

P2P Similarity Search Structures

Pavel Zezula

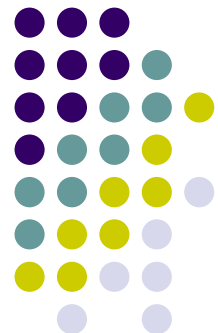
Faculty of Informatics, Masaryk University

Brno, Czech Republic

SEBD 2006

Fourteenth Italian Symposium on Advanced Database Systems

Portonovo (AN), Italy, June 18-21, 2006



Searching



- One of the oldest and important data processing operations
- The problem is constrained by definitions of:
 - **where** to search \Rightarrow *domain (collection) of data*
 - **how** to search \Rightarrow *comparison criterion on objects*
 - **what** to retrieve \Rightarrow *query specification of data subsets*

Digital Data Explosion



- Everything we **write**, **see**, or **hear** can now be **digital** form!!
- Estimations:
 - 93% of produced data is digital
 - 1% is text
 - multimedia, scientific, sensor, etc. is **prevalent**

SEBD06, Portonovo, June, 2006

Change of the Search Paradigm



- Traditional YES-NO **keyword** search will not suffice
- New types of data need **gradual** comparison and/or ranking based on:
 - similarity
 - dissimilarity
 - proximity, etc.

SEBD06, Portonovo, June, 2006

The Concept of Similarity

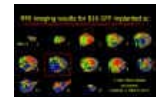


- Social Psychology:
 - *“An ability to assess similarity lies close to the core of cognition. The sense of sameness is the very keel and backbone of our thinking. An understanding of problem solving, categorization, memory retrieval, inductive reasoning, and other cognitive processes require that we understand how humans assess similarity.”*
- Quotation from: *MIT Encyclopedia of the Cognitive Sciences, Cambridge, MA, MIT Press 2006, pp. 763-765*

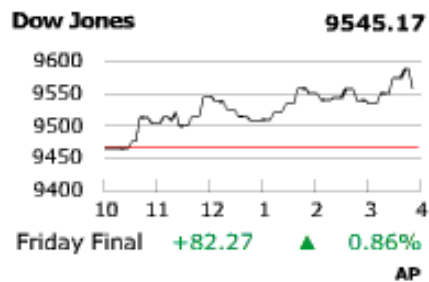
Requirements of New Applications



Medicine: *Magnetic Resonance Images (MRI)*

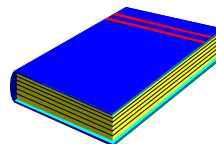


Finance: *stocks with similar time behavior*



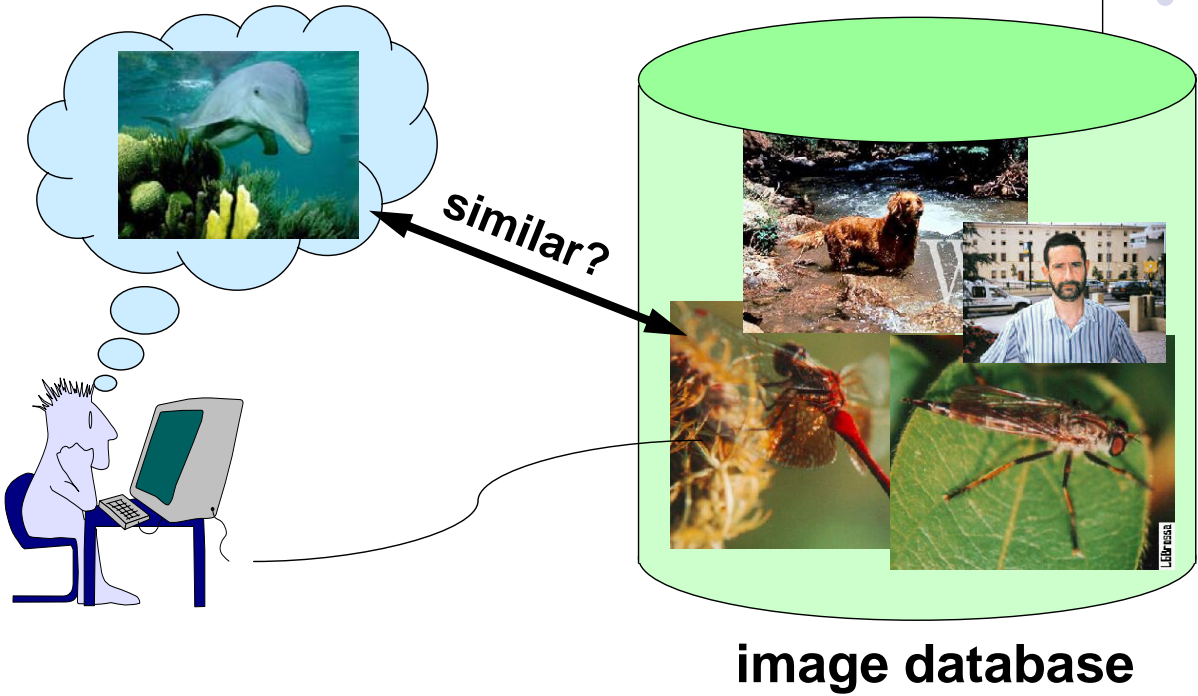
Digital library:

- *test retrieval*
- *multimedia information retrieval*



SEBD06, Portonovo, June, 2006

Image Similarity Search

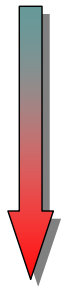


SEBD06, Portonovo, June, 2006

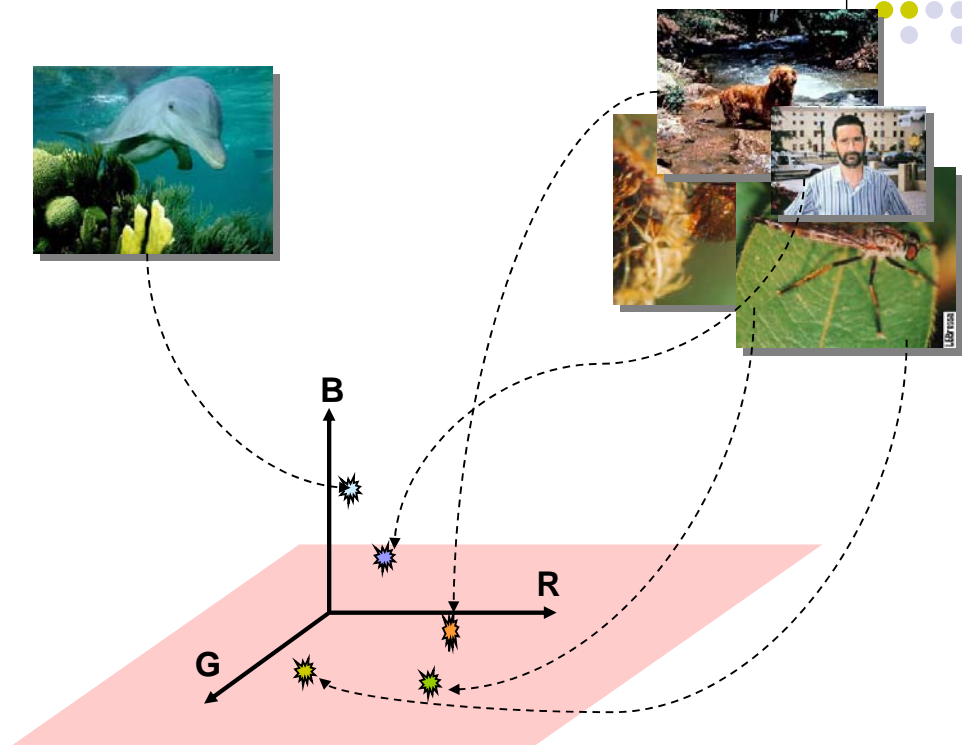
Similarity Based on Features



image layer



feature layer



SEBD06, Portonovo, June, 2006

Similarity Searching



- Effectiveness
 - the way of formulating the similarity measures - a model of human perception
- Efficiency
 - the way of achieving the required performance over huge volumes of data – index structure

SEBD06, Portonovo, June, 2006

Metric Space

an Abstraction of Similarity



- Metric space: $\mathcal{M} = (\mathcal{D}, d)$

- \mathcal{D} – domain
- distance function $d(x, y)$

$\forall x, y, z \in \mathcal{D}$

- $d(x, y) > 0$ - non-negativity
- $d(x, y) = 0 \Leftrightarrow x = y$ - identity
- $d(x, y) = d(y, x)$ - symmetry
- $d(x, y) \leq d(x, z) + d(z, y)$ - triangle inequality

SEBD06, Portonovo, June, 2006

Examples of Distance Functions



- L_p metric functions (for vectors)

- L_1 – city-block distance

$$L_1(x, y) = \sum_{i=1}^n |x_i - y_i|$$

- L_2 – Euclidean distance

$$L_2(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

- L_∞ – infinity

$$L_\infty(x, y) = \max_{i=1}^n |x_i - y_i|$$

- edit distance (for strings)

- minimal number of insertions, deletions and substitutions
- $d(\text{'application'}, \text{'applet'}) = 6$

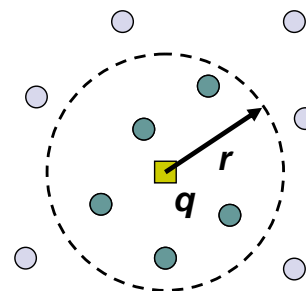
- Jaccard's coefficient (for sets A,B) $d(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$

SEBD06, Portonovo, June, 2006

Similarity Search Problem



- For $\mathcal{X} \subseteq \mathcal{D}$ in metric space \mathcal{M} ,
pre-process \mathcal{X} so that the similarity queries
are executed efficiently.
- similarity queries
 - range search
 - $R(q,r) = \{ x \in \mathcal{X} \mid d(q,x) \leq r \}$
 $q \in \mathcal{D}, r \geq 0$



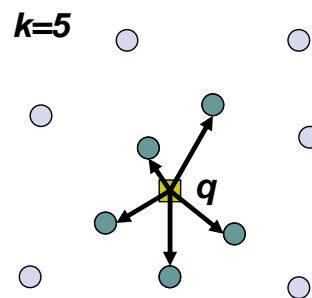
SEBD06, Portonovo, June, 2006

Similarity Queries



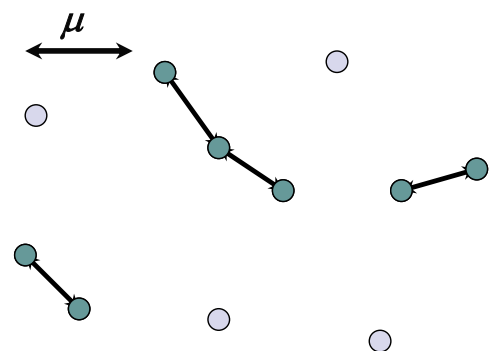
- k -nearest neighbours

- $NN(q,k) = A, q \in \mathcal{D}, k > 0$
- $A \subseteq \mathcal{X}, |A| = k$
- $\forall x \in A, y \in \mathcal{X} - A, d(q,x) < d(q,y)$



- similarity join

- $X = \{x_1, x_2, \dots, x_N\}, Y = \{y_1, y_2, \dots, y_M\}$
- $\{(x_i, y_j) \mid d(x_i, y_j) < \mu\}$
- similarity „self“ join $\Leftrightarrow X = Y$



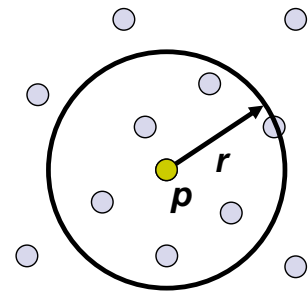
SEBD06, Portonovo, June, 2006

Basic Partitioning Principles



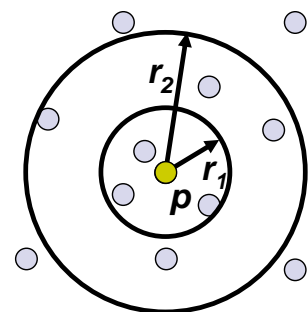
- ball partitioning

- $\{x \in \mathcal{X} \mid d(p,x) \leq r\}$
- $\{x \in \mathcal{X} \mid d(p,x) \geq r\}$



- multiple ball partitioning

- $\{x \in \mathcal{X} \mid d(p,x) \leq r_1\}$
- $\{x \in \mathcal{X} \mid d(p,x) > r_1 \text{ and } d(p,x) \leq r_2\}$
- $\{x \in \mathcal{X} \mid d(p,x) > r_2\}$



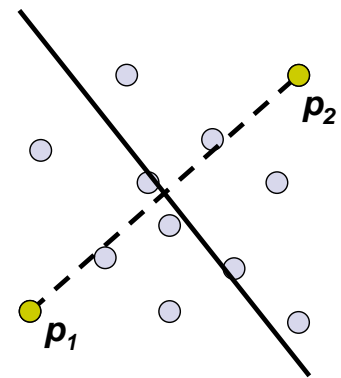
SEBD06, Portonovo, June, 2006

Basic Partitioning Principles



- generalised hyperplane

- $\{x \in \mathcal{X} \mid d(p_1, x) \leq d(p_2, x)\}$
- $\{x \in \mathcal{X} \mid d(p_1, x) > d(p_2, x)\}$



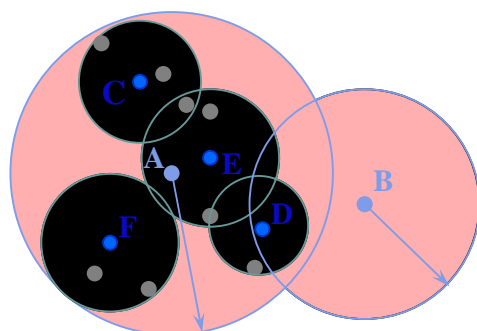
The M-tree [Ciaccia, Patella, Zezula, VLDB 1997]



