Using the Real Dimension of the Data

Christian Zirkelbach
University Gh Kassel - Fachbereich 17
Heinrich-Plett-Str. 40
D-34109 Kassel , Germany
phone: (+49 561) 804 4101, fax: (+49 561) 804 4199,
e-mail: czi@db.informatik.uni-kassel.de

10th February 1999

Abstract

This paper presents a method for extracting the real dimension of a large data set in a high-dimensional data cube and indicates its use for visual data mining. A similarity measure structures a data set in a general, but weak sense. If the elements are part of a high-dimensional host space (primary space), for instance a data warehouse cube, the resulting structure doesn't necessarily reflect the real dimension of the embedded (secondary) space. We show that a metric-structured set has, in general, a fractal dimension. This means that the data set is a finite subset of a fractal secondary space of lower dimension. Mapping the set into the secondary space of lower dimension will not result in loss of information with regard to the semantics defined by the measure. However, it helps to reduce storage and computing efforts. Additionally, the secondary space itself reveals much about the set's structure and can facilitate data mining. The main problem with the secondary space is that it is unknown, and if it is not a linear sub-space of the $\mathbb{R}^d$, then there is no algorithm to determine it. We make a proposal for adding the property of a dimension to a metric and show that this is compatible to our customized understanding of a dimension. We present an algorithm which computes, in optimal time, an index on the data set. The index can be regarded as a materialized view for representing the structure of the elements with respect to the given metric. The algorithm works independently of the dimension of the primary space, i.e. pumping up the dimension of the primary space will not affect the result and the computational effort. Thus, besides supporting navigation and answering neighborhood queries, the algorithm also determines the real dimension of the underlying data set, even if the set is not linearly structured. We prove that the algorithm is robust with respect to skewed data or a distortion of the distance measure.

Keywords: Data Warehouse, OLAP, Data Mining, Indexing, Query processing

1 INTRODUCTION

Ten years ago, Richard E. Stamm, CEO of Knowledge Discovery Systems, stated that "similarity searching is an extremely useful technique for navigating among data that do not have a precise scheme for indexing". Unintentionally, he raised the issue of indexing data by means of similarity methods.

Within the research community, Nick Roussopoulos [20] recently pointed out the multi-faced form of views: "What is a relational view ? Is is a program ? Is it data ? Is it an index ? Is it an OLAP aggregate ?"

Here we propose to combine the properties of both notion. "Viewing" a data set under a metric results in a projection of those attributes that are relevant to the metric. On the other hand, it creates an algorithm and an access structure for navigating among data or to answer proximity queries with respect to the underlying metric. Data miners already make implicitly full use of this combination [15].

Initial suggestions reach back to the design of algorithms for geometric data-structures and an early attempt was the *Bisector Tree* of Kalantary and McDonald [11]. In this context the triangle inequality of a metric can be used for pruning or accepting complete clusters by comparing a query object with a representative element of the cluster. In doing so, the metric is used as a black-box.

This makes this approach universal in contrast to the various R-Tree derivatives. Applying this idea recursively results in hierarchical data-structures, so called *Metric Trees*, where each node represents a data cluster and is described by a representative (existing or artificial) element (called the *center of the cluster*) and a cluster radius defined by the maximum distance between the center and a cluster element. Whenever the queried cluster is too coarse (i.e. has a too large radius) to answer a neighborhood-query, the query is recursively passed down to the refined sub-clusters. The resulting tree structure may be generated on the fly or by preprocessing. Within this data structure, navigation over data spaces with aggregation is supported, as it is known from NF^2-tables [27]. This mechanism was published first in [17] and independently in [22] and the fact was stressed that, due to the general character of this type of data structures, a host of absolute and relative neighborhood queries can be supported by a single data structure.

In the context of data warehousing and multimedia, those ideas resurfaced [6], [5], [2], [28], [3]. Intensive theoretical and experimental studies were undertaken to establish a proper choice of cluster-centers in different metric spaces [18] of different application areas (for instance convex distance functions in $\mathbb{R}^d$, Waterman distance measure in sequence spaces, Mahalanobis-distance, Canberra-metric, distance measures of Sokal&Sneath and Czekanowski&Dice in non-discrete spaces, and the measures of Maron, Kulczynski, and Kendall in discrete spaces), in order to minimize distance computing by means of a chain estimation based on the triangulation inequality [26], and to develop a dynamic paging concept [24]. However, the later papers that appeared in the context of data warehouses did not draw much from these experiences, but proved the results to be true by their own benchmarks.

Besides choosing the cluster-centers in a way to keep the tree balanced, much attention was paid to choose cluster-centers that ensure a steep decrease of the cluster-radii which corresponds to a fast gain of locality or fineness. Additionally, the trade-off between those two goals was examined [29], [25].

As stated in [11], a precondition for small clusters is the separation of a father cluster by the bisector of the cluster-centers of the (two or more) son clusters. In the query process, besides checking a queried cluster against a ball (defined by the underlying metric) a queried cluster can now be additionally checked against a bisector. The bisector of two points is the separation sheet between the two areas that are closer to one point [9]. Thus, the bisector is the subset of points which are at the same distance from each of the two cluster-centers. There are three ways for handling point elements that are located on the separation-sheet: avoiding this situation by means of the selection algorithm [18], creating recursively a subtree of lower dimension [23], and storing those points redundantly. Whenever there are more than two points portioning the space, the bisector is generalized by the *Voronoi-diagram* [19].

