varying in the way how to give up the query result precision for the sake of improved search efficiency. The techniques were applied to approximate kNN search.

The first technique[7] considers a user-defined *relative distance error* $\epsilon \geq 0$. The error ϵ states that the distance between Q and an approximation of k-th nearest neighbor O_A^k *must* be no more than $(1+\epsilon)$ times farther than the real k-th nearest neighbor O_N^k, i.e. $\delta(Q, O_A^k) \leq (1+\epsilon)\delta(Q, O_N^k)$. Then O_A^k is called the $(1+\epsilon)$ k-th nearest neighbor. In order to provide approximate search, the kNN query algorithm of M-tree is adjusted in a way that query radius r'_Q, used to discard non-relevant regions, is $1+\epsilon$ times smaller than the proper dynamic query radius r_Q (i.e. $r'_Q = r_Q/(1+\epsilon)$), see Figure 9a.

The second technique proposed is a usage of distance distribution $F_Q(x)$ for approximate kNN search. More specifically, $F_Q(x)$ represents the fraction of objects in the dataset, for which the distance to Q is less than or equal to x. Provided there are n objects in the dataset, $n \cdot F_Q(x)$ objects should have distance to Q not greater than x. Given a kNN candidate O_A^k, $F_Q(\delta(Q, O_A^k))$ determines the fraction of the best approximations of kNN, while a sufficiently small $F_Q(\delta(Q, O_A^k))$ has been used for an early termination stop condition of approximate kNN search.

The third approximation heuristic proposed stops the kNN algorithm processing as soon as the intermediate results change only slowly. In such cases, most of

[7] A similar approach, but oriented to BBD-trees, was proposed in [Arya et al. 1998].

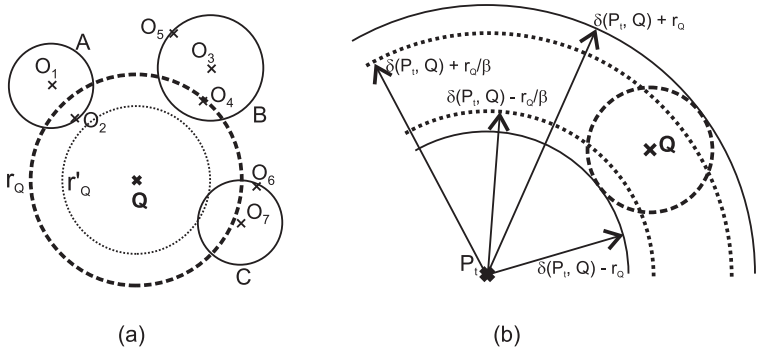

Fig. 9. (a) $(1+\epsilon)$ kNN in M-tree (b) Probabilistic LAESA

the kNN algorithm runtime is spent on negligible changes in the result set, where $\delta(Q, O_A^k)$ is changing steadily slower.

3.3.2 *Space Transformations & Mapping Methods.* The contractive mapping methods (see Section 3.2.1) can be easily utilized for faster but approximate search. The idea is to omit the second phase of refiltering candidate objects such that the false hits are not filtered, so the result is only approximate. This approach is simple and can be implemented by various spatial access methods (e.g. the R-tree), but there is a risk of very bad results if the number of false hits is too large – they can, in fact, dominate the true hits and make them false dismissals. Another way to approximate search can be non-contractive mapping of the objects, such that the distances in the target space are preserved only approximately. Both means (contractive/non-contractive) are often accomplished by mapping to some low-dimensional space.

3.3.3 *Clustering Techniques.* Several access methods are based on clustering techniques, especially those designed for the vector space case, e.g. the *Clindex* [Li et al. 2002] or *VQ-file* [Tuncel et al. 2002]. A variant for metric spaces is the *buoy indexing* approach [Volmer 2002], where the dataset is partitioned into disjoint clusters bounded by ball regions, where the cluster representative is called a buoy. Clusters are built by assigning objects to the cluster with the closest buoy. The buoys are iteratively refined so that the sum of radii of cluster balls in the result set gets minimized. The approximate kNN search is provided by processing only a limited number of the nearest clusters.

A similar approach is presented in [Goldstein and Ramakrishnan 2000], where the *P-sphere tree* – a two-level structure – is used. The data in leaves (clusters) are referenced by entries in the root node. Each root entry contains a covering sphere consisting of a center object and a radius. The approximate NN algorithm sequentially searches the leaf (cluster) which is nearest to the query object.

3.3.4 *Probabilistic Search.* Given a user-defined threshold $\theta \in \langle 0, 1\rangle$, the probabilistic methods guarantee the probability of a false dismissal is at most θ. The user can tune the θ parameter in order to achieve an optimal efficiency vs. accuracy trade-off.

A probabilistic approach to LAESA-like methods has been proposed in [Chávez and Navarro 2001], where the triangle inequality is "stretched" by a multiplier $\beta \geq 1$. Specifically, the filtering condition $|\delta(O_i, P_t) - \delta(Q, P_t)| > r_Q$ is stretched by β, see Figure 9b. In order to preserve the user-defined probability θ of a false drop, the upper-bound for β is computed as $\beta \leq \frac{r_Q \sqrt{1-(1-\theta)^{\frac{1}{p}}}}{\sqrt{2}\sigma}$, where p is the number of pivots, and σ^2 is variance of the dataset's distance distribution.

Recently, an approach [Amato et al. 2003] has been proposed, predicting the probability that an intersection of a query region and a data region contains some indexed objects. If the probability is lower than a user-defined threshold, the data region (e.g. M-tree node) is not processed. The probability is predicted by means of *region proximity*, computed using the dataset's distance distribution.

A method of probabilistic metric search based on *compact partitions* has been introduced in [Bustos and Navarro 2004]. The idea is to fix in advance the limit of distance computations allowed to answer a query. Moreover, an advanced version exploits a ranking of the regions to be searched, so that the most promising regions are processed at first.

3.3.5 *PAC Queries.* When searching in a high-dimensional dataset, even the approximately correct methods (see Section 3.3.1) work inefficiently. In [Ciaccia and Patella 2000] the authors propose *probably approximately correct* (PAC) nearest neighbor search, even more reducing the search costs at the expense of only probabilistic search. The method extends the $(1 + \epsilon)$ nearest neighbor search by a user-defined confidence parameter θ. The method searches so that the retrieved object is a $(1 + \epsilon)$ nearest neighbor with probability at least θ. The NN algorithm is enhanced by a stop condition which terminates the NN search if the distance to the current $(1 + \epsilon)$ NN candidate falls below r_Q^θ, where (Q, r_Q^θ) is a region which is guaranteed to be empty with probability at least θ. The PAC-NN algorithm has been applied to M-tree as well as to sequential index.

3.4 Exact Search in Non-Metric Spaces

The existing approaches to exact non-metric search are designed for particular access methods and also for a particular distance.

As a classic technique, the *inverted index* [Baeza-Yates and Ribeiro-Neto 1999] is widely used in fulltext systems for vector query processing. Whenever a vector query is to be processed, only such lists in the inverted index are scanned, the ids of which correspond to ids of nonzero query weights. However, we must realize the skipping of zero-weight lists is only possible due to the usage of cosine measure or inner product as the similarity measure (where the multiplying by zero query weight leads to zero). Thus, we cannot use translation-invariant distance (e.g. any L_p distance), since there are no coordinate multiplications and so we cannot skip the zero-weighted lists. To overcome this limitation, the *IGrid* [Aggarwal and Yu 2000] was proposed – a structure inspired by the inverted index – usable with a kind of robust fractional L_p distances, where the values in each dimension are quantized into equi-width buckets (a bucket stores a list, similarly like an inverted index entry). The query processing then exploits a similar idea as the inverted index, the substraction of particular query and vector coordinate values (used to compute the

robust L_p) is considered as nonzero just in case the compared values fall into the same bucket – otherwise the index bucket is skipped.

To support similarity search by a non-metric distance d_Q, the *QIC-M-tree* [Ciaccia and Patella 2002] has been proposed as an extension of the M-tree (the key idea is applicable also to other MAMs). The M-tree index is built by use of an index distance d_I, which is a metric *lower-bounding* the query distance d_Q (up to a scaling constant $S_{I \to Q}$), i.e. $d_I(O_i, O_j) \leq S_{I \to Q}\ d_Q(O_i, O_j), \forall O_i, O_j \in \mathbb{U}$. As d_I lower-bounds d_Q, a query can be partially processed by d_I (which, moreover, could be computationally much cheaper than d_Q), such that many non-relevant classes of objects (subtrees in M-tree) are filtered out. All objects in the non-filtered classes are compared against Q using d_Q. Actually, this approach is similar to the usage of contractive mapping methods (d_I is an analogy to δ_V), but here the objects generally need not to be mapped into a vector space. However, this approach has two major limitations. First, for a given non-metric distance d_Q there is no general way how to find the metric d_I. Although d_I could be found "manually" for a particular d_Q (as in [Bartolini et al. 2005]), this is not easy for d_Q given as a black box (an algorithmically described one). Second, the lower-bounding metric should be as tight approximation of d_Q as possible, because this "tightness" heavily affects the intrinsic dimensionality, the number of MAMs' filtered classes, and so the retrieval efficiency.

3.5 Approximate Search in Non-Metric Spaces

To the best of our knowledge, there was not proposed a specialized access method for approximate non-metric search. Nevertheless, this task can be indirectly carried out by either mapping methods or classification.

3.5.1 *Mapping Methods.* As in the case of approximate metric search, the non-metric case can be also accomplished by mapping methods. To better preserve the "exotic" non-metric distances in Euclidean spaces, there were specialized techniques proposed, we refer to [Athitsos et al. 2005; Kruskal 1964; Farago et al. 1993].

3.5.2 *Classification.* Quite many attempts to non-metric nearest neighbor (NN) search have been tried out in the classification area. Let us recall the basic three steps of classification. First, the dataset is organized in classes of similar objects (by user annotation or clustering). Then, for each class a description consisting of the most representative object(s) is created; this is achieved by *condensing* [Hart 1968] or *editing* [Wilson 1972] algorithms. Third, the NN search is accomplished as a classification of the query object. Such a class is searched, to which the query object is "nearest" – there is an assumption the nearest neighbor is located in the "nearest class". For non-metric classification there have been proposed methods enhancing the description of classes (step 2). In particular, condensing algorithms producing *atypical points* [Goh et al. 2002] or *correlated points* [Jacobs et al. 2000] have been successfully applied.

In [Roth et al. 2002], the authors transform a matrix of pair-wise non-metric distances into a matrix of metric distances which is equivalent with respect to the effectiveness of subsequent clustering/classification. The "metrization" is accomplished by a simple distance shift such that the triangle inequality becomes satisfied. However, although the modified matrix is suitable for clustering purposes,

the constant shifting of distances is problematic for efficient indexing and search (as discussed later in Section 5.2). Moreover, this approach is not much "database-friendly", since it assumes a static database (matrix); an insertion/deletion requires recomputation of the entire model.

The drawbacks of classification-based methods reside in static indexing and limited scalability, while the querying is restricted just to approximate (k-)NN.

4. DISTANCE MODIFICATIONS FOR SIMILARITY SEARCH

The proposed framework is based on a kind of dissimilarity transformation, however, this one is based on dissimilarity-to-dissimilarity transformation (rather than a dissimilarity-to-vector space one, mentioned in the previous sections).

4.1 Assumptions

We assume δ satisfies at least reflexivity and non-negativity but, as we have mentioned in Section 2.2, these are the less restrictive properties and can be handled easily. The *non-negativity* is satisfied by a shift of the distances, while for the *reflexivity* property we require every two non-identical objects are at least d^--distant (d^- is some positive distance lower bound). The lower-bound distance (d^-) and upper-bound distance (d^+) could be provided by the distance measure in case the structure of input universe $\mathbb{U}$ is known. Or, if it is not, we can sample a number of distances $\delta(O_i, O_j), O_i, O_j \in \mathbb{S}$ and determine the approximate lower-/upper-bound distances. The outlier distances (exceeding the upper-bound distance or falling below the lower-bound distance) can be represented directly by d^- or d^+, and two objects falling into the "d^--bucket" are regarded as *at most* d^--distant. Similarly, two objects falling into the "d^+-bucket" are regarded as *at least* d^+-distant. When searching, the possibly relevant objects involved in outlier distances $\delta(Q, O_i)$ (where Q is a query object) are filtered sequentially in the original space.

Furthermore, searching by an *asymmetric measure* d could be partially provided by a symmetric measure δ, e.g. $\delta(O_i, O_j) = min\{d(O_i, O_j), d(O_j, O_i)\}$. Using the symmetric measure some non-relevant objects can be filtered out, while the original asymmetric measure δ is then used to rank the remaining non-filtered objects.

