Indexing High-Dimensional Space:

Database Support for Next Decade´s Applications

Stefan Berchtold ***AT&T Research***
berchtol@research.att.com

Daniel A. Keim ***University of Halle-Wittenberg***
keim@informatik.uni-halle.de

Modern Database Applications

- Multimedia Databases
  - – large data set
  - – content-based search
  - – feature-vectors
  - – high-dimensional data
- Data Warehouses
  - – large data set
  - – data mining
  - – many attributes
  - – high-dimensional data

Overview

Effects in High-Dimensional Spaces

- Exponential dependency of measures on the dimension
- Boundary effects
- No geometric imagination
  - ⇨ Intuition fails

The Curse of Dimensionality

Assets

- N data items
- d dimensions
- data space $[0, 1]^d$
- q query (range, partial range, NN)
- uniform data
- but not: N exponentially depends on d

Exponential Growth of Volume

- Hyper-cube

$$Volume_{cube}(edge, d) = edge^d$$

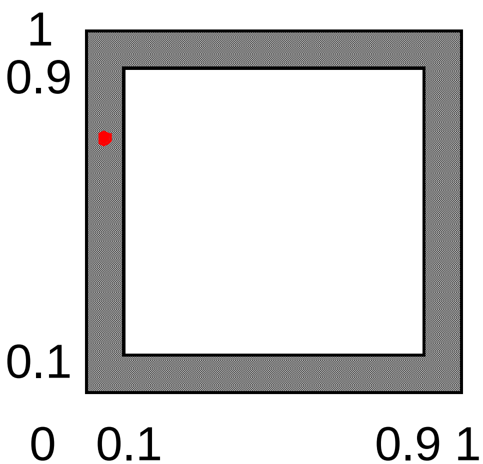
$$Diagonal_{cube}(edge, d) = edge \cdot \sqrt{d}$$

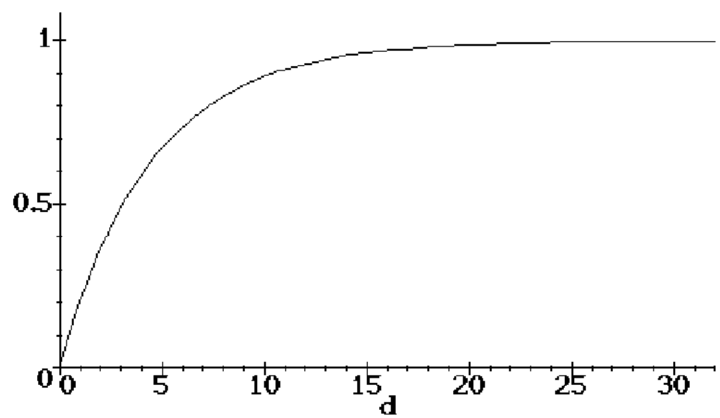
- Hyper-sphere

$$Volume_{sphere}(radius, d) = radius^d \cdot \frac{\sqrt{\pi^d}}{\Gamma(d/2+1)}$$

The Surface is Everything

- Probability that a point is closer than 0.1 to a (*d*-1)-dimensional surface

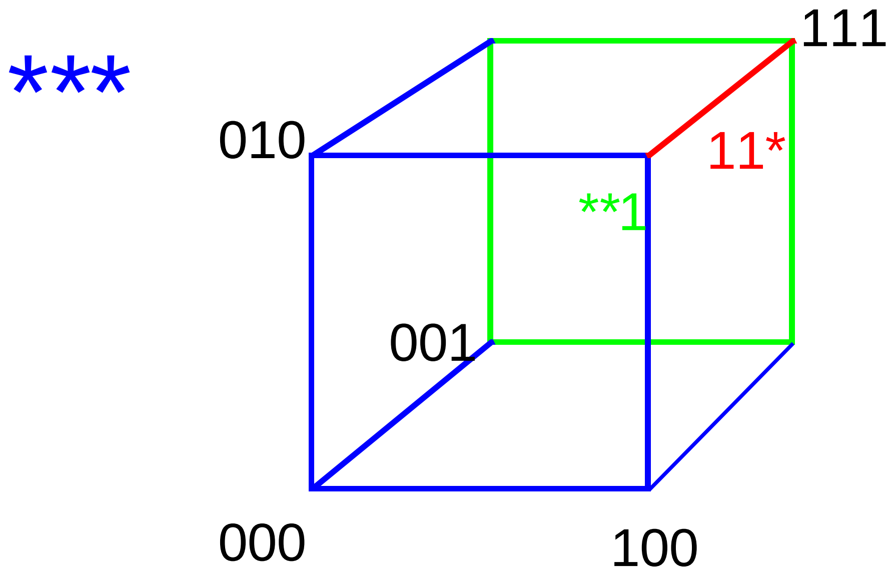




Number of Surfaces

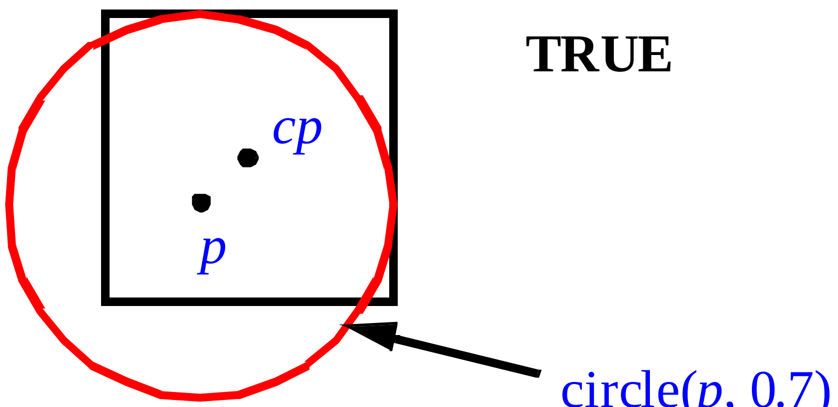
- How much *k*-dimensional surfaces has a *d*-dimensional hypercube $[0..1]^d$?

$$\binom{d}{k} \cdot 2^{(d-k)}$$



"Each Circle Touching All Boundaries Includes the Center Point"

- d-dimensional cube $[0, 1]^d$
- cp = (0.5, 0.5, ..., 0.5)
- p = (0.3, 0.3, ..., 0.3)
- 16-d: circle (p, 0.7), distance (p, cp)=0.8



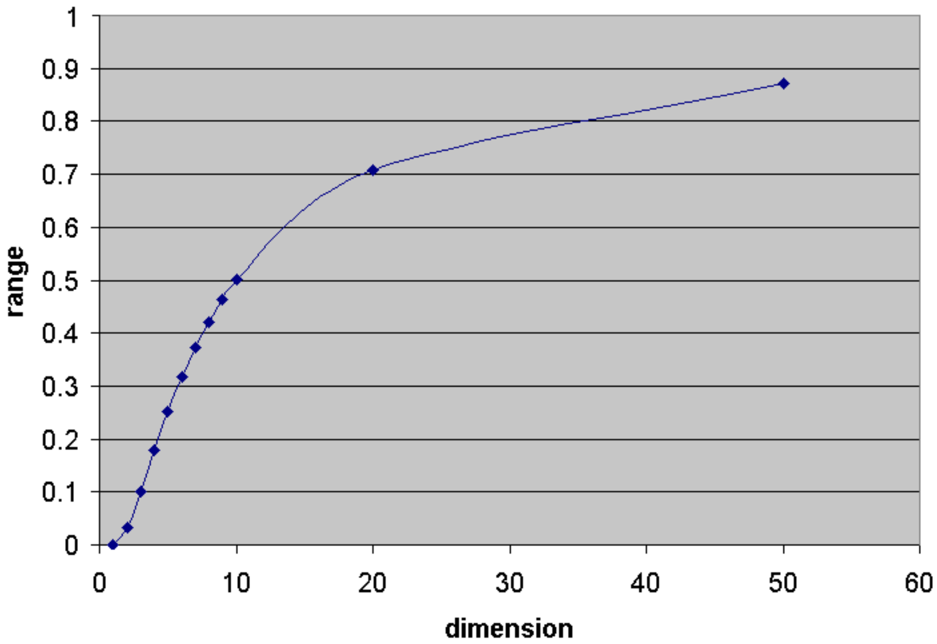
Database-Specific Effects

- Selectivity of queries
- Shape of data pages
- Location of data pages

Selectivity of Range Queries

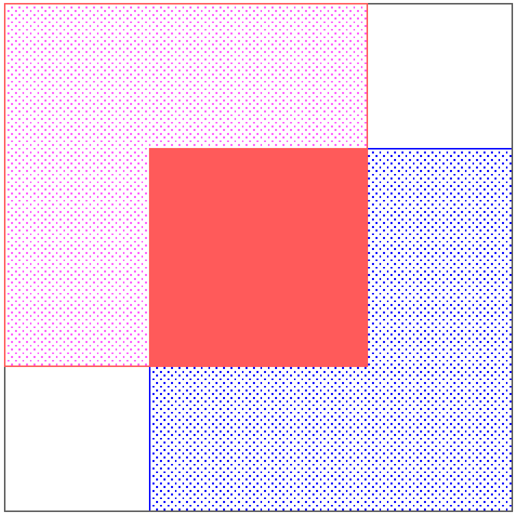
- The selectivity depends on the volume of the query

$$e = \sqrt[d]{Vol_{cube}}$$



Selectivity of Range Queries

- In high-dimensional data spaces, there exists a region in the data space which is affected by ANY range query (assuming uniformity)



Shape of Data Pages

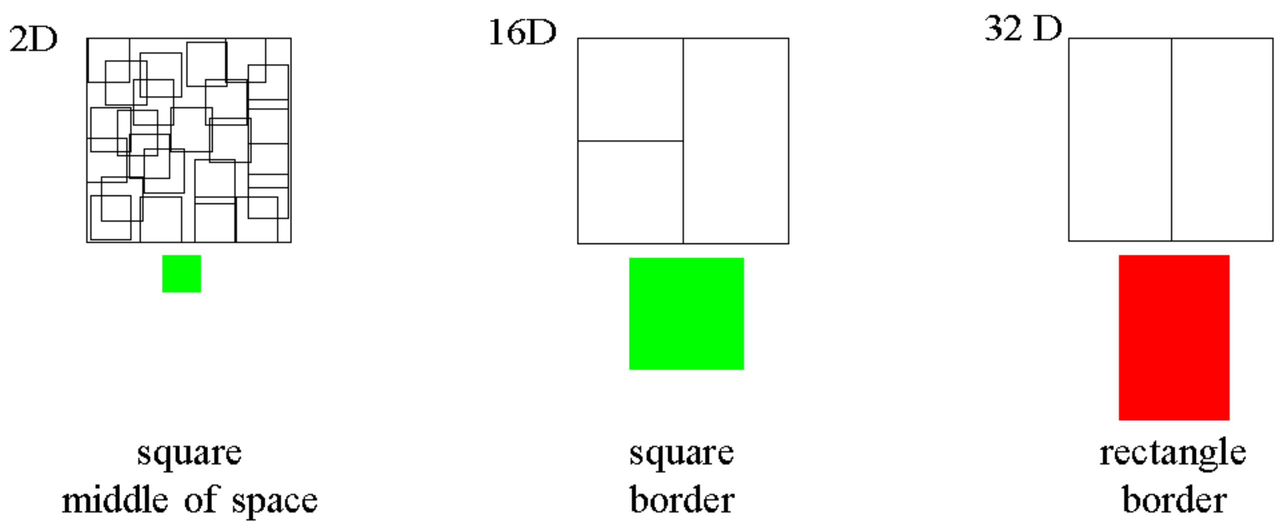
- uniformly distributed data
  ⇨ each data page has the same volume
- split strategy: split always at the 50%-quantile
- number of split dimensions:

$$d' = \log_2\left(\frac{N}{C_{eff}(d)}\right)$$

- extension of a “typical” data page: 0.5 in *d’* dimensions, 1.0 in (*d*-*d’*) dimensions

Location and Shape of Data Pages

- Data pages have large extensions
- Most data pages touch the surface of the data space on most sides



Models for High-Dimensional Query Processing

- Traditional NN-Model [FBF 77]
- Exact NN-Model [BBKK 97]
- Analytical NN-Model [BBKK 98]
- Modeling the NN-Problem [BGRS 98]
- Modeling Range Queries [BBK 98]

Traditional NN-Model

- Friedman, Finkel, Bentley-Model [FBF 77]

  Assumptions:

  – number of data points N goes towards infinity
    (⇨ unrealistic for real data sets)

  – no boundary effects
    (⇨ large errors for high-dim. data)

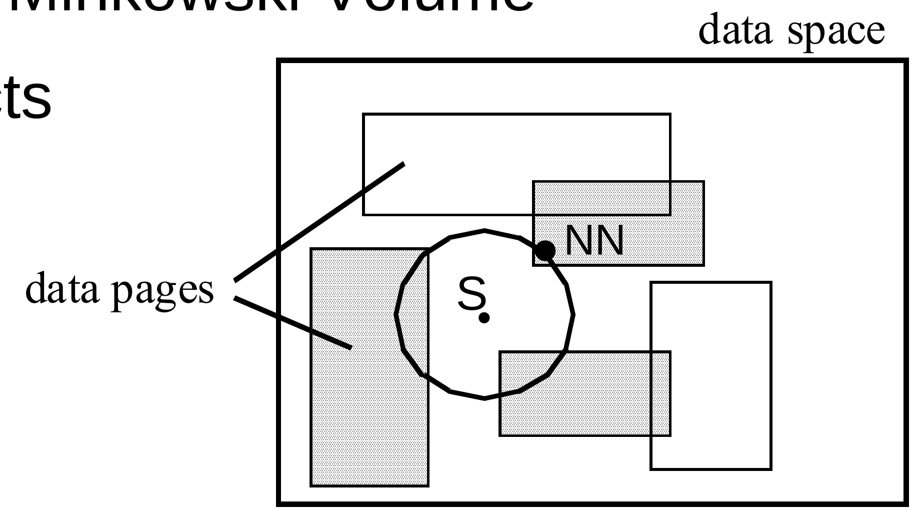
Exact NN-Model [BBKK 97]

- Goal: Determination of the number of data pages which have to be accessed on the average
- Three Steps:

1. Distance to the Nearest Neighbor
2. Mapping to the Minkowski Volume
3. Boundary Effects

Exact NN-Model

1. **Distance to the Nearest Neighbor**
2. Mapping to the Minkowski Volume
3. Boundary Effects



Distribution function

$$P(NN-dist=r)=1-P(None\ of\ the\ N\ points\ intersects\ NN\text{-}sphere)$$

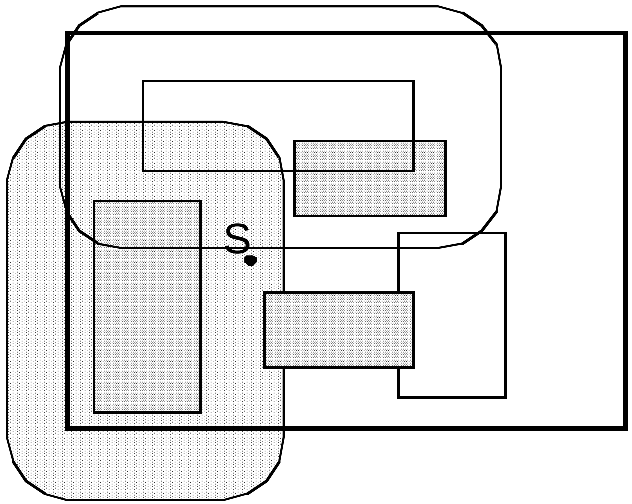
$$=(1-(1-\mathrm{Vol}^{d}_{avg}(r))^{N})$$

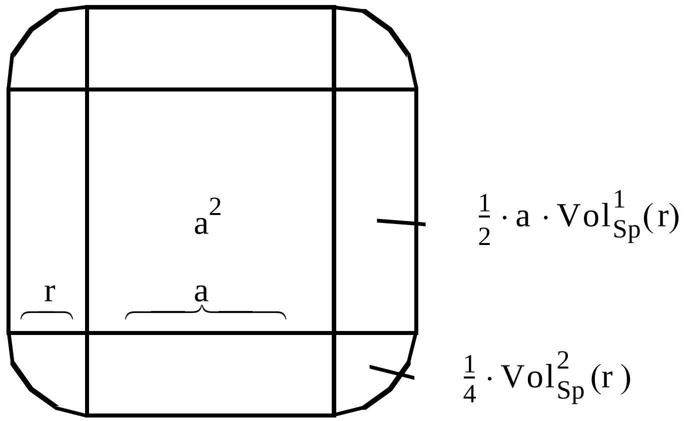
Density function

$$\frac{d}{dr}P(NN-dist=r)=\frac{d}{dr}Vol^{d}_{avg}(r)\cdot N\cdot\left(1-Vol^{d}_{avg}(r)\right)^{N-1}$$

Exact NN-Model

1. Distance to the Nearest Neighbor
2. **Mapping to the Minkowski Volume**
3. Boundary Effects



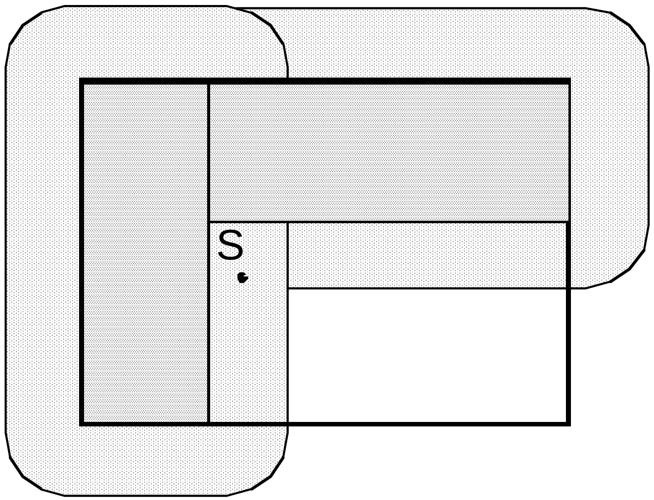


Minkowski Volume:

$$Vol^d_{Mink}(r) = \sum_{i=0}^{d} \binom{d}{i} \cdot a^{d-i} \cdot Vol^i_{Sp}(r)$$

Exact NN-Model

1. Distance to the Nearest Neighbor
2. Mapping to the Minkowski Volume
3. **Boundary Effects**



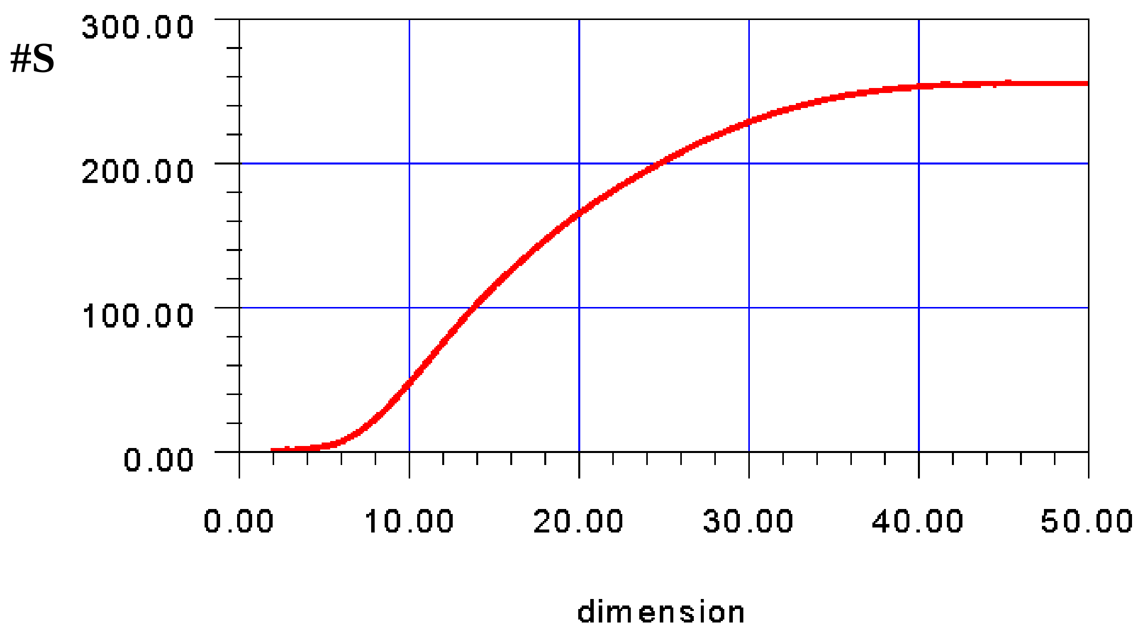
Generalized Minkowski Volume with boundary effects:

$$\#S(r) = \sum_{k=0}^{d'} \sum_{\{i_1, \ldots, i_k\} \in P(\{1, \ldots, d'\})} Vol(SP^k([a_{i_1}, \ldots, a_{i_k}], r) \cap DS) \quad \text{where } d' = \left\lceil \log_2\left(\frac{N}{C_{eff}}\right) \right\rceil$$

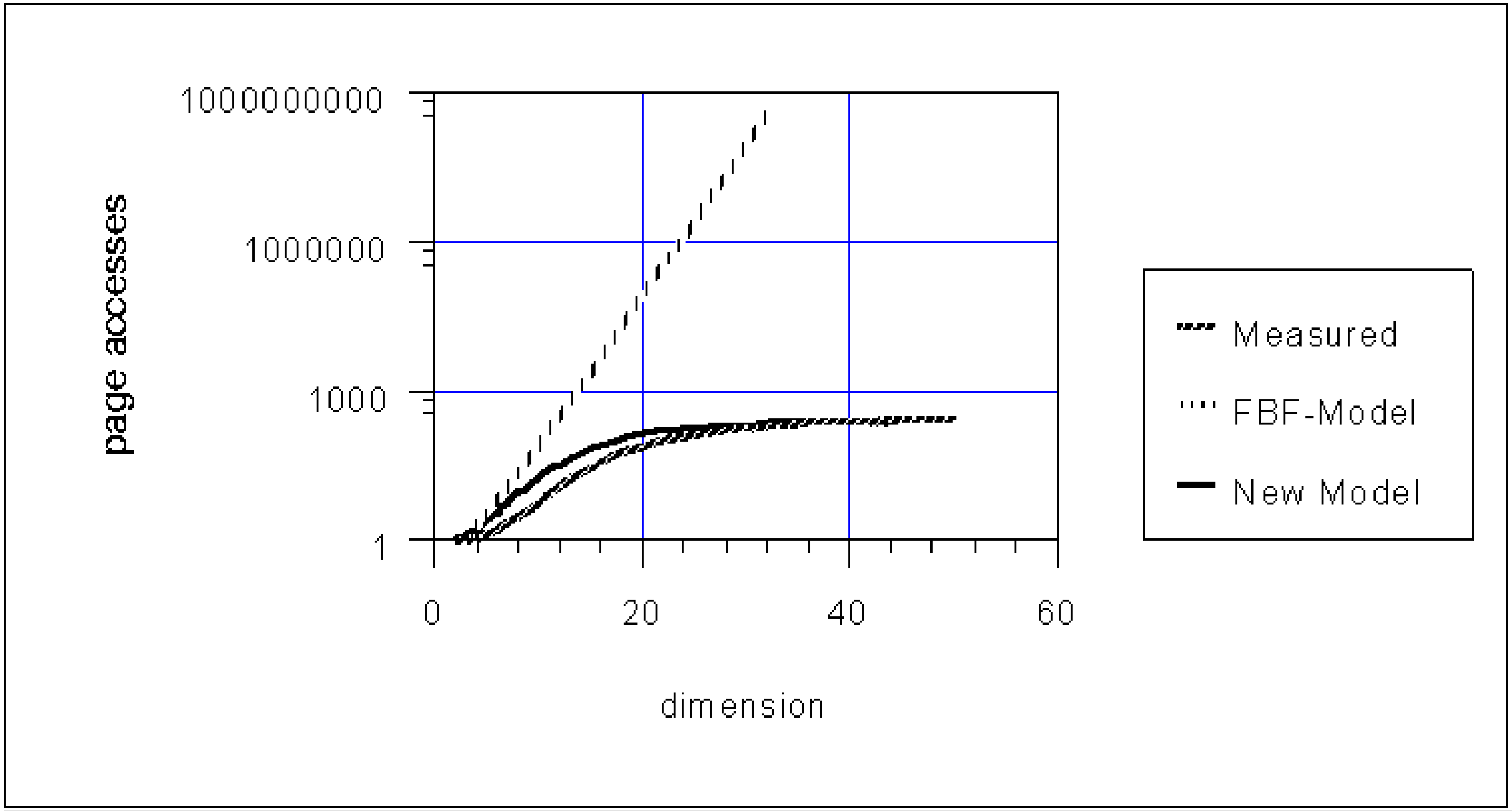
Exact NN-Model

$$E(\#S) = \int_0^\infty \#Pages(r) \cdot p(r)\, dr$$

$$= N \cdot \int_0^\infty \frac{d}{dr} Vol_{avg}^{d}(r) \cdot \left(1 - Vol_{avg}^{d}(r)\right)^{N-1} \cdot \sum_{k=0}^{d'} \sum_{\{i_1, \ldots, i_k\} \in P(\{1, \ldots, d'\})} Vol(SP^k([a_{i_1}, \ldots, a_{i_k}], r) \cap DS)\, dr$$



