Chapter 7

Similarity Based Retrieval

Motivation (1)

- In data base retrieval the access to the desired data is done by presenting a certain key.
- In information retrieval systems also a key word is presented. The problem with this is that one gets either no answer (silence) or very many answers (noise).
- There are several situations in E-C where no exact queries can be raised, e.g.:
  - Which is the best available product with respect to customer demands ?
  - Which is the best profile covering a given customer ?
  - What is the most likely reason for the customer‘s complaint ?

Motivation (2)

- The task of similarity based retrieval is to deal with such problems by allowing inexact queries. They are ultimately transferred into an exact one (e.g. a data base query).
- The answer is only an approximate one which hopefully close to an optimal answer. The similarity concepts allows to restrict the number of answers as one wants.
- The properties of a retrieval function are closely connected to the structure of the base.
- In more general search situations retrieval functions are called as black-box functions.

Efficient Retrieval

- Effiency and accuracy are the two important issues for retrieval.
- The efficiency of the different retrieval methods depends very much on the following:
  - The representation of the objects
  - The base structure
  - The similarity measure
  - The accuracy of the intended answer
- These characteristics depend on the other hand on the domain and the specific properties of the application.

Efficient Retrieval

- Central task of the retrieval:
  - given:
    - A base CB = $\{F_1,...,F_n\}$ (a case base, a product base, ...) and a similarity measure sim
    - Query: Q (new problem, new product etc.)
  - wanted either:
    1. The most similar object F_i OR
    2. the *m* most similar objects $\{F_1,...,F_m\}$ (ordered or unordered) OR
    3. All objects $\{F_1,...,F_m\}$ which have to Q at least a similarity sim_{min}
- Problem: How to organize a case base for efficient retrieval?

Retrieval Methods

Typ	*Method*	*Retriction w.r.t. Similarity*	*Appropriate for ...*
brute force	**sequential search**	no	small case bases simple similarity
index based	**kd-tre** *(Wess et al., 1993)*	reflexivity, monotony no class similarity	small number of attributes large case bases
	Fish & Shrink *(Schaaf, 1996)*	reflexivity, nonotony triangle inequality	complex similarity small case bases
	Retrieval Nets *(Burkhard & Lenz, 1996)*	monotony, no class similarity	few numeric attributes large case bases
dyn. database queries	**SQL Approximation** *(Schumacher & Bergmann, 2000)*	monotony no class similarity linear aggregation functions	simple similarity large case bases dynamic case bases

Sequential Retrieval

Data structures

```
Types:
    SimObject = RECORD
                    object: object;
                    similarity: [0..1]
                END;
    SimOjectQueue = ARRAY[1..m] OF SimObject;
Variables:
    scq: SimObjectQueue          (* m most similar objects *)
    cb: ARRAY [1..n] OF object   (* Object base *)
```

Retrieval algorithm

```
scq[1..m].similarity := 0
FOR i:=1 to n DO
    IF sim(Q,cb[i]) >  scq[m].similarity
      THEN insert cb[i] in scq RETURN scq
```

Properties of Sequential Retrieval

- Complexity of sequential retrieval: O(n)
- Disadvantages:
  - Problems if base very large
  - Retrieval effort is independent of the query
  - Retrieval effort is independent of m
- Avantages:
  - Simple implementation
  - No additional index structures needed
  - Arbitrary similarity measures can be used

Two Level Retrieval

- Idea: Two levels

  Mac/Fac (Many are called; Few are chosen)

  – 1. Preselection of possible candidates $M_Q \subseteq CB$

    $M_Q = \{F \in CB \mid SIM(Q,F)\}$

  – 2. Ordering of the (few) candidates according to sim

    Application of sequential retrieval

- Problem: To define the predicate SIM

Examples for Preselection Predicates

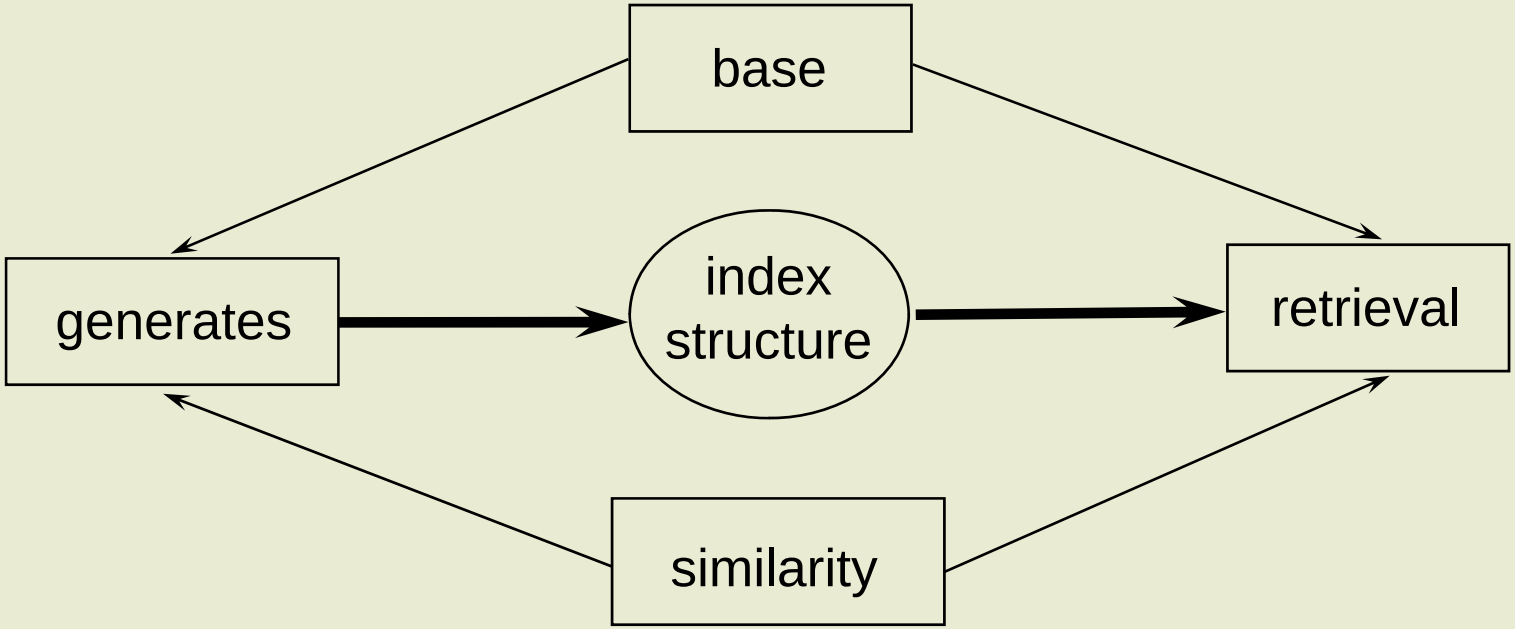
- Partial equality:
  - SIM(Q,F) if Q and F coincide in at least one attribute value
  - Local similarity
  - SIM(Q,F) if Q and F are sufficiently similar with respect to each local measure
  - Partial local similarity
  - SIM(Q,F) if Q and F are sufficiently similar with respect to one local measure
- User defined preselection is often more efficient and correct

Properties of Two Level Retrieval

- Advantages:
  - more effient if only few objects are preselected
- Disadvantages:
  - retrieval errors possible, i.e.
  - α-errors: Sufficiently similar objects may not be preselected
    $\rightarrow$ retrieval may be incomplete
  - The definition of the predicate SIM is usually difficult

Index Oriented Retrieval

- Preprocessing: Generates an index structure
- Retrieval: Makes use of the index structure for efficient retrieval



Retrieval with kd-Trees *(S. Wess)*

- k-dimensional binary search tree (Bentley, 1975).
- Idea: Decompose data (i.e. case base) iteratively in smaller parts
- Use a tree structure
- Retrieval:
  - Searching in the tree top-down with backtracking

Definition: kd-Tree

- Given:
  - k ordered domains $T_1,...,T_k$ of the attributes $A_1,...,A_k$,
  - a base $CB \subseteq T_1 x ... x T_k$. and
  - some parameter b (bucket size).

  A *kd-tree* T(CB) for the base CB is a binary tree recursively defined as:
  - if $|CB| \leq$ b: T(CB) is a leave node (called *bucket*) which is labelled with CB
  - if |CB| > b: T(CB) then T(CB) is a tree with the properties:
    - the root is labelled with an attribute A_i and a value $v_i \in T_i$
    - the root has two kd-trees$T_{\leq}$ ($CB_{\leq}$) and $T_{>}$($CB_{>}$) as successors
    - where $CB_{\leq} := \{(x_1,...,x_k) \in CB \mid x_i \leq v_i\}$ and
      $CB_{>} := \{(x_1,...,x_k) \in CB \mid x_i > v_i\}$