In the following we assume the measure δ is a bounded semimetric (which includes also full metrics). The bounding assumption is introduced just for clarity of the following presentation – the forthcoming principles can be extended to the general case without restrictions. Finally, as δ is bounded by d^+, we can further normalize the semimetric such that it assigns distances from $\langle 0, 1\rangle$. This can be achieved simply by scaling the original value $\delta(O_i, O_j)$ to $\frac{\delta(O_i,O_j)}{d^+}$. The same way a range query radius r_Q must be scaled to $\frac{r_Q}{d^+}$, when searching.

4.2 Similarity-Preserving Modifiers

The cornerstone of the proposed transformation is an employment of so-called *similarity-preserving modifiers* which, given a query object Q and the dataset $\mathbb{S}$, induce classes of equivalent similarity orderings (with respect to Q and $\mathbb{S}$).

DEFINITION 3. (similarity-preserving modification)
Given a dissimilarity measure δ, we call $\delta^f(O_i, O_j) = f(\delta(O_i, O_j))$ a *similarity-preserving modification of* δ (or *SP-modification*), where f, called the *similarity-*

preserving modifier (*SP-modifier*), is a strictly increasing function for which $f(0) = 0$. Again, for clarity reasons we assume f is bounded, i.e. $f : \langle 0, 1\rangle \mapsto \langle 0, 1\rangle$. □

DEFINITION 4. (similarity ordering)

We define $SimOrder_\delta : \mathbb{U} \mapsto 2^{\mathbb{U}\times\mathbb{U}}$, $\forall O_i, O_j, Q \in \mathbb{U}$ as $\langle O_i, O_j\rangle \in SimOrder_\delta(Q) \Leftrightarrow \delta(Q, O_i) < \delta(Q, O_j)$, i.e. $SimOrder_\delta$ orders objects by their distances to Q. □

Although an SP-modification δ^f is not topologically equivalent to δ (because the distances induce different systems of open sets), we can observe that any SP-modification of δ induces the same similarity ordering (i.e. all SP-modifications of δ are *SO-equivalent*), as follows.

LEMMA 1. *(similarity ordering (SO-) equivalence)*

Given a dissimilarity δ and any δ^f, $SimOrder_\delta(Q) = SimOrder_{\delta^f}(Q), \forall Q \in \mathbb{U}$.

Proof: As f is increasing, then $\forall Q, O_i, O_j \in \mathbb{U}$ it follows that $\delta(Q, O_i) > \delta(Q, O_j) \Leftrightarrow f(\delta(Q, O_i)) > f(\delta(Q, O_j))$. ■

The SO-equivalence can be seen as a weaker alternative to the topological equivalence – it says two distances represent the same *similarity retrieval model*. In other words, if a query is processed sequentially (by comparing all objects in $\mathbb{S}$ to the query object Q), then it does not matter if we use either δ or any δ^f, because both ways induce the same similarity ordering. Naturally, the radius r_Q of a range query must be modified to $f(r_Q)$, when searching by δ^f.

4.3 Triangle-Generating Modifiers

Among the infinite number of possible SP-modifiers, the class of triangle-generating modifiers has an exceptional property – the ability of (partial) triangle inequality enforcement. More specifically, the class of triangle-generating modifiers itself is a special subclass of so-called *metric-preserving functions*, studied in metric spaces theory [Corazza 1999]. Within the class of metric-preserving functions some are simultaneously SP-modifiers, as follows.

DEFINITION 5. (metric-preserving SP-modifier)

An SP-modifier f is *metric-preserving* if for every metric δ the SP-modification δ^f preserves the triangle inequality, i.e. δ^f is also metric.[8] □

The concave SP-modifiers are always metric-preserving.

LEMMA 2. *(concave SP-modifier)*

(a) Every concave SP-modifier f is metric-preserving.
(b) Let (a, b, c) be a triangular triplet and f be a metric-preserving SP-modifier, then $(f(a), f(b), f(c))$ is a triangular triplet as well.

Proof: For the proof and for more about general metric-preserving functions (not only the continuous/monotonous ones) we refer to [Corazza 1999]. ■

DEFINITION 6. (triangle-generating modifier)

Let a strictly concave SP-modifier f be called a *triangle-generating modifier* (or *TG-modifier*). Having a TG-modifier f, let a δ^f be called a *TG-modification*. □

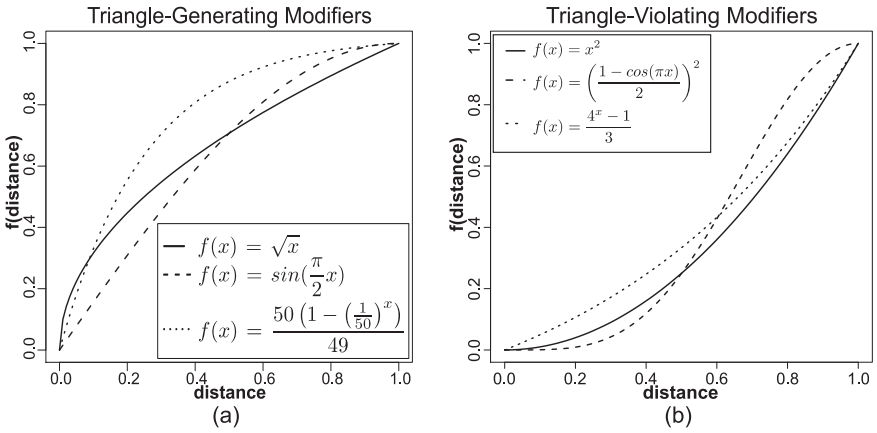

Fig. 10. Several T-modifiers: (a) TG-modifiers (b) TV-modifiers

The concavity property of *any* TG-modifier (see examples in Figure 10a) not only guarantees the preservation of the triangle inequality with respect to *any* metric being modified, it does strengthen the "triangle inequality fulfillment" of *any* semimetric. Moreover, it can even fully enforce the triangle inequality for a semimetric – turning it into a metric – as follows.

THEOREM 1. *(turning semimetric into metric)*
Given a semimetric δ, then there always exists a TG-modifier f, such that the SP-modification δ^f is a metric.

Proof: We show that every ordered triplet (a, b, c) generated by δ can be turned by a single TG-modifier f into an ordered triangular triplet.

1. As every semimetric is reflexive and non-negative, it generates ordered triplets just of forms $(0, 0, 0)$, $(0, c, c)$, and (a, b, c), where $a, b, c > 0$. Among these, just the triplets (a, b, c), $0 < a \leq b < c$, can be non-triangular. Hence, it is sufficient to show how to turn such triplets by a TG-modifier into triangular ones.

2. Suppose an arbitrary TG-modifier f_1. From TG-modifiers' properties it follows that $\frac{f_1(a)}{f_1(c)} > \frac{a}{c}$, $\frac{f_1(b)}{f_1(c)} > \frac{b}{c}$, hence $\frac{f_1(a)+f_1(b)}{f_1(c)} > \frac{a+b}{c}$ (theory of concave functions). If $(f_1(a) + f_1(b))/f_1(c) \geq 1$, the triplet $(f_1(a), f_1(b), f_1(c))$ becomes triangular (i.e. $f_1(a) + f_1(b) \geq f_1(c)$ is true). In case there still exist triplets which have not become triangular after application of f_1, we take another TG-modifier f_2 and compose f_1 and f_2 into $f^*(x) = f_2(f_1(x))$. The compositions (or nestings) $f^*(x) = f_i(\ldots f_2(f_1(x))\ldots)$ are repeated until f^* turns all triplets generated by δ into triangular ones – then f^* is the single TG-modifier f we are looking for. ■

4.3.1 *Notes to the proof of Theorem 1.* The constructive nesting of TG-modifiers in the second phase can be done in a finite number of steps if we assume the distance a not tending to zero. Otherwise, the process of nesting could be (but not necessarily) infinite. Nevertheless, in computer we model data types in discrete

[8] *Note:* Such an SP-modifier is additionally *subadditive* $(f(x) + f(y) \geq f(x + y), \forall x, y \in \mathbb{R}_0^+)$.

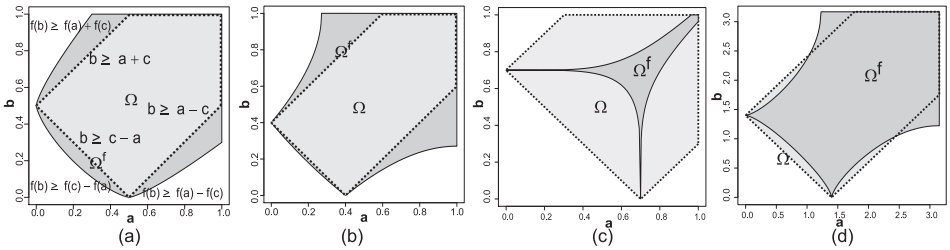

Fig. 11. (a,b) Triplet regions of a TG-modification (c,d) Triplet regions of a TV-modification

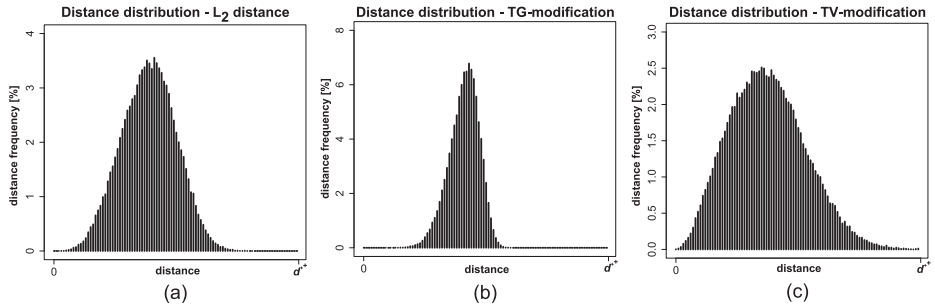

Fig. 12. (a) L_2-DDH (b) L_2^f-DDH, $f(x) = x^{\frac{1}{2}}$ (c) L_2^f-DDH, $f(x) = x^{\frac{3}{2}}$

domains (i.e. also the distances are discrete approximations), so in practice the nesting is always finite.

The proof shows the more concave TG-modifier we apply, the more triplets become triangular. This effect can be visualized by 3D regions in the space $\langle 0, 1\rangle^3$ of all possible distance triplets, where the three dimensions represent the distance values a,b,c, respectively. In Figures 11a,b see examples of region[9] Ω of all triangular triplets as the dotted-line-bounded area. The super-region Ω^f (the solid-line-bounded area) represents all the triplets which become (or remain) triangular after the application of TG-modifier $f(x) = x^{\frac{3}{4}}$ and $f(x) = sin(\frac{\pi}{2}x)$, respectively.

The more concave TG-modifier, the more the objects forming a triplet become equidistant, i.e. the more the respective triangles become equilateral. This observation also implies the mean of distances increases and the variance decreases thus, the intrinsic dimensionality of TG-modified distance δ^f is always greater than that of δ (with respect to $\mathbb{S}$). In Figure 12b see an example of distance distribution regarding to TG-modified L_2 distance (for the not modified L_2 distance distribution see Figure 12a). The classic *curse of dimensionality* problem [Böhm et al. 2001] refers to the same behavior in vector spaces – the higher dimensionality, the more the vectors become equidistant, so the relation between high intrinsic dimensionality and high embedding dimensionality is obvious.

[9]The 2D representations of Ω and Ω^f regions are c-cuts of the real 3D regions.

4.4 Triangle-Violating Modifiers

As a counterpart to TG-modifiers, we can define the triangle-violating modifiers and modifications.

DEFINITION 7. (triangle-violating modifier)

Let a strictly convex SP-modifier f be called a *triangle-violating modifier* (or *TV-modifier*). Having a TV-modifier f, let a δ^f be called a *TV-modification*. □

Unlike TG-modifiers, the TV-modifiers (see Figure 10b) cause the triangle inequality of a TV-modified metric could be no more satisfied, i.e. the metric could be turned into a semimetric (or a semimetric is still kept semimetric). This can be stated as the opposite to Theorem 1, as follows.

THEOREM 2. *(turning metric into semimetric)*

Given a metric δ, then there always exists a TV-modifier f, such that the SP-modification δ^f is a semimetric.

Proof: The proof is exactly the opposite to that of Theorem 1. By composing TV-modifiers we increase the variance among distances within distance triplets, so that some of them sooner or later become NOT triangular triplets (we omit the exotic case where all the objects in $\mathbb{U}$ are equidistant). ■

In consequence, the properties of TG-modifiers discussed in the previous section hold inversely for TV-modifiers. In particular, the more convex TV-modifier we apply, the more triplets become non-triangular. Like for the TG-modifiers, in Figures 11c,d see examples of region Ω of all triangular triplets[10]. The region Ω^f represents all the triplets which remain triangular after the application of TV-modifier $f(x) = x^5$ or $f(x) = (\frac{1-cos(x)}{2})^{\frac{7}{10}}$ (actually, this one is a partially convex and partially concave SP-modifier – note the Ω^f is neither super- nor sub-region of Ω).