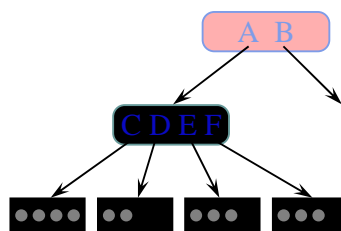
- 1) Paged organization
- 2) Dynamic
- 3) Suitable for arbitrary metric spaces
- 4) I/O and CPU optimization - computing d can be time-consuming

SEBD06, Portonovo, June, 2006

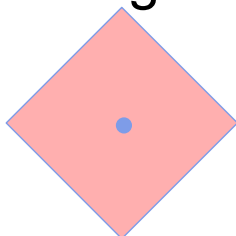
The M-tree Idea



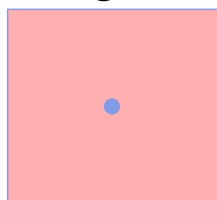
Metric: L_2 (Euclidean)



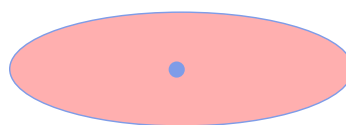
- Depending on the metric, the “shape” of index regions changes



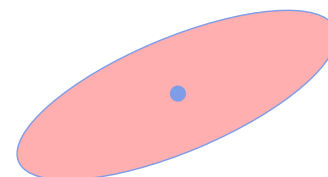
L_1 (city-block)



L_∞ (max-metric)



weighted-Euclidean



quadratic form

SEBD06, Portonovo, June, 2006

The M-tree on the Web



- Home page: <http://www-db.deis.unibo.it/Mtree/>



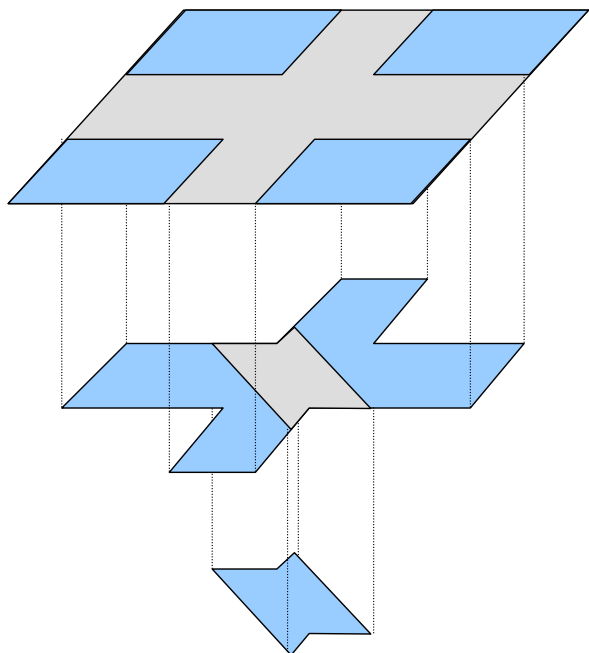
M-tree software can be freely downloaded

- Based on GiST package (Berkeley Univ.)

Google Scholar: 478 citations in June 2006

SEBD06, Portonovo, June, 2006

D-Index [Dohnal, Gennaro, Zezula, MTA 2002]