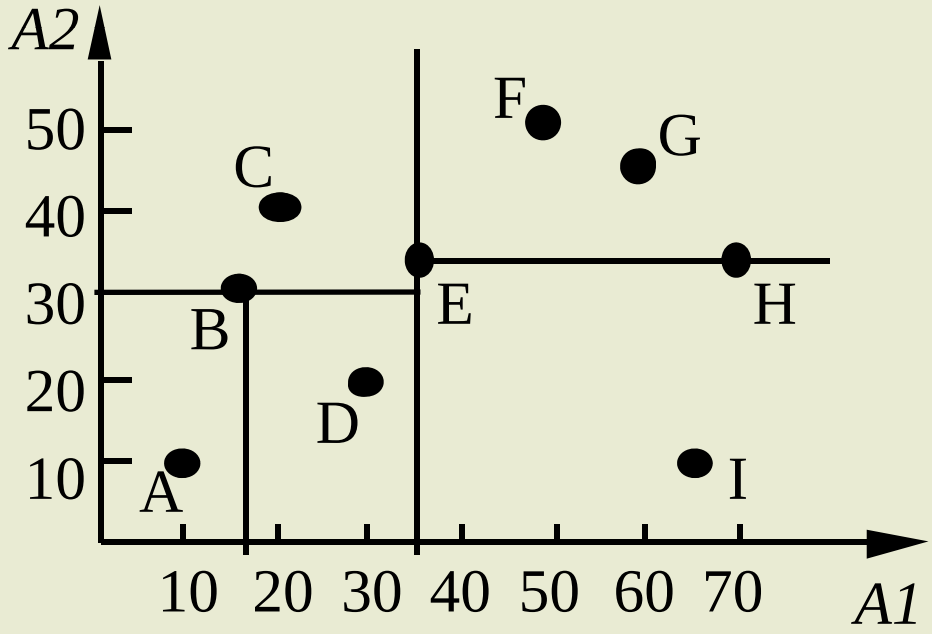
Properties of a kd-Tree

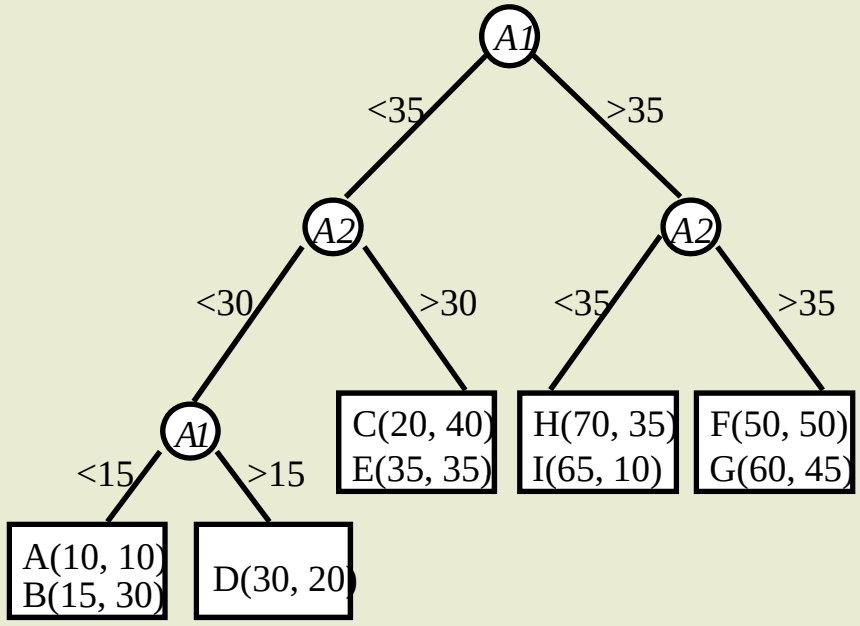
- Ein kd-tree partitions a base:
  - the root represents the whole base
  - a leave node (bucket) represents a subset of the base which is not further partitioned
  - at all inner nodes the base is further partitioned s.t. base is divided according to the value of some attribute.

e

Example for a kd-Tree

CB={A,B,C,D,E,F,G,H,I}





Generating kd-Trees (1)

Algorithm:

PROCEDURE CreateTree(CB) : kd-tree

IF $|CB| \leq b$

THEN RETURN leave node with case base CB

ELSE

A_i := choose_attribute(CB);

v_i := choose_Value(CB,A_i)

RETURN

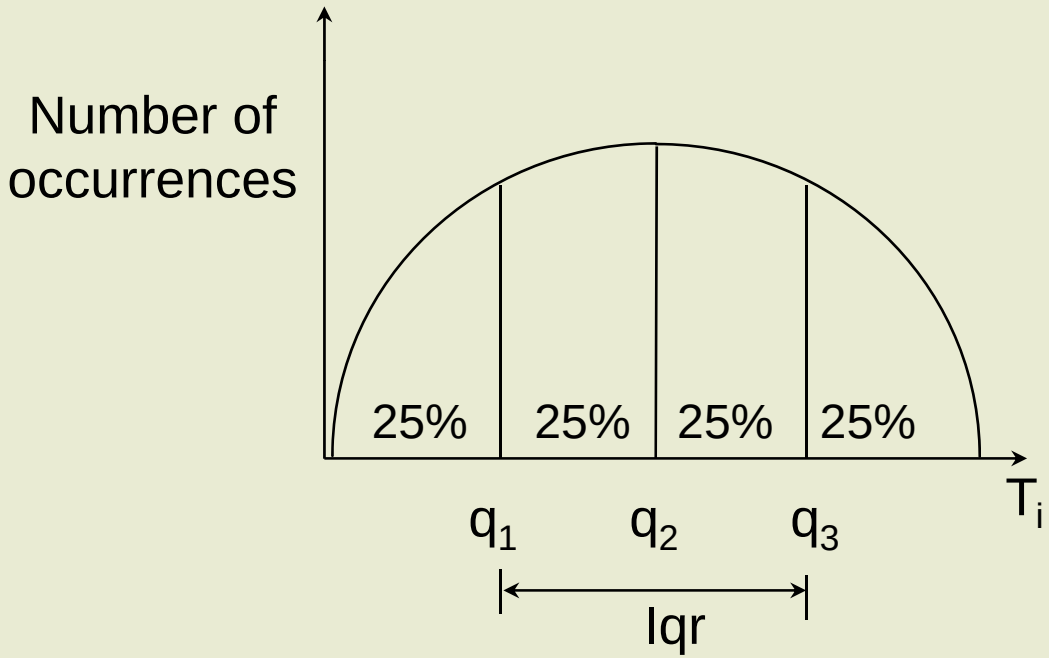
Tree with root labelled with A_i and v_i and which has the two subtrees

CreateTree({$(x_1,...,x_k) \in CB \mid x_i \leq v_i$}) and

CreateTree({$(x_1,...,x_k) \in CB \mid x_i > v_i$})

Selection of Attributes

- Many techniques possible, e.g. the use of entropy
- typical: Interquartil distance

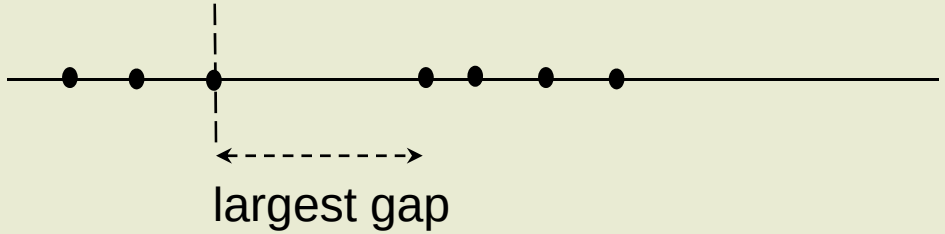


Interquartil distance iqr = $d(q_1,q_2)$

Selection of attribute with largest interquartil distance

Selection of the Values

- Two methods:
  - Median splitting: choose the median d_2 as partition point
  - Maximum splitting:



Retrieval with kd-Trees

Idea for an algorithm:

1. Search the tree top-down to a leave

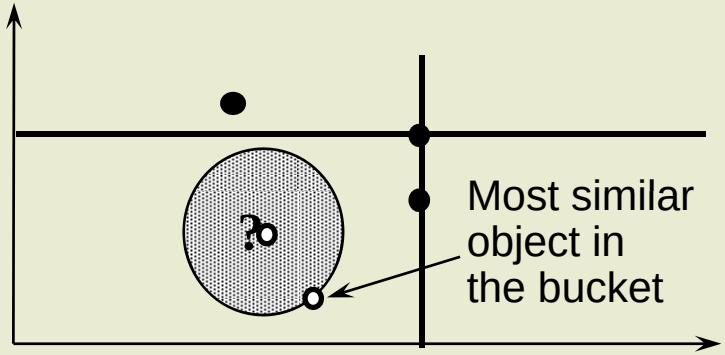
2. compute similarity to objects found

3. Determine termination by BWB-test
3. Determine additional candidates using a BOB-test

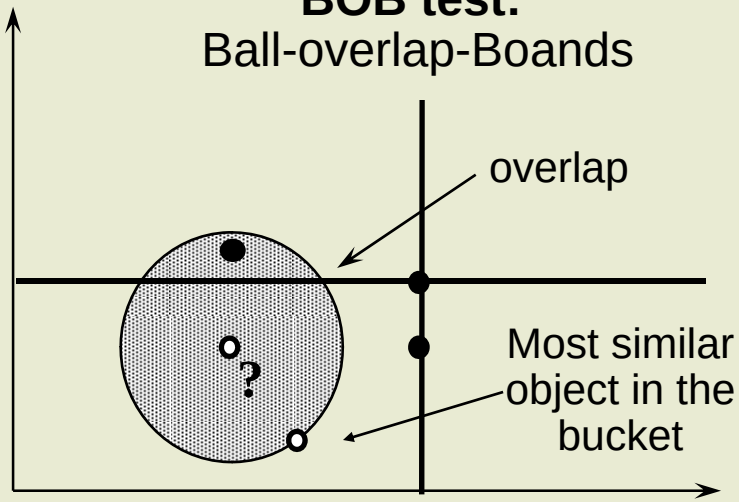
4. If overlapping buckets exist then search in alternative branches (back to step 3)