Comparison
with Traditional Model and Measured Performance



Approximate NN-Model [BBKK 98]

1. Distance to the Nearest-Neighbor

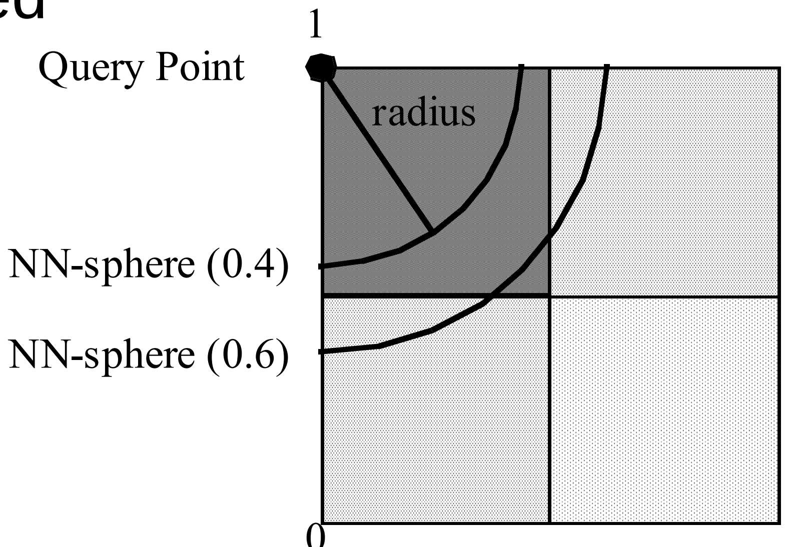
Idea:

Nearest-neighbor Sphere contains 1/*N* of the volume of the data space

$$\mathrm{Vol}^{d}_{Sp}(\mathit{NN\text{-}dist}) = \frac{1}{N} \quad \Rightarrow \mathit{NN\text{-}dist}(N, d) = \frac{1}{\sqrt{\pi}} \cdot \sqrt[d]{\frac{\Gamma(d/2+1)}{N}}$$

Approximate NN-Model

2. Distance threshold which requires more data pages to be considered



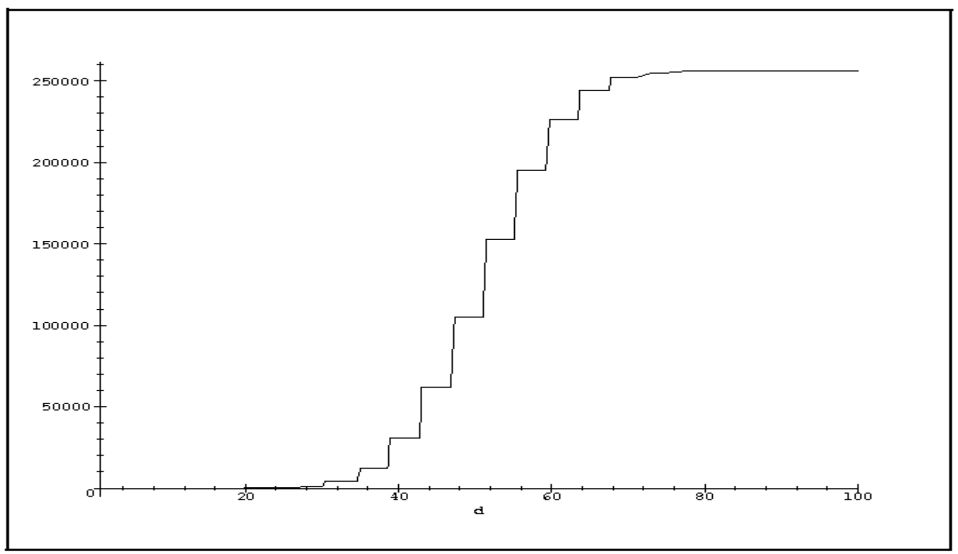
$$\mathit{NN\text{-}dist}(\mathrm{N}, \mathrm{d}) = 0.5 \cdot \sqrt{\mathrm{i}}$$

$$\Leftrightarrow \mathrm{i} = \left(\frac{\frac{1}{\sqrt{\pi}} \cdot \sqrt[\mathrm{d}]{\frac{\Gamma(\mathrm{d}/2 + 1)}{\mathrm{N}}}}{0.5} \right)^2 \Rightarrow \mathrm{i} \approx \frac{2 \cdot \mathrm{d}}{\mathrm{e} \cdot \pi} \cdot \sqrt[\mathrm{d}]{\frac{\pi \cdot \mathrm{d}^3}{4 \cdot \mathrm{N}^2}}$$

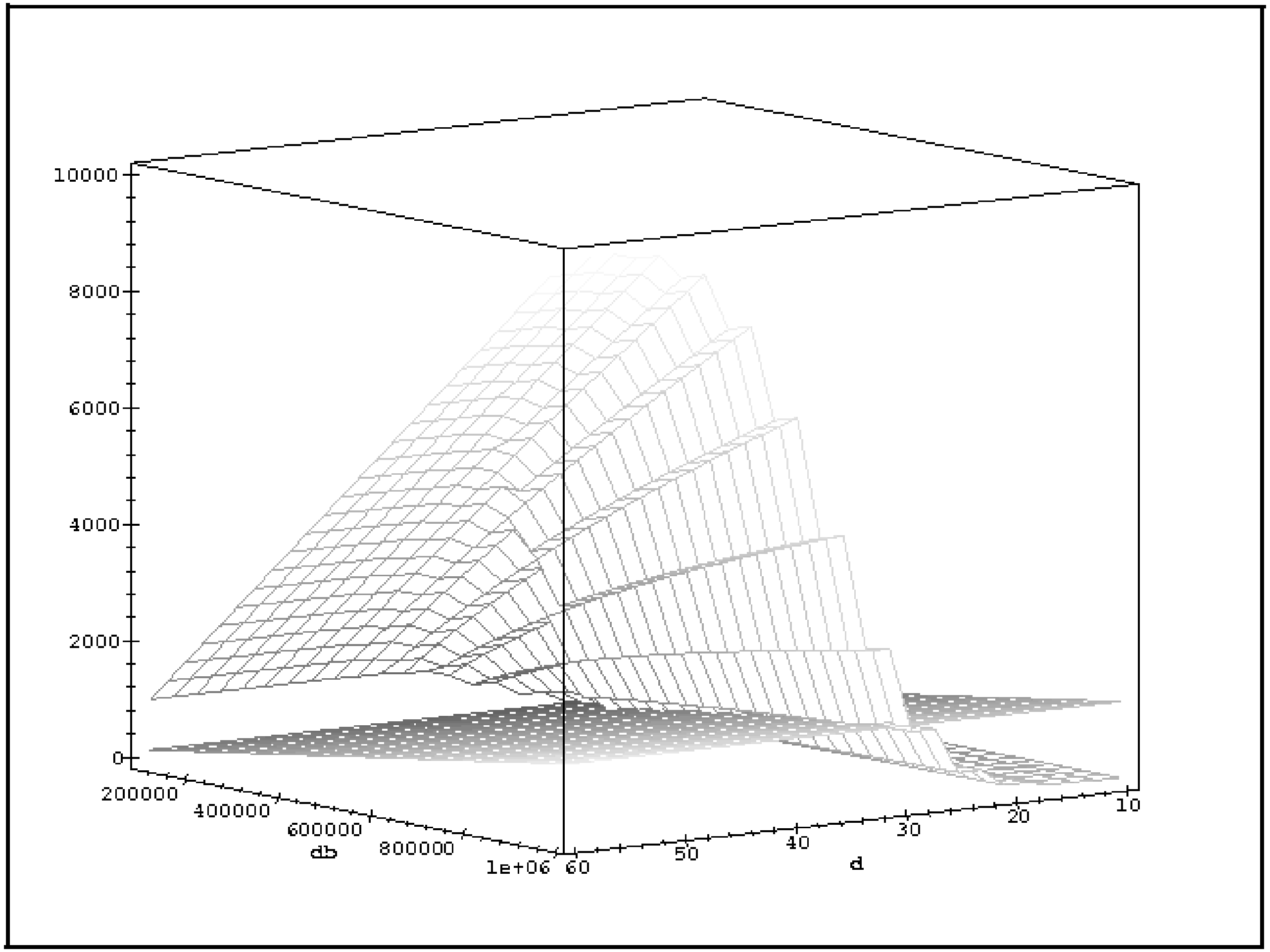
Approximate NN-Model

3. Number of pages

$$\#S(d) = \sum_{k=0}^{\frac{2 \cdot d}{e \cdot \pi} \cdot \sqrt[d]{\frac{\pi \cdot d^3}{4 \cdot N^2}}} \binom{d'}{k} = \sum_{k=0}^{\frac{2 \cdot d}{e \cdot \pi} \cdot \sqrt[d]{\frac{\pi \cdot d^3}{4 \cdot N^2}}} \binom{\left\lceil \log_2\left(\frac{N}{C_{eff}}\right) \right\rceil}{k}$$



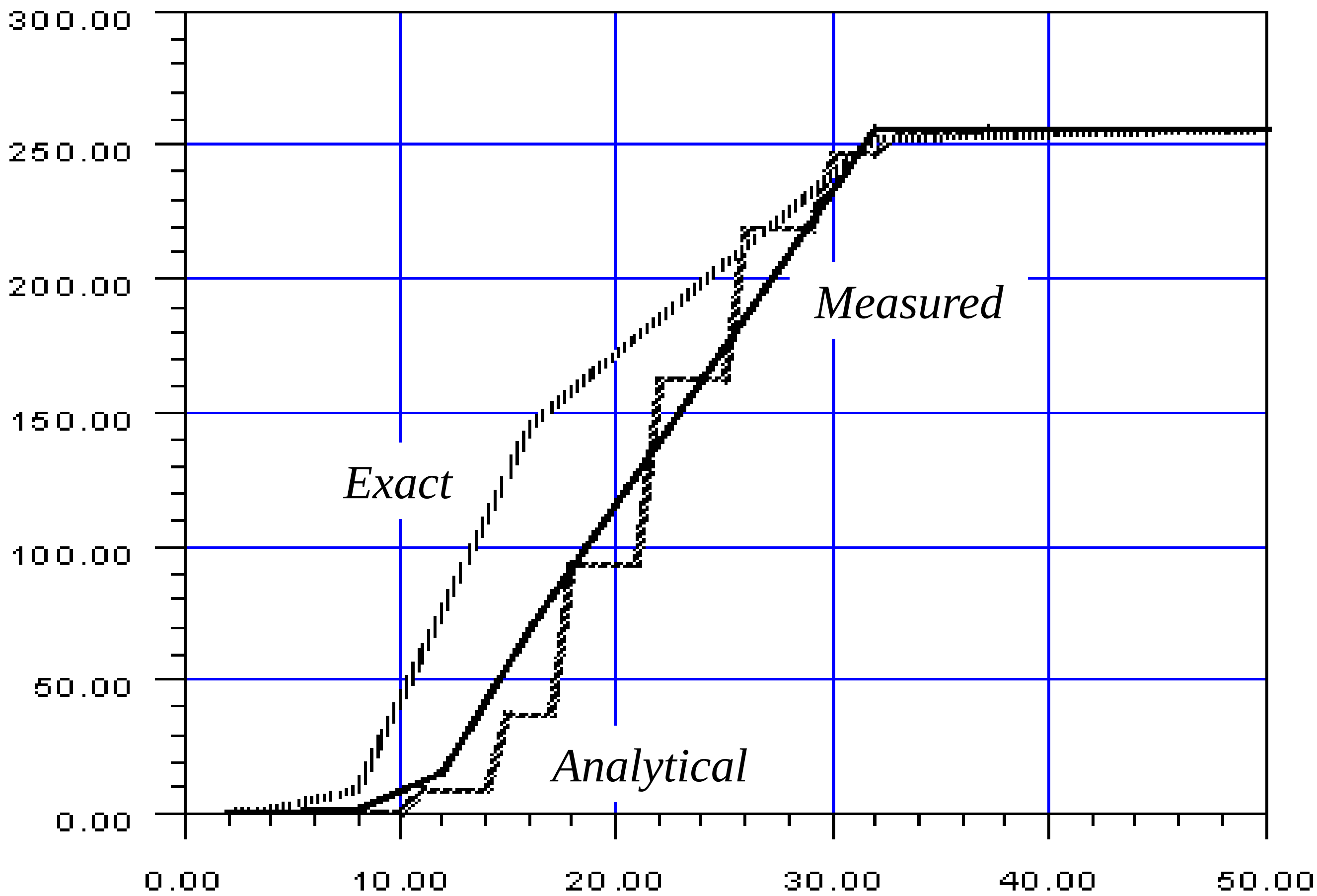
Approximate NN-Model



(depending on the database size and the dimension)

Comparison

with Exact NN-Model and Measured Performance



The Problem of Searching the Nearest Neighbor [BGRS 98]

- *Observations:*
  - When increasing the dimensionality, the nearest-neighbor distance grows.
  - When increasing the dimensionality, the farest-neighbor distance grows.
  - The nearest-neighbor distance grows FASTER than the farest-neighbor distance.
  - For $d \to \infty$, the nearest-neighbor distance equals to the farest-neighbor distance.