Furthermore, every (strictly convex) TV-modification exhibits lower intrinsic dimensionality than the original (semi)metric δ (with respect to $\mathbb{S}$), see the DDH in Figure 12c (compare with Figure 12a).

In the following, we will call a TG-modifier or TV-modifier f simply a *T-modifier*.

5. EXACT AND APPROXIMATE INDEXING & SEARCH

The T-modifiers can be utilized to perform an exact or approximate, metric or non-metric similarity search. As a T-modifier can either increase or decrease the "amount" of triangle inequality of a particular dissimilarity measure, we can utilize metric access methods as a general tool for similarity search. In other words, the abilities of MAMs are no more limited to the metric search case but now include also the non-metric fields.

In particular, we can use a TG-modifier to fully or partially enforce the triangle inequality for a semimetric (at the cost of higher intrinsic dimensionality obtained), or we can use a TV-modifier to "de-metricate" a metric to achieve lower intrinsic

[10] Actually, the SP-modifier used in Figure 11d and the second one in Figure 10b are not strict TV-modifiers since they are only partially convex.

dimensionality and thus faster but approximate retrieval. We can also even worsen the violation of triangle inequality of a semimetric measure, providing this will not significantly affect the retrieval effectiveness.

Unlike all the approaches presented in Section 3, the "modification approach" proposed in this paper has three advantages:

—An employment of T-modifiers covers both preferences of similarity search, the topological and precision preferences, and their combinations. We can perform exact metric, exact non-metric, approximate metric and approximate non-metric search.

—The modifications are not restricted to be used by a specific access method, they are generally applicable to any MAM. After obtaining a suitable T-modification (we will discuss how to do this in the next section), we can use any MAM for the similarity search. We must just additionally modify the query radius r_Q to $f(r_Q)$ when issuing a range query (the kNN search needs no adjustment at all).

—The distance measure is always treated as a black box, i.e. we do not require any analytical information on the distance semantics. The information used by modification is obtained purely on statistical basis (by sampling a number of distance triplets).

5.1 Retrieval Error Model

When given a TV-modifier f, we could determine the "amount" of triangle inequality violation (however, just for case the modified distance is a *metric*) as the *triangle-violation error*

$$E^f_{TV} = 1 - \frac{V(\Omega^f)}{V(\Omega)}$$

where $V(\Omega^f)$ is the volume of region Ω^f and $V(\Omega)$ is the volume of region Ω (defined in the previous section). The volume of Ω can be determined as

$$V(\Omega) = \int_{c=0}^{d^+} 2d^+c - \frac{3}{2}c^2 \; \mathbf{dc} = \frac{d^{+3}}{2}$$

i.e. the volume is one half of the cube $\langle 0, d^+\rangle^3$. Since f is increasing, there exists the inverse function f^{-1}, and the volume of Ω^f can be determined as

$$V(\Omega^f) = d^{+3} - \int_{c=0}^{d^+}\int_{a=0}^{c} f^{-1}(f(c)-f(a))\mathbf{da}\,\mathbf{dc} - 2\int_{c=0}^{d^+}\int_{a=c}^{d^+} f^{-1}(f(a)-f(c))\mathbf{da}\,\mathbf{dc}$$

However, although the E^f_{TV} can be determined easily (because of its distance- and data- *independence*, and due to analytical form of f), its utilization is limited just to TV-modified metrics. Moreover, the E^f_{TV} error is very rough indicator of how the triangle inequality of a TV-modified metric got deteriorated. For example, considering $f(x) = x^2$, the actual triangle inequality is highly violated in L^f_2 distance, however, in δ^f ($\delta = \sqrt[3]{L_2}$) it is fully preserved. This is in contradiction with E^f_{TV} error which reports violation (even of the same size) in both cases.

Hence, instead of triangle-violation error, we introduce the *T-error*, a distance- and data- *dependent* measure providing a more appropriate estimation of how a metric access method will behave when used with a T-modified distance.

5.1.1 *T-Error.* An important question is how to measure the "amount" of triangle inequality preservation/violation. Since the triangle-violation error is not appropriate, we propose the *triangle error (T-error)*, defined as the fraction of distance triplets being non-triangular (the triplets are sampled from the dataset), i.e.

$$\varepsilon_{\delta,\mathbb{S}} = \frac{m_{nt}}{m}$$

where m is the total number of sampled distance triplets and m_{nt} is the number of triplets being non-triangular. Obviously, the total number of triplets m acquired from a dataset sample $\mathbb{S}^* \subseteq \mathbb{S}$ affects the precision of T-error. Ideally, we would like to obtain $\varepsilon_{\delta,\mathbb{S}} = \varepsilon_{\delta,\mathbb{U}}$, where $\varepsilon_{\delta,\mathbb{U}}$ is the T-error computed on the entire universe $\mathbb{U}$ by incorporating all possible triplets, i.e. $m = \binom{|\mathbb{U}|}{3}$ (which includes also all possible query objects). Of course, evaluation of $\varepsilon_{\delta,\mathbb{U}}$ is impracticable, so to keep the T-error as precise as possible, we must incorporate suitable sampling techniques which minimize the discrepancy between $\varepsilon_{\delta,\mathbb{U}}$ and $\varepsilon_{\delta,\mathbb{S}}$, while retain the size of $\mathbb{S}^*$ and the triplet count m as low as possible (we will discuss the sampling techniques later in Section 6.3).

We expect the T-error can generally predict the real retrieval error exhibited by *any* metric access method when used with a non-metric T-modification (or any other semimetric distance). The real retrieval error can be defined as relative precision and recall, or more simply as a kind of *Jaccard distance*, the normed overlap distance E_{NO} between the query result QR_{MAM} returned by a MAM and the correct query result QR_{SEQ} obtained by sequential search of the dataset, i.e.

$$E_{NO} = 1 - \frac{|QR_{MAM} \cap QR_{SEQ}|}{max(|QR_{MAM}|, |QR_{SEQ}|)}$$

Moreover, we can employ a user-defined *T-error tolerance threshold* θ, in order to control the T-error of a T-modification we are choosing for our retrieval task, i.e. we search for a modifier f such that $\varepsilon_{\delta^f,\mathbb{S}} \leq \theta$.

5.2 Suitable T-modifiers

In addition to minimizing the T-error, we aim to use a T-modification which keeps the indexability indicators (i.e. the intrinsic dimensionality and/or the ball-overlap factor, see Section 2.6) as good as possible. For an example why this is important, consider a TG-modifier

$$f(x) = \begin{cases} 0 & (\text{for } x = 0) \\ \frac{x+d^+}{2} & (\text{otherwise}) \end{cases}$$

which turns every d^+-bounded semimetric into a metric, keeping the T-error zero. Unfortunately, such a metric is useless for searching, since all classes of objects maintained by a MAM are overlapped by every query, so the retrieval always deteriorates to sequential search. This behavior is reflected by maximal ball-overlap factor, i.e. $\mathrm{BOF}_k(\mathbb{S}, \delta^f) = 1$. The intrinsic dimensionality $\rho(\mathbb{S}, \delta^f)$ is also high, however, it is not appropriate here since it does not recognize whether f is totally useless for indexing or the dataset is just bad-structured – this was actually the reason why the BOF was introduced.

In fact, we look for an *optimal* T-modifier, i.e. a T-modifier which turns (or keeps) only such non-triangular triplets into triangular ones, which are generated by δ. The

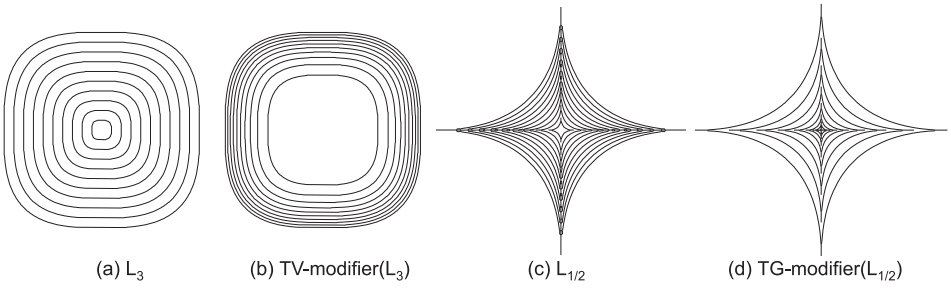

Fig. 13. Structure of distance density inside data regions (balls).

non-triangular triplets which are not generated by δ should remain non-triangular (the white areas in Figures 11a–d), since such triplets represent the "decisions" used by MAMs for filtering of non-relevant objects or classes. The more often such decisions occur, the more efficient the search is (and the better the indexability indicators are). As an example, given two vectors u, v of dimensionality n, the optimal "zero-error" TG-modifier (with respect to the entire universe $\mathbb{U}$) for semi-metric $\delta(u,v) = \sum_{i=1}^{n} |u_i - v_i|^2$ is $f(x) = \sqrt{x}$, turning δ into the Euclidean (L_2) distance (the f's "optimality" follows from the definition of L_2 distance).

From another point of view, the concavity/convexity of f determines how much the object clusters (MAMs' classes respectively) become loose/tight (overlapped by other clusters/classes).

In Figure 13a see an L_3-ball of radius $r = 1$, partitioned among 10 "onion-ring" regions. The k-th ring (counting from the center) represents the volume of points in the space which are within L_3 distance $\langle (k-1) * 0.1, k * 0.1 \rangle$ – we can call them *equi-distant rings*. In Figure 13b see how the rings change[11] if the L_3 distance is TV-modified by $f(x) = x^2$. Much of the volume is now gathered in the central ring, the respective points have distance to the center lower than 0.1. The remaining rings cover less and less of the space. That is, the respective DDH would contain tall low-distance bins and short high-distance bins, which leads to lower mean of distances, higher variance, and thus lower intrinsic dimensionality.

On the other side, in Figure 13c see a $L_{\frac{1}{2}}$-ball and in Figure 13d its modification by TG-modifier $f(x) = \sqrt{x}$. The effect is exactly opposite to the TV-modifier case, i.e. most of the volume gets distant ("inflated" to the ball surface[12]), so the mean of distances increases, variance decreases and the intrinsic dimensionality is higher.

When considering MAMs using ball partitioning (e.g. M-tree), the above example shows how the modifications impact the indexability and retrieval efficiency. The TG-modified data regions (balls) represent indistinct clusters of data, hence the indexing and searching will suffer from overlaps among them. Conversely, the TV-modified regions will tend to be disjoint (keeping the data distributed among tight separated clusters), hence the indexing becomes more efficient.

[11] By the way, we can observe the region shape does not change when modified by any SP-modifier. Moreover, in this case also the unitary ball radius has not changed, so the ball covers the same space volume as in the non-modified case.

[12] This corresponds to the effects of the "classic" curse of dimensionality.

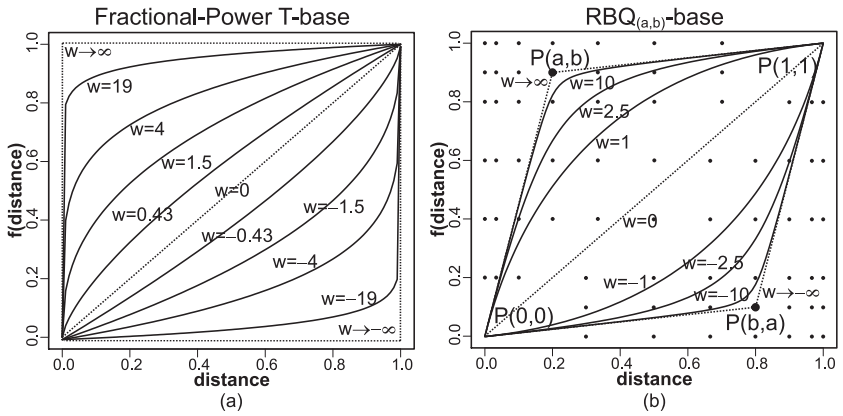

Fig. 14. Two T-base: (a) Fractional-power T-base (b) Rational-Bézier-quadratic T-base

6. AUTOMATED DISTANCE TRANSFORMATIONS

The question is how to find the optimal T-modifier f for a given measure δ, a dataset $\mathbb{S}$, and a T-error tolerance θ. Since we suppose no analytical form of δ is known, we have to obtain a guidance by sampling a number of distance triplets for a dataset sample $\mathbb{S}^* \subseteq \mathbb{S}$, and by use of some predefined T-modifier generators.

6.1 T-bases

In order to easily find a T-modifier which fits the required properties, we make use of so-called *T-base* – a T-modifier additionally parametrized by a concavity/convexity weight w.