5. Stop if no overlapping buckets

BWB test
Ball-within-Boands



BOB test:
Ball-overlap-Boands



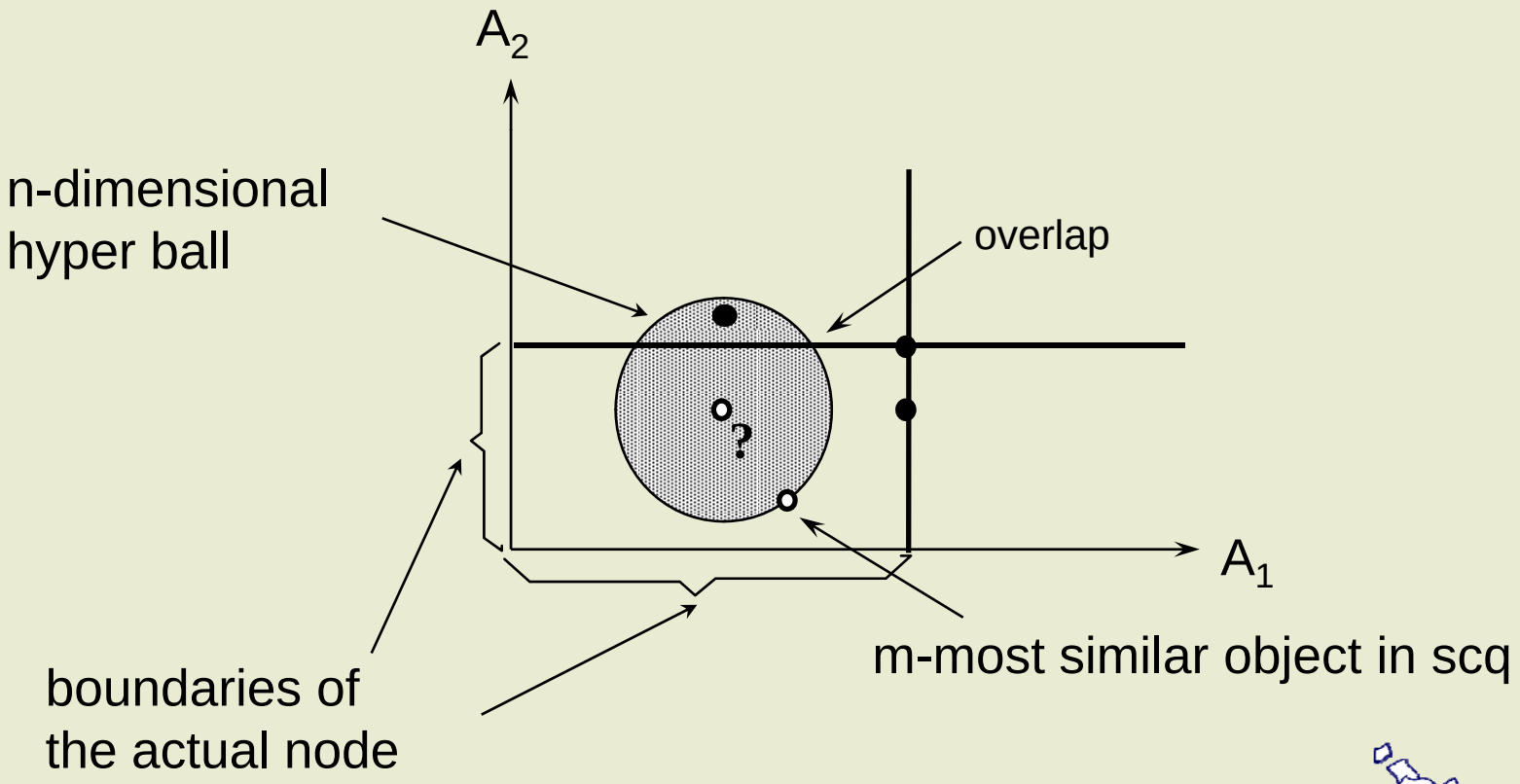
Retrieval with kd-Trees

Algorithm

PROCEDURE retrieve(K: kd-tree)
IF K is leave node
THEN
FOR each object F of *K* DO
IF *sim*(Q,F) > scq[m]. THEN insert F in scq e
ELSE *(* inner node *)*
If A_i is the attribute and v_i the value that label K
IF $Q[A_i] \leq v_i$
THEN
retrieve($K_{\leq}$)
IF *BOB-Test is satisfied* THEN retrieve($K_{>}$)
ELSE
retrieve($K_{>}$)
IF *BOB-Test is satisfied* THEN retrieve($K_{\leq}$)
IF *BWB-Test is satisfied* THEN terminate retrieval with scq ELSE RETURN

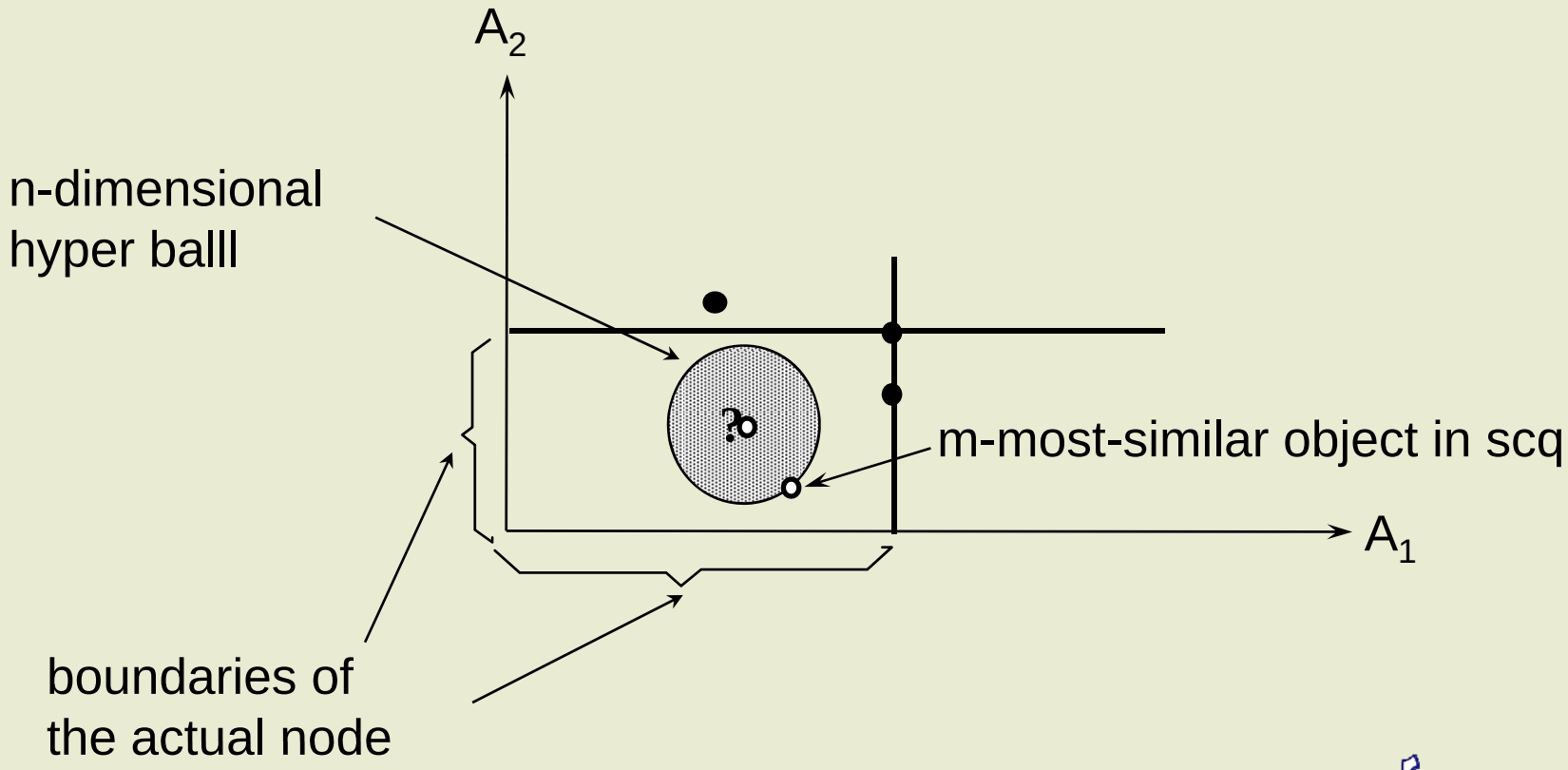
BOB-Test
Ball-Overlap-Bounds

Are there more similar objects then the m-most similar object found in the neighbor subtree ?



BWB-Test
Ball-Within-Bounds

Is there no object in the neighboring subtree
that is more similar than the m-most-object?



Restrictions to the Applicable Similarity Measures

- retrieval with a kd-tree guarantees the m-most similar object if the similarity measure satisfies the following restrictions:
  - Compatability with the ordering and monotonicity
    - $\forall\ x_1,\ldots, x_n, x_i', x_i''$ if $x_i <_i x_i' <_i x_i''$ then
      $sim((x_1,\ldots,x_n) , (x_1,\ldots,x_i',\ldots,x_n)) \geq sim((x_1,\ldots,x_n) , (x_1,\ldots,x_i'',\ldots,x_n))$

Properties of Retrieval with kd-Trees

- Disadvantages:
  - Higher effort to built up the index structure
  - Restrictions for kd-trees:
    - only ordered domains
    - problems with unknown values
    - Only monotonic similarity measures compatible with the ordering
- Advantages:
  - Efficient retrieval
  - Effort depends on the number m of objects to find
  - Incremental extension possible if new objects arise
  - Storage of the objects in a data base possible
- There are improvements of kd-trees (INRECA)

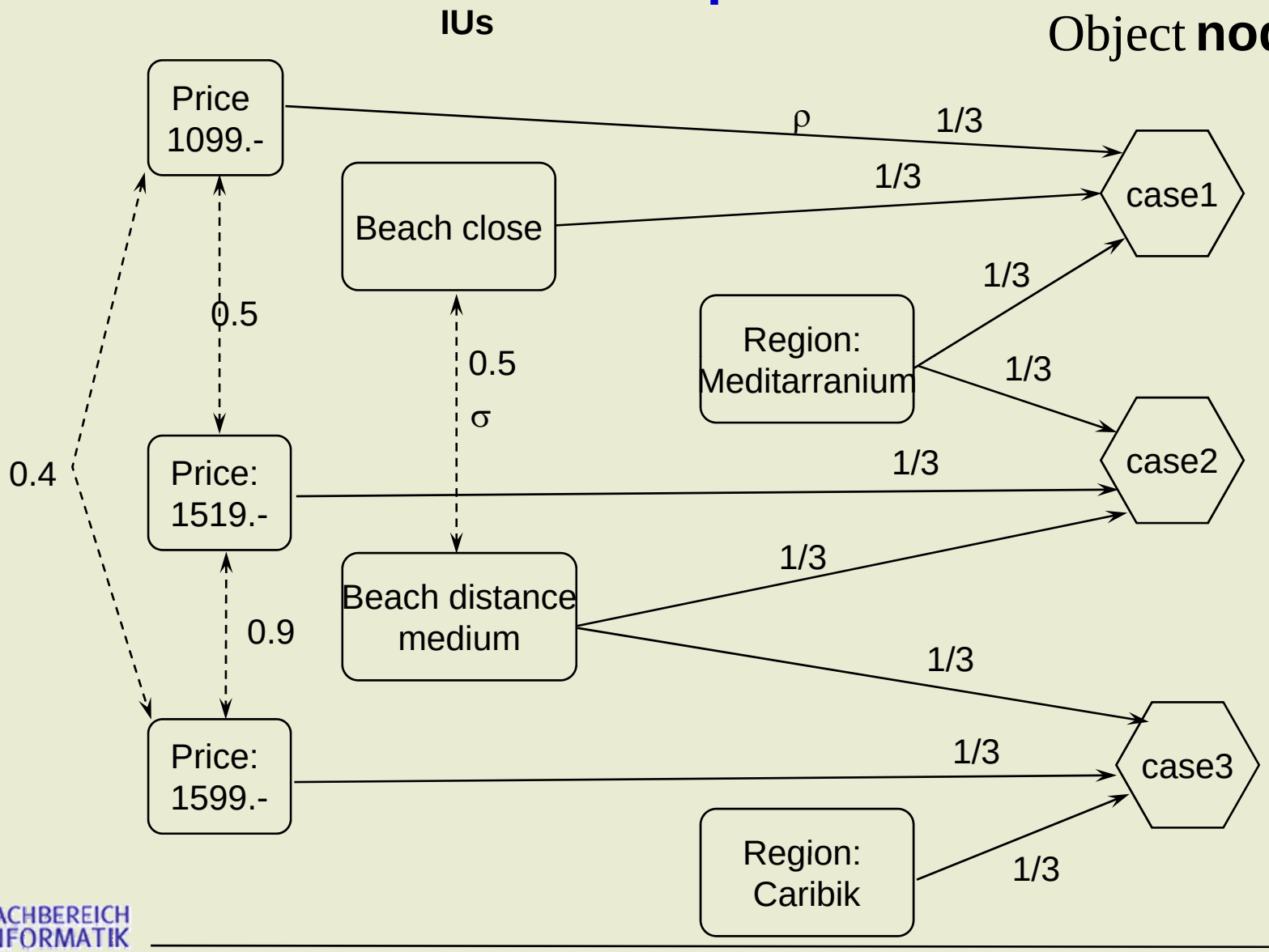
Case Retrieval Nets
(Lenz & Burkhard)

- Partitioning of object information in information units (e.g. attribute-value-pair)
  - Each information unit is a node in the net
  - Each object is a node in the net
- information units which have a similarity > 0 are connected with strenght = similarity.
- For the retrieval information units of the query are activated.
- The activity is propagated through the unil object nodes are reached .
- The activity at the object nodes reflects the similarity to the query.