The next result in this line of research stressed the point that it is advantageous for the set of (at least two) son-clusters to inherit the cluster center from the father cluster. This is a simple way to ensure that the radii of the son clusters decrease [8]. Otherwise this property is much harder to handle, because the decrease is not anchored within the tree structure and thus has to be guaranteed by the algorithm that chooses the son clusters. In other words, if the cluster-centers are inherited from the father cluster, the cluster-radii on a path from the root down to a leaf generate a monotonously decreasing sequence. Because of this property, one resulting data structure was called a *Monotonic Bisector Tree* [17].

Finally, there was an suggestion to choose those elements of the cluster that define the cluster-radius i.e. the center and its farthest element. Experimental results of this simple technique revealed that the decrease of the radii corresponds directly to the real dimension of the data set [24]. For instance, the cluster radii of points distributed on a one-dimensional curve in $\mathbb{R}^d$ decreased by a smaller factor q than those scattered arbitrarily in a 2-dimensional manifold. At the same time q is typical for sets of same dimension under same L_p-metric and is independent of the dimension d of the host space.

The main intension of this paper is to add theory to this observation and to prove that the decrease of the cluster radii is an indicator for the fractal dimension of a data set, which is independent of the dimension of the host-space. Thus, roughly speaking, a special type of Metric Tree can reveal the fractal dimension of a large data set in a warehouse cube of high dimension.

The remainder of this paper is organized as follows. After giving some formal definitions and properties in Chapter 2, we develop the basic idea in Chapter 3 and review briefly Monotonic Bisector Trees. Chapter 4 demontrates the support of relative and absolute neighborhood queries, combining principles of Metric Trees and Binary-Space-Partitioning (BSP) Trees [16]. Chapter 5 proves the strong relationship between the fractal dimension of a data set and its Monotonic Bisector Tree. Chapter 6 makes a proposal for applying the theory in visual data mining and Chapter 7 closes with a summary and an outlook on future activities. The paper is without proofs. Please note that the proofs can be found in the full version of this paper (http://www.db.informatik.uni-kassel.de/ czi/paper.ps).

2 Metric Spaces and Fractal Dimension

Let a *quasi-metric space* (M, d) be composed of a non-empty data set M and a similarity measure d: $M \times M \longrightarrow \mathbb{R}_{\geq 0}$ enhanced with the following properties:

1. $\exists_{C_1 \geq 1} \, \forall_{a,b \in M} \;\; d(a,b) \leq C_1 \, d(b,a)$ (Quasi-Symmetry)
2. $\exists_{C_2 \geq 1} \, \forall_{a,b,c \in M} \;\; d(a,c) \leq C_2 \, [d(a,b) + d(b,c)]$ (Quasi-Triangulation Inequality)
3. $\exists_{l \in \mathbb{N}} \, \forall_{a \in M} \forall_{r \geq 0} \, \exists_{\{a_1, \dots, a_l\} \subset M} \;\; K_r(a) \subset \bigcup_{1 \leq i \leq l} K_{\frac{r}{2}}(a_i)$ (Covering)

Thereby, for $z \in M$ and $r \geq 0$ let $K_r(z) := \{x \in M \mid d(z,x) \leq r\}$ denote the *ball* with radius r, centered at z. In order to estimate the runtime of our algorithms, we additionally demand

4. $\forall_{a,b \in M} \;\; d(a,b)$ can be computed in constant time (efficiently computable)

Using $C_1 = C_2 = 1$ provides a real metric. The quasi-properties in 1 and 2 are only for technical reason and will be used to quantify the robustness of our algorithms with respect to skewed data or a distortion of the distance measure. For this reason we define:

$$\varepsilon := \frac{1}{C_2 \, (C_1 + 1)}$$

In the metric case, $\varepsilon = \frac{1}{2}$ holds. We assume C_1, C_2 to be close to 1. Please note that $d(a,a) = 0$ is not explicitly demanded since a quasi-non-negativity

$$\forall_{a,b \in M} \;\; d(a,a) \leq C_2 \, (d(a,b) + d(b,a)) \;\leq\; C_2 \, (d(a,b) + C_1 \, d(a,b)) \;=\; \varepsilon^{-1} \, d(a,b)$$

is implied by 1 and 2, depending on the density of the underlying data set.

The coverage property reveals a fractal Dimension ($d = \log_2 l$ according to Mandelbrot's theory [14]) of the metric space. Intuitively we are familiar with this idea since a hypercube in $\mathbb{R}^d$ (i.e. a circle under L_∞-metric) with edge-length s can be covered with 2^d sub-cubes of edge-length $s/2$. Besides this approach, space-filling (fractal) curves have been already used for the refined stretching of the $\mathbb{R}^d$ [13], [1].

Lemma 1 *The four properties are independent.*

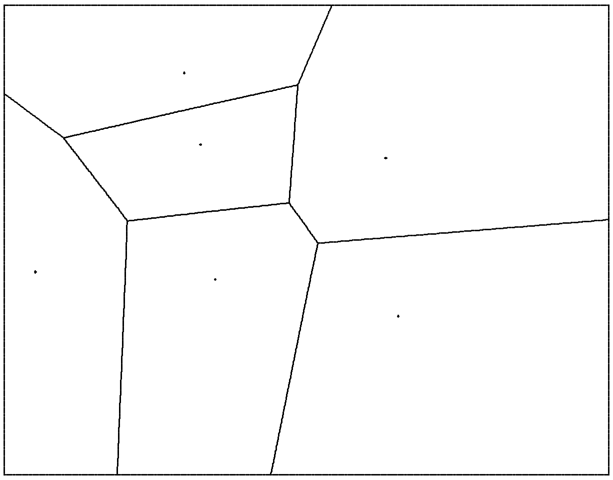

Figure 1: Voronoi-diagram of 6 points in the Euclidean plane

Definition 1 *Let $S \subset M$ be finite. For each $p \in S$ the set*

$$V_S(p) := \{x \in M \mid d(p,x) = 0 \ \vee \ \forall_{s \in S \setminus \{p\}} \ d(p,x) < d(s,x)\}$$