Definition 8. (T-base)

Let $g : \langle 0,1\rangle \times \mathbb{R} \mapsto \mathbb{R}_0^+$ such that $g(x,0) = x$, $g(x,w)$ is a TG-modifier for $w > 0$ and it is a TV-modifier for $w < 0$, where w is called the *concavity-convexity weight* (CC-weight). Furthermore, if $w_1, w_2 > 0 \wedge w_1 > w_2$, then $g(x,w_1) > g(x,w_2), \forall x \in (0,1)$. Conversely, if $w_1, w_2 < 0 \wedge w_1 < w_2$, then $g(x,w_1) < g(x,w_2), \forall x \in (0,1)$. We also require g is continuous in sense $lim_{w_1 \to w_2} g(x,w_1) = g(x,w_2), \forall x \in \langle 0,1\rangle$. Then we call g a *base of T-modifiers* (or T-base). □

The parameter w will serve us to smoothly adjust a particular T-modifier in order to decrease/increase the concavity/convexity, so that the T-error will match the user-defined T-error tolerance. Simultaneously, among the T-modifications matching the T-error tolerance (the candidates), we pick the one which promises the best indexing performance (i.e. exhibits minimal intrinsic dimensionality or ball-overlap factor).

In Figure 14 see two T-bases – the *Fractional-power T-base* (FP-base), defined as

$$\mathrm{FP}(x,w) = \begin{cases} x^{\frac{1}{1+w}} & \text{for } w > 0 \\ x^{1-w} & \text{for } w \leq 0 \end{cases}$$

and the *Rational-Bézier-Quadratic T-base* (RBQ-base), defined as

$$\mathrm{RBQ}_{(a,b)}(x,w) = \begin{cases} rbq(x,w,a,b) & \text{for } w > 0 \\ rbq(x,-w,b,a) & \text{for } w \leq 0 \end{cases}$$

$$rbq(x,w,a,b) = \frac{-(\Psi - x + wx - aw)\cdot(-2bwx + 2bw^2x - 2abw^2 + 2bw - x + wx - aw + \Psi(1-2bw))}{(-1 + 2aw - 4awx - 4a^2w^2 + 2aw^2 + 4aw^2x + 2wx - 2w^2x + 2\Psi(1-w))}$$

where $\Psi = \sqrt{-x^2 + x^2w^2 - 2aw^2x + a^2w^2 + x}$.

The simpler FP-base has the advantage there always exists a CC-weight w for which either a modified semimetric becomes metric ($w > 0$) or a modified metric becomes semimetric ($w < 0$). Furthermore, when using the FP-base, the dissimilarity δ needs not to be d^+-bounded. A particular disadvantage of the FP-base is that its concavity/convexity is controlled globally, just by the CC-weight w. Hence, to fit a user-defined T-error tolerance, the FP-base has to be possibly "inflated" by a high CC-weight on the whole interval $(0, d^+)$, although a small highly "inflated" subinterval on the distance domain would suffice to fit the overall T-error tolerance, keeping the rest of the distance domain less inflated (and so achieving better indexability).

The RBQ-bases are more sophisticated T-bases, allowing to control the concavity/convexity locally, by the second Bézier point of the underlying rational Bézier quadratic curve. To derive a proper T-base from the curve[13], the three Bézier points are specified as $P_1(0,0), P_2(a,b), P_3(1,1)$. The additional RBQ parameters a, b (such that $a < b$) are treated as constants (the second Bézier point $P_2(a,b)$), i.e. for various a, b values we get multiple RBQ-bases (see the dots in Figure 14b), which are all individual T-bases. To obtain a regular T-base for any pair of values $a < b$, either a TG-modifier is used in case $w > 0$ (considering $P_2(a,b)$), or a TV-modifier is used in case $w < 0$ (considering $P_2(b,a)$). The advantage of RBQ-bases is the place of maximal concavity/convexity can be controlled locally by the choice of $P_2(a,b)$, hence, for a given CC-weight w we can achieve lower value of either an indexability indicator or the T-error just by choosing different $P_2(a,b)$.

As a particular limitation, for usage of RBQ-bases the distance δ must be d^+-bounded (due to the third Bézier point (1,1)). On the other side, the d^+ bound can be obtained when ensuring the reflexivity and non-negativity of δ, as mentioned in Section 4.1. Furthermore, for an RBQ-base with $P_2(a,b) \neq (0,1)$ the T-error could be generally greater than the T-error tolerance θ, even in case $w \to \infty$ (and conversely, even in case $w \to -\infty$ the modified dissimilarity can still remain metric or "insufficiently spoiled" semimetric). Nevertheless, with FP-base or $\mathrm{RBQ}_{(0,1)}$-base in our pool of T-bases, there can always be found a T-modifier which fits the T-error tolerance. In addition to the properties required in Definition 8, the FP- and RBQ- T-bases are also symmetric in sense a TG-modifier obtained for $w_1 > 0$ is inverse to the TV-modifier obtained for $w_2 = -w_1$, i.e. $g(x,w) = g^{-1}(x,-w)$.

[13]To keep the $rbq(x,w,a,b)$ evaluation correct, a possible division by zero or $\Psi^2 < 0$ is prevented by a slight shift of a or w.

6.2 The TriGen Algorithm

Given a pool of T-bases $\mathcal{F}$, searching for the optimal T-modifier can be automated by sliding the CC-weight of each T-base in the pool and keeping the best found so far.

LISTING 1. *(the generalized TriGen algorithm)*

Input: (semi)metric δ, pool $\mathcal{F}$ of T-bases, sample $\mathbb{S}^*$, T-error tolerance threshold θ, iteration limit iterLimit, number of sampled triplets m
Output: optimal f, w

```
f = w = null; maxIndexability = −∞                                                          1

T = SampleTriplets(m, S*, δ)                                                                2

for each f* in F                                                                            3
    w* = 0; iter = 0                                                                        4
    error = TriangleError(f*, w*, T)        // compute the initial error of non-modified δ, i.e. w* = 0   5
    if error < θ then                       // TV-modifiers will be used                    6
        w_LB = 0; w_UB = −∞; w_best = 0                                                     7
        err_best = error; w* = -1                                                           8
    else                                    // TG-modifiers will be used                    9
        w_LB = 0; w_UB = ∞; err_best = 0                                                    10
        w_best = -1; w* = 1                                                                 11
    end if                                                                                  12

    while (iter < iterLimit)                // the main algorithm – halving/doubling the weight w*   13
        error = TriangleError(f*, w*, T)                                                    14
        if w_UB > 0 then                                                                    15
            if error ≤ θ then                                                               16
                w_UB = w_best = w*                                                          17
                err_best = error;                                                           18
                w* = (w_LB + w_UB) / 2                                                      19
            else                                                                            20
                w_LB = w*                                                                   21
                if w_UB ≠ ∞ then                                                            22
                    w* = (w_LB + w_UB) / 2                                                  23
                else                                                                        24
                    w* = 2 * w*                                                             25
                end if                                                                      26
            end if                                                                          27
        else                                                                                28
            if error ≤ θ then                                                               29
                w_LB = w_best = w*                                                          30
                err_best = error;                                                           31
                if w_UB ≠ −∞ then                                                           32
                    w* = (w_LB + w_UB) / 2                                                  33
                else                                                                        34
                    w* = 2 * w*                                                             35
                end if                                                                      36
            else                                                                            37
                w_UB = w*                                                                   38
                w* = (w_LB + w_UB) / 2                                                      39
            end if                                                                          40
        end if                                                                              41
        iter++                                                                              42
    end while                                                                               43

    if (w_UB > 0 and w_best > -1) or (w_UB < 0 and w_best < 1) then // store the best (f,w) found   44
        w* = w_best                                                                         45
        indexability = ComputeIndexability(f*, w*, T)                                       46
        if indexability > maxIndexability then                                              47
            f = f*; w = w_best                                                              48
            maxIndexability = indexability                                                  49
        end if                                                                              50
    end if                                                                                  51
end for each                                                                                52
```

Since we know the more concave (convex) T-modifier, the lower (higher) the T-error, the search can be accelerated by halving a CC-weight interval (or doubling one of the bounds). In such a way we can quickly (in $O(log\ n)$ time) find the CC-weight for each T-base such that the respective T-modifier fits the T-error tolerance. Then, among the candidate T-modifiers, the one is picked which exhibits the best indexability indicator.

In Listing 1, see the *TriGen* algorithm which seeks for the best T-modifier, as outlined above. Initially, for each T-base the zero weight is checked whether the non-modified dissimilarity's T-error $\varepsilon_{\delta,\mathbb{S}}$ already fits the T-error tolerance θ. If so, the dissimilarity is being TV-modified until the T-error $\varepsilon_{\delta^f,\mathbb{S}}$ is about to exceed θ. If not, the dissimilarity is being TG-modified until the $\varepsilon_{\delta^f,\mathbb{S}}$ gets below or equal to θ. In such a way for each T-base we obtain (at most) one candidate for the best T-modifier. Finally, we pick such T-modifier among the candidates, which has the smallest intrinsic dimensionality (or ball-overlap factor).

To provide a scalability trade-off when searching for a candidate, the main loop is bounded by a fixed number of iterations (the iterLimit constant). The TriangleError method computes the T-error, having the actual T-modifier derived from a T-base (the sampling of distance triplets needed for the T-error evaluation is described in the next section). The ComputeIndexability method makes use of the previously sampled distance triplets (computed in the TriangleError method) and computes (negative value of) an indexability indicator on them (either intrinsic dimensionality or BOF).

6.3 Sampling of Distance Triplets

The distance triplets used for the T-error computation can be sampled randomly; this should suffice especially if $\theta > 0$. However, the random sampling could miss some "anomalous" ordered triplets (a, b, c), where the ratio $\frac{a+b}{c}$ is extremely low. Such a not-sampled anomalous triplet remains non-triangular even if we force the other ones (by the TriGen algorithm) to be triangular. Hence, if we require zero T-error (i.e. $\theta = 0$ for "really" exact search), we should sample also some of the anomalous triplets, in addition to the randomly sampled ones.

In Listing 2, a heuristic algorithm for sampling anomalous triplets is proposed. First, all pair-wise distances δ^f in a dataset sample $\mathbb{S}^*$ are computed and the respective object pairs are sorted according to their distances. Then, for each of the last tripletCount pairs of objects (O_k, O_l) in Ordering, the algorithm searches for the third object O_m such that for the respective ordered triplet (a, b, c) the ratio $\frac{a+b}{c}$ is minimized. To keep the tracking of the best O_m scalable, the number of performed attempts is limited.

It is maybe worth noting, the searching for anomalous distance triplets is a kind of analogy to searching for pivots when indexing by pivot-based methods (e.g. LAESA) [Bustos et al. 2003]. As good pivots are outliers in the metric space, the anomalous triples can be viewed as outliers in the "triplet space".

6.3.1 *Implementation Issues.* Initially, we have n objects in the dataset sample $\mathbb{S}^*$. Then we create an $n \times n$ distance matrix for storage of pair-wise distances $\delta_{ij} = \delta(O_i, O_j)$ between the sampled objects. In such a way we are able to obtain up to $\binom{n}{3}$ distance triplets for at most $\frac{n(n-1)}{2}$ distance computations. Thus, to

obtain a sufficiently large number of distance triplets, the dataset sample $\mathbb{S}^*$ needs to be quite small. Naturally, the values in the matrix could be computed "on-demand", just in the moment a distance is requested. Since δ is symmetric, the sub-diagonal half of the matrix can be used for storage of the T-modified distances $\delta^g_{ji} = g(\delta_{ij}(\cdot,\cdot), w)$, however, these are repeatedly recomputed for each particular $g(\cdot, w)$. As in case of distances, also the modified distances can be computed "on-demand".

LISTING 2. *(the SampleAnomalousTriplets algorithm)*

```
Input:   number of triplets to be sampled tripletCount, a dataset sample S*, distance function δ
Output:  set of triplets T
// step 1 - sort object pairs
allocate an array Ordering of pairs (i,j) such that i > j ∧ 1 ≤ i,j ≤ |S*|                    1
sort all pairs (i,j) in Ordering according to δ(O_i,O_j) in ascending order                   2
// step 2 - fetch objects (O_k, O_l) and find O_m such that the largest
//          distance in (δ(O_k,O_l), δ(O_l,O_m), δ(O_k,O_m)) is maximized
for (i = |Ordering|; i > |Ordering| − tripletCount; i−−)                                      3
    pair = Ordering[i]                                                                        4
    k = pair.first                                                                            5
    l = pair.second                                                                           6
    m = l      // start search of O_m from index l                                           7
    maxVariance = 0                                                                           8
    candidateM = m                                                                            9
    initialized = false                                                                      10
    for (j = 0; j < attempts; j++)     // attempts = number of attempts to find the best O_m 11
        if l < |S*|/2 then m++ else m−−                                                      12
        triplet = order(δ(O_k,O_l), δ(O_l,O_m), δ(O_k,O_m))                                  13
        if triplet.isRegular() then      // isRegular() is true if triplet.a > 0             14
            variance = triplet.c / (triplet.a + triplet.b)                                   15
            if not initialized                                                               16
                initialized = true                                                           17
                candidateM = m                                                               18
            end if                                                                           19
            if variance > maxVariance then                                                   20
                candidateM = m                                                               21
                maxVariance = variance                                                       22
            end if                                                                           23
        end if                                                                               24
    end for                                                                                  25
    add triplet (δ(O_k,O_l), δ(O_l,O_candidateM), δ(O_candidateM,O_k)) to T                  26
end for                                                                                      27
```

6.4 Time Complexity Analysis

Let $|\mathbb{S}^*|$ be the number of objects in the sample $\mathbb{S}^*$, m be the number of sampled triplets, $\mathcal{F}$ the pool of T-bases, and $O(\delta)$ be the complexity of single distance computation. The complexity of a single $g(x, w)$ computation is supposed $O(1)$. The overall complexity of TriGen algorithm is then

$$O(|\mathbb{S}^*|^2 * O(\delta) + iterLimit * |\mathcal{F}| * m)$$

i.e. the distance matrix computation plus the main algorithm.