Concepts (1)

- An information unit (IU) is an atomic knowledge unit, e.g. an attribute-value-pair.
- An is a set of information units.
  A *retrieval Net (Basic Case Retrieval Net, BCRN)* is a 5-tupel N=(E, C, σ, ρ, Π) with:
  - E is a finite set of information units
    C is a finite set of object nodes
  - σ is a similarity measure: σ: E x E -> IR
    σ (e,e') describes the *local similarity* between two IUs e, e'
  - ρ is a relevance function: ρ: E x C -> IR
    ρ(e,c) describes the relevance (*weight*) of the IU for the object c
  - Π is a set of propagation functions π_n: IR^E -> IR for each node
  - $n \in N \cup C$.

Example


IUs
Object node
Price 1099.-
Beach close
Region: Meditarranium
Price: 1519.-
Beach distance medium
Price: 1599.-
Region: Caribik
case1
case2
case3
ρ
σ
1/3
0.4
0.5
0.9

Concepts (2)

- An *activation* of a BCRN is a function α: N $\cup$ C -> IR.

  The activation α(e) of an IU e describes the relevance of this IU for the actual problem.

- The activation at time t is a function α_t: N $\cup$ C -> IR defined by:
  - IUs: $\alpha_{t+1}(e) = \pi_e(\sigma(e_1,e)\cdot\alpha_t(e_1),\ldots,\sigma(e_s,e)\cdot\alpha_t(e_s))$
  - objects: $\alpha_{t+1}(c) = \pi_c(\sigma(e_1,c)\cdot\alpha_t(e_1),\ldots,\sigma(e_s,c)\cdot\alpha_t(e_s))$

Retrieval

- Presented: Query Q, consising of some set of IUs
- retrieval:
  1. Determination of the activation α_0 by:

$$\alpha_0(e) = \begin{cases} 1 \text{ if IE e occurs in the Query} \\ 0 \text{ otherwise} \end{cases}$$

  2. Similarity propagation: for all e $\in$ E, which have a connection to some activated IU.

$$\alpha_1(e) = \pi_e(\sigma(e_1,e)\cdot\alpha_0(e_1),\ldots,\sigma(e_s,e)\cdot\alpha_0(e_s))$$

  3. Relevance propagation:for all object nodes c $\in$ C, which have a connection to some activated IU.

$$\alpha_2(c) = \pi_c(\sigma(e_1,c)\cdot\alpha_1(e_1),\ldots,\sigma(e_s,c)\cdot\alpha_1(e_s))$$

Properties of Retrieval Nets

- Disadvantages:
  - High costs to construct the net.
  - New query nodes may be necessary for numerical attributes.
  - Many propagation steps necessary if the degree of connectivity is high (i.e. many similar IUs)
  - Advantages:
  - Efficient retrieval
  - Effort depends on the number of activated IUs of the query
  - Inkremental extension possible if new object s arise
- There are improvements of this approach (e.g.. Lazy Spreading Activation)

Retrieval with “Fish and Shrink “
Schaaf (1996)

- Object representation is partinioned in different *aspects*.
- aspect is a complex property

  Examples for aspects in medicine: *EKG*, *X-ray image.*
- for each aspect there are individual similarity measures defined.
- Assumption: similarity computation for aspects is costly.
- Total similarity is computed by weighted combinations (aggregation) of aspect similarites.
- Weights are not static but given by the query.
- Approach for retrieval:
  - look ahead computation of the aspect similarities between certain object s
  - Net construction:
  - Test for similarity between the query and the test object
  - Conclusion for similarity for other object s possible without computation

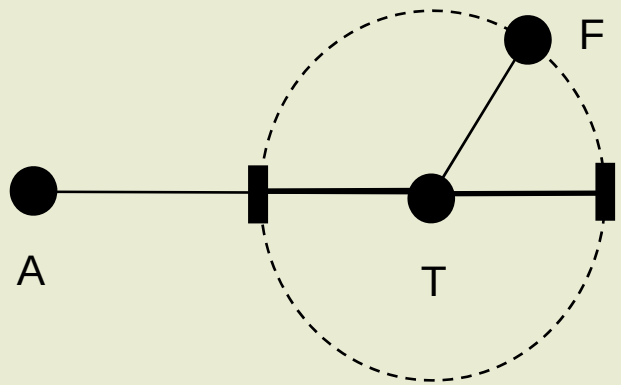
Concepts (1)

- An object F consists of several *aspects* $\alpha_i(F)$
- for each aspect there is an *aspect distance function* $\delta_i(\alpha_i(F_x), \alpha_i(F_y)) \in [0..1]$ (short: $\delta_i(F_x, F_y)$).
- a *view* is a weight vector $W=(w_1,...,w_n)$ mit $w_1+...+w_n=1$

  Observe: The view is part of the query and presented by the user.
- A *view distance* for two objects and a view is a function $VD(F_x,F_y,W) = w_1\ \delta_1(F_x,F_y) +...+ w_n\ \delta_n(F_x,F_y)$
- A case F_x is *view neighbor of the* object F_y w.r.t. a view distance VD, if $VD(F_x,F_y,W) < c$ holds (e.g. c=1).

Assumptions for Fish & Shrink

- If an object F is not similar to query A it is an indication that other object s F'which are similar to F are also not similar to query A.
- More precise: Assumption of the triangle inequality is made:



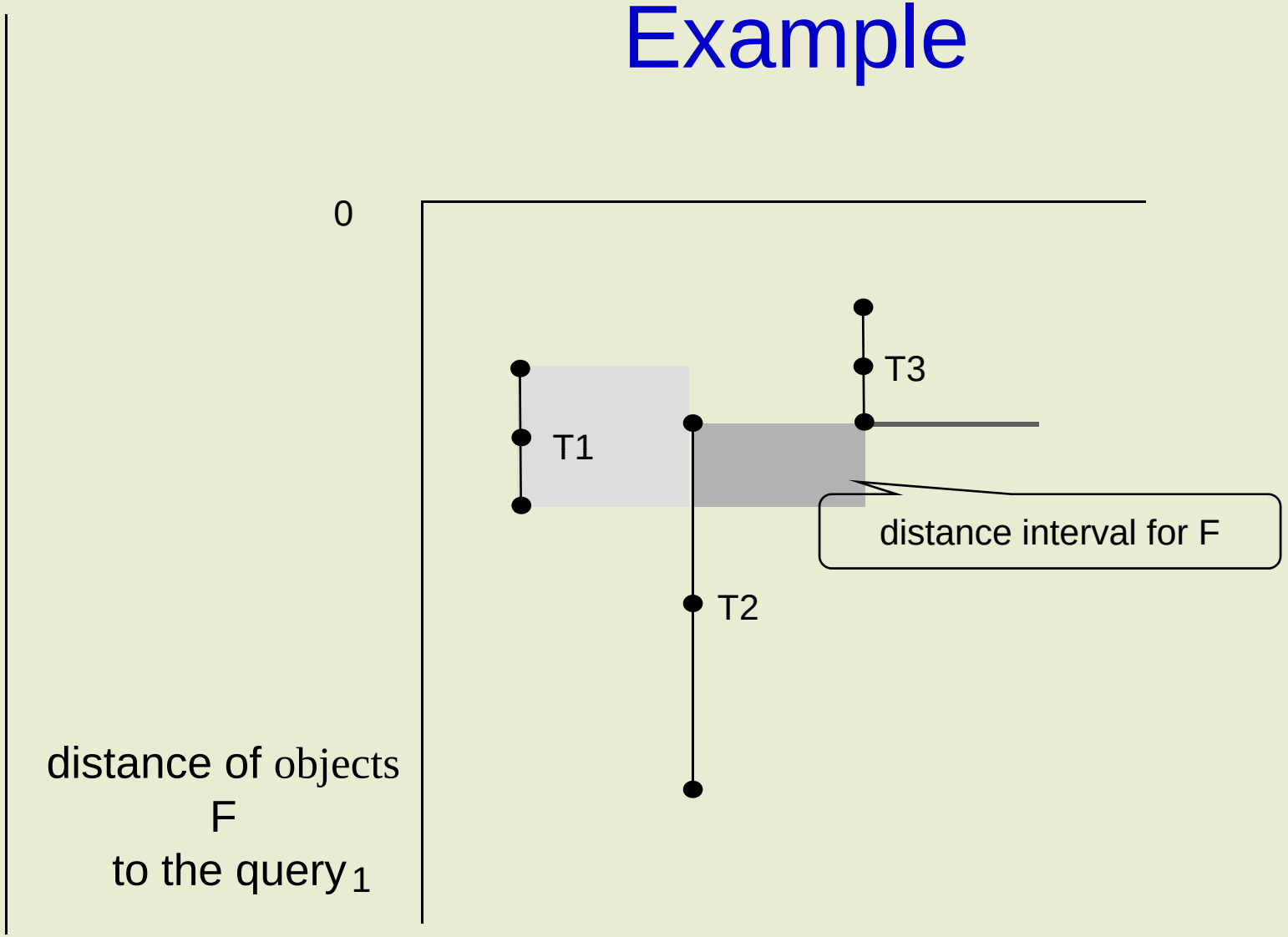
1. $\delta_i(A,F) \geq \delta_i(A,T) - \delta_i(T,F)$