$$\bar{V}_S(p) := \{x \in M \mid \forall_{s \in S} \ d(p,x) \leq d(p,s)\}$$

is called the open (closed) Voronoi-Region of p with respect to S. For $a, b \in M$ the set

$$H(a,b) := \{x \in M \,|\, d(a,x) \leq d(b,x)\} = \bar{V}_{\{a,b\}}(a)$$

$$H^\circ(a,b) := \{x \in M \,|\, d(a,x) = 0 \ \vee \ d(a,x) < d(b,x)\} = V_{\{a,b\}}(a)$$

is called the open (closed) Half-space of a with respect to $\{a,b\}$. For $a, b \in M$ the bisector of a and b is defined:

$$B(a,b) := H(a,b) \cap H(b,a)$$

A bisector $B(a,b)$ separates *two sets $A, B \subset M$ iff $A \subset H(a,b)$ and $B \subset H(b,a)$.*

The set of all closed Voronoi-regions

$$VD(S) := \{\bar{V}_S(p) \mid p \in S\}$$

is called the Voronoi-diagram *of S. Figure 1 gives an example.*

The following lemma is easy to prove:

Lemma 2 *Let $S \subset M$ be finite and let $p \in S$. It holds:*

$$\bigcup_{s \in S} \bar{V}_S(s) = M \quad \text{and} \quad \bar{V}_S(p) = \bigcap_{s \in S} H(p,s)$$

Lemma 3 *Voronoi-regions are invariant with respect to a strictly monotonic transformation (i.e. a transformation of d to $d_\Phi := \Phi \circ d$, using a a strict monotonic mapping $\Phi : \mathbb{R}_{\geq 0} \longrightarrow \mathbb{R}_{\geq 0}$, $\Phi(0) = 0$) of the distance measure d.*

Lemma 3 proofs the partitioning of (M, d) to be invariant under a class of distance measures. Additionally, we have to examine the impact of a monotonic transformation on the topological constants (C_1, C_2 on page 3).

Lemma 4 *Applying a strictly monotonic increasing transformation $d_\Phi := \Phi \circ d$ on d will not worsen (heighten) C_1, C_2 if*

$$\frac{\Phi}{id} : \mathbb{R}_{>0} \longrightarrow \mathbb{R}_{>0}, x \longmapsto \frac{\Phi(x)}{x}$$

is monotonically decreasing, whereby l can not improve (lessen). Conversely, l will not worsen and C_1, C_2 will not improve if $\frac{\Phi}{id}$ is monotonically increasing.

Besides proving a robustness with respect to skewed data, the last two lemmata demonstrate, how a carefully chosen, neighborhood-respecting distortion of the distance measure (for instance an acceleration) can be used to improve the properties of the measure. This might be of interest while experimenting with different distance functions.

Definition 2 *Let a* Cluster *be a tuple (S, z), defined by the* cluster center *$z \in M$ and the finite* base set *$S \subset M$. For each (S, z) let*

$$r(S, z) := \max\{d(z, s) \mid s \in S\}$$

denote the radius*. Let a k-clustering ($k \in \mathbb{N}$) of $S \subset M$ denote the set $\{(S_1, z_1), \ldots, (S_k, z_k)\}$, such that $\bigcup_{1 \le i \le k} S_i = S$ and $S_i \cap S_j = \emptyset$ for $i \neq j$.*

With respect to the distance measure, a center acts as a representative of a cluster and the radius quantifies this relationship. In this context, the following theorem proves the superiority of a bisector partitioning.

Theorem 1 *For every k-Clustering $\{(S_1, z_1), (S_2, z_2), \ldots, (S_k, z_k)\}$ of a finite set $S \subset M$ into k clusters with centers $z_1, \ldots, z_k \in M$ and radii $r_1, \ldots, r_k$ there is a k-Clustering of S into k clusters $\{(S'_1, z_1), (S'_2, z_2), \ldots, (S'_k, z_k)\}$ with radii $r'_1, \ldots, r'_k$ based on the Voronoi-diagram of S with the following feature:*

$$\max_{1 \le i \le k} r'_i \;\le\; \max_{1 \le i \le k} r_i$$

3 An Index Based on a Hierarchical Clustering Method

At first, we point to a simple observation: Let the distance measure d be any L_p-metric. Imagine a finite set S of points scattered in the one-dimensional interval $[-1, 1]$ and let $0, 1 \in S$. Taking the 0 for the cluster center we gain a cluster $(S, 0)$ with radius $r(S, 0) = 1$. Next, we partition S by means of the bisector of 0 and 1, yielding two sub-clusters $(S_0, 0)$ and $(S_1, 1)$ with radii $r_0 = 1$ and $r_1 = 0.5$ where $S_1 = \{s \in S \,|\, d(0, s) \le d(1, s)\}$ and $S_2 = \{s \in S \,|\, d(0, s) > d(1, s)\}$. In the consecutive step S_0 is split by the bisector of its farthest point in S_0 which results in radii $r_{00}, r_{01} \le 0.5$. Applying this simple partitioning method recursively in a depth-first-search manner, provides a reduction of the actual cluster-radius by factor 0.5 after at most two consecutive separation steps. After mapping $[0, 1]$ into the plane (for instance by $x \longmapsto (x, sinx)$) this method results in the same reduction rate. But, after scattering the points of S arbitrarily in the plane (making full use of the two dimensions), there are about 8 consecutive steps necessary (depending on the used p) to halve the radii. Mapping the points into a fractal of dimension 1.x (for instance by $x \longmapsto (x, sin1/x)$) results in a value between both thresholds. In order to examine this observation, we reintroduce a data structure which is directly implied by the mechanism mentioned above.