The size of T-base pool $|\mathcal{F}|$ as well as the number of iterations (variable iterLimit) are assumed as (small) constants, hence we get

$$O(|\mathbb{S}^*|^2 * O(\delta) + m)$$

The size of $\mathbb{S}^*$ and the number m affect the precision of T-error and indexability indicator values, so we can trade off the TriGen's complexity and the precision by choosing $|\mathbb{S}^*| = O(1)$, $O(log|\mathbb{S}|)$, or $O(|\mathbb{S}|)$ and $m = O(1)$, $O(|\mathbb{S}^*|)$, or e.g. $O(|\mathbb{S}^*|^2)$.

7. EXPERIMENTAL RESULTS

To examine the proposed method, we have performed extensive testing of the TriGen algorithm as well as evaluation of the generated distance measures with respect to the effectiveness and efficiency of retrieval by two MAMs (the M-tree and PM-tree).

We would like to say this experimentation was not intended to compare the competitiveness of various access methods against the TriGen-aided MAMs. Although the specifically developed index structures will always perform better (e.g. the IGrid), they are limited just to a single dissimilarity measure (or a class of highly related measures). Instead, we would like to show the universality of the proposed approach on a general MAM (here on the M-tree and PM-tree), resulting in a fact the TriGen algorithm can be used as a common front end when implementing an effective and efficient multimedia retrieval system which is richly customizable (in sense of employment of any similarity measure provided by the user).

7.1 The TestBed

We have examined 26 dissimilarity measures (all described in Section 2) on four datasets (images, polygons, protein sequences and time series), while the distances were considered as black-box semimetrics. The dataset of images consisted of 65615 8-dimensional *Corel features* [Hettich and Bay 1999] (the color moments were used). We have tested 5 semimetrics and 4 metrics on the images: three fractional L_p distances ($p = 0.25, 0.5, 0.75$, denoted **L***p*), the 3-median L_2 distance (**3med L2**), the squared L_2 distance (**L2square**), three Minkowski L_p distances ($p = 1, 2, 5$, denoted **L***p*) and the angle distance (**Angle**).

As second we have sampled 50,000 strings of protein sequences (of lengths 50-100) from the *GenBank* file `rel147` [Benson et al. 2000]. The edit distance (denoted **Edit**) and the longest common subsequence distance (**LCSS**) were used to index the GenBank dataset. The LCSS similarity measure has been turned into a distance as $\text{LCSS}(\cdot,\cdot) = 100 - \text{LCSS}_{sim}(\cdot,\cdot)$.

Third, we created a synthetic dataset of 100,000 2D polygons, each consisting of 5 to 10 vertices. We have tested 2 semimetrics on the polygons: the 3-median and 5-median Hausdorff distances (denoted **3med Hausdorff L2**, **5med Hausdorff L2**, where the partial d distance was the L_2 metric on vertices), and the dynamic time warping distance (DTW) with d chosen as L_2 on vertices (denoted **DTWpoly**), and one metric: the Hausdorff distance (denoted **Hausdorff L2**).

Finally, we used 10,000 154-dimensional time series created by Keogh et al concatenating 10 diverse datasets from the UCR time series archive [Keogh and Ratanamahatana 2005]. The 10 datasets are foetal ecg, steam generator, space

shuttle, Photon Burst, Standard and Poor 500, ocean, power demand, leleccum, Koski ECG, and infrasound beams. The subsequences we used for our experiments were drawn at random from this pool, making sure that all 10 seed time series contribute equally. As distance functions we employed band-constrained DTW (denoted **DTW(b)**), where b is the half-width of Sakoe-Chiba band, i.e. $b = 0$ stands for L_2 distance and $b = 77$ is unconstrained DTW (with respect to 154 dimensions),i.e. for $b > 0$ the DTW is semimetric.

All the distances in all tests were normed to return values from $\langle 0, 1 \rangle$.

	T-error tolerance $\theta = 0.00$ winning T-modifier				T-error tolerance $\theta = 0.1$ winning T-modifier			
dissimilarity	T-base RGB(a,b)/FP	idim(ρ)	BOF	TG/TV	T-base RGB(a,b)/FP	idim(ρ)	BOF	TG/TV
3med L2	(0.005, 0.6)	58.8	0.96	TG	(0.075, 0.2)	3.95	0.015	TG
L2square	FP	3.47	0.057	TG	(0.015, 0.1)	2.39	0.015	TG
L0.25	(0.015, 0.55)	10.7	0.38	TG	(0.075, 0.1)	3.01	0.04	TG
L0.5	(0.075, 0.55)	6.84	0.17	TG	FP	2.71	0.047	TG
L0.75	(0.075, 0.15)	4.3	0.08	TG	(0, 0.05)	2.53	0.036	TV
L1	any	3.09	0.054	$w = 0$	any	3.09	0.054	$w = 0$
L2	(0, 0.05)	3.5	0.057	TV	(0, 0.05)	2.36	0.02	TV
L5	(0, 0.05)	3.71	0.056	TV	(0.015, 0.05)	2.77	0.024	TV
Angle	(0, 0.05)	3.11	0.044	TV	(0, 0.05)	2.21	0.018	TV
Edit	any	22.2	0.71	$w = 0$	(0, 0.5)	2.85	0.103	TV
LCSS	any	66.5	0.899	$w = 0$	(0, 0.65)	3.23	0.037	TV
3med Hausdorff L2	(0, 0.15)	4.26	0.054	TG	(0.155, 0.2)	2.24	0.0083	TG
5med Hausdorff L2	(0.005, 1)	73.48	0.798	TG	FP	2.26	0.0105	TG
DTWpoly	(0.015, 0.6)	12.16	0.0794	TG	(0.155, 0.3)	3.059	0.0143	TG
Hausdorff L2	any	2.347	0.015	$w = 0$	(0, 0.05)	2.23	0.0141	TV
DTW(0)=L_2	(0, 0.5)	5.3	0.055	TV	(0, 0.4)	3.6	0.0067	TV
DTW(2)	(0.015, 0.1)	20.70	0.40	TG	(0, 0.35)	3.976	0.004	TV
DTW(4)	(0.015, 0.1)	18.92	0.32	TG	(0, 0.35)	3.68	0.0037	TV
DTW(8)	(0.035, 0.2)	21.37	0.42	TG	(0, 0.3)	3.82	0.003	TV
DTW(20)	(0.075, 0.3)	21.61	0.61	TG	(0, 0.25)	3.50	0.0039	TV
DTW(40)	(0.075, 0.3)	18.90	0.58	TG	(0, 0.2)	3.47	0.0049	TV

Table 1. T-modifiers found by BOF-driven TriGen.

7.2 TriGen Setup

The TriGen algorithm was used to generate the optimal T-modifier for each dissimilarity measure (considering the respective dataset and a T-error tolerance θ). To examine the relation between the retrieval error of MAMs and the T-error, we have constructed several T-modifiers for each distance measure, considering different values of T-error tolerance $\theta \geq 0$. The TriGen's pool of T-bases $\mathcal{F}$ was populated by the FP-base and 116 RBQ-bases parametrized by all such pairs (a, b) that $a \in \{0, 0.005, 0.015, 0.035, 0.075, 0.155\}$, where for a value of a the values of b were multiples of 0.05 limited by $a < b \leq 1$. The dataset sample $\mathbb{S}^*$ used by TriGen consisted of only $n = 500$ randomly selected objects, i.e. 0.76% of Corel dataset, 1% of the GenBank dataset, 0.5% of the Polygons dataset and 5% of the Time series dataset. The distance matrix built on the respective dataset sample $\mathbb{S}^*$ was used to form $m = 10^5$ distance triplets. Unless otherwise stated, 95% of the triplets were sampled randomly and the remaining 5% were sampled by the "anomalous triplets" heuristics (see Section 6.3). The ball-overlap factor (BOF) was computed as BOF_1 (1-NN balls ratio) in all experiments.

In Table 1 see the optimal T-modifiers found for the dissimilarity measures by *BOF-driven* TriGen (i.e. a TriGen configuration where the BOF was used to rank

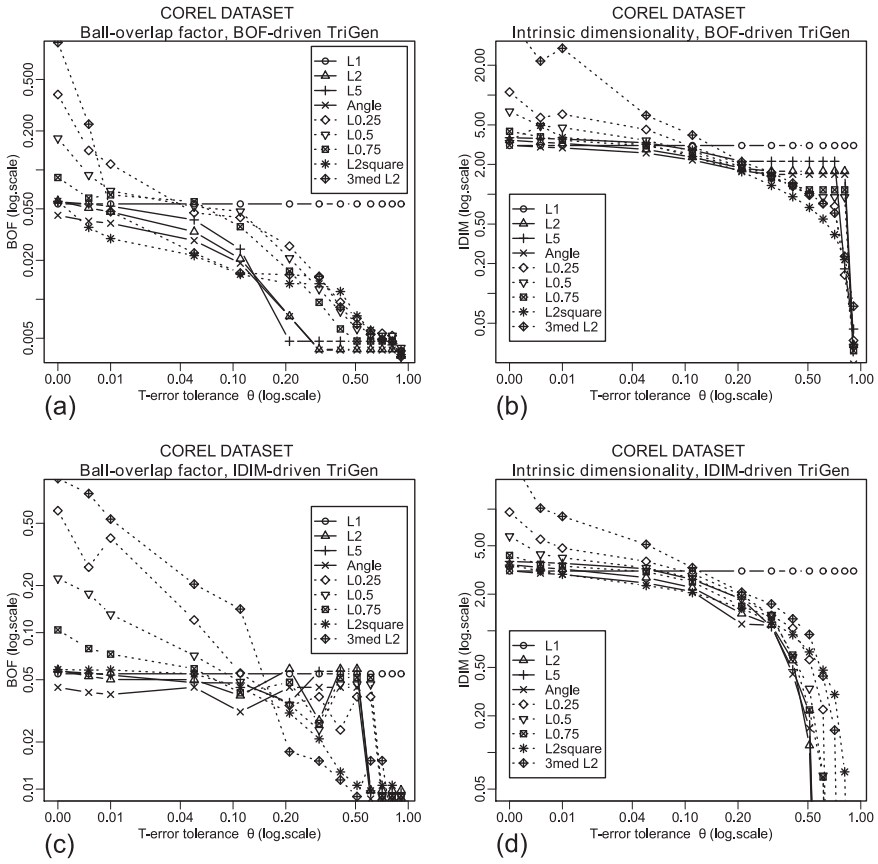

Fig. 15. (a,c) Ball-overlap factors of T-modifiers obtained by BOF- or IDIM-driven TriGen, respectively (b,d) Intrinsic dimensionalities of T-modifiers obtained by BOF- or IDIM-driven TriGen

the T-modifier candidates), considering $\theta = 0$ and $\theta = 0.1$, respectively. The winning T-modifier parameters for a particular T-error tolerance θ (the best in sense of lowest BOF) are denoted as RBQ/FP (either the (a, b) parameters of RBQ or just the FP label), idim (the intrinsic dimensionality), BOF (ball-overlap factor), TG/TV (the "direction" of the winning T-modifiers, i.e. either TG- or TV-modifier). We can observe the L_1 distance cannot be modified at all. Note that metrics **L2**, **L5**, **Angle** and **DTW(0)** were TV-modified without "harming" by the T-error. Furthermore, we can observe the semimetric **L0.75** is TG-modified to fit $\theta = 0$, but for $\theta = 0.1$ the triangle inequality is satisfied more than enough, so it is even TV-modified. As expected, the RBQ-bases are the most winning ones.

Considering the Corel dataset, in Figures 15a,b see the BOFs and intrinsic dimensionalities for BOF-driven TriGen with respect to the growing T-error tolerance θ, in Figures 15c,d see the opposite, i.e. BOFs and intrinsic dimensionalities for IDIM-driven TriGen.

The Figure 16a shows BOFs for BOF-driven TriGen on the GenBank dataset, while in Figure 16b see the same for Polygons dataset.