2. $\delta_i(A,F) \leq \delta_i(A,T) + \delta_i(T,F)$

Algorithm (Idea)

- Given:
  - A base with precomputed aspect distances between certain objects (not necessarily between all objects)
  - query A and view W (weight vector)
- Idea of the Algorithm
  - Determine for each object of the base a distance interval (initially [0..1])
  - Choose a test object T and determine the view distance between A and T expensive).
  - determine for most objects of the base the new smaller distance interval by using inequalities 1) and 2).
  - Iterate these steps until the intervals are small enough

Example


0
T1
T2
T3
distance interval for F
distance of objects
F
to the query 1

Algorithm

- Given: base CB, query A, view W
- Algorithm:

FOR EACH F ∈ CB **DO** F.mindis:= 0 F.maxdis := 1 **END**

WHILE NOT *OK* OR Interupted **DO**

determine precision line PL

Choose (fish) an object T with T.mindis = PL

T.mindis, T.maxdis := VD(A,T,W) (*view distance*)

FOR EACH F ∈ CB **AND** F view neighbor of T
AND F.mindis ≠ F.maxdis **DO**

F.mindis := max(testdis-basedis, F.mindis) (*Shrink*)

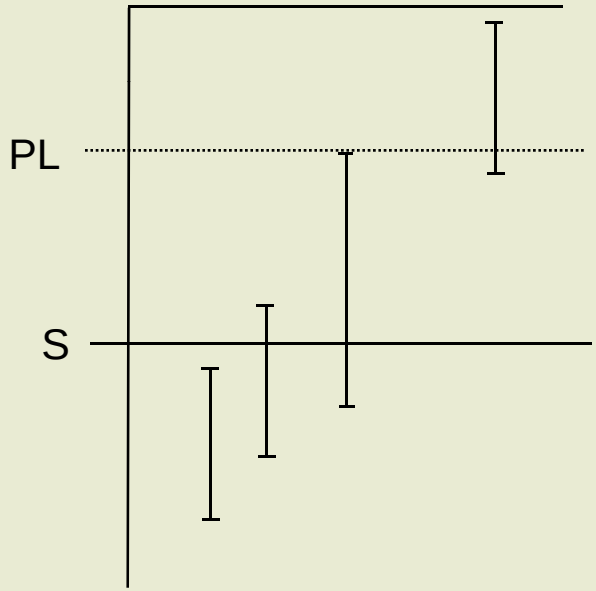
F.maxdis := min(testdis-basedis, F.maxdis)

END

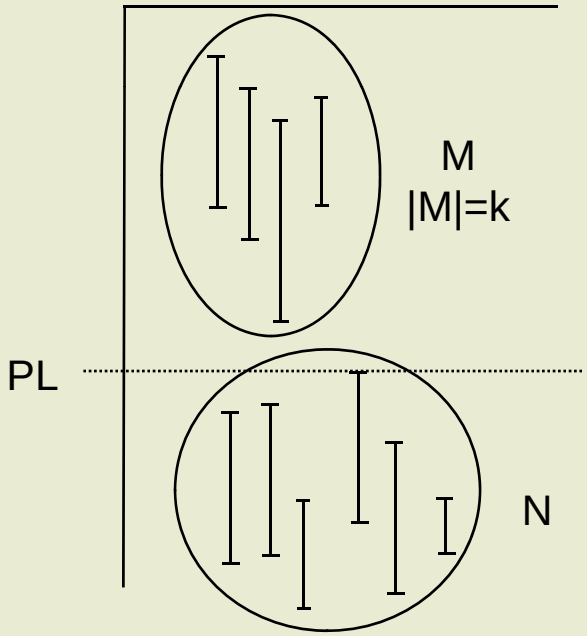
END

Predicate *OK* and *Precision Line (1)*

- By a suitable choices of *OK* and *precision line PL* different retrieval tasks can be satisfied:

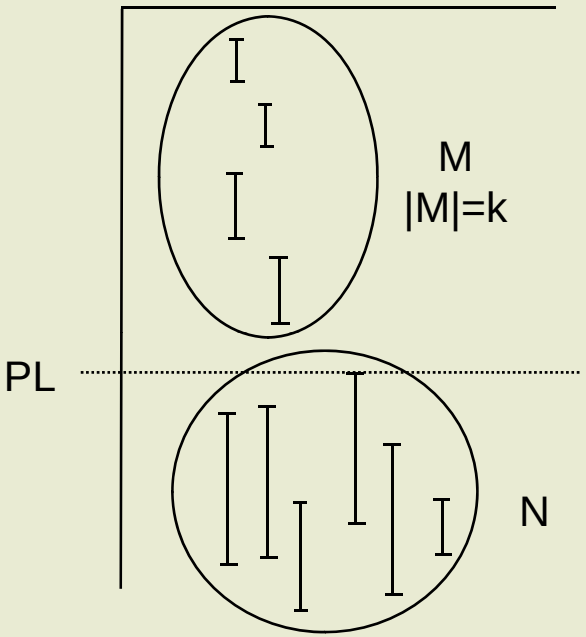


a) All objects better than a threshold S

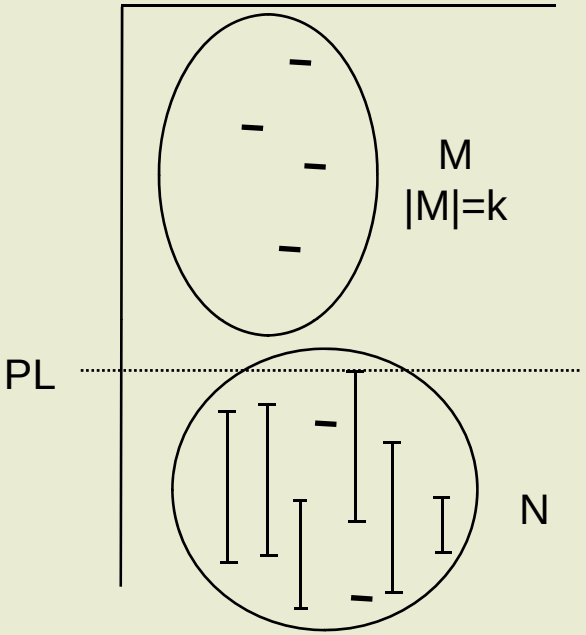


b) The best k objects unordered

Predicate *OK* and *Precision Line (2)*



c) The best k objects ordered, but no exact distances



d) The best k objects ordered with exact distances

Predicate *OK* and *Precision Line (3)*

- Formal definition for OK and PL for variation a) (all cases better than a threshold)

$$\mathrm{OK} := \forall F \in \mathrm{FB} : \neg \mathrm{overlap}(F, S) \vee \mathrm{zero\_interval}(F)$$

$$\mathrm{PL} := F.\mathrm{mindist} : \mathrm{overlap}(F, S) \wedge \mathrm{min\_mindist}(F, \mathrm{FB}) \wedge \neg \mathrm{zero\_interval}(F)$$

Properties of Fish & Shrink

- Disadvantages :
  - aspect distances between objects have to precomputed
  - distance function has to satisfy the triangle inequality
  - Advantages:
  - Flexible distance computation to the query (views)
  - different retrieval tasks can be performed
  - Efficient because many distance computations can be saved
  - Can be used as anytime-algorithm
- Suitable If very expensive similarity computations occur (e.g. for graph representations)

Retrieval by SQL Approximation: Application Scenario

(Schumacher & Bergmann, 2000)

- Product database exists and is used for many services in the business processes
- Very huge number of products
- Product database changes continously and rapidly
- Product representation is very simple: usually a list of attribute values

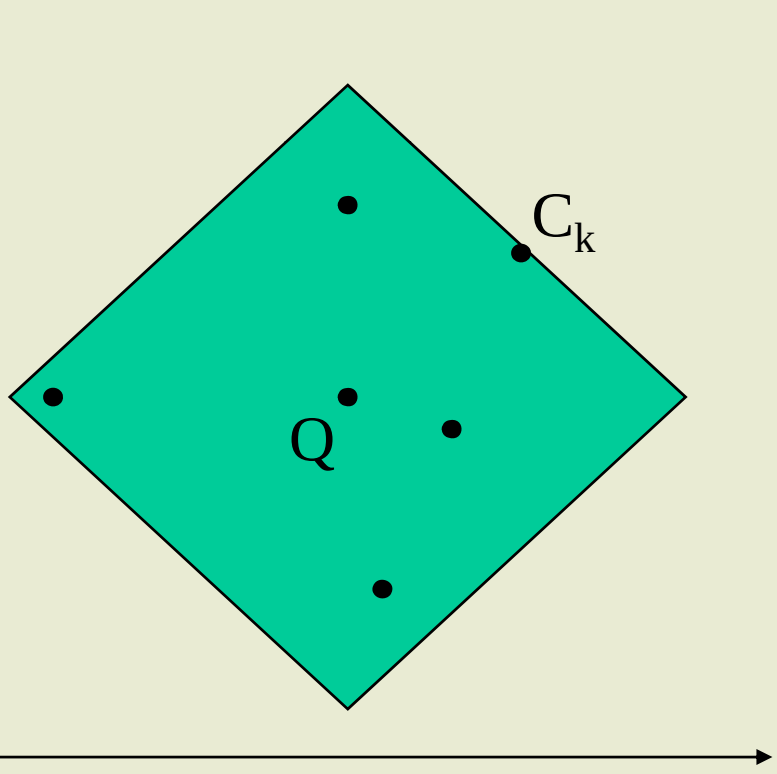
Basic Approaches for Linking with Databases

- Retrieval inside the database
  - Solution depends on database system
- Retrieval on top of the database
  - Bulk-loading all products: duplication of data, inefficient, consistency problems
  - Building an index: consistency problem remains
  - Approximating similarity-based retrieval with SQL queries

Foundations

- Query and objects: List of Attribute Values
- Objects are stored in a single database table
- Each attribute corresponds to one column
- Similarity calculated as a weighted average of local similarities of attribute values
- Assumption: Objects are distributed equally across *n*-dimensional space