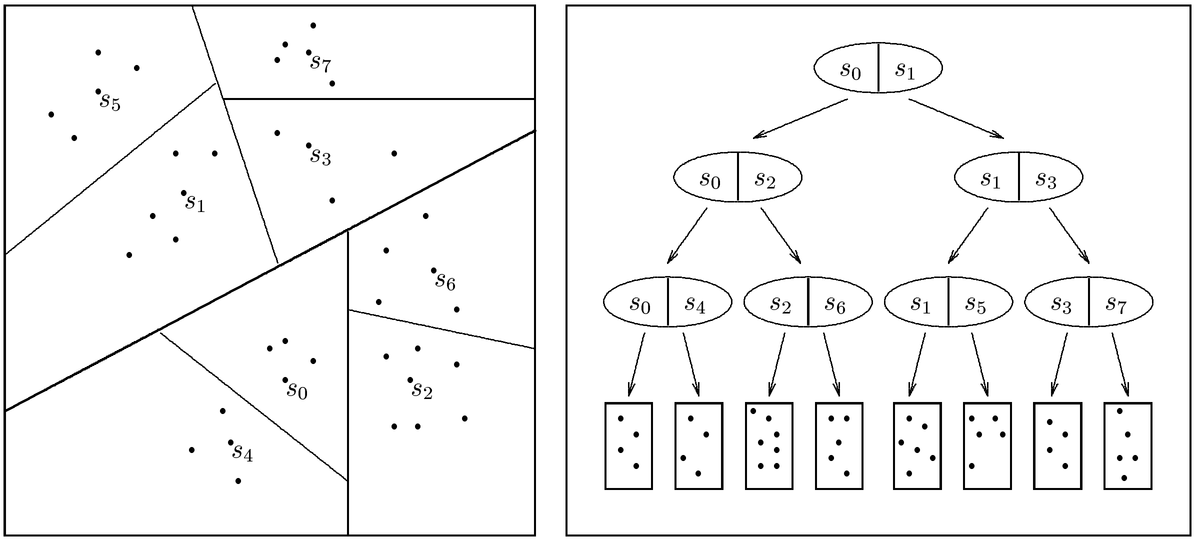

Figure 2: Monotonic Bisector Tree in the Euclidean plane

Definition 3 *Let (S, s_0) be a cluster where $S \subset M$ and $s_0 \in S$. Let a: $\mathcal{P}(S) \times S \longrightarrow \{0,1\}$ be a so called* truncation function, *equipped with the following properties:*

$$\forall_{s \in S} \forall_{S' \subset S} \; |S'| \leq 2 \implies a(S', s) = 1 \quad \textit{and}$$

$$\forall_{s \in S} \forall_{S'' \subset S' \subset S} \;\; a(S', s) = 1 \implies a(S'', s) = 1.$$

A Monotonic-Bisector-Tree *$MBT(S, s_0)$ is a rooted binary tree having the following features:*

1. *If $a(S, s_0) = 1$, then $MBT(S, s_0)$ consists of a single node containing the elements of S.*
2. *If $a(S, s_0) = 0$, then $MBT(S, s_0)$ is a binary tree with root w. w contains s_0 and $s_1 \in S \setminus \{s_0\}$. The sons of w are the sub-roots w_0 and w_1, corresponding to the two subtrees $MBT(S_0, s_0)$ and $MBT(S_1, s_1)$, where for $i \in \{0, 1\}$ holds:*

$$S_i \cap S_{1-i} = \emptyset \quad \wedge \quad S \cap H^\circ(s_i, s_{1-i}) \subset S_i \subset S \cap H(s_i, s_{1-i})$$

In this context, the radius *of a $MBT(S, s)$ denotes the radius of the corresponding cluster (S, s). The* radius *of a node denotes the radius of the corresponding subtree. The clusters given by the leaf-nodes of the tree (buckets) are called* terminal clusters.

Obviously, this data structure stores $n + 1$ different points in n nodes. Thus, the storage needed is linear in $|S|$. Figure 2 gives an example. Please note, that the redundancy implied by the inheritance of cluster-centers is just of logical nature and can be physically avoided by storing only the new center in a node and using a simple stack operation to retrieve the cluster center of an ancestor node.

The terminal function a determines the fineness of the decomposition of (S, s_0) based on $|S|$ and $r(S, s)$. We assume in the following that $a(S, s)$ can be computed in $\mathcal{O}(|S|)$ time and storage. We call a Monotonic Bisector Tree *completely developed*, iff $a(S, s) = 1 \implies |S| \leq 2$ holds.

Lemma 5 *MBTs do not have eccentric sons. The radii of the elements stored in nodes which appear on a path from the root to a leaf generate a monotonously decreasing sequence (justifying the term: monotonic tree).*

Up to now, the initial cluster center of a MBT was taken for granted. This is not the general case. The trivial determination of the best (with smallest radius) center results in $\mathcal{O}(|S|^2)$ distance computations. This might be unbearable in practice. The Computational Geometry community has brought forth faster algorithms which work in special spaces [19], [9]. On the other hand, however, the deviation caused by a randomly chosen initial center is bounded, since the triangulation inequality holds:

$$d(x,y) \leq C_2\,(d(x,z_{opt}) + d(z_{opt},y)) \leq C_2\,(C_1\,d(z_{opt},x) + d(z_{opt},y)) \leq \frac{r_{opt}}{\varepsilon}$$

where $x, y \in S$ and z_{opt} denotes the best initial center with radius r_{opt}.