4 separable buckets at the first level



2 separable buckets at the second level

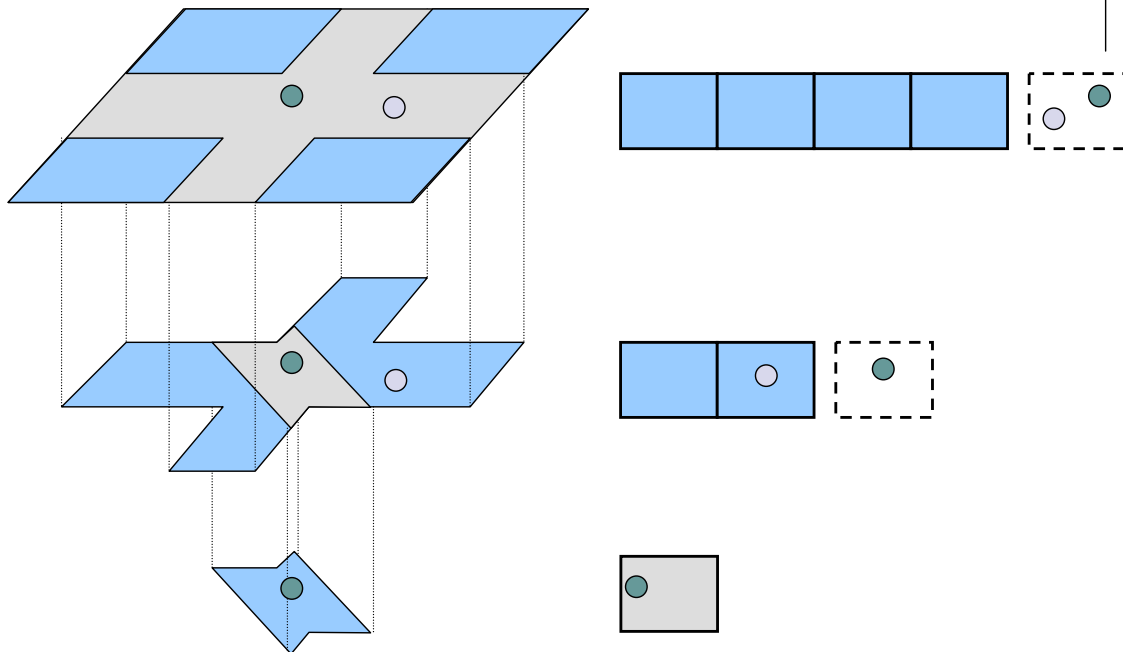


exclusion bucket of the whole structure



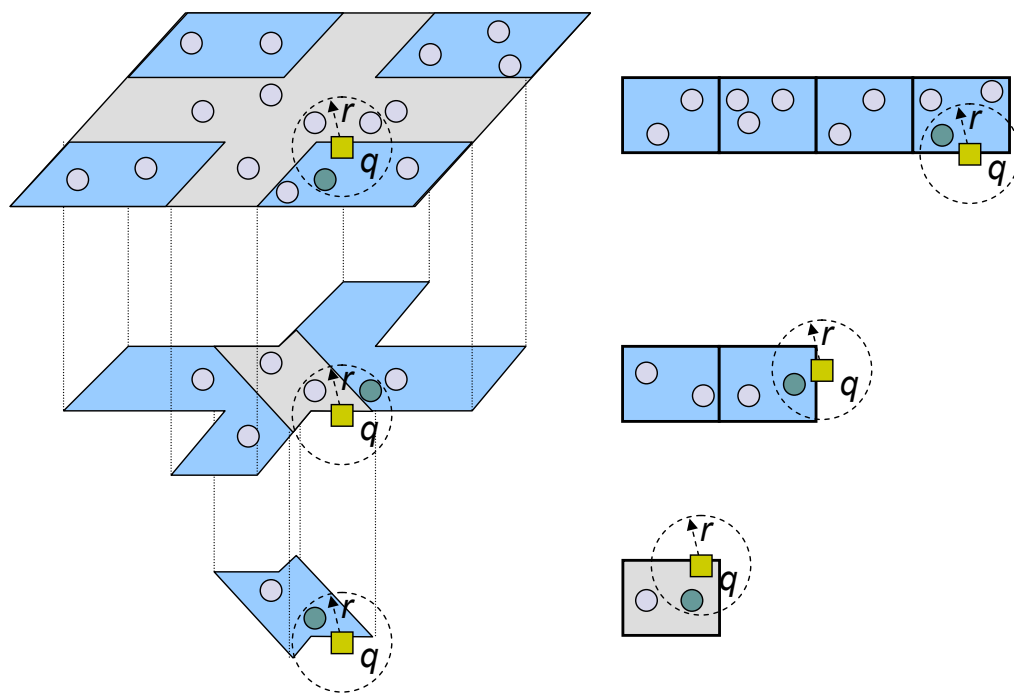
SEBD06, Portonovo, June, 2006

D-index: Insertion



SEBD06, Portonovo, June, 2006

D-index: Range Search



SEBD06, Portonovo, June, 2006

Metric Space Index Structures



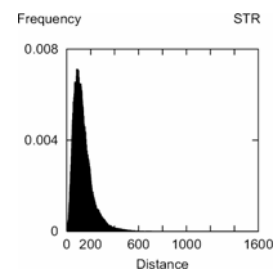
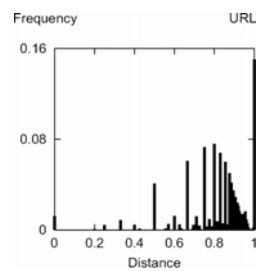
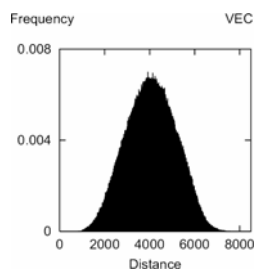
- Metric space $\mathcal{M}=(\mathcal{D},d)$ approach is extensible:
 - exact (partial) match, range search, vector space, edit distance, Jaccard coef., Hausdorff distance,
 - due to the *positivity*, *symmetry*, and *triangular ineq.* of distances, the metric space **is indexable**
- but:
“the response time grows with the data size”

SEBD06, Portonovo, June, 2006

Performance of Centralized Indexes



- Real-life data sets
 - 45-dimensional vectors (quadratic form distance)
 - sets of URL addresses from IS MUNI (Jaccard's coefficient)
 - sentences from the Czech corpus (edit distance)



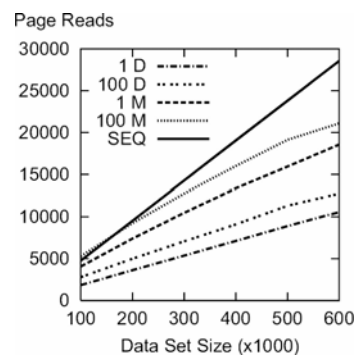
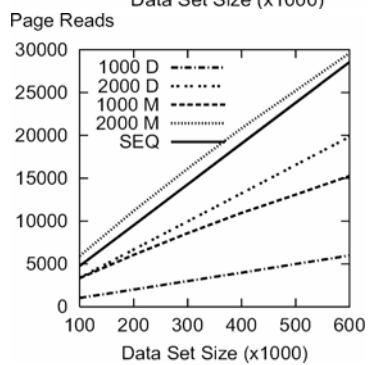
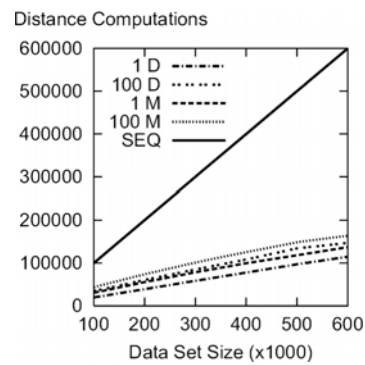
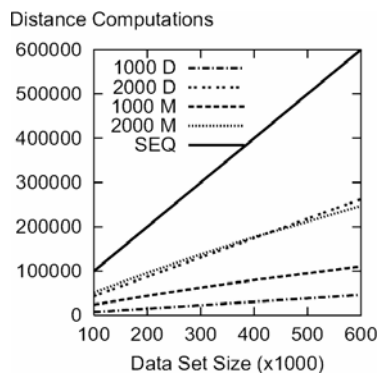
SEBD06, Portonovo, June, 2006

Scalability: D-index, M-tree, and Seq.



range search

k-NN search



SEBD06, Portonovo, June, 2006

Implementation Postulates of Distributed Indexes



- **scalability** – nodes (computers) can be added (removed)
- **no hot-spots** – no centralized nodes, no flooding by messages
- **update independence** – network update at one site does not require an immediate change propagation to all the other sites

SEBD06, Portonovo, June, 2006

Distributed Similarity Search Structures



- Native metric structures:
 - GHT* (Generalized Hyperplane Tree)
 - VPT* (Vantage Point Tree)
- Transformation approaches:
 - M-CAN (Metric Content Addressable Network)
 - M-Chord (Metric Chord)

SEBD06, Portonovo, June, 2006

GHT* Architecture



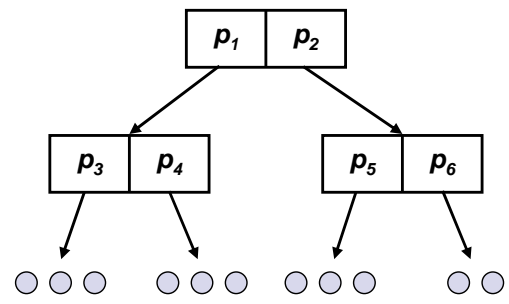
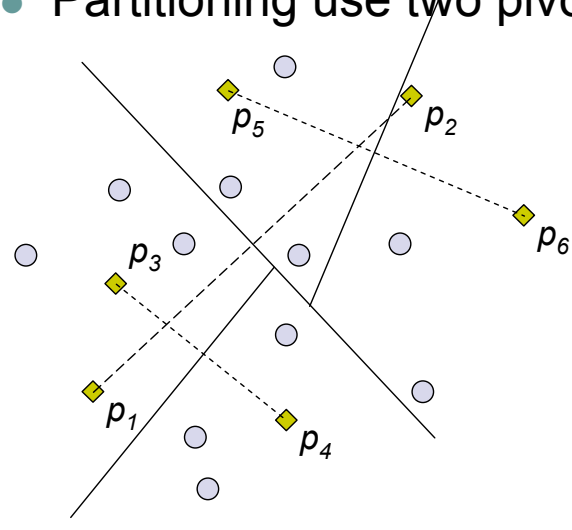
- Peers
 - pose queries, inserts and updates of objects
 - store data in buckets - process queries
 - unique identifier *NNID*
- Buckets
 - limited storage for metric data
 - multiple buckets per peer
 - identified by *BID*, unique within a peer
- Address search tree (AST)
 - tree-based navigation structure present in every peer
 - object localization in several subsequent steps