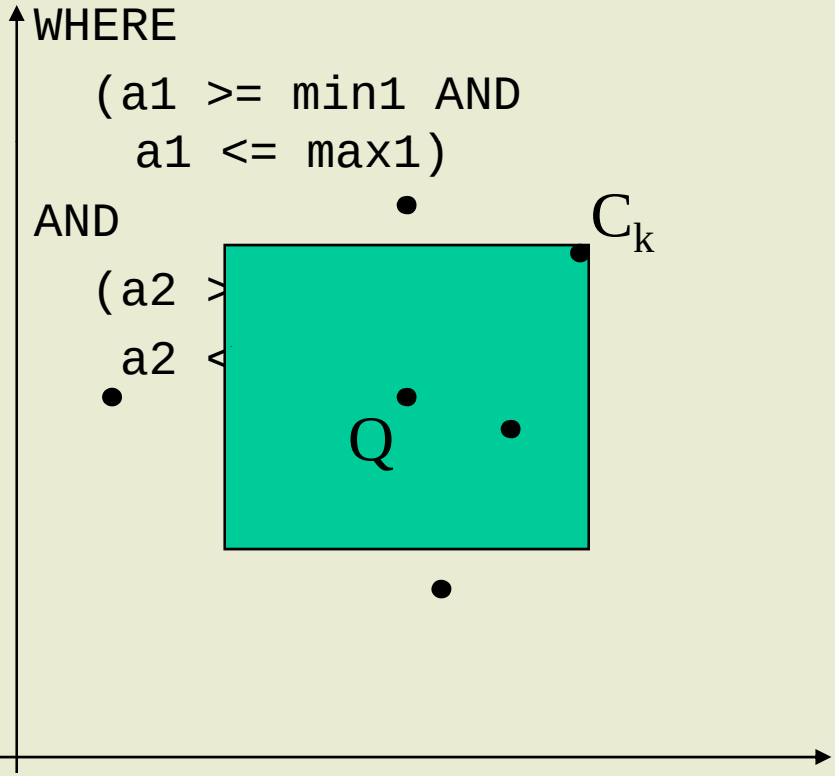
Similarity-based Retrieval



- The most similar objects lie within a hyper-rhombus centered on the query
- Less similar objects are all outside the rhombus

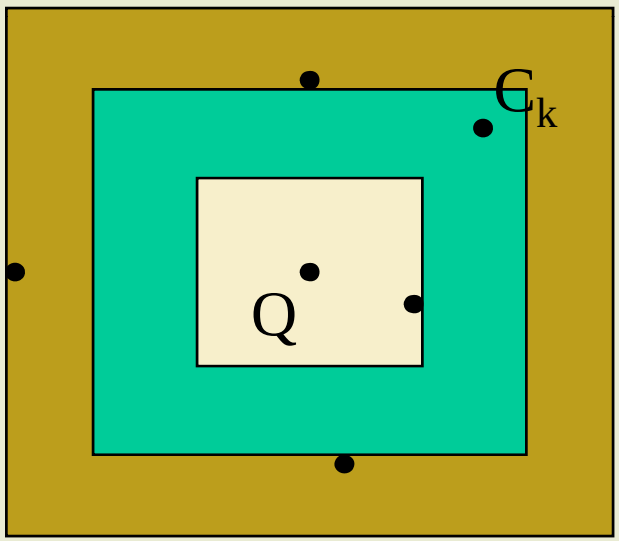
SQL Query

```
SELECT a1, a2
FROM table
WHERE
   (a1 >= min1 AND
    a1 <= max1)
AND
   (a2 >
    a2 <
```



- Selected objects lie within a hyper-rectangle
- All objects within the rectangle have a minimum similarity
- More similar objects can lie outside the rectangle

SQL-Approximation of Rhombus



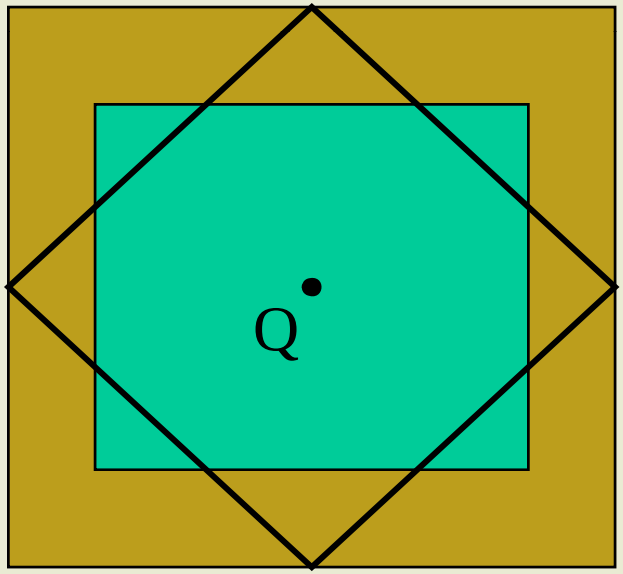
- *Query Relaxation:* Generate a series of SQL-queries (“rings”) and compute similarity of retrieved objects
- Repeat until “enough” objects have been evaluated

Basic Algorithm

```
ALGORITHM DBRetrieve
  INPUT :  Object query, Integer k
  OUTPUT : List of Object result[1...k]

  BEGIN
    step = 0
    WHILE (NotFinished(query, result)) DO
      sqlQuery = RelaxQuery(query, step, k)
      dbResult = ExecuteSQLQuery(sqlQuery)
      newObjects = SortObjects(query, dbResult)
      merge(result, newObjects)
      step = step + 1
  END
```

Completeness vs. Efficiency

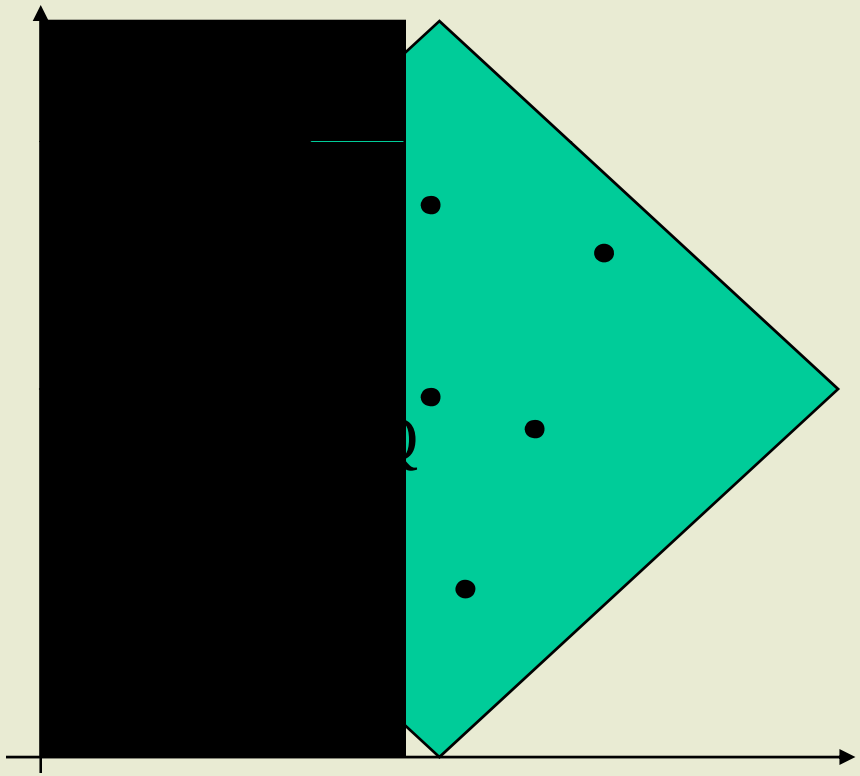


- For each ring there are objects more similar than the selected ones, but outside the ring
- Ensuring complete-ness can mean to load too many useless objects

Query Relaxation in a Nutshell

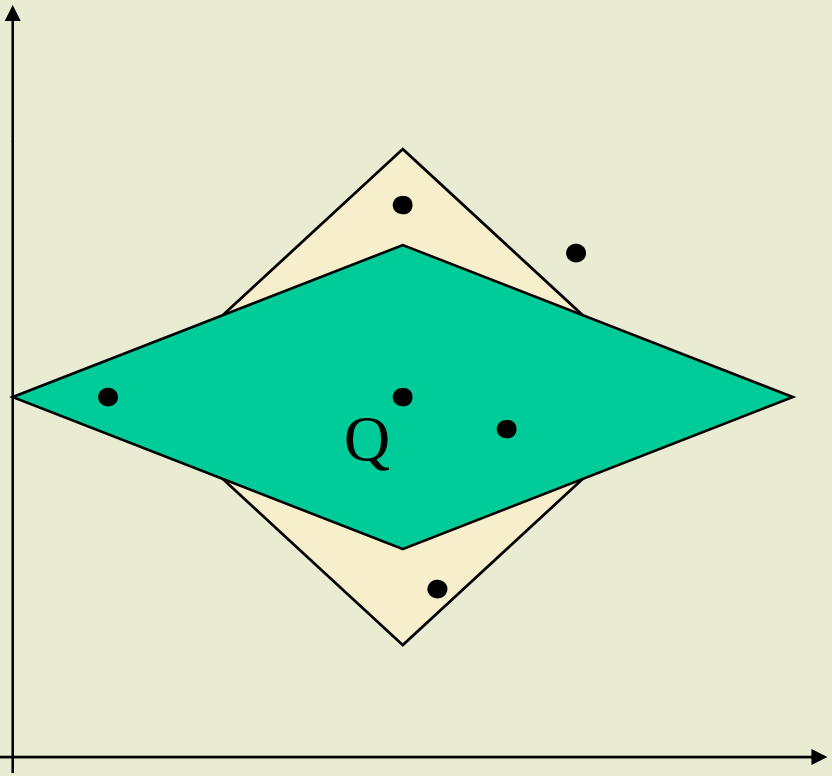
- Goal: Volume of first ring big enough to cover complete retrieval result
- Calculate number of needed objects from number object requested in query: $N_{max}(k)$
- Equal distribution of objects $\rightarrow$ calculation of volume to cover this number
- Equal relaxation in all attributes $\rightarrow$ Compute bounds of ring from volume by using local similarity measures

Advanced Features: Filter



- Hard constraints on the results cases
- Cuts off half of similarity rhombus
- Double volume of ring so that same number of objects is loaded

Advanced Features: Weights



- Express preferences of user for attributes
- Rhombus becomes smaller for attributes with heigher weights
- Adapt ring in the same way so that same volume is covered