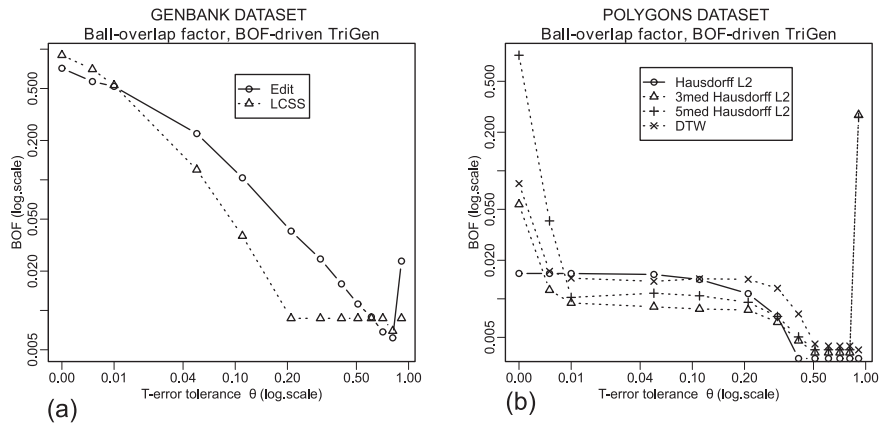

Fig. 16. BOFs of T-modifiers obtained by BOF-driven TriGen on (a) GenBank (b) Polygons

7.3 Indexing & Querying

In order to evaluate the efficiency and effectiveness of search when using T-modified dissimilarity measures, we have utilized the M-tree [Ciaccia et al. 1997] and the PM-tree [Skopal et al. 2005]. For either of the datasets several M-tree and PM-tree indices were built, which differentiate in the T-modification employed – for each dissimilarity measure and each θ value a δ^f was found by TriGen, and an index created. The setup of (P)M-tree indices is summarized in Table 2 (for technical details see [Ciaccia et al. 1997; Skopal et al. 2003; Skopal et al. 2005]).

disk page (tree node) size: 2–8 kB	avg. page utilization: approx. 68%
PM-tree pivots:	32 inner node pivots, 16 leaf pivots
Corel index sizes:	15 MB (M-tree) 20 MB (PM-tree)
Polygons index sizes:	14.5 MB (M-tree) 19 MB (PM-tree)
GenBank index sizes:	12.5 MB (M-tree) 15.2 MB (PM-tree)
Time series index sizes:	11.8 MB (M-tree) 12 MB (PM-tree)
Construction method:	MinMax + SingleWay

Table 2. M-tree and PM-tree setup

All the (P)M-tree indices were used to process kNN queries. Since the TriGen-generated modifications are generally approximations (especially when $\theta > 0$), the filtration of (P)M-tree branches was affected by a retrieval error E_{NO} (see Section 5.1).

To examine retrieval efficiency, the computation costs needed for query evaluation were compared to the costs spent by sequential search. Every query was repeated for 100 query objects (randomly selected from the dataset), and the results were averaged.

In Figure 17 see the costs and the retrieval errors of 10NN queries processed on Corel M-tree indices, depending on the growing θ and considering BOF-driven TriGen. Since the BOFs/IDIMs decrease, the searching becomes more efficient (e.g. down to 1% of costs spent by sequential search for $\theta = 0.2$ and T-modifications of metric L_p distances). On the other hand, for $\theta = 0$ the T-modifications of

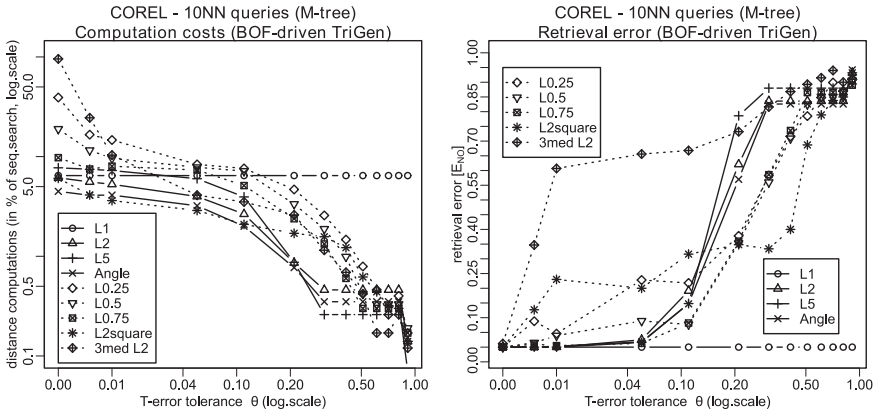

Fig. 17. Computation costs and retrieval errors observed for 10NN queries on M-tree Corel indices, considering BOF-driven TriGen generating of T-modifiers.

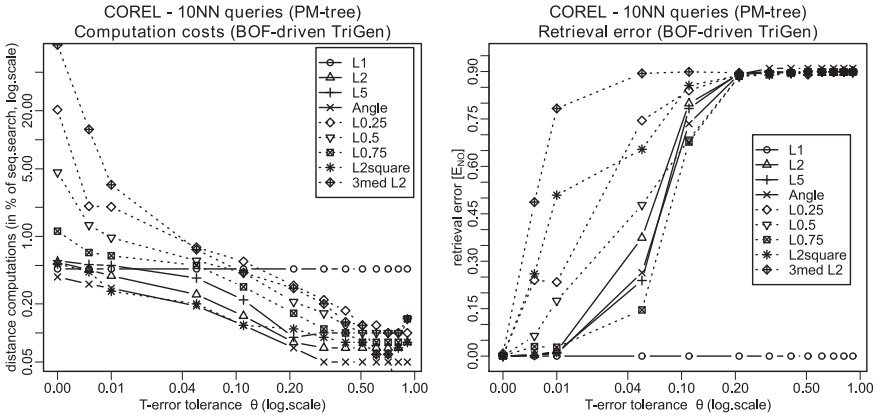

Fig. 18. Computation costs and retrieval errors observed for 10NN queries on PM-tree Corel indices, considering BOF-driven TriGen generating of T-modifiers.

3med L2 and **L0.25** imply high BOFs/IDIMs, so the retrieval deteriorates (to almost sequential search in case of **3med L2**). In Figure 18 the same results are presented for PM-tree indices. The results show PM-tree is sometimes an order of magnitude faster than the respective M-tree in the same situation.

For a comparison, in Figure 19 see the scenario from Figure 18 but now with indices built using T-modifications created by IDIM-driven TriGen (the results for L_1 indices are the same in both cases so we can use them as a clue). The BOF-driven TriGen produces T-modifications which perform better (up to $\theta = 0.2$). On the other hand, the T-modifications produced by BOF-TriGen lead to higher retrieval errors.

The costs and the errors for kNN querying on Corel M-trees are presented in Figures 20. The retrieval error is not considerably changing with increasing k, so we could expect stable behavior under various query selectivities.

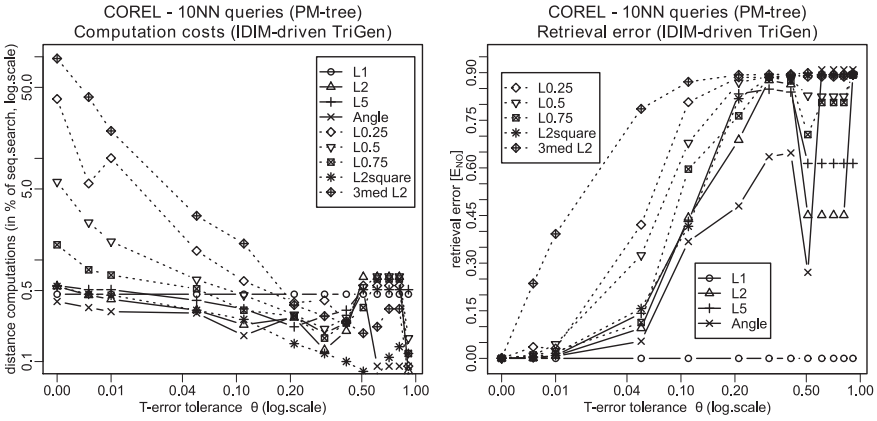

Fig. 19. Computation costs and retrieval errors observed for 10NN queries on PM-tree Corel indices, considering IDIM-driven TriGen generating of T-modifiers.

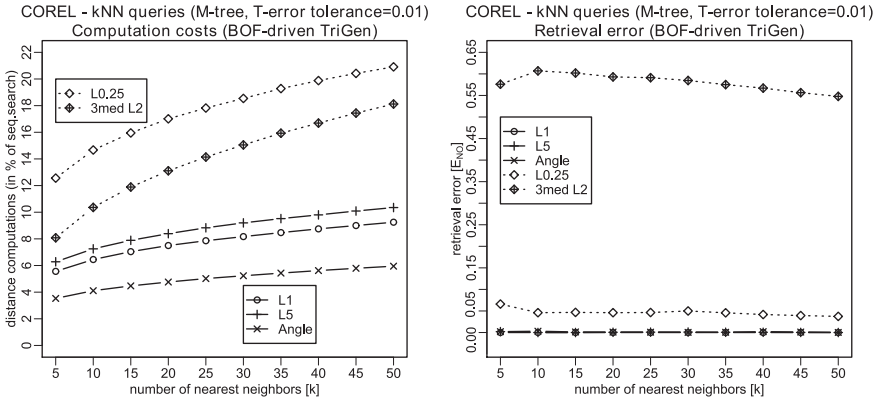

Fig. 20. Computation costs and retrieval errors for kNN queries on Corel M-tree indices.

To examine also the other datasets, in Figure 21 see the results for 10NN queries on GenBank indices, with respect to the growing θ. The edit distance has proved its bad exact indexability, more than 20% of PM-tree and almost 100% of M-tree indices was fetched to answer a query. Simultaneously, the respective retrieval errors are high for already small θ. Although a non-metric, the LCSS behaves similarly like the edit distance.

In Figure 22 see the results for Polygons indices. The indexing by **Hausdorff L2** distance is quite efficient, but also the respective retrieval errors for the growing θ are very small (even zero up to $\theta = 0.1$ when indexed by M-tree). The non-metric distances are quite efficiently indexable as well, the **5med Hausdorff L2** is an exception. On the other hand, the **DTWpoly** distance exhibits significant retrieval error even in case $\theta = 0$.

In Figure 23a,b see the computation costs (retrieval error, respectively) for 10-NN queries on the high-dimensional Time series dataset, where the observed parameter

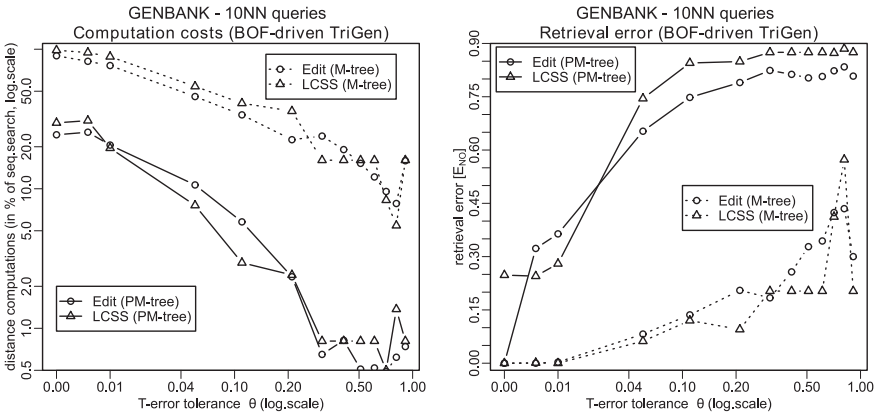

Fig. 21. Computation costs and retrieval errors observed for 10NN queries on (P)M-tree GenBank indices.

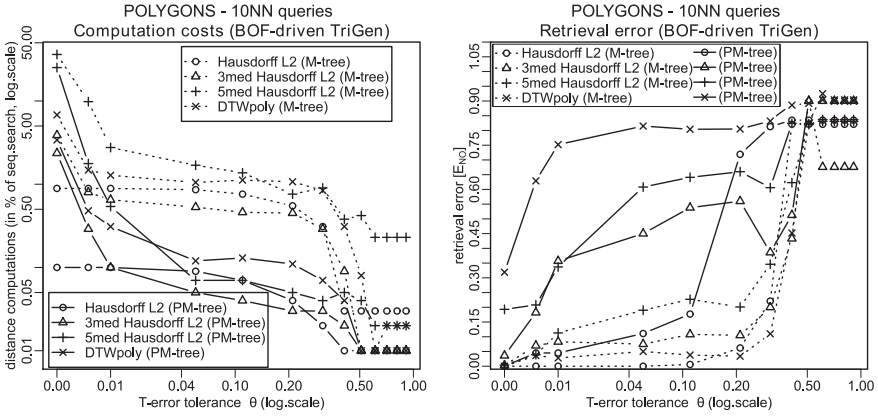

Fig. 22. Computation costs and retrieval errors observed for 10NN queries on (P)M-tree Polygons indices.

w is the half-width of the Sakoe-Chiba's band. The T-error tolerance was fixed to 0. For $w = 0$ the DTW becomes an ordinary L_2 metric, so the retrieval is fast, since no TG-modification is needed. However, for $w > 0$ the costs are much higher, no matter what actual value of w is. This observation is particularly interesting, since one would expect the higher w the higher T-error, and also the higher computation costs (enforced by a "heavy" TG-modification). Considering the T-error tolerance and **DTW(5)**, the computation costs and retrieval error on Time series are presented in Figure 24. Generally, the high-dimensional time series are hard to index for zero retrieval error, nevertheless, for T-error tolerance ≤ 0.01 the performance improves quite significantly, while the retrieval error is reasonably small.