SEBD06, Portonovo, June, 2006



GHT* Address Search Tree

- Based on the Generalized Hyperplane Tree
 - Partitioning use two pivots per inner node

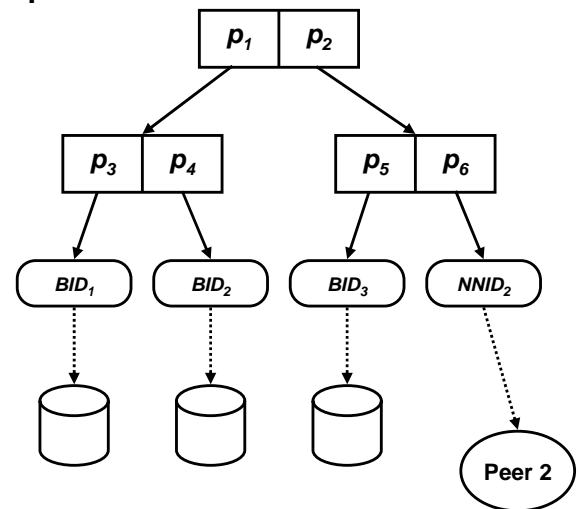


SEBD06, Portonovo, June, 2006

GHT* Address Search Tree

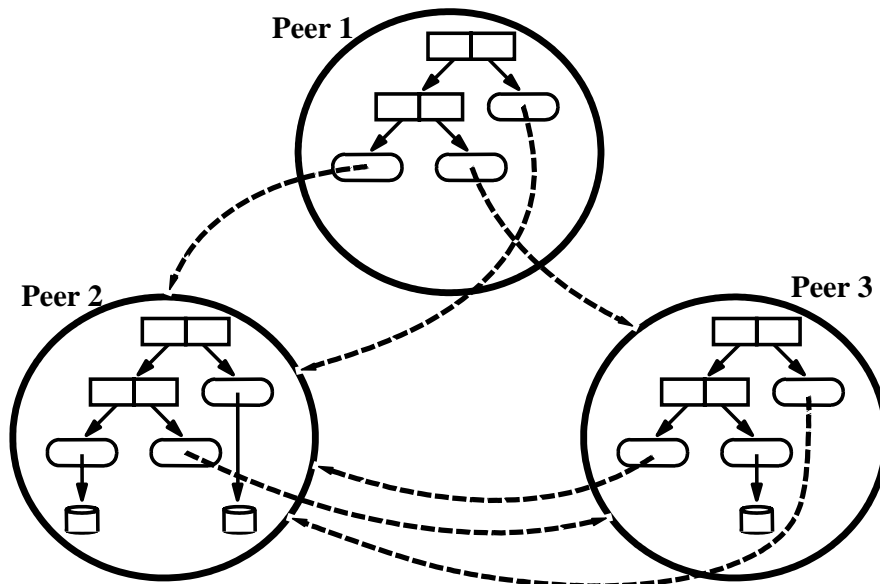


- Based on the Generalized Hyperplane Tree
 - partitioning using two pivots per inner node
- Leaf node
 - *BID* pointer to a bucket
 - data partition stored on this peer
 - *NNID* pointer to a peer
 - data partition stored on a different peer



SEBD06, Portonovo, June, 2006

GHT* Address Search Tree

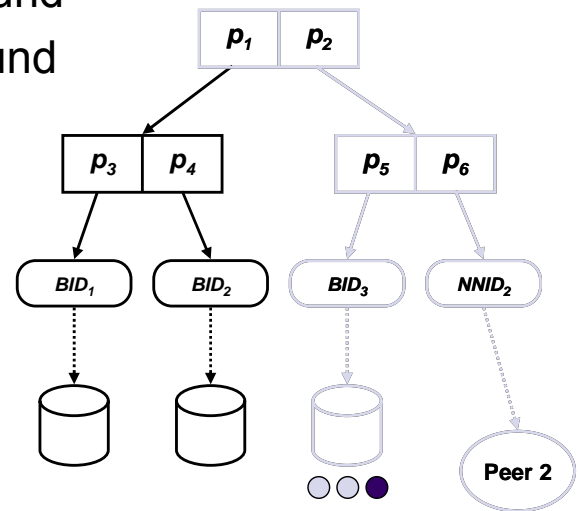
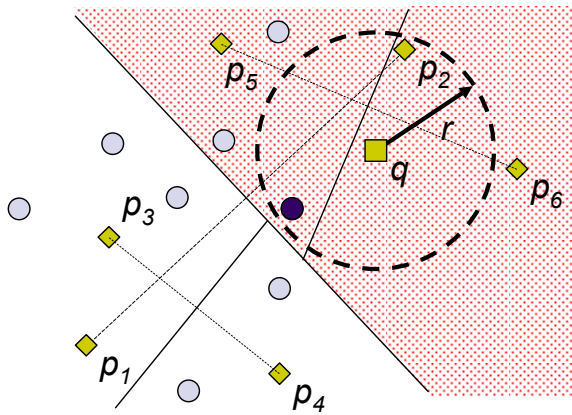


SEBD06, Portonovo, June, 2006



GHT* Range Query

- Range query $R(q,r)$
 - traverse peer's own AST
 - search buckets for all *BIDs* found
 - forward query to all *NNIDs* found

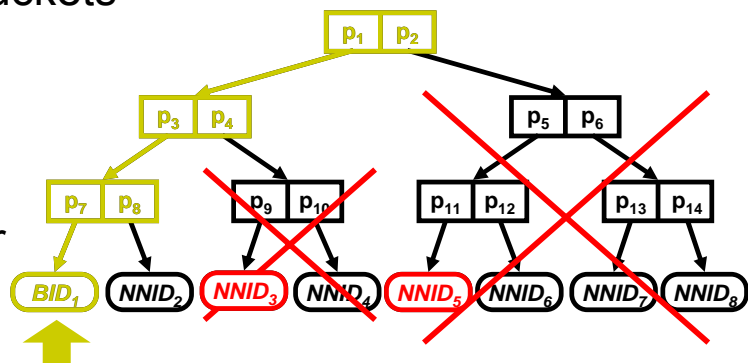


SEBD06, Portonovo, June, 2006



AST – Logarithmic replication

- Full AST on every peer is space consuming
 - Many pivots must be replicated at each peer
- Only a limited AST stored
 - All paths to local buckets
 - Nothing more
- Unknown parts
 - Replaced by *NNID* of the leftmost peer