Sales Support Applications

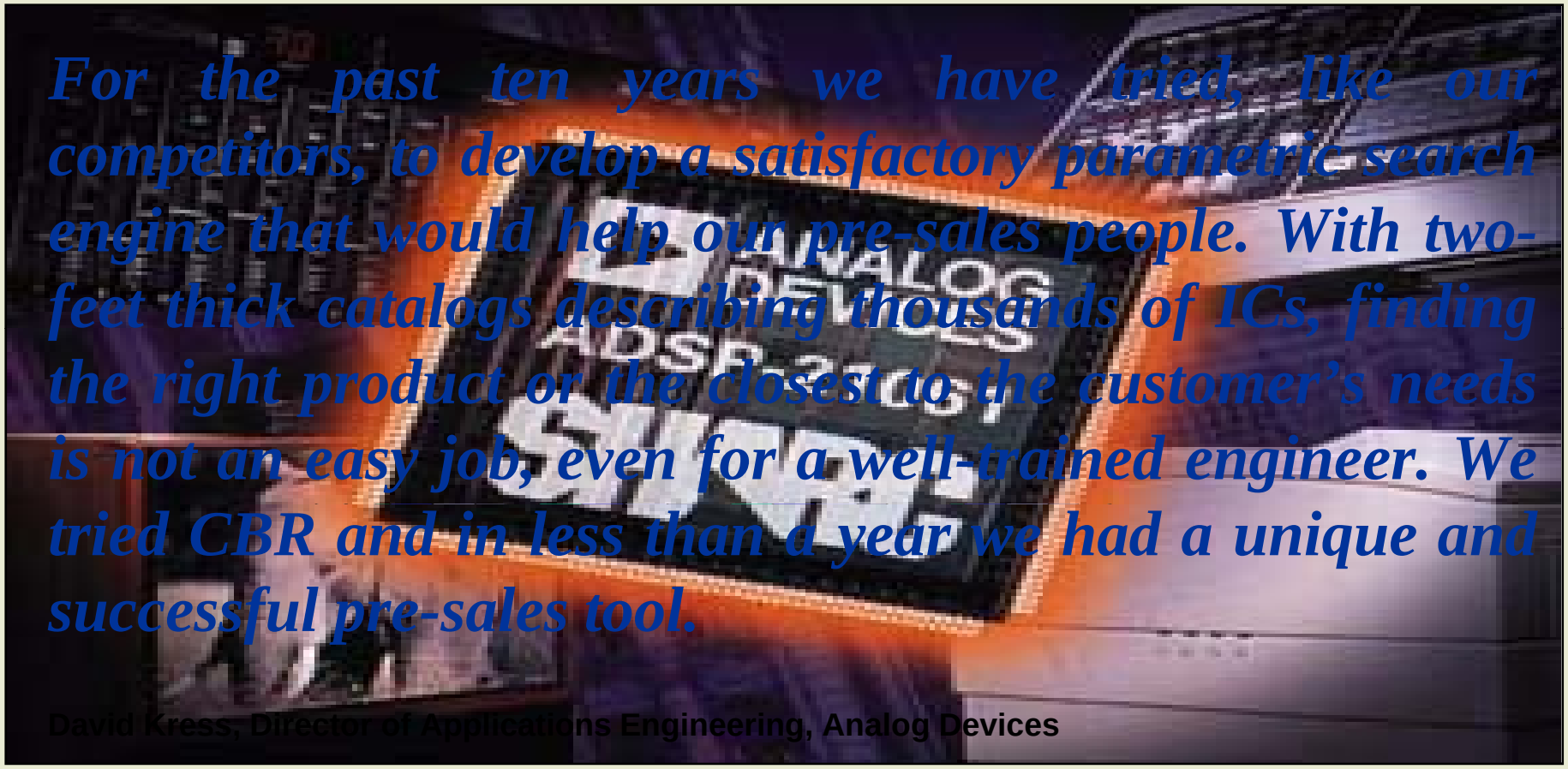
A company has described a catalog of electronic products according to the different sales parameters. Parameters are used to describe

- technical characteristics of the product, as well as
- sales characteristics.

The chance of retrieving a product that is identical to the user's query is very low. Standard database queries almost never retrieve any products from the catalog database.

⇒ **The application's goal is to find the best matching product based on the characteristics the customer enters for his or her desired product.**

Analog Devices (I)



For the past ten years we have tried, like our competitors, to develop a satisfactory parametric search engine that would help our pre-sales people. With two-feet thick catalogs describing thousands of ICs, finding the right product or the closest to the customer's needs is not an easy job, even for a well-trained engineer. We tried CBR and in less than a year we had a unique and successful pre-sales tool.

David Kress, Director of Applications Engineering, Analog Devices

Analog Devices (II)

- Leading provider of precision, high-performance integrated circuits used in analog and digital signal processing applications
- More than 50000 customers worldwide
- Thousands of products:
  - catalogues and data sheets up to two feet thick each year
  - cost of printing and shipping $3 million per year
  - more than 50 parameters to describe the product
  - most products in a line just differ by one element

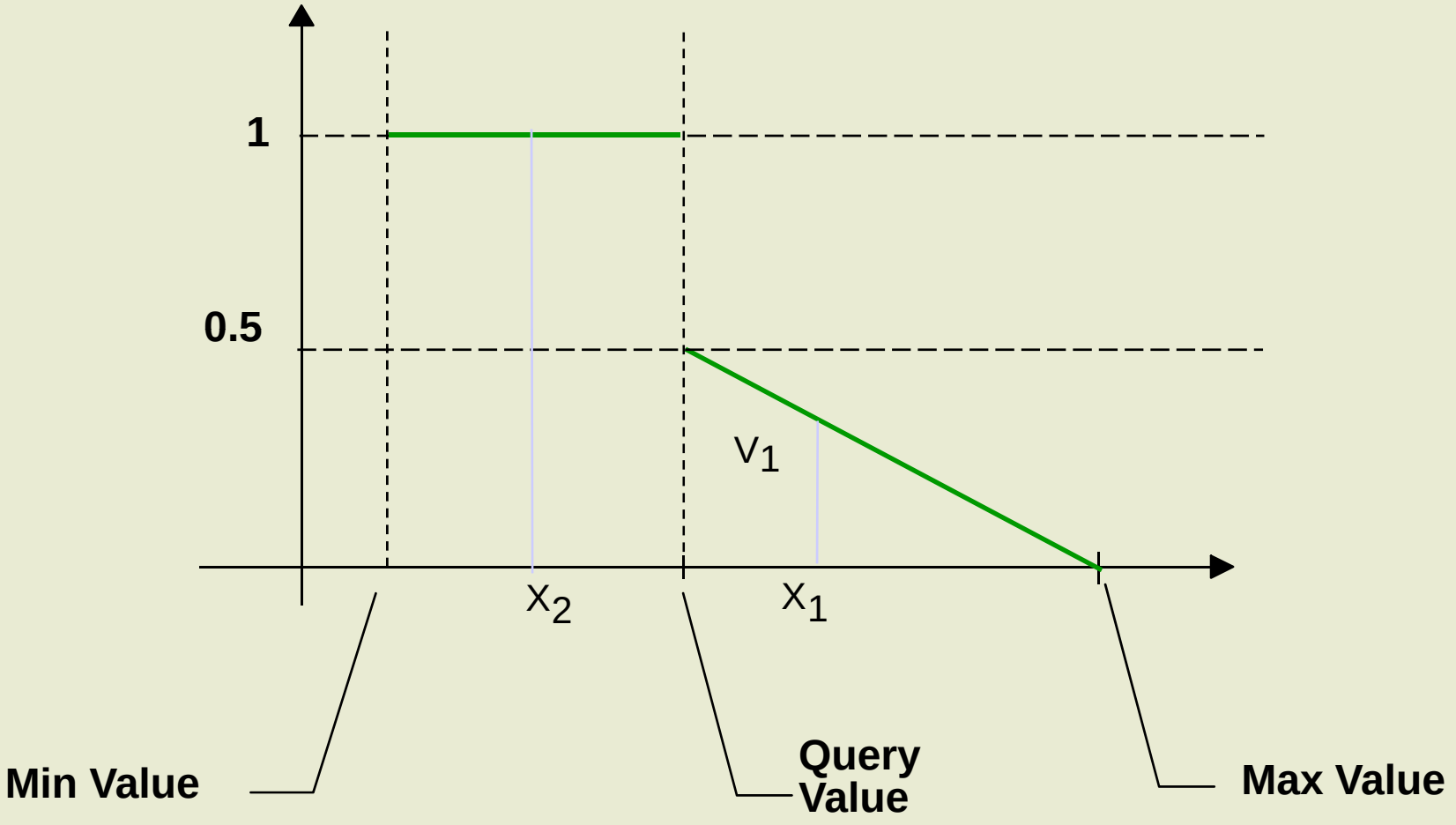
Analog Devices (III)

- How technical support worked:
  - Pre-sales engineer takes customer's requirements over the phone
  - Tries to find a match in the Analog Devices product range
- Lengthy and complex process that involves weighting dozens of constraints while interacting with the customer at the same time
- Tried solution: usage of standard database management $\Rightarrow$ SQL-style searches