A coarse index built-up algorithm, called Algorithm 1, is now outlined. In order to map this primary storage structure into secondary storage (hard-disk), Verbarg developed a paging concept based on the celltree [24]. It supports dynamic operations (insert, update, delete) efficiently.

```
TYPE  nodeptr       =  ↑nodetype
      nodetype      =  RECORD
                          p0,p1: point
                          radius: real
                          l_son,r_son: nodeptr, leafptr
                       END
TYPE  leafptr       =  ↑term_cluster
      term_cluster  =  RECORD
                          z: point
                          radius: real
                          set: set_of_point
                       END
VAR   w, w'         :  nodeptr
VAR   c             :  leafptr
```

Algorithm 1

FUNCTION MBTconstruct(VAR S: set_of_point, VAR s: point): nodeptr, term_cluster
BEGIN
    $r := \max\{d(s,x) \mid x \in S\}$
    IF $a(S,s) = 0$ *THEN*
        $s_0 := s$
        Select $s_1 \in S$ *such that* $d(s_0, s_1) = r$
        $S_0 := \{x \in S \mid d(s_0,x) \leq d(s_1,x)\}$ *(** $= S \cap H(s_0,s_1)$ **)*
        $S_1 := \{x \in S \mid d(s_0,x) > d(s_1,x)\}$ *(** $= S \cap H^\circ(s_1,s_0)$ **)*
        new(w)
        WITH $w\uparrow$ *DO*
            BEGIN
                $p0 := s_0$
                $p1 := s_1$
                $radius := r$
                $l\_son :=$ *MBTconstruct*(S_0, s_0)
                $r\_son :=$ *MBTconstruct*(S_1, s_1)
            END
        MBTconstruct $:= w$

```
ELSE (* (S, s) is a terminal cluster *)
        new(c)
        WITH c↑ DO
            BEGIN
                z := s
                radius := r
                set := S
            END
        MBTconstruct := c
    ENDIF
END
```

The following lemma is easy to prove by means of the master theorem [7]:

Lemma 6 *Let (S, s) be a cluster. The invocation root := $MBTconstruct(S, s)$ returns the root of $MBT(S, s)$ requiring $\mathcal{O}(|S|)$ storage. This still holds if s_1 is arbitrarily chosen from $S \setminus \{s\}$.*

4 Querying the Index

Roughly speaking, there are relative and absolute neighborhood queries. A relative neighborhood query, for instance, is the nearest-neighbor-query (i.e. searching for the nearest neighbor of a query item). An absolute neighborhood query is the fixed-radius-near-neighbor query (i.e. searching for all data items whose distance form a given query point is less than a fixed threshold value).

We can use the triangle inequality of a metric for pruning or accepting complete clusters by comparing a query point with a representative element of the cluster or with the bisector of two cluster centers. This reduces the number of distance computations. Thus, a MBT supports the BSP-like pruning mechanism as well as those of Metric trees. Whenever the radius of the queried cluster is too big to answer the query across the board, the query is recursively reached down to the corresponding subtrees. Now, we apply this idea exemplarily on the already mentioned examples.

Let (S, z) be a cluster represented by a MBT. In order to handle the case of noise in data we additionally define:

$$\begin{aligned} r_1(S,z) &:= \max\{d(z,s) \mid s \in S\} = r(S,z) \\ r_2(S,z) &:= \max\{d(s,z) \mid s \in S\} \end{aligned}$$

The checks for pruning ore accepting are in detail:

1. Fixed-radius-near-neighbor-search

   After giving $q \in M$ and $R \geq 0$ we search the set $\{s \in S \mid d(q,s) \leq R\}$.

   1.1 Accepting

   For each $s \in S$

   $$\begin{aligned} d(q,s) \leq C_2\ (d(q,z) + d(z,s)) &\leq C_2\ (d(q,z) + r_1(S,z)) \\ &\leq C_2\ (d(q,z) + C_1\, r_2(S,z)) \end{aligned}$$

   holds. Thus, (S, z) can be accepted, if

   $$\boxed{C_2\ (d(q,z) + r_1(S,z)) \leq R \quad \text{or} \quad C_2\ (d(q,z) + C_1\, r_2(S,z)) \leq R}$$

Searching for $\{s \in S \mid d(s,q) \leq R\}$, we have to check:

$$\boxed{C_2\ (d(z,q) + r_2(S,z)) \leq R \quad \text{or} \quad C_2\ (d(z,q) + C_1\, r_1(S,z)) \leq R}$$