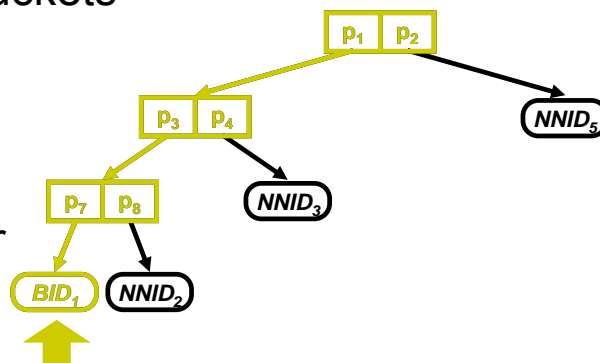


SEBD06, Portonovo, June, 2006



AST – Logarithmic replication

- Full AST on every peer is space consuming
 - Many pivots must be replicated at each peer
- Only a limited AST stored
 - All paths to local buckets
 - Nothing more
- Unknown parts
 - Replaced by *NNID* of the leftmost peer

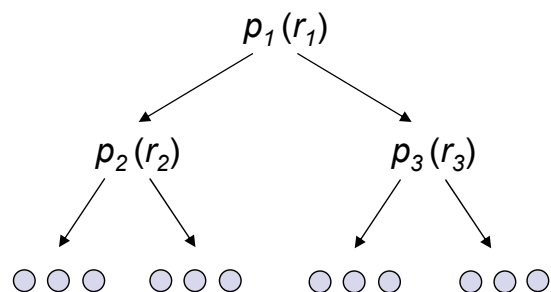
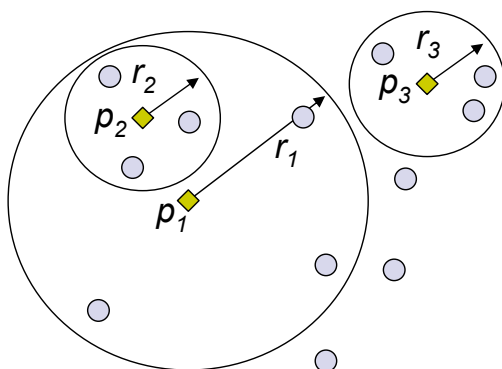


SEBD06, Portonovo, June, 2006

VPT* Structure



- Similar to the GHT* - ball partitioning is used
 - Based on the Vantage Point Tree
 - inner nodes have one pivot and a radius
 - different traversing conditions



SEBD06, Portonovo, June, 2006



M-CAN: The Metric CAN

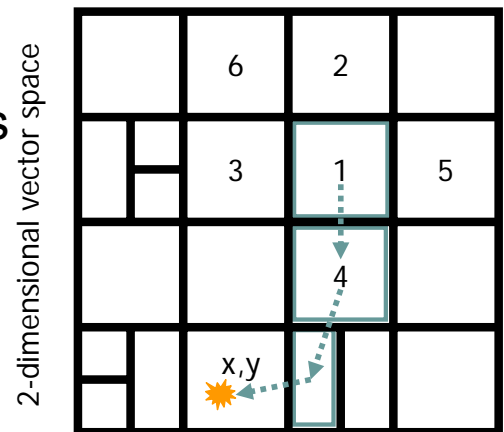
- Based on the Content-Addressable Network (CAN)
 - a DHT navigating in an N -dimensional vector space
- The Idea:
 1. Map the metric space to a *vector space*
 - given N pivots p_1, \dots, p_N
$$F : D \rightarrow R^N, F(o) = (d(o, p_1), d(o, p_2), \dots, d(o, p_N))$$
 2. Use CAN to
 - distribute the vector space zones among the nodes
 - navigate in the network

SEBD06, Portonovo, June, 2006

CAN: Principles & Navigation



- CAN – the principles
 - the space is divided in zones
 - each node “owns” a zone
 - nodes know their neighbors
- CAN – the navigation
 - greedy routing
 - in every step – move to the neighbor *closer* to target location



SEBD06, Portonovo, June, 2006

M-CAN: Contractiveness & Filtering



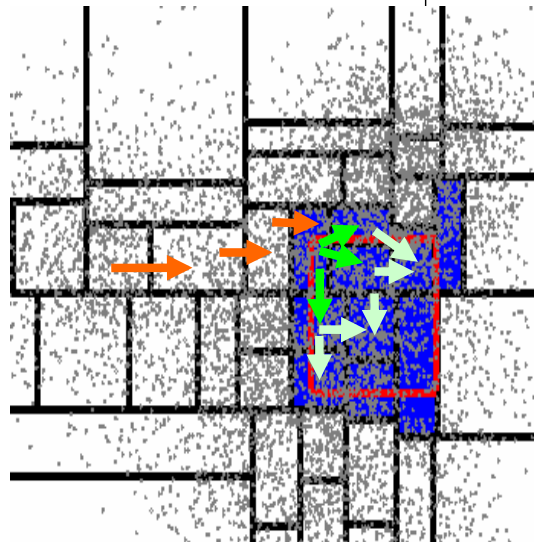
- Use the L_∞ as a distance measure
 - mapping F is *contractive*
$$L_\infty(F(x), F(y)) \leq d(x, y)$$
- More pivots \Rightarrow better filtering
 - but, CAN routing is better for less dimensions
- Additional filtering
 - some pivots are only used for *filtering* data (inside the explored nodes)
 - not used for mapping into CAN vector space

SEBD06, Portonovo, June, 2006

M-CAN: Range Query Execution



- Range query $R(q,r)$
 - map the q on $F(q)$
 - route the query towards $F(q)$
- Reach regions with candidate objects
 - $L_{\infty}(F(x),F(q)) \leq r$
- Propagate the query over the candidate regions
 - using a multicast algorithm of CAN
- Check objects using d



SEBD06, Portonovo, June, 2006

M-Chord: The Metric Chord



- Transform metric space to one-dimensional domain
 - use a generalized version of the *iDistance*
- Divide the domain into intervals
 - assign each interval to a peer
- Use the *Chord* P2P protocol for navigation
- Object filtering
 - inside the explored peers
 - use more pivots

SEBD06, Portonovo, June, 2006

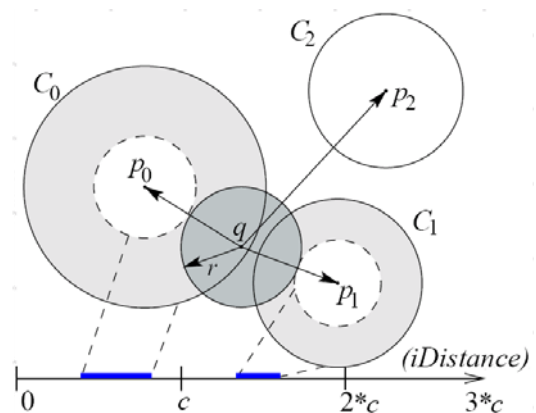
M-Chord: Indexing the Distance



- *iDistance* – indexing technique for vector domains
 - partition the vector space & identify reference points p_i
 - assign *iDistance* keys to objects $x \in C_i$

$$iDist(x) = d(p_i, x) + i \cdot c$$

- range query $R(q,r)$: identify *intervals of interest*
- Generalization to metric spaces
 - select pivots $\{p_1, \dots, p_n\}$
 - then partition: *Voronoi-style*