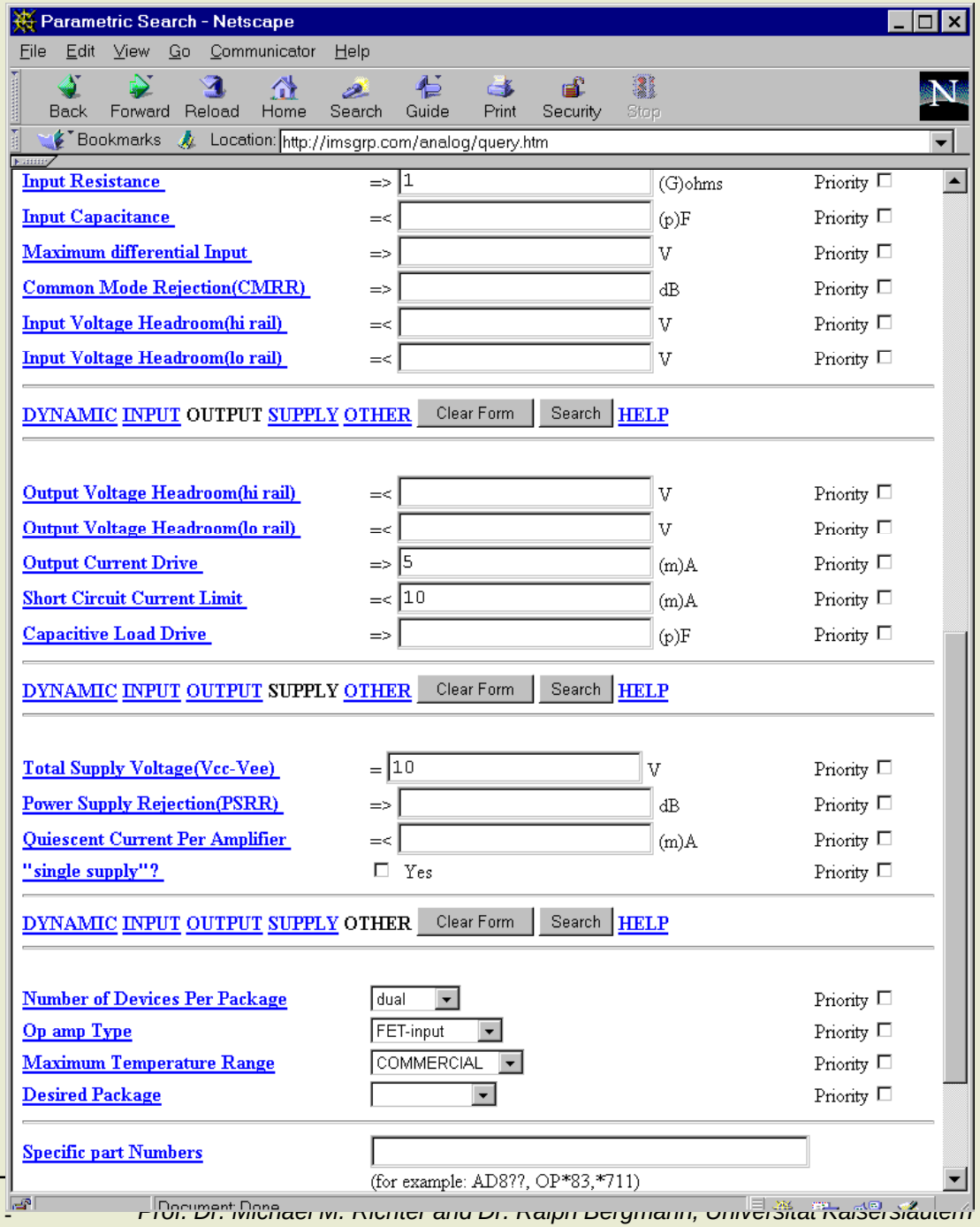
Analog Devices (IV)

- CBR solution provided by IMS (Ireland) in less than six months
- Parametric search:
  - customers specify interactively their product requirements
  - system finds the right product or one as close as possible to the customers' needs
  - always an answer provided
  - if the customer is not satisfied a new search can be started
- System available on CD-ROM catalogue; used by customers as well as pre-sales engineers
- Savings of $2 million per year
- The quality of the service makes the difference

Similarity Measures Contain much Knowledge

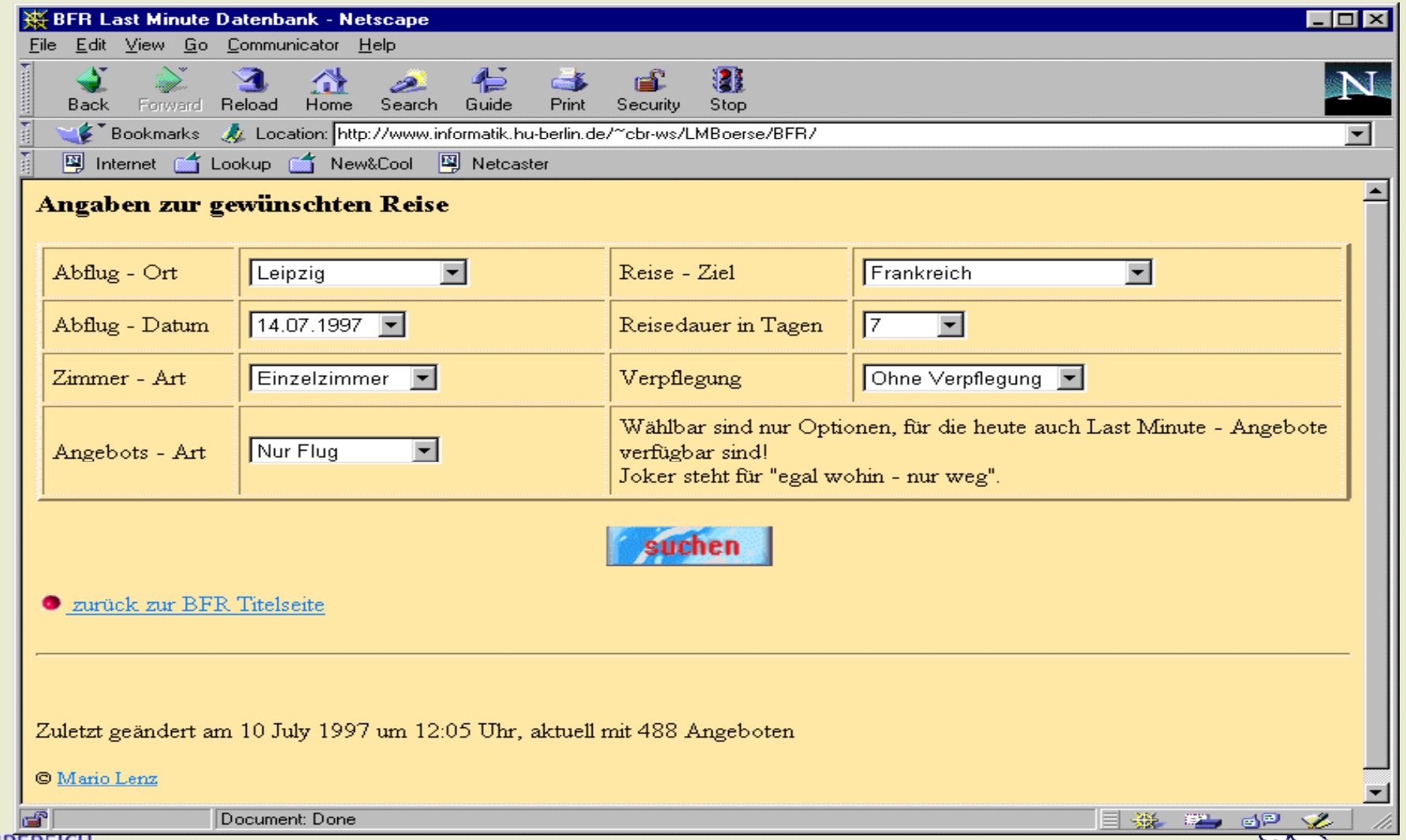


Analog Devices (V)

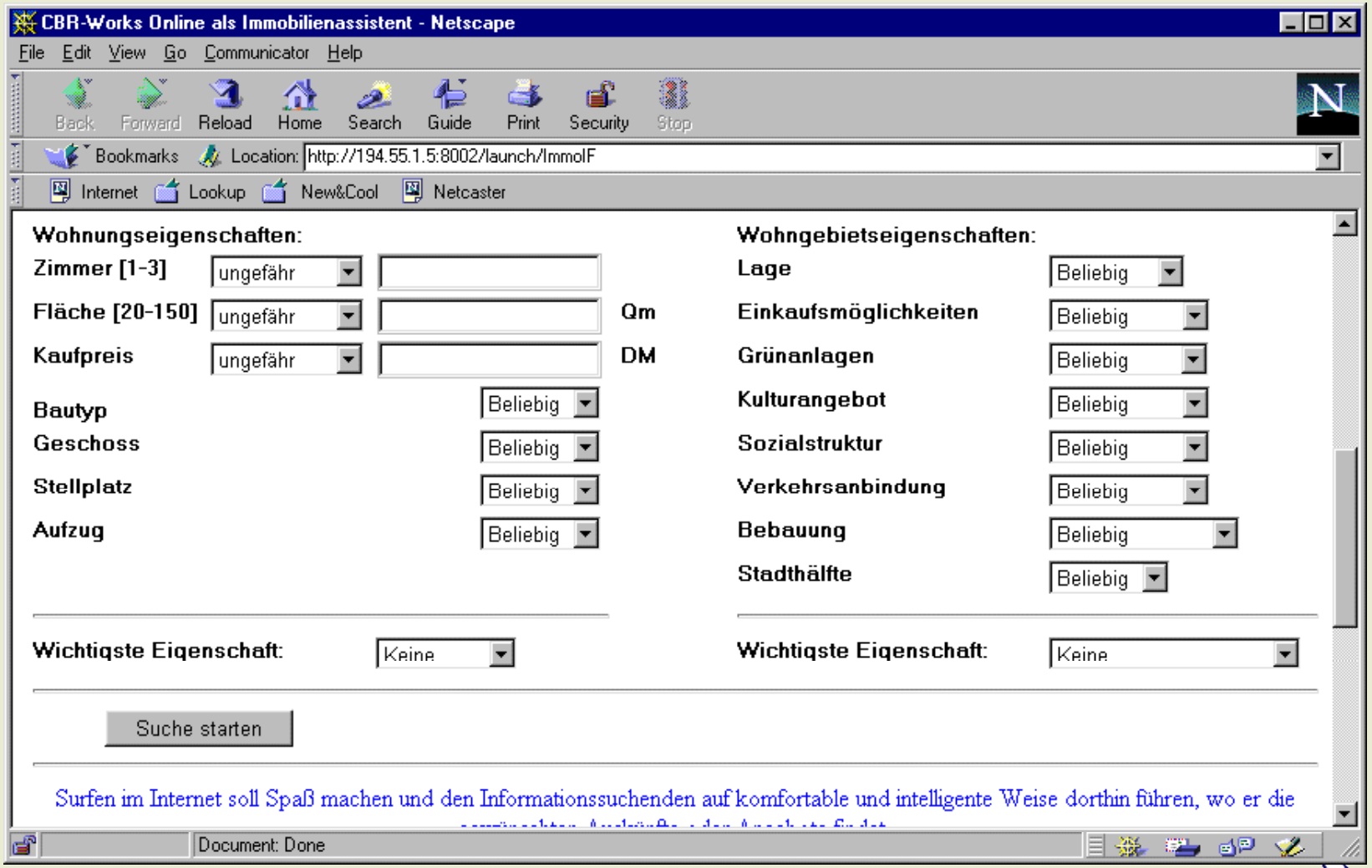


FACHBEREICH INFORMATIK

Example: Last Minute Travel



Example: Appartment Rental



Summary

- For approximate retrieval data base queries as well as common information retrieval techniques are insufficient.
- The main retrieval techniques are:
  - linear search, two level search, index oriented retrieval
  - retrieval with k-d trees
  - case-retrieval nets
  - fish and sink
  - SQL-approximation
- They have been developed in CBR. Examples were shown.