If d is a metric, the check is

$$\boxed{d(z,q) + r(S,z) \leq R}$$

1.2 Pruning

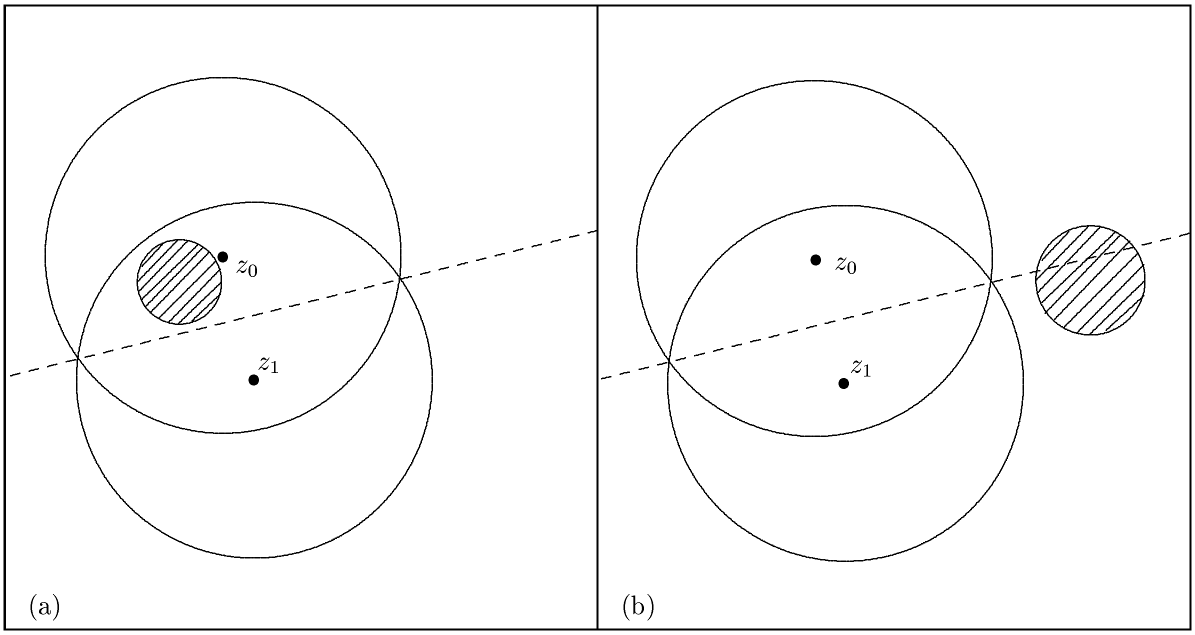

Figure 3: Pruning of subtrees

Here, we can check against the cluster or the bisector.

(a) Checking against the cluster (S,z):

For every $s \in S$

$$\begin{aligned} d(q,s) \geq \frac{d(q,z)}{C_2} - d(s,z) \quad &\geq \quad \frac{d(q,z)}{C_2} - r_2(S,z) \\ &\geq \quad \frac{d(q,z)}{C_2} - C_1\, r_1(S,z) \end{aligned}$$

holds. Thus, we can prune (S,z), if

$$\boxed{\tfrac{d(q,z)}{C_2} - r_2(S,z)\ > R \quad \text{or} \quad \tfrac{d(q,z)}{C_2} - C_1\, r_1(S,z)\ > R}$$

Searching for $\{s \in S \mid d(s,q) \leq R\}$, we have to check:

$$\boxed{\tfrac{d(z,q)}{C_2} - r_1(S,z)\ > R \quad \text{or} \quad \tfrac{d(z,q)}{C_2} - C_1\, r_2(S,z)\ > R}$$

If d is a metric, the check is

$$\boxed{d(q,z) - r(S,z)\ > R}$$

(b) Checking against the bisector:

Let (S, z) be split into the sub-clusters $(S_0, z_0), (S_1, z_1)$. For each $i \in \{0, 1\}$ and $p \in \{s \in S \mid d(q, s) \leq R\}$

$$d(z_i, p) \leq C_2\ (d(z_i, q) + d(q, p)) \quad \leq \quad C_2\ (d(z_i, q) + R)$$

and

$$d(z_i, p) \geq \frac{d(z_i, q)}{C_2} - d(p, q) \quad \geq \quad \frac{d(z_i, q)}{C_2} - C_1\, R$$

holds. Thus, we prune (S_i, z_i), if

$$\boxed{C_2\ (d(z_{1-i}, q) + R) < \tfrac{d(z_i, q)}{C_2} - C_1\, R}$$

holds. Searching for $\{s \in S \mid d(s, q) \leq R\}$, the check is:

$$\boxed{C_2\ (d(z_{1-i}, q) + C_1\, R) < \tfrac{d(z_i, q)}{C_2} - R}$$

If d is a metric, the check is

$$\boxed{d(z_{1-i}, q) + R < d(z_i, q) - R}$$

Note: If the bisector is explicitly known, this condition may be improvable by checking the query range directly against the bisector.

2. Nearest-neighbor-search

After giving $q \in M$ we search for the set $\{s \in S \mid \forall_{x \in S}\ d(q, s) \leq d(q, x)\}$. The method is straightforward, but now we check against the shrinking region: $\{s \in S \mid d(q, s) \leq R_{cur}\}$, where $R_{cur} > 0$ denotes the currently best match. We start with $R_{cur} = \infty$.

(a) Checking against the current Cluster (S, z):
Searching for a closer candidate we prune (analogical to 1.2 (a)) (S, z), if

$$\boxed{\tfrac{d(q,z)}{C_2} - r_2(S, z)\ > R_{cur} \quad \text{oder} \quad \tfrac{d(q,z)}{C_2} - C_1\, r_1(S, z)\ > R_{cur}}$$

holds. Searching for $\{s \in S \mid \forall_{x \in S}\ d(s, q) \leq d(s, x)\}$, the check is:

$$\boxed{\tfrac{d(z,q)}{C_2} - r_1(S, z)\ > R_{cur} \quad \text{oder} \quad \tfrac{d(z,q)}{C_2} - C_1\, r_2(S, z)\ > R_{cur}}$$

If d is a metric, the check is

$$\boxed{d(q, z) - r(S, z)\ > R_{cur}}$$

(b) Checking against the bisector: Let (S, z) be split into the sub-clusters $(S_0, z_0), (S_1, z_1)$. As stated in case 1.2 (b), we prune (S_i, z_i), if

$$\boxed{C_2\ (d(z_{1-i}, q) + R_{cur}) < \tfrac{d(z_i, q)}{C_2} - C_1\, R_{cur}}$$

holds. If we search for $\{s \in S \mid \forall_{x \in S}\ d(s, q) \leq d(s, x)\}$, the check is

$$\boxed{C_2\ (d(z_{1-i}, q) + C_1\, R_{cur}) < \tfrac{d(z_i, q)}{C_2} - R_{cur}}$$

If d is a metric, the check is

$$\boxed{d(z_{1-i}, q) + R_{cur} < d(z_i, q) - R_{cur}}$$

5 Discovering the Fractal Dimension

Theorem 2 *For each $0 < q < 1$ there is a $l := l(q) \in I\!N$, such that algorithm 1 creates on every cluster (S, s) with $S \subset M$ and center $s \in S$ a MBT with the following feature:*

$$\textit{For every node } v \textit{ of height } h \textit{ it holds:} \quad r(v) \leq q^{\frac{h}{l}}\, r(S, s)\,.$$

Thereby the required storage is $\mathcal{O}(|S|)$.