7.3.1 *Aggregate Results.* To briefly summarize the overall indexability of all four datasets, in Figure 25 see the computation costs of 10NN queries directly against the real retrieval error (instead of just T-error tolerance). The four graph pairs in

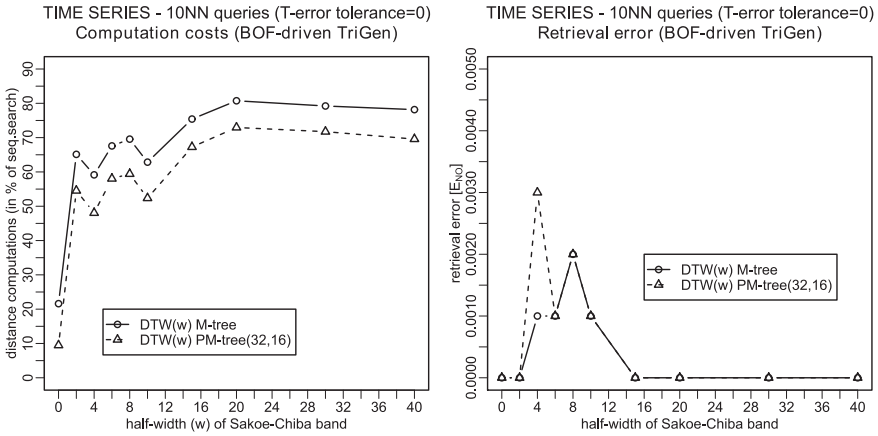

Fig. 23. Computation costs and retrieval errors for 10NN queries on (P)M-tree Time series indices, according to the growing half-width of Sakoe-Chiba band used in DTW. T-error tolerance set to zero.

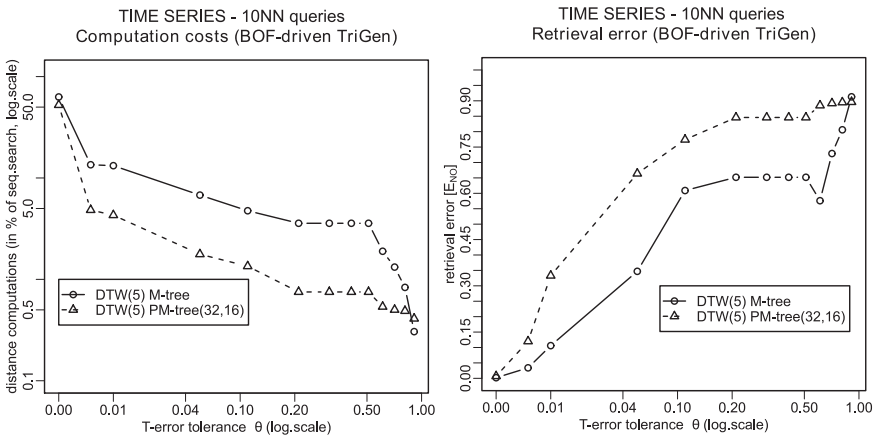

Fig. 24. Computation costs and retrieval errors observed for 10NN queries on (P)M-tree Time series indices.

Figures 18, 21, 22, 24 are re-plotted as four single graphs, showing the trade-off between efficiency and error for each dataset. The best values are the bottom-left ones, and vice versa.

7.3.2 *Sample size & anomalous triplets.* In the above results we can observe several cases where the retrieval error is high even for small or zero T-error tolerance θ (see **L0.25** in Figure 18, **3med L2** in Figure 20, **LCSS** in Figure 21, and **DTWpoly** in Figure 22). Hence, in the last experiment we test the impact of the triangular triplets' structure used by determining the T-error. In Figure 26 see computation costs and the retrieval errors according to the growing proportion of anomalous triplets in the set of 100,000 triplets used for T-error evaluation. We can observe the retrieval errors decrease, considering all the T-modified dissimilarity measures.

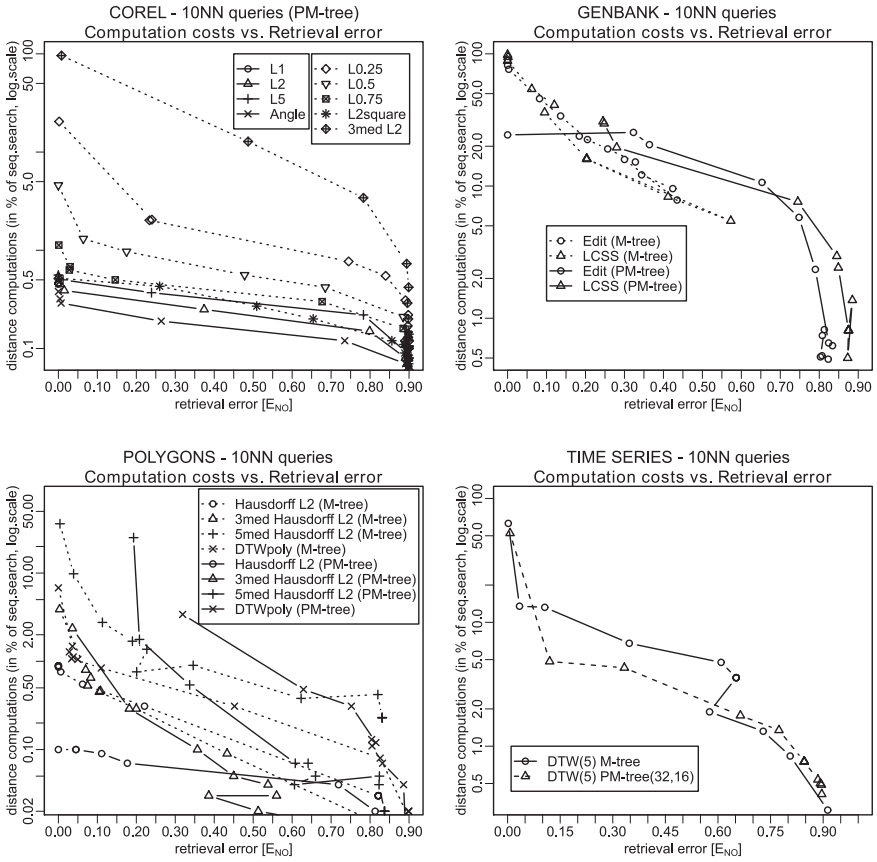

Fig. 25. Aggregate results for all datasets: 10NN queries, efficiency/error trade-off.

The drop is visible especially for the **L0.25**. However, at some point (here at 8% of anomalous triplets) the retrieval error gets stabilized, so we cannot further refine the T-error evaluation. This is due to the size of the dataset sample $\mathbb{S}^*$ and the distance triplet set, which are both relatively small. Thus, applications with requirements on a strictly guaranteed level of retrieval error have to use larger sizes of both.

To demonstrate the impact of the dataset size and the number of distance triplets, see Figure 27 to observe the dependency of retrieval error on the size $n = |\mathbb{S}^*|$ of the dataset sample (as proportion of the dataset size). The number of sampled distance triplets m was increasing super-linearly, as $m = n^{\frac{7}{4}}$. The error for **L0.25** is stable from approx. 2% (there is a nonzero T-error tolerance required). The error for **3med L2** steadily decreases as the sample size gets larger (up to 7% wherefrom the error is 0). We also observe the errors for **DTWpoly** and **LCSS** fall quickly, so the increased dataset sample in this case is better than an increased ratio of anomalous triplets (compare with the previous figure). Remarkably, for sample sizes above 2% of the dataset size we can observe the computation costs do not increase (actually, this is relevant for **L0.25** and **DTWpoly** since the other two got almost 100%).

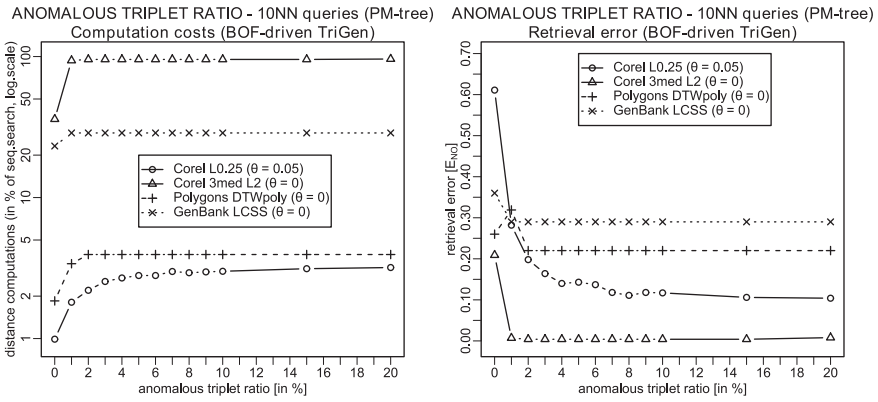

Fig. 26. An impact of increasing ratio of anomalous distance triplets sampled for the T-error evaluation.

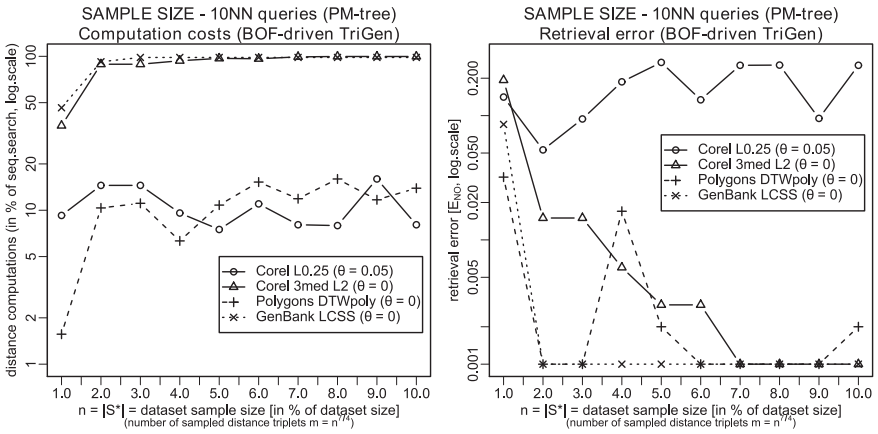

Fig. 27. An impact of size of dataset sample $\mathbb{S}^*$ as well the number of distance triplets on the retrieval error.

8. CONCLUSIONS

In this paper we have proposed a unified framework for exact and approximate search in general dissimilarity spaces by use of metric access methods (MAMs), which can be utilized especially in the area of multimedia retrieval. We have shown the triangle inequality property is not restrictive for similarity search and can be enforced for every semimetric (modifying it to a metric) in order to achieve exact search. Or, conversely, the triangle inequality can be intentionally violated for the sake of better structuring of the indexed dataset, so we achieve faster, but imprecise, retrieval. Furthermore, we have introduced the TriGen algorithm for automatic turning of any black-box dissimilarity into an approximation of metric just by use of distance distribution in a fraction of the database. Such a "TriGen-approximated metric" can be safely used to search the database by any MAM, while the similarity orderings with respect to a query object (the retrieval effectiveness)

are correctly preserved.

8.1 Future Work & Open Problems

We would like to carry out some research concerning further improvements of TriGen's "tracking abilities" for the optimal T-modifier. We consider two ways – utilizing more complex combinations of the T-bases, or, alternatively, extending the class of T-bases to all *partially* concave/convex SP-modifications.

Another of our efforts is aimed to a proposal of dynamic techniques for improvement of a T-modification which was already used to build a (P)M-tree index. With such a feature, the index could be repaired "on-the-fly" to achieve more or less approximate results for future queries, so full re-indexing of the dataset will not be required (at least to some extent).

Acknowledgments

This research has been supported in part by grants GAČR 201/05/P036 provided by the Czech Science Foundation, and "Information Society" grant number 1ET100300419.

REFERENCES

Aggarwal, C. C., Hinneburg, A., and Keim, D. A. 2001. On the surprising behavior of distance metrics in high dimensional spaces. In *ICDT*. LNCS, Springer.