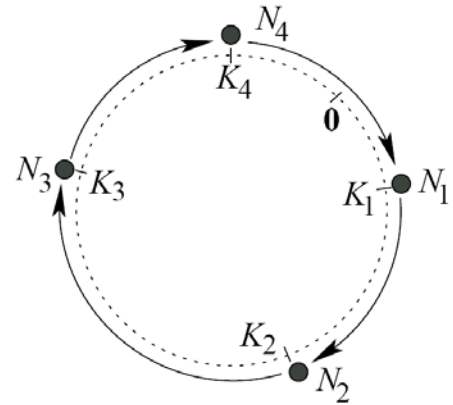


SEBD06, Portonovo, June, 2006



M-Chord: Chord Protocol

- Peer-to-Peer navigation protocol
- Peers are responsible for *intervals* of keys
- $O(\log n)$ hops to localize a node storing a given *key*
- M-Chord
 - set the *iDistance* domain
 - make it *uniform*: function h
$$mchord(x) = h(d(p_i, x) + i \cdot c)$$
- Use *Chord* on this domain



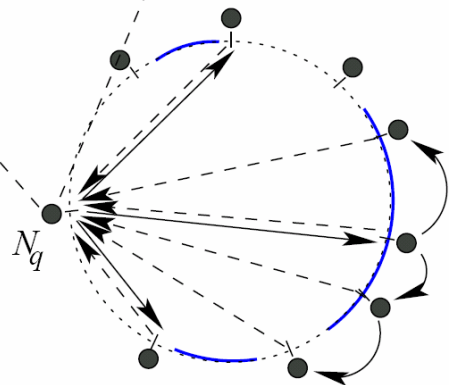
SEBD06, Portonovo, June, 2006



M-Chord: Range Query

- Node N_q initiates the search
- Determine intervals
 - generalized iDistance
- Forward requests to peers on intervals
- Search the nodes
 - using additional filtering
- Merge the received partial answers

range(q, r):
intervals(q, r)
forward(q, r)



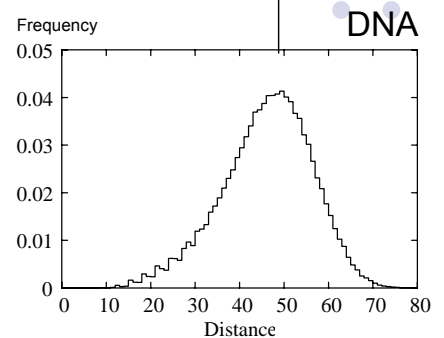
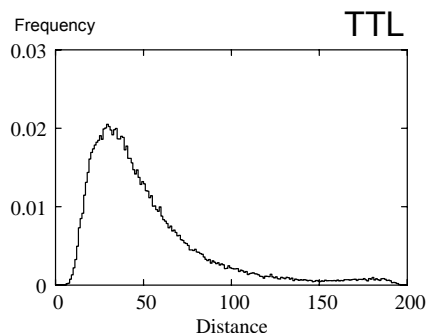
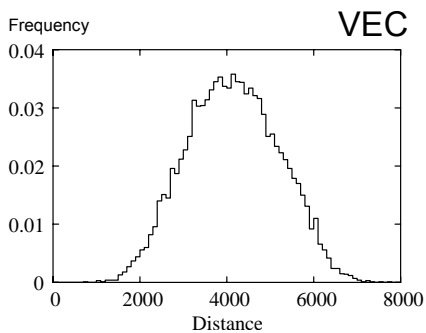
Scalability Performance Evaluation (INFOSCALE 2006)



- Computing infrastructure
 - 300 workstations connected by 100Mbps network
 - Memory based storage
 - Maximum of 5000 objects per peer
- Datasets
 - **VEC**: 45-dimensional vectors of image color features compared by the *quadratic distance* measure
 - **TTL**: titles and subtitles of Czech books and periodicals compared using an *edit distance*
 - **DNA**: protein symbol sequences of length sixteen compared using a *weighted edit distance* (Needleman-Wunsch algorithm)

SEBD06, Portonovo, June, 2006

Datasets: Distance Distribution



- Distribution of the distances within the datasets
 - VEC: distances distributed practically normally
 - TTL: skewed distribution
 - DNA: discrete metric function with small variety of values

SEBD06, Portonovo, June, 2006

Measurements: CPU Costs



- Total distance computations
 - on all peers engaged
 - represents the CPU costs of a centralized structure
- Parallel distance computations
 - the maximal number of distance evaluations performed in a sequential manner
 - represents the real CPU costs

SEBD06, Portonovo, June, 2006

Measurements: Communication Costs



- Total number of messages
 - the number of all messages (requests and responses) sent during a query execution
 - represents the overall network communication load
- Maximal hop count
 - the maximal number of messages sent in a serial way to complete the query
 - represents the parallel communication costs

SEBD06, Portonovo, June, 2006

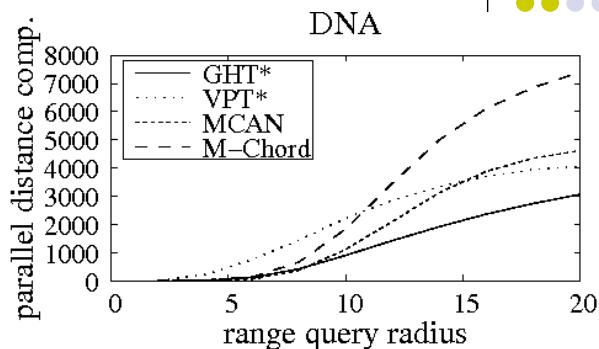
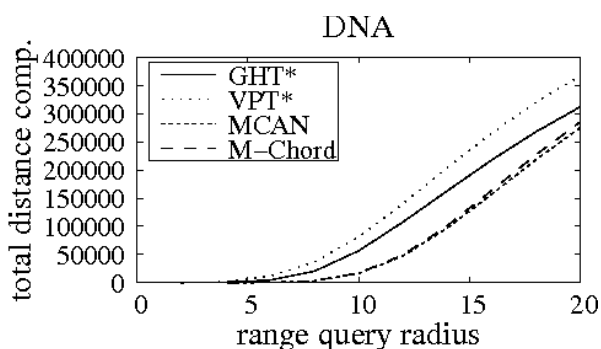
Changing the Query Size



- Range queries with changing radii
 - values averaged over 100 different query objects
 - **VEC** dataset: query radii between 200 and 2,000
 - **TTL** and **DNA** datasets: query radii between 2 and 20
- All four structures filled with 500,000 objects
- Peer storage load factors 60-70%
- Approximately 150 active nodes

SEBD06, Portonovo, June, 2006

Total & Parallel Distance Computations



- Total costs comparable to centralized structures
 - pivot filtering differentiates the structures
- The parallel costs are bounded by 5000 dist. computations
 - limited storage per peer
 - M-Chord in DNA: very small and discrete domain causes multiple mapping collisions (mapping to one key)