In summary we can say that this simple algorithm forces the cluster-radii of a MBT which appear on a path from the root down to a leaf to generate a geometrically decreasing sequence. Due to the fact that this holds for any host space, the decreasing rate is depending of the covering feature of the host space with lowest dimension, in general a fractal dimension. Now, after observing the reduction rate $l(\varepsilon^2 q)$, we can compute this dimension:

$$\boxed{d = \log_{1/\varepsilon^2 q} \left(l(\varepsilon^2 q) + 1\right)}$$

Although this statement is the basic intension of the paper, we additionally state some features which ease the handling of a MBT.

Now we are faced with the problem, that the number of cluster centers that can be computed is bounded (c-means-clustering [4]). Of course, we vary algorithm 1 and store the current cluster not in a lrw-stack, but in a max-heap. Thus, the partitioning step selects recursively the leaf-cluster with biggest radius (maximum error method). This modified algorithm is denoted $1'$. Theorem 2 reveals a trade-off between the balance of the tree and the quality of the resulting clusters, since for every leaf-node it holds:

$$q^{\frac{h-1}{l}}\, r(S, s) \geq r(vater(v)) \geq \max\{r(b) \mid b \text{ is a leaf}\}$$

The next theorem examines the resolution of a MBT.

Theorem 3 *Let (S, s) be the initial cluster. If the truncation function $a\colon \mathcal{P}(S) \times S \longrightarrow \{0, 1\}$ satisfies*

$$\frac{r(S, s)}{r(S', s')} \geq C \geq 1 \quad \Longrightarrow \quad a(S', s') = 1$$

the height of the resulting tree can be estimated $h \in \mathcal{O}(\log C)$.

The next theorem proofs the asymptotic optimal performance of the built-up algorithms of a MBT with respect to its height:

Theorem 4 *Algorithm 1 and $1'$ need $\mathcal{O}((1 + h)\,|S|)$ time to create a MBT on a cluster (S, s) with height h .*

Because the height of a MBT is bounded by its cardinality $|S|$, we derive from theorem 4 and theorem 3:

Theorem 5 *Let $C \geq 1$ be given. Algorithm 1 and $1'$ need $\mathcal{O}(\min\{\log C + 1, |S|\}\,|S|)$ time for the creation of a MBT on (S, s) such that $\frac{r(S,s)}{r(b)} \geq C$ holds for any leaf b.*

In order to ease the experimental handling of different distance measure we state the last theorem.

Theorem 6 *MBTs created by algorithm 1 and $1'$ are invariant with respect to a monotonic transformation of the distance measure.*

6 User Interface for Visual Data Mining

Based on the concept of coarsen and refinement, we suggest the visualization of a MBT for interactive visual data mining. Visualization is based on the development of a MBT with the help of algorithm 1 or $1'$ until a threshold value for the number of leaf nodes is reached. This can be controlled by a suitable truncation function a, which the user can adjust.

Next, the resulting tree is mapped onto the screen: The cluster center of the root is associated with the middle of the window, the origin. The sub-clusters are arranged as discs in circular order around the origin. There are three degrees of freedom to be assigned: The size of a disc, its distance from the origin and the position of a disc in circular ordering around the origin. We suggest to associate the volume (fatness) of a disc to its significance (for instance the cardinality of the corresponding cluster) and to order the discs around the origin with respect to the corresponding cluster radius. The distance of a discs from the origin may reflect its distance from the center of the root. In this context, the distance of two cluster may be defined by the distance of the centers or the min/max-distance of the corresponding sets. Of course, the assignment of quantitative tree features to the three degrees of freedom is within the user.

The interaction is carried out by a generalization of the finger concept [27] as it is known from the navigation in NF^2-tables. For meaningful illustrations of NF^2-tables, compare [10] and [12]. A finger is a cursor for navigating in complex (structured) objects. Instead of a linear cursor, it offers three dimensions for incremental positioning: zoom in/out, next/last item of a list, next/last attribute within a record. Here, however, the hierarchy is of quantitative nature instead of a NF^2-tables qualitative nature. Mouse clicks and double-clicks on the discs can be used to roll-up/drill-down or zoom/unzoom the corresponding cluster. The circular order of the discs and the fatness of the discs can be orthogonally used to activate a "successing" cluster by pressing an arrow-key.

7 Summary and Outlook

In order to overcome rigid R-Tree derivatives, we have decided to index our space by means of a metric. We have shown that a simple Metric-tree-technique can be used to discover the real dimension of a large data set and to support proximity queries. We made a proposal to apply the finger technique on this data structure in order to support visual data mining. We demonstrated, that this approach is an alternative to plane projections, because the navigator can make full use of the real dimension of a data set and, by using polar coordinates, can choose a third degree of freedom. The main advantage is that the navigation works sensitive with respect to the given metric.