Aggarwal, C. C. and Yu, P. S. 2000. The IGrid index: reversing the dimensionality curse for similarity indexing in high dimensional space. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM Press, New York, NY, USA, 119–129.

Amato, G., Rabitti, F., Savino, P., and Zezula, P. 2003. Region proximity in metric spaces and its use for approximate similarity search. *ACM Transactions on Information Systems 21,* 2, 192–227.

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. Y. 1998. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM (JACM) 45,* 6, 891–923.

Ashby, F. and Perrin, N. 1988. Toward a unified theory of similarity and recognition. *Psychological Review 95,* 1, 124–150.

Athitsos, V., Hadjieleftheriou, M., Kollios, G., and Sclaroff, S. 2005. Query-sensitive embeddings. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM Press, New York, NY, USA, 706–717.

Baeza-Yates, R. A. and Ribeiro-Neto, B. 1999. *Modern Information Retrieval*. Addison-Wesley Longman Publishing.

Bartolini, I., Ciaccia, P., and Patella, M. 2005. WARP: Accurate Retrieval of Shapes Using Phase of Fourier Descriptors and Time Warping Distance. *IEEE Pattern Analysis and Machine Intelligence 27,* 1, 142–147.

Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., Rapp, B. A., and Wheeler, D. L. 2000. Genbank. *Nucleic Acids Res 28,* 1 (January), 15–18.

Böhm, C., Berchtold, S., and Keim, D. 2001. Searching in High-Dimensional Spaces – Index Structures for Improving the Performance of Multimedia Databases. *ACM Computing Surveys 33,* 3, 322–373.

Bozkaya, T. and Özsoyoglu, M. 1999. Indexing large metric spaces for similarity search queries. *ACM Transactions on Database Systems 24,* 3, 361–404.

Brambilla, C., Ventura, A. D., Gagliardi, I., and Schettini, R. 1999. Multiresolution wavelet transform and supervised learning for content-based image retrieval. *icmcs 01*, 9183.

Brin, S. 1995. Near neighbor search in large metric spaces. In *Proceedings of the 21th International Conference on Very Large Data Bases*. Morgan Kaufmann Publishers Inc., 574–584.

Bustos, B., Keim, D., and Schreck, T. 2005. A pivot-based index structure for combination of feature vectors. In *Proc. 20th Annual ACM Symposium on Applied Computing, Multimedia and Visualization Track (SAC-MV'05)*. ACM Press, 1180–1184.

Bustos, B., Keim, D. A., Saupe, D., Schreck, T., and Vranic, D. V. 2005. Feature-based similarity search in 3d object databases. *ACM Computing Surveys 37,* 4, 345–387.

Bustos, B. and Navarro, G. 2004. Probabilistic proximity search algorithms based on compact partitions. *Journal of Discrete Algorithms 2,* 1, 115–134.

Bustos, B., Navarro, G., and Chávez, E. 2003. Pivot selection techniques for proximity searching in metric spaces. *Pattern Recognition Letters 24,* 14, 2357–2366.

Bustos, B. and Skopal, T. 2006. Dynamic Similarity Search in Multi-Metric Spaces. In *Proceedings of ACM Multimedia, MIR workshop*. ACM Press, 137–146.

Chávez, E. and Navarro, G. 2001. A Probabilistic Spell for the Curse of Dimensionality. In *ALENEX'01, LNCS 2153*. Springer, 147–160.

Chávez, E., Navarro, G., Baeza-Yates, R., and Marroquín, J. L. 2001. Searching in metric spaces. *ACM Computing Surveys 33,* 3, 273–321.

Ciaccia, P. and Patella, M. 2000. The M^2-tree: Processing Complex Multi-Feature Queries with Just One Index. In *DELOS Workshop: Information Seeking, Searching and Querying in Digital Libraries*. Zurich, Switzerland.

Ciaccia, P. and Patella, M. 2002. Searching in metric spaces with user-defined and approximate distances. *ACM Database Systems 27,* 4, 398–437.

Ciaccia, P., Patella, M., and Zezula, P. 1997. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *VLDB'97*. 426–435.

Corazza, P. 1999. Introduction to metric-preserving functions. *American Mathematical Monthly 104,* 4, 309–23.

Corboy, A., Raicu, D., and Furst, J. 2005. Texture-based image retrieval for computerized tomography databases. In *CBMS '05: Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05)*. IEEE Computer Society, Washington, DC, USA, 593–598.

Dohnal, V., Gennaro, C., Savino, P., and Zezula, P. 2003. D-index: Distance searching index for metric data sets. *Multimedia Tools and Applications 21,* 1, 9–33.

Donahue, M., Geiger, D., Liu, T., and Hummel, R. 1996. Sparse representations for image decomposition with occlusions. In *CVPR*. 7–12.

Faloutsos, C. and Kamel, I. 1994. Beyond uniformity and independence: analysis of r-trees using the concept of fractal dimension. In *PODS '94: Proceedings of the thirteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. ACM Press, New York, NY, USA, 4–13.

Faloutsos, C. and Lin, K. 1995. Fastmap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets. In *SIGMOD*.

Farago, A., Linder, T., and Lugosi, G. 1993. Fast nearest-neighbor search in dissimilarity spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence 15,* 9, 957–962.

Filho, R. F. S., Traina, A. J. M., Traina, C., and Faloutsos, C. 2001. Similarity search without tears: The OMNI family of all-purpose access methods. In *ICDE*.

Freeman, M. 2006. Evaluating dataflow and pipelined vector processing architectures for fpga co-processors. In *DSD '06: Proceedings of the 9th EUROMICRO Conference on Digital System Design*. IEEE Computer Society, Washington, DC, USA, 127–130.

Goh, K.-S., Li, B., and Chang, E. 2002. DynDex: a dynamic and non-metric space indexer. In *ACM Multimedia*.

Goldstein, J. and Ramakrishnan, R. 2000. Contrast plots and p-sphere trees: Space vs. time in nearest neighbour searches. In *VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 429–440.

Guo, G. D., Jain, A. K., Ma, W. Y., and Zhang, H. J. 2002. Learning similarity measure for natural image retrieval with relevance feedback. *IEEE Neural Networks 13,* 4, 811–820.

Hart, P. 1968. The condensed nearest neighbour rule. *IEEE Transactions on Information Theory 14,* 3, 515–516.

Hettich, S. and Bay, S. 1999. The UCI KDD archive [http://kdd.ics.uci.edu].

Hjaltason, G. R. and Samet, H. 2003. Properties of embedding methods for similarity searching in metric spaces. *IEEE Patt.Anal. and Mach.Intell. 25,* 5, 530–549.

Howarth, P. and Ruger, S. 2005. Fractional distance measures for content-based image retrieval. In *ECIR 2005*. LNCS 3408, Springer-Verlag, 447–456.

Huttenlocher, D., Klanderman, G., and Rucklidge, W. 1993. Comparing images using the hausdorff distance. *IEEE Patt. Anal. and Mach. Intell. 15,* 9, 850–863.

Jacobs, D., Weinshall, D., and Gdalyahu, Y. 2000. Classification with nonmetric distances: Image retrieval and class representation. *IEEE Pattern Analysis and Machine Intelligence 22,* 6, 583–600.

Jesorsky, O., Kirchberg, K. J., and Frischholz, R. 2001. Robust face detection using the hausdorff distance. In *AVBPA*. LNCS 2091, Springer-Verlag, 90–95.

Kao, D., Bergeron, R., and Sparr, T. 1997. Mapping metric data to multidimensional spaces, technical report tr 97-13, dept. of computer science, univ. of new hampshire.

Keogh, E. J. and Ratanamahatana, C. A. 2005. Exact indexing of dynamic time warping. *Knowledge and Information Systems 7,* 3, 358–386.

Krumhansl, C. L. 1978. Concerning the applicability of geometric models to similar data: The interrelationship between similarity and spatial density. *Psychological Review 85,* 5, 445–463.

Kruskal, J. B. 1964. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika 29,* 1, 1–27.

Li, C., Chang, E., Garcia-Molina, H., and Wiederhold, G. 2002. Clustering for approximate similarity search in high-dimensional spaces. *IEEE Transactions on Knowledge and Data Engineering 14,* 4, 792–808.

Li, H., Shi, R., Chen, W., and Shen, I.-F. 2006. Image tangent space for image retrieval. *18th IEEE International Conference on Pattern Recognition Vol 2*, 1126–1130.

Mandl, T. 1998. Learning similarity functions in information retrieval. In *EUFIT*.

Micó, M. L., Oncina, J., and Vidal, E. 1992. An algorithm for finding nearest neighbour in constant average time with a linear space complexity. In *Int. Cnf. on Pattern Recog.*

Mukherjee, A. 1989. Hardware algorithms for determining similarity between two strings. *IEEE Trans. Comput. 38,* 4, 600–603.

Nierman, A. and Jagadish, H. V. 2002. Evaluating structural similarity in XML documents. In *Proceedings of the Fifth International Workshop on the Web and Databases (WebDB 2002)*. Madison, Wisconsin, USA.

Rosch, E. 1975. Cognitive reference points. *Cognitive Psychology 7*, 532–47.

Roth, V., Laub, J., Buhmann, J., and Muller, K. 2002. Going metric: Denoising pairwise data, in nips02, 2002. submitted.

Rothkopf, E. 1957. A measure of stimulus similarity and errors in some paired-associate learning tasks. *J. of Experimental Psychology 53,* 2, 94–101.

Rubner, Y., Puzicha, J., Tomasi, C., and Buhmann, J. M. 2001. Empirical evaluation of dissimilarity measures for color and texture. *Comput. Vis. Image Underst. 84,* 1, 25–43.

Rubner, Y., Tomasi, C., and Guibas, L. J. 2000. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision 40,* 2, 99–121.

Sankoff, D. and Kruskal, J., Eds. 1983. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley, Reading, MA.

Santini, S. and Jain, R. 1999. Similarity measures. *IEEE Pattern Analysis and Machine Intelligence 21,* 9, 871–883.

Skopal, T. 2006. On fast non-metric similarity search by metric access methods. In *Proc. 10th International Conference on Extending Database Technology (EDBT'06)*. LNCS 3896. Springer, 718–736.

Skopal, T., Moravec, P., Pokorný, J., and Snášel, V. 2004. Metric Indexing for the Vector Model in Text Retrieval. In *SPIRE, Padova, Italy*. LNCS 3246, Springer, 183–195.

Skopal, T., Pokorný, J., Krátký, M., and Snášel, V. 2003. Revisiting M-tree Building Principles. In *ADBIS, Dresden*. LNCS 2798, Springer, 148–162.

Skopal, T., Pokorný, J., and Snášel, V. 2005. Nearest Neighbours Search using the PM-tree. In *DASFAA '05, Beijing, China*. LNCS 3453, Springer, 803–815.

Traina Jr., C., Traina, A., Seeger, B., and Faloutsos, C. 2000. Slim-Trees: High performance metric trees minimizing overlap between nodes. *Lecture Notes in Computer Science 1777*.

Tuncel, E., Ferhatosmanoglu, H., and Rose, K. 2002. Vq-index: an index structure for similarity searching in multimedia databases. In *MULTIMEDIA '02: Proceedings of the tenth ACM international conference on Multimedia*. ACM Press, New York, NY, USA, 543–552.

Tversky, A. 1977. Features of similarity. *Psychological review 84,* 4, 327–352.

Tversky, A. and Gati, I. 1982. Similarity, separability, and the triangle inequality. *Psychological Review 89,* 2, 123–154.

Uhlmann, J. K. 1991. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters 40,* 4, 175–179.

Volmer, S. 2002. Buoy indexing of metric feature spaces for fast approximate image queries. In *Proceedings of the sixth Eurographics workshop on Multimedia 2001*. Springer-Verlag New York, Inc., New York, NY, USA, 131–140.

Weber, R., Schek, H.-J., and Blott, S. 1998. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*.

Wilson, D. L. 1972. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics 2,* 3, 408–421.

Yi, B.-K., Jagadish, H. V., and Faloutsos, C. 1998. Efficient retrieval of similar time sequences under time warping. In *ICDE '98*. 201–208.

Yianilos, P. N. 1993. Data structures and algorithms for nearest neighbor search in general metric spaces. In *ACM SIAM SODA*. 311–321.

Zezula, P., Amato, G., Dohnal, V., and Batko, M. 2005. *Similarity Search: The Metric Space Approach (Advances in Database Systems)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Zezula, P., Savino, P., Amato, G., and Rabitti, F. 1998. Approximate Similarity Retrieval with M-Trees. *VLDB Journal 7,* 4, 275–293.

Zhou, X., Wang, G., Xu, J. Y., and Yu, G. 2003. M^+-tree: A New Dynamical Multidimensional Index for Metric Spaces. In *Proceedings of the Fourteenth Australasian Database Conference - ADC'03, Adelaide, Australia*.