When Is Nearest Neighbor meaningful?

- Statistical Model:
- For the *d*-dimensional distribution holds:

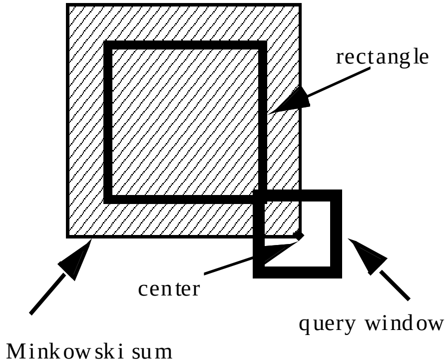
$$\lim_{d \to \infty} (\mathrm{var}(D_d{}^p) / E(D_d{}^p)^2) = 0$$

where *D* is the distribution of the distance of the query point and a data point and we consider a L_p metric.

- This is true for synthetic distributions such as normal, uniform, zipfian, etc.
- This is NOT true for clustered data.

Modeling Range-Queries [BBK 98]

- Idea: Use Minkowski-sum to determine the probability that a data page (URC, LLC) is loaded



$$P_{\text{no\_bound\_eff}}(q) = \sum_{i} \prod_{j=0}^{d-1} (\text{URC}_{i,j} - \text{LLC}_{i,j} + q)$$

$$P_{\text{bound\_eff}}(q) = \sum_{i} \prod_{j=0}^{d-1} \frac{\min(\text{URC}_{i,j}, 1-q) - \max(\text{LLC}_{i,j} - q, 0)}{1-q}$$

Indexing High-Dimensional Space

- Criterions
- kd-Tree-based Index Structures
- R-Tree-based Index Structures
- Other Techniques
- Optimization and Parallelization

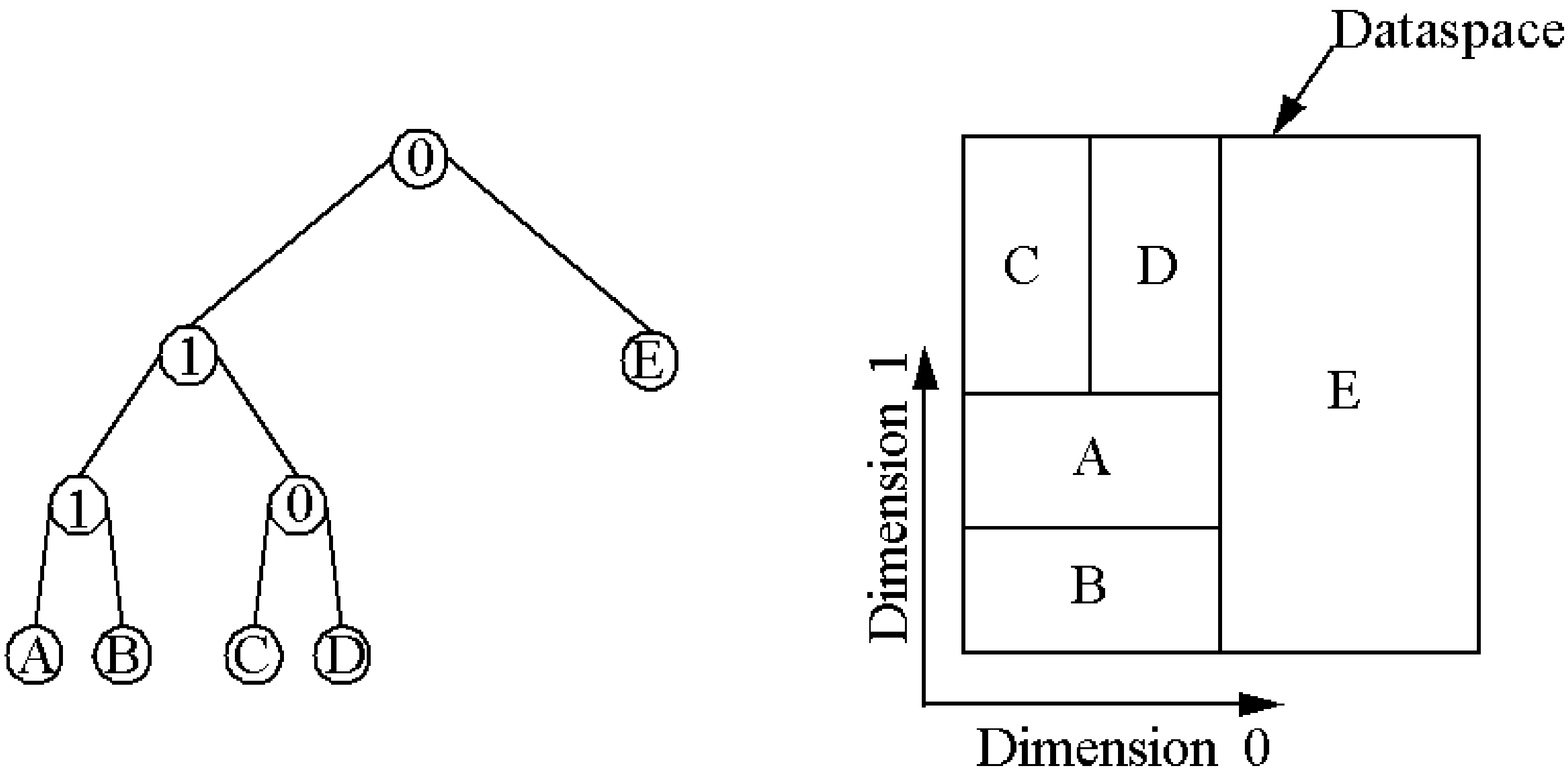
Criterions

- Structure of the Directory
- Overlapping vs. Non-overlapping Directory
- Type of MBR used
- Static vs. Dynamic
- Exact vs. Approximate

The kd-Tree [Ben 75]

- *Idea:*
  Select a dimension, split according to this dimension and do the same recursively with the two new sub-partitions
- *Problem:*
  The resulting binary tree is not adequate for secondary storage
- Many proposals how to make it work on disk (e.g., [Rob 81], [Ore 82] [See 91])

kd-Tree - Example



The kd-Tree

- *Plus:*
  - fanout constant for arbitrary dimension
  - fast insertion
  - no overlap
- *Minus:*
  - depends on the order of insertion (e.g., not robust for sorted data)
  - dead space covered

The kdB-Tree [Rob 81]

- *Idea:*
  – Aggregate kd-Tree nodes into disk pages
  – Split data pages in case of overflow (B-Tree-like)
- *Problem:*
  – splits are not local
  – forced splits

The LSD^h-Tree [Hen 98]

- Similar to kdB-Tree
  (forced splits are avoided)
- Two-level directory:
  first level in main memory
- To avoid dead space:
  only actual data regions are coded

The LSD^h-Tree

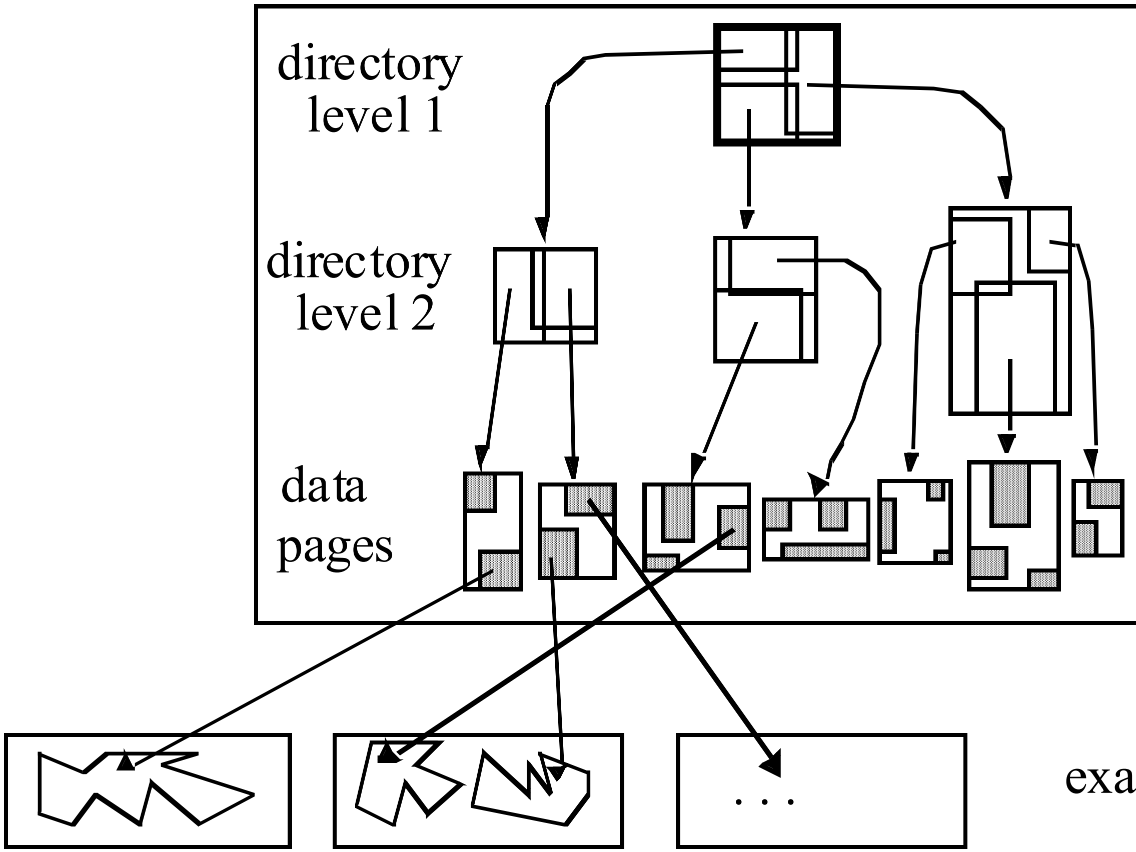
- Fast insertion
- Search performance (NN) competitive to X-Tree
- Still sensitive to pre-sorted data
- Technique of CADR (Coded Actual Data Regions) is applicable to many index structures

The VAMSplit Tree [JW 96]

- Idea:
  Split at the point where maximum variance occurs (rather than in the middle)
- sort data in main memory
- determine split position and recurse
- Problems:
  - data must fit in main memory
  - benefit of variance-based split is not clear

R-Tree: [Gut 84]

The Concept of Overlapping Regions



exact representation

Variants of the R-Tree

Low-dimensional

- R^+-Tree [SRF 87]
- R^*-Tree [BKSS 90]
- Hilbert R-Tree [KF94]

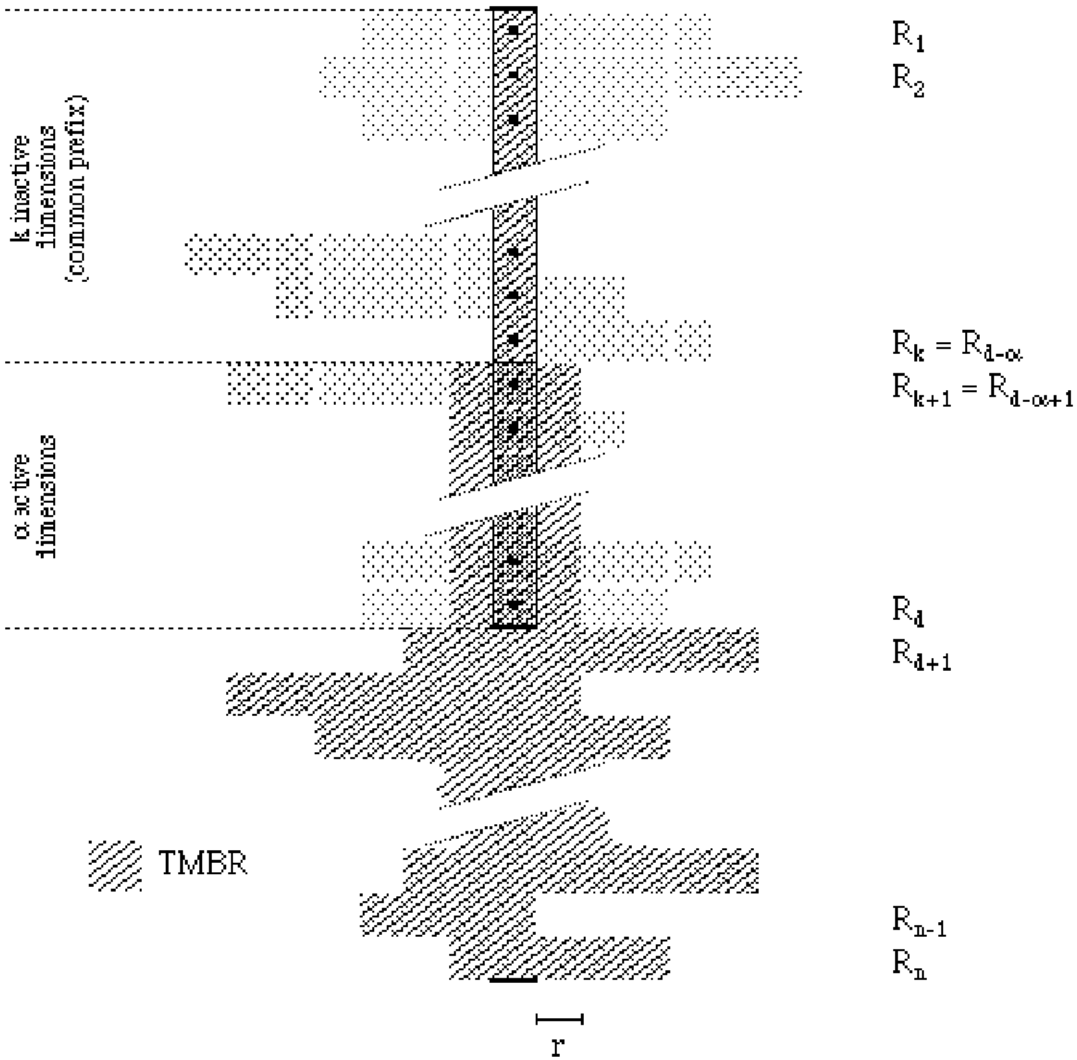
High-dimensional

- TV-Tree [LJF 94]
- X-Tree [BKK 96]
- SS-Tree [WJ 96]
- SR-Tree [KS 97]

The TV-Tree [LJF 94]
(Telescope-Vector Tree)

- Basic Idea: Not all attributes/dimensions are of the same importance for the search process.
- Divide the dimensions into three classes
  - attributes which are shared by a set of data items
  - attributes which can be used to distinguish data items
  - attributes to ignore

Telescope Vectors

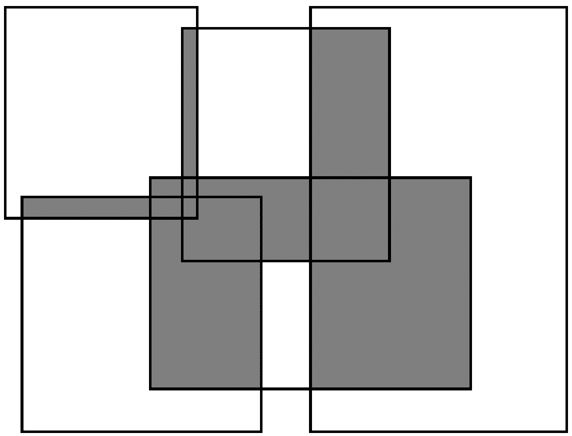


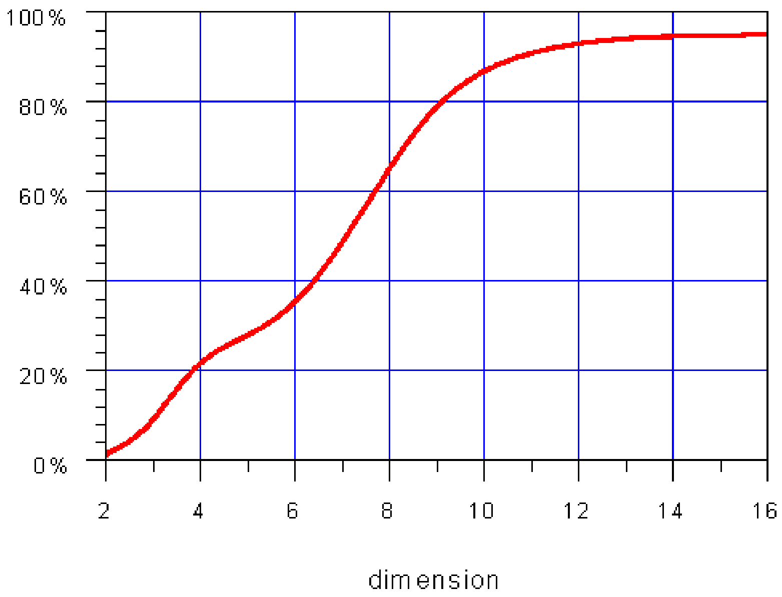
The TV-Tree

- Split algorithm:
  either increase dimensionality of TV or split in the given dimensions
- Insert algorithm: similar to R-Tree
- Problems:
  - how to choose the right metric
  - high overlap in case of most metrics
  - complex implementation

The X-Tree [BKK 96]
(eXtended-Node Tree)

- Motivation:
  Performance of the R-Tree degenerates in high dimensions
- Reason: overlap in the directory



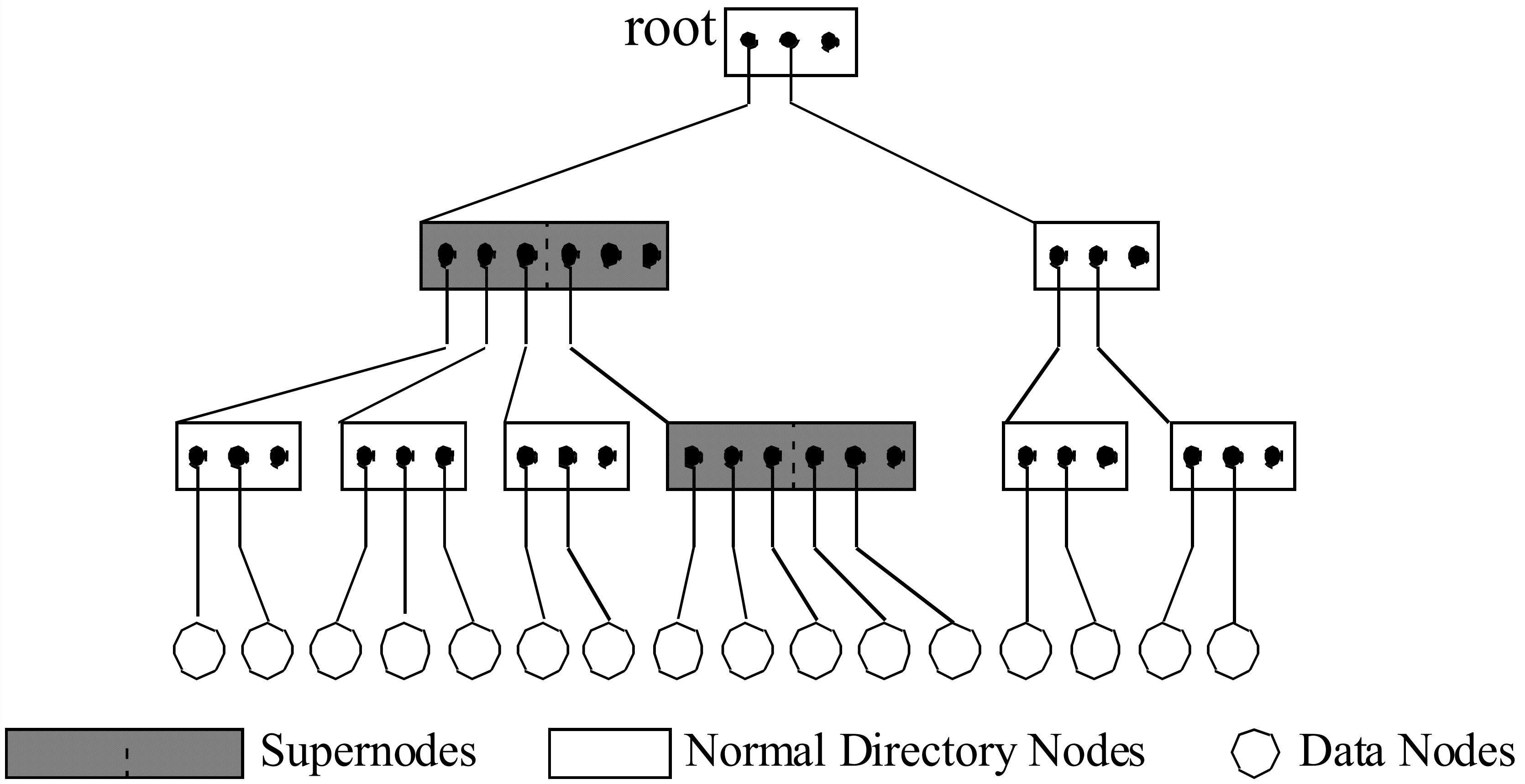


The X-Tree

Idea:

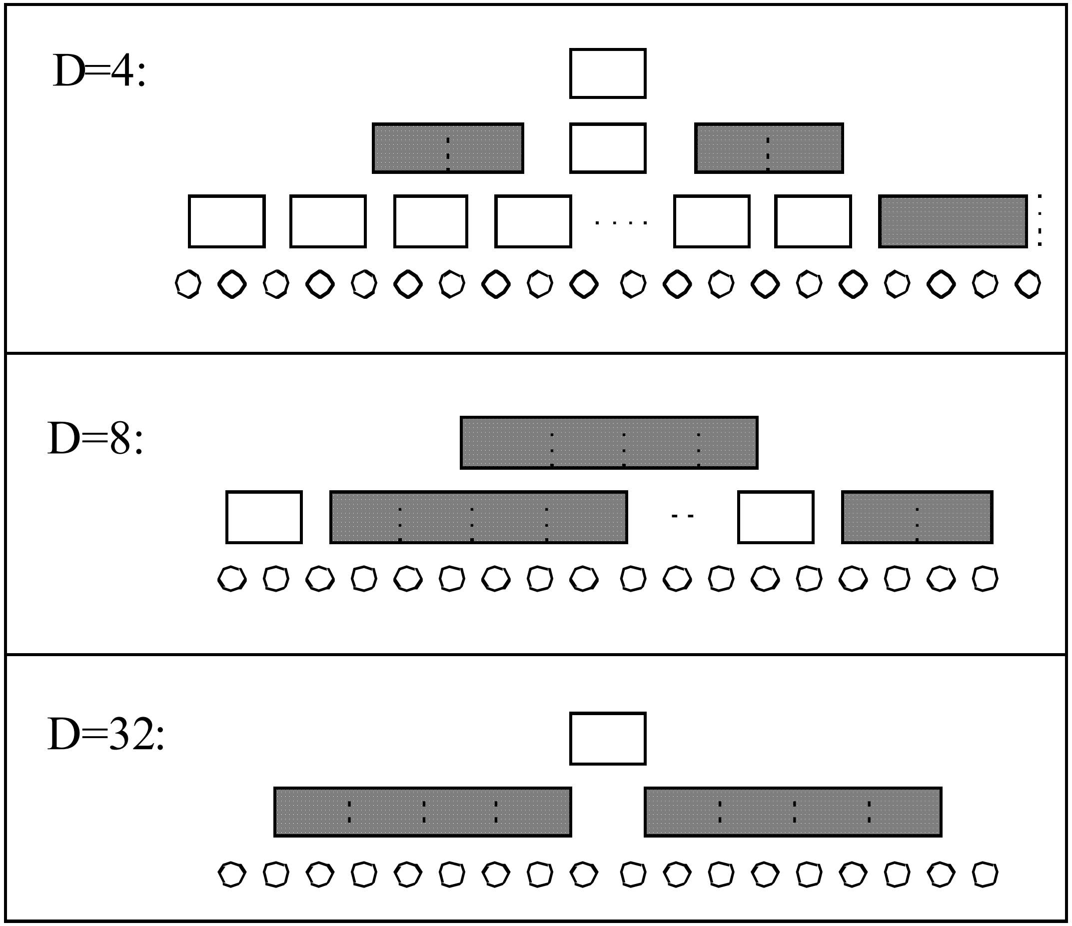
- X-tree avoids overlap in the directory by using
  - an overlap-free split
  - the concept of supernodes
- properties of the X-tree
  - directory nodes have no overlap
  - X-tree is a hybrid of a hierarchical and a linear structure
    - hierarchical organization is best for low dimensionality ($D \leq 5$)
    - linear organization is best for very high dimensionality ($D \geq 32$)
    - X-tree dynamically provides best possible combination for any dimensionality

The X-Tree



The X-Tree

Examples for X-Trees with different dimensionality



The X-Tree

Overlap-Free Split

- **Definition (Split):**

  The split of a node $S = \{mbr_1, \ldots, mbr_n\}$ into two subnodes S_1 and S_2 ($S_1 \neq \emptyset$ and $S_2 \neq \emptyset$) is defined as

  $$Split(S) = \{(S_1, S_2) \mid S = S_1 \cup S_2 \;\wedge\; S_1 \cap S_2 = \emptyset\}.$$

  The split is called

  (1) overlap-minimal iff $\|MBR(S_1) \cap MBR(S_2)\|$ *is minimal*

  (2) overlap-free iff $\|MUR(S_1) \cap MUR(S_2)\| = 0$

  (3) balanced iff $-\varepsilon \leq |S_1| - |S_2| \leq \varepsilon$ (for small ε).

The X-Tree

- **Lemma 1** (for uniformly distributed point data)

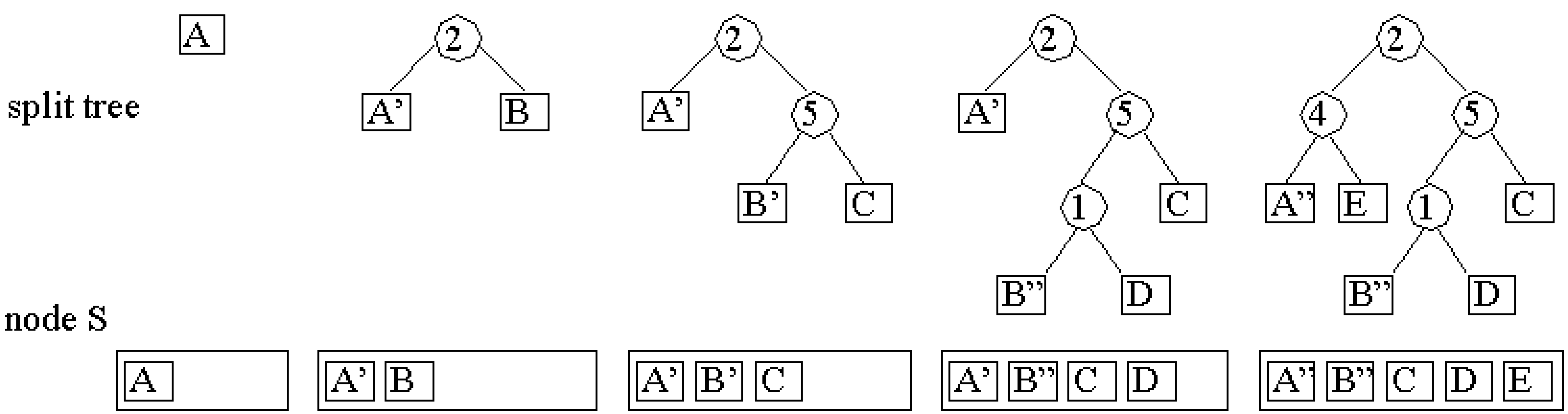
  Split(S) is overlap-free $\Leftrightarrow$

  $\exists\, d \in \{1, \ldots, D\}\ \forall mbr \in S:$ *mbr has been split according to d*

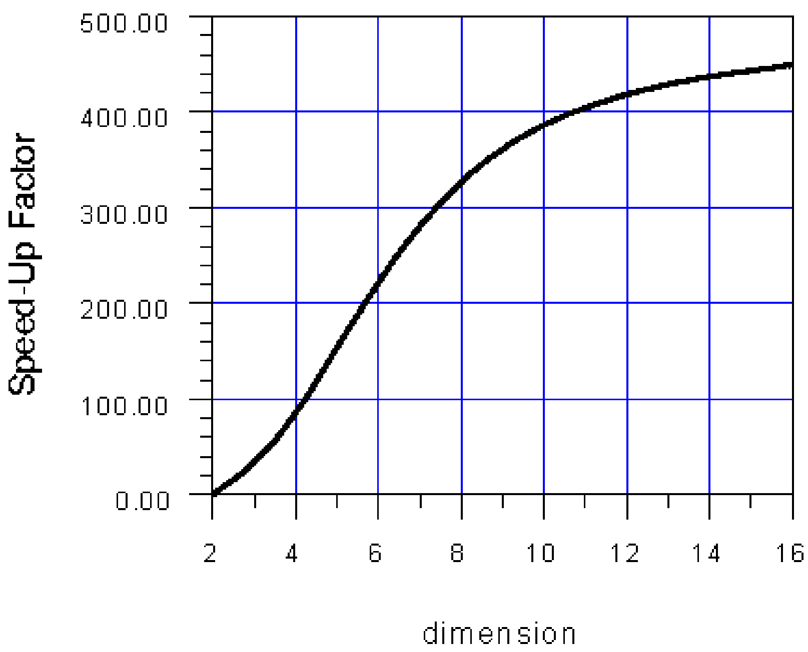
- **Lemma 2**

  For point data, an overlap-free split always exists.

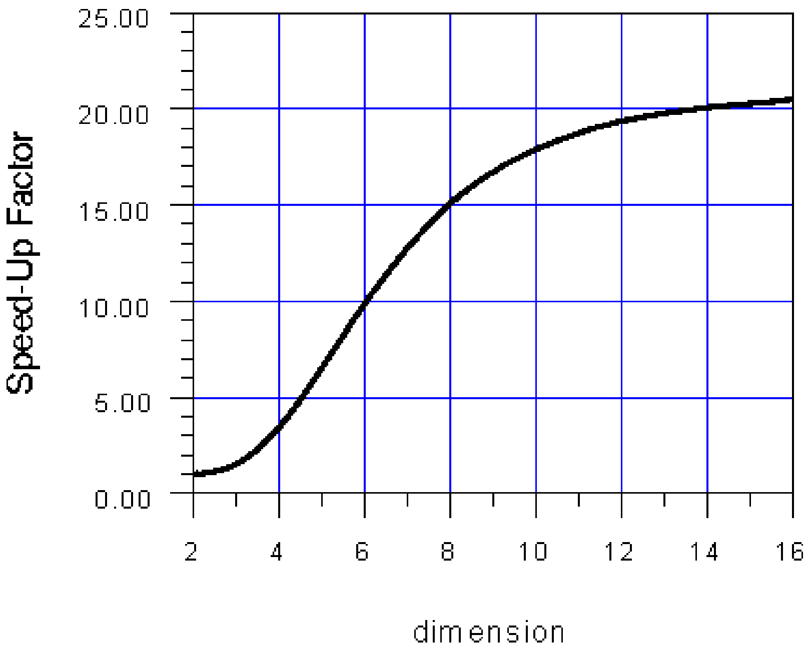
<u>*Example split history:*</u>



Speed-Up of X-Tree over the R*-Tree

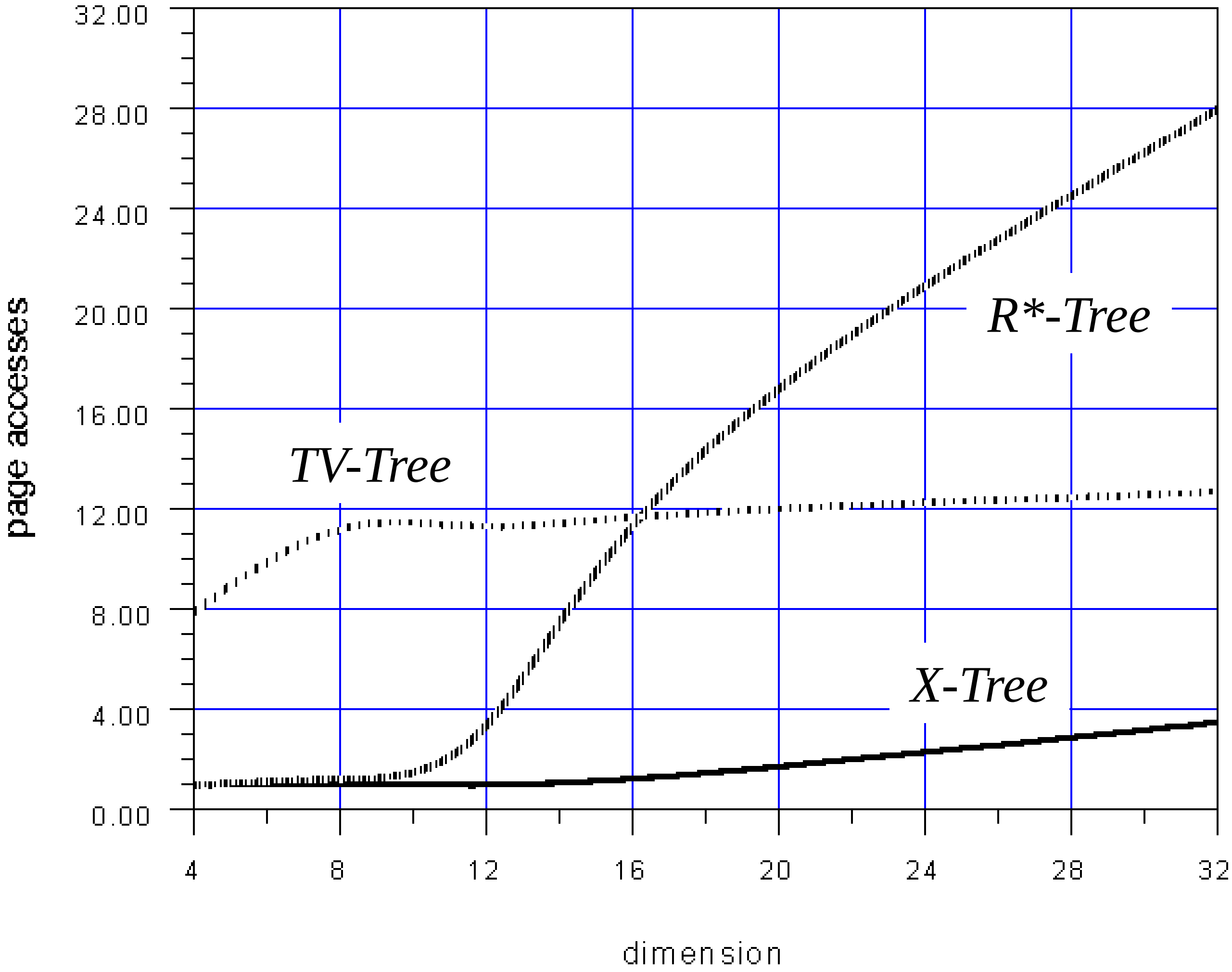


Point Query



10 NN Query

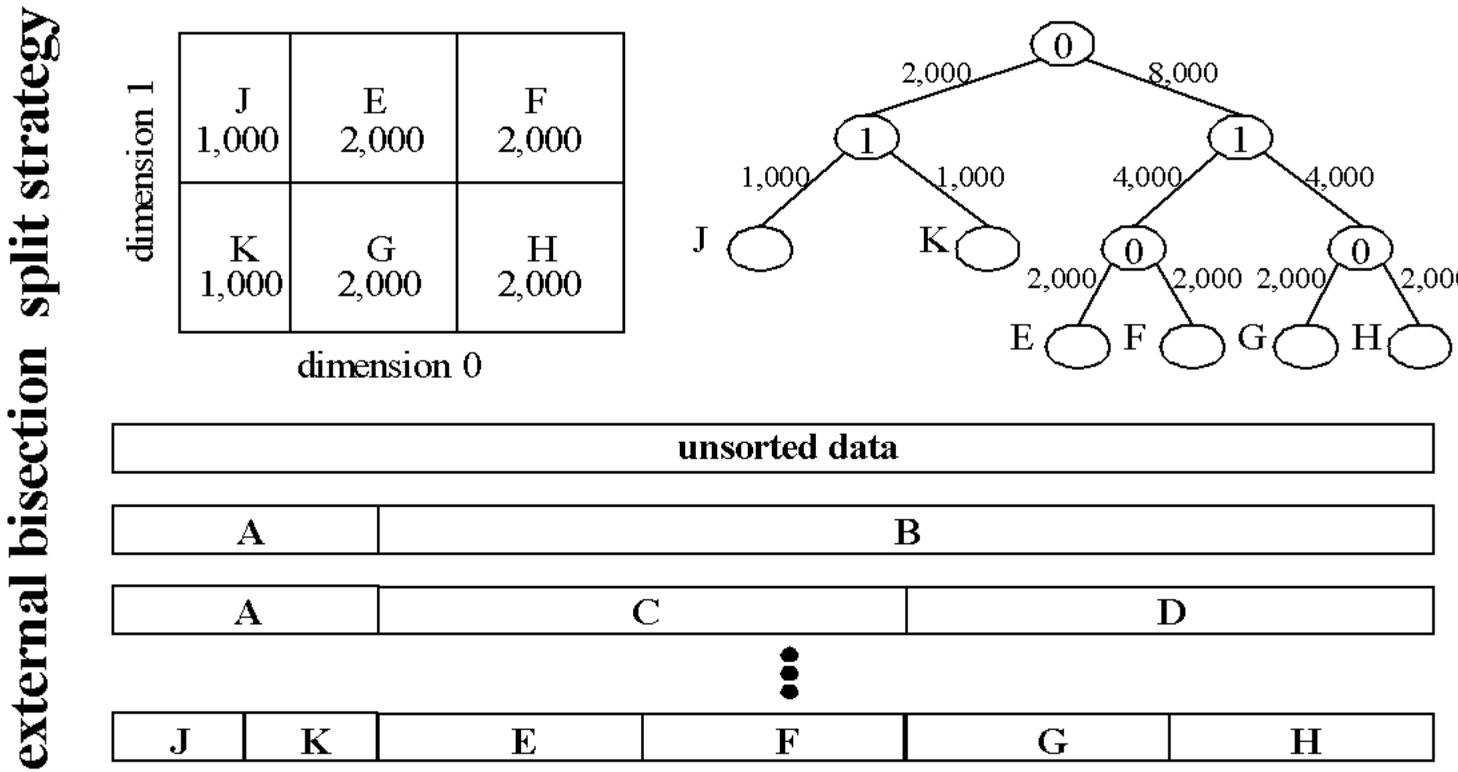
Comparison with R*-Tree and TV-Tree



Bulk-Load of X-Trees [BBK 98a]

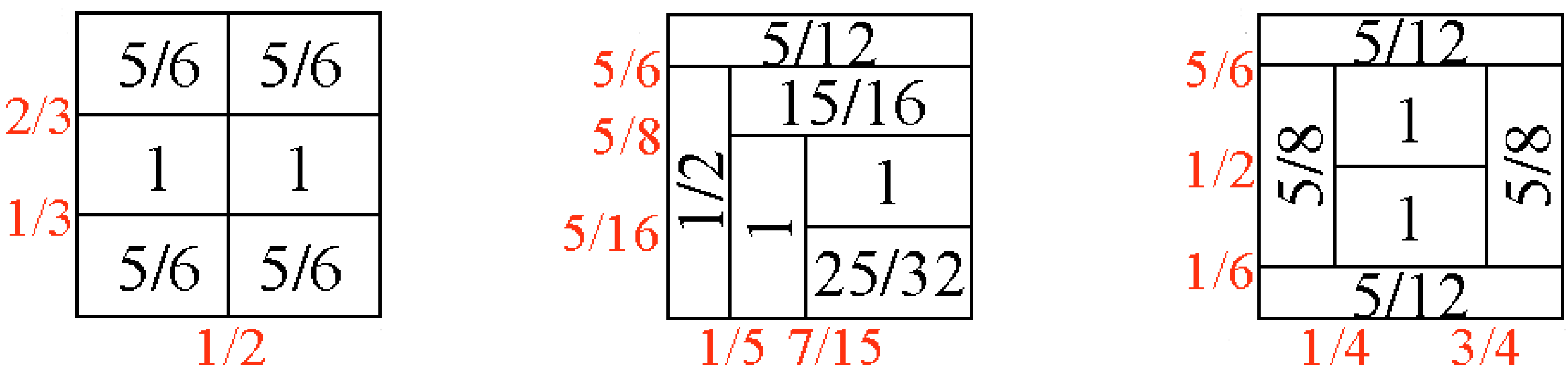
- Observation:
  In order to split a data set, we do not have to sort it
- Recursive top-down partitioning of the data set
- Quicksort-like algorithm
- Improved data space partitioning

Example


external bisection split strategy
dimension 1
J 1,000
E 2,000
F 2,000
K 1,000
G 2,000
H 2,000
dimension 0
0
2,000
8,000
1
1
1,000
1,000
4,000
4,000
J
K
0
0
2,000
2,000
2,000
2,000
E
F
G
H
unsorted data
A
B
A
C
D
J
K
E
F
G
H

Unbalanced Split

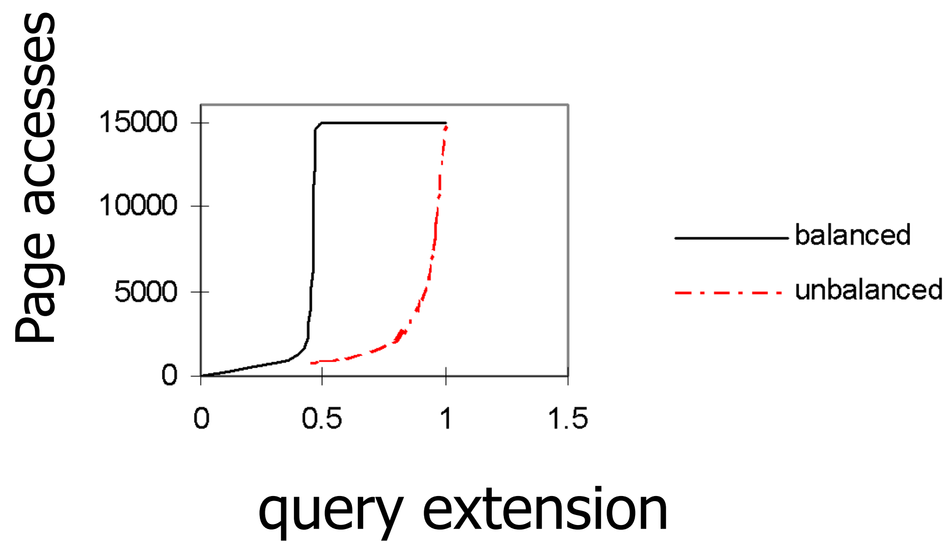
- Probability that a data page is loaded when processing a range query of edge length 0.6 (for three different split strategies)



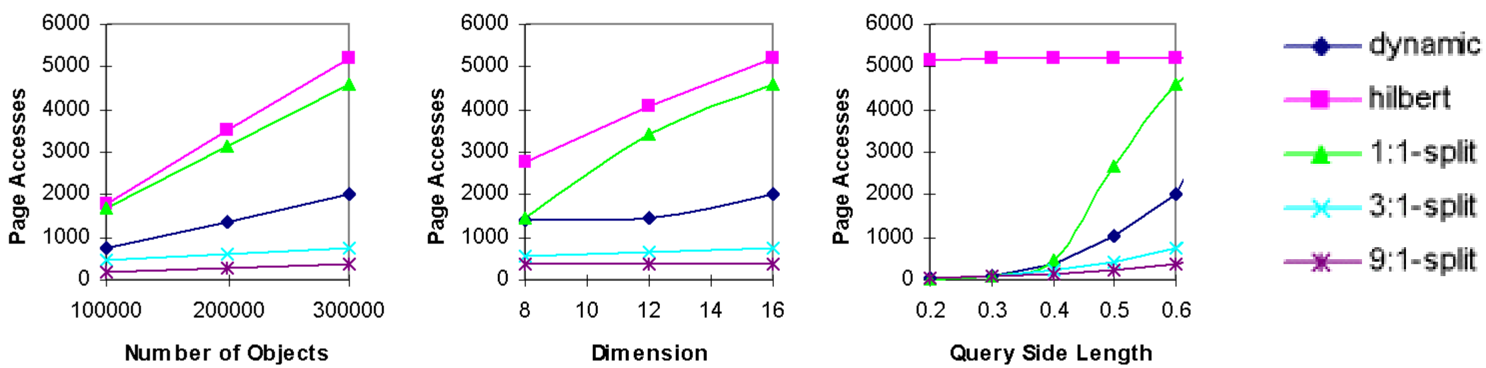
$P_1(0.6) = 5.33$ $P_2(0.6) = 4.64$ $P_3(0.6) = 4.08$

Effect of Unbalanced Split

In Theory:



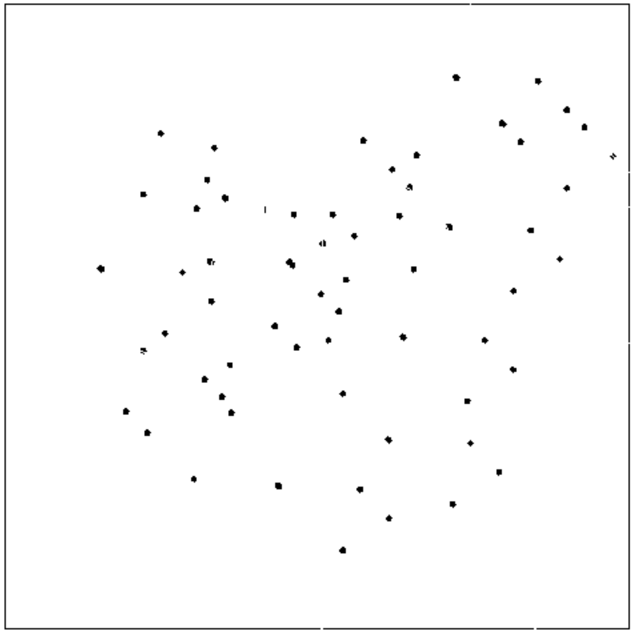
In Practice:



The SS-Tree [WJ 96]

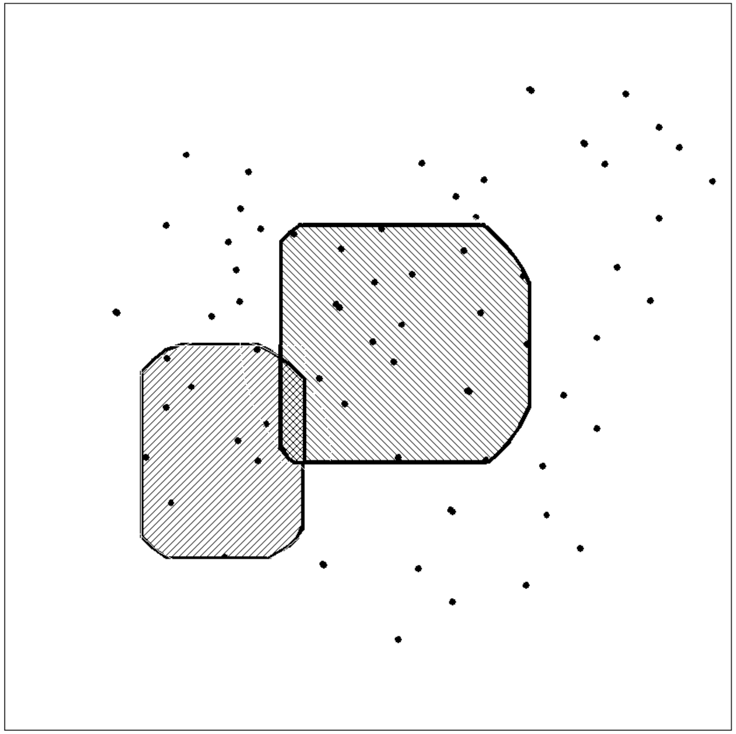
(Similarity-Search Tree)

- Idea:
  Split data space into spherical regions
- small MINDIST
- high fanout
- Problem: overlap



The SR-Tree [KS 97]
(Similarity-Search R-Tree)

- Similar to SS-Tree, but:
- Partitions are intersections of spheres and hyper-rectangles
- Low overlap



Other Techniques

- Pyramid-Tree [BBK 98]
- VA-File [WSB 98]
- Voroni-based Indexing [BEK+ 98]

The Pyramid-Tree [BBK 98]

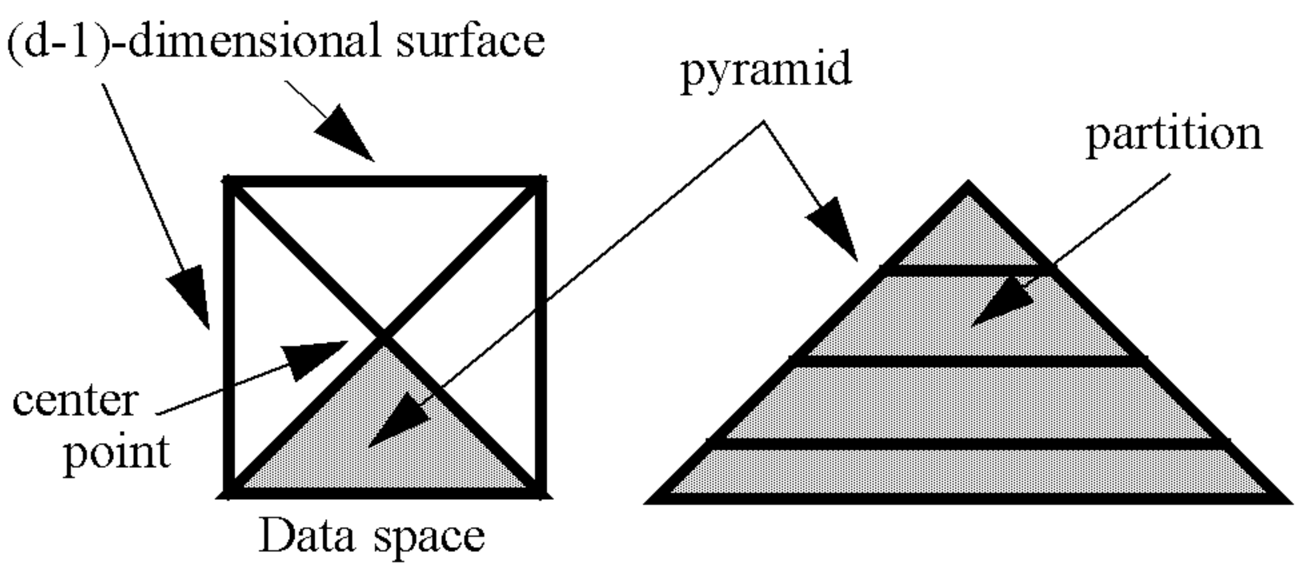
- Motivation:

  Index-structures such as the X-Tree have several drawbacks
  - the split strategy is sub-optimal
  - all page accesses result in random I/O
  - high transaction times (insert, delete, update)
- Idea:

  Provide a data space partitioning which can be seen as a mapping from a *d*-dim. space to a 1-dim. space and make use of B^+-Trees

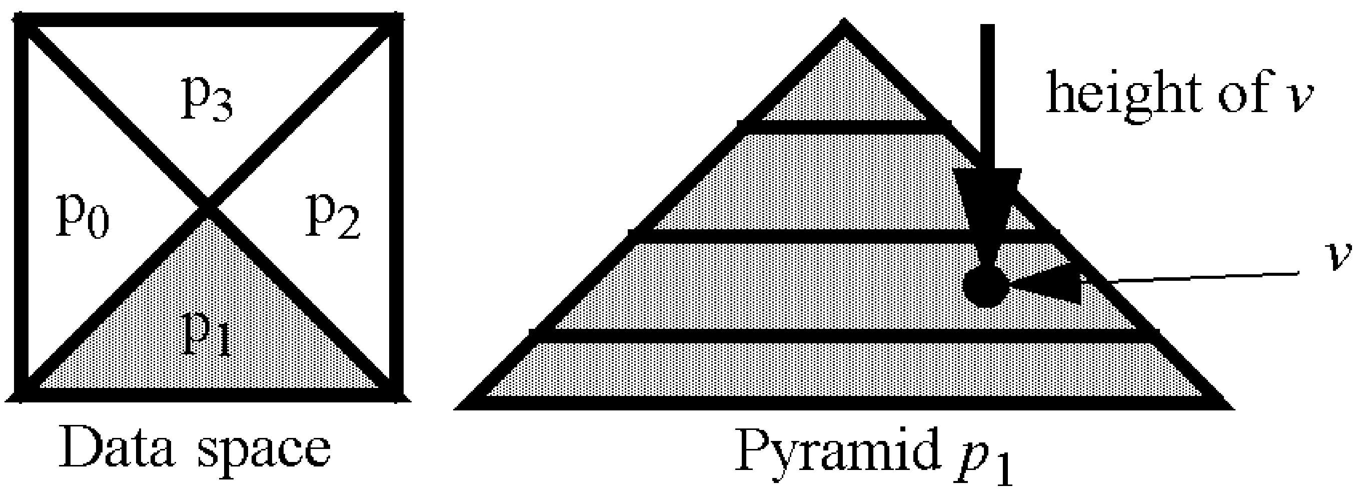
The Pyramid-Mapping

- Divide the space into 2d pyramids
- Divide each pyramid into partitions
- Each partition corresponds to a B^+-Tree page



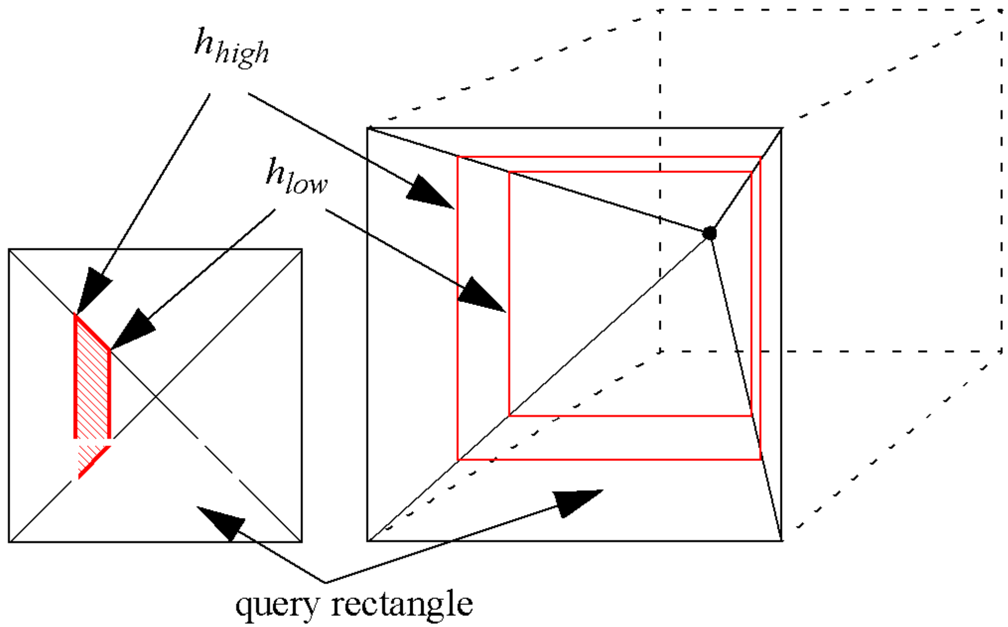
The Pyramid-Mapping

- A point in a high-dimensional space can be addressed by the number of the pyramid and the height within the pyramid.

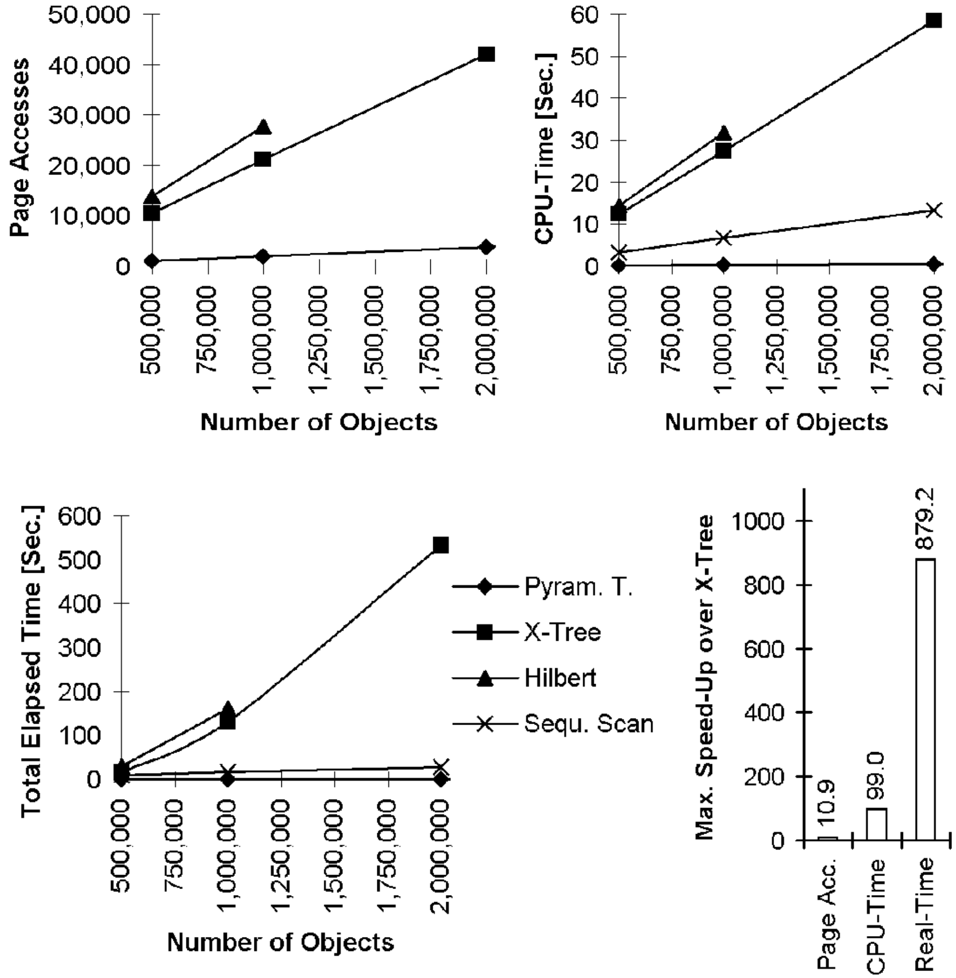


Query Processing using a Pyramid-Tree

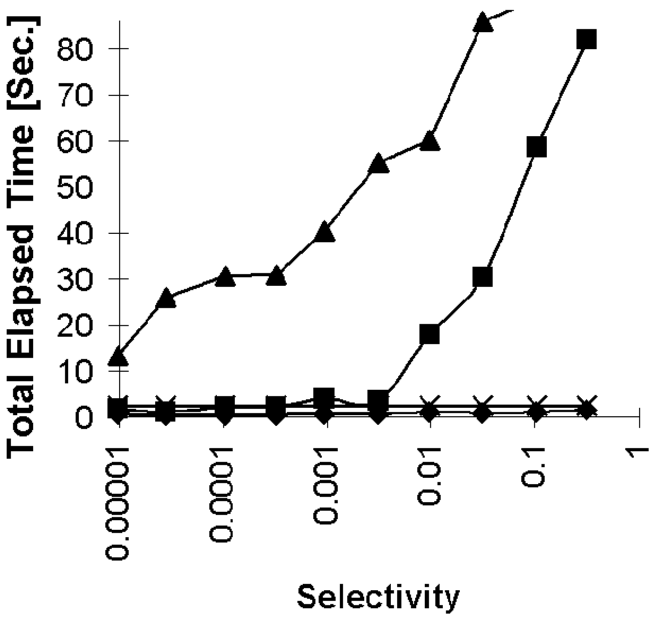
- Problem:
  Determine the pyramids intersected by the query rectangle and the interval [h_{high}, h_{low}] within the pyramids.