SEBD06, Portonovo, June, 2006

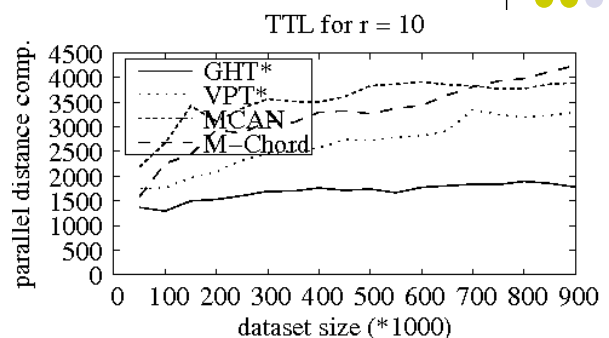
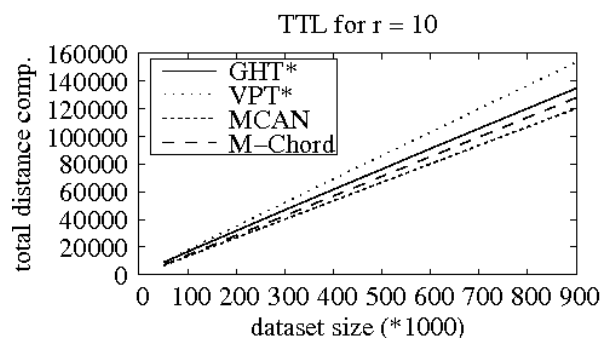
Changing the Dataset Size



- Data volume size from 50,000 to 1,000,000 objects
 - increased in blocks of 50,000 objects
- Range queries executed after every block insertion
 - values averaged over 100 different query objects
 - **VEC** dataset: query radii fixed to 500, 1000 and 1500
 - **TTL** and **DNA** datasets: query radii fixed to 5, 10 and 15
- Measures:
 - distance comp. and messaging costs
 - only selected results with typical behavior shown

SEBD06, Portonovo, June, 2006

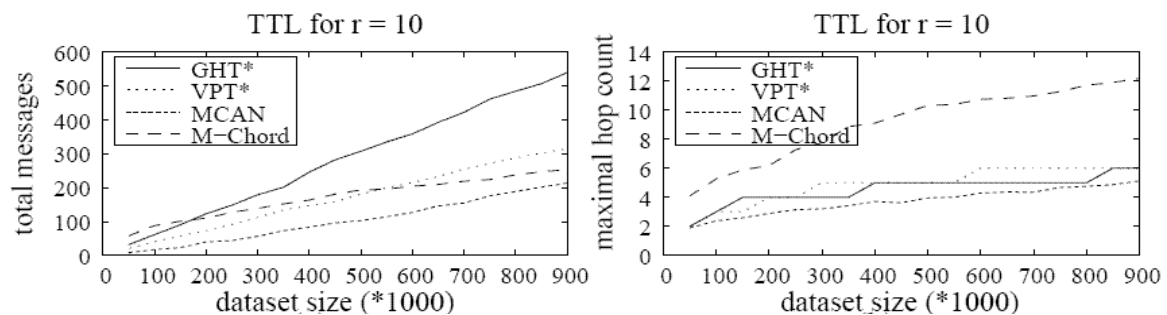
Total & Parallel Distance Computations



- Total distance comp. costs increases linearly
 - M-Chord, MCAN filtering with 50 pivots is most efficient
- Very slow increase in parallel distance computations
 - response times do not increase significantly
 - M-Chord & MCAN: higher costs - relevant objects clustered on peers

SEBD06, Portonovo, June, 2006

Total Messages and Hop Count



- Total messages correlate with visited peers
 - highest for GHT* – many peers visited (declustering)
- Hop counts practically negligible
 - M-Chord significantly worse
 - sequential hops during adjacent clusters search

SEBD06, Portonovo, June, 2006

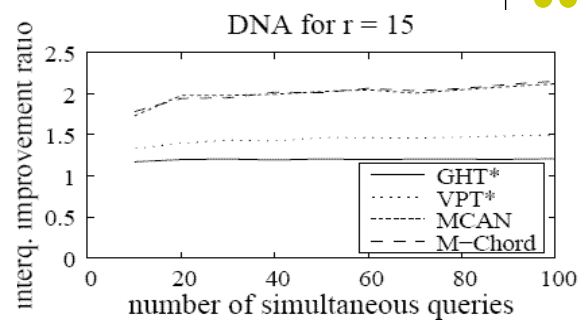
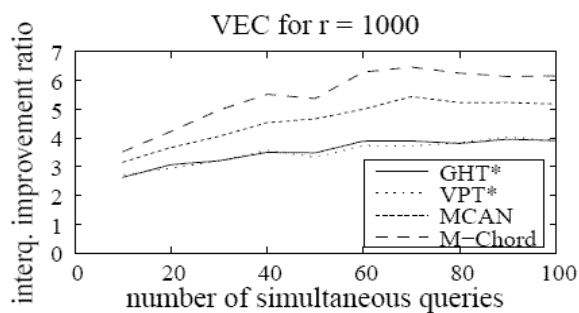
Number of Simultaneous Queries



- Simultaneous queries
 - More queries from different peers at the same time
 - Evaluated in parallel if possible
 - sequential processing of queries at individual peers
- Groups from 10 to 100 range queries
 - Average over multiple groups (of equal size)
 - VEC dataset: query radii fixed to 500, 1000 and 1500
 - TTL and DNA datasets: query radii fixed to 5, 10 and 15
- Data volume of size 500,000 objects

SEBD06, Portonovo, June, 2006

Interquery Improvement Ratio



- Number of simultaneous queries processed without performance degradation
- Influenced by the spread of objects among peers
 - GHT* visits the most peers for each query thus it has the lowest interquery improvement

SEBD06, Portonovo, June, 2006

Concluding Summary



- Compared 4 distributed similarity search structures
 - Query size scalability
 - Dataset size scalability
 - Capability of simultaneous query processing

	single query	multiple queries
GHT*	excellent	poor
VPT*	good	satisfactory
MCAN	satisfactory	good
M-Chord	satisfactory	very good

SEBD06, Portonovo, June, 2006

Recent Book by Springer



Similarity Search

The Metric Space Approach

Series: [Advances in Database Systems](#), Vol. 32

Zežula, P., Amato, G., Dohnal, V., Batko, M.

2006, XVIII, 220 p., Hardcover

ISBN: 0-387-29146-6

January 2006

SEBD06, Portonovo, June, 2006

Recent Book by Springer



Table of contents:

Dedication.- Foreword.- Preface.- Acknowledgements.

Part I Metric Searching in a Nutshell:

- *Foundations of Metric Space Searching.*
- *Survey of Existing Approaches.*

Part II Metric Searching in Large Collections of Data:

- *Centralized Index Structures.*
- *Approximate Similarity Search.*
- *Parallel and Distributed Indexes.*

References.- Author Index.- Index.- Abbreviations.

SEBD06, Portonovo, June, 2006