We have already implemented the finger concept on NF^2-tables by using Tcl/Tk [21]. Now, we plan to expand this user interface to handle metric navigation as described in the last section. We will also try to make use of a metric as a compact interface for mining within a distributed environment. Instead of a SQL-query the metric itself is passed down from a client to its servers. The metric will be transmitted to the servers in the form of a PL/SQL- or a SQLJ-function.

References

[1] T. Asano, D. Ranjan, T. Roos, and E. Welzl. Space filling curves and their use in the design of geometric data structures. *Lecture Notes in Computer Science*, 911:36–??, 1995.

[2] Tolga Bozkaya and Meral Ozsoyoglu. Distance-based indexing for high-dimensional metric spaces. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 26(2):357–??, may 1997.

[3] S. Brin. Near neighbor search in large metric spaces. In Umeshwar Dayal, Peter M. D. Gray, and Shojiro Nishio, editors, *VLDB '95: proceedings of the 21st International Conference on*

Very Large Data Bases, Zurich, Switzerland, Sept. 11–15, 1995, pages 574–584, Los Altos, CA 94022, USA, 1995. Morgan Kaufmann Publishers.

[4] Robert L. Cannon, Jitendra V. Dave, and James C. Bezdek. Efficient implementation of the fuzzy C-means clustering algorithms. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-8(2):248–255, March 1986.

[5] P. Ciaccia, M. Patella, and P. Zezula. Processing complex similarity queries with distance-based access methods. *Lecture Notes in Computer Science*, 1377:9–??, 1998.

[6] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases*, pages 426–435, 1997.

[7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, Mass., 1996.

[8] Frank K. H. A. Dehne and Hartmut Noltemeier. Voronoi trees and clustering problems. *Information Systems*, 12(2):171–175, 1987.

[9] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer, Berlin, 1987.

[10] A. Heuer and G. Saake. *Datenbanken: Konzepte und Sprachen.* Thomson Publishing, Bonn, 1995.

[11] G. McDonald I. Kalantari. A data structure and an algorithm for the nearest point problem. *IEEE Transactions on Software Engineering*, 9(5), 1983.

[12] G. Lausen and G. Vossen. *Objektorientierte Datenbanken: Modelle und Sprachen.* Oldenbourg, München, 1996.

[13] K.-J. Li and R. Laurini. The spatial locality and a spatial indexing method by dynamic clustering in hypermap systems. In Oliver Guenther and Hans-Joerg Schek, editors, *Proceedings of Advances in Spatial Databases (SSD '91)*, volume 525 of *LNCS*, pages 207–224, Berlin, Germany, August 1991. Springer.

[14] Benuoit B. Mandelbrot. *The Fractal Geometry of Nature.* W. H. Freeman and Company, New York, 1983.

[15] Gholamreza Nakhaeizadeh and Alexander Schnabl. Development of multi-criteria metrics for evaluation of data mining algorithms. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, page 37. AAAI Press, 1997.

[16] Bruce F. Naylor and William C. Thibault. Application of BSP trees to ray tracing and CSG evaluation. *Tech. Rep. GIT-ICS 86/03*, February 1986. true BSP-trees (not just generalized octrees) used for efficiency.

[17] H. Noltemeier, T. Roos, and C. Zirkelbach. Partitioning of complex scenes of geometric objects. In P. Kall, editor, *Proceedings of the 15th IFIP Conference on System Modelling and Optimization*, volume 180 of *LNCIS*, Berlin, Germany, September 1991. Springer.

[18] H. Noltemeier, K. Verbarg, and C. Zirkelbach. Monotonous bisector trees : A tool for efficient partitioning of complex scenes of geometric objects. In Burkhard Monien and Thomas Ottmann, editors, *Proceedings of Data Structures and Efficient Algorithms*, volume 594 of *LNCS*, pages 186–203, Berlin, Germany, September 1992. Springer.

[19] F. P. Preparata and M. I. Shamos. *Computational Geometry : An Introduction.* Springer-Verlag, 1985.

[20] N. Roussopoulos. Materialized views and data warehouses. *SIGMOD Record*, 27(1):21–26, March 1998.

[21] Jens Thamm and Lutz Wegner. What you see is what you store: Database-driven interfaces. In *Proceedings of the 4th IFIP Working Conference on Visual Database Systems, L'Aquila, Italy May 27–29, 1998*, 1998. To appear.

[22] Uhlmann. Satisfying general proximity / similarity queries with metric trees. *IPL: Information Processing Letters*, 40, 1991.

[23] G. Vanecek and V. Ferrucci. A spatial index for convex simplical complexes in d dimensions. *Lecture Notes in Computer Science*, 525:361–??, 1991.

[24] K. Verbarg. Räumliche indizes - celltrees. Master's thesis, University Würzburg, 1992.

[25] Knut Verbarg. Approximate center points in dense point sets. *Information Processing Letters*, 61(5):271–278, ???? 1997.

[26] T. L. Wang and D. Shasha. Query processing with distance metrics. In *Proc. Int'l. Conf. on Very Large Data Bases*, page 602, Brisbane, Australia, August 1990.

[27] L. M. Wegner. Let the fingers do the walking: Object manipulation in an NF^2 database editor. In Hermann Maurer, editor, *Proceedings of New Results and New Trends in Computer Science*, volume 555 of *LNCS*, pages 337–358, Berlin, Germany, June 1991. Springer.

[28] Pavel Zezula, Paolo Ciaccia, and Fausto Rabitti. M-tree: A dynamic index for similarity queries in multimedia databases. Technical Report TR96-0002, Progetto Mediterraneo, CNR-Pisa, 1996.

[29] C. Zirkelbach. *Repräsentatives Clustern - ein metrischer Ansatz*. PhD thesis, University Würzburg, 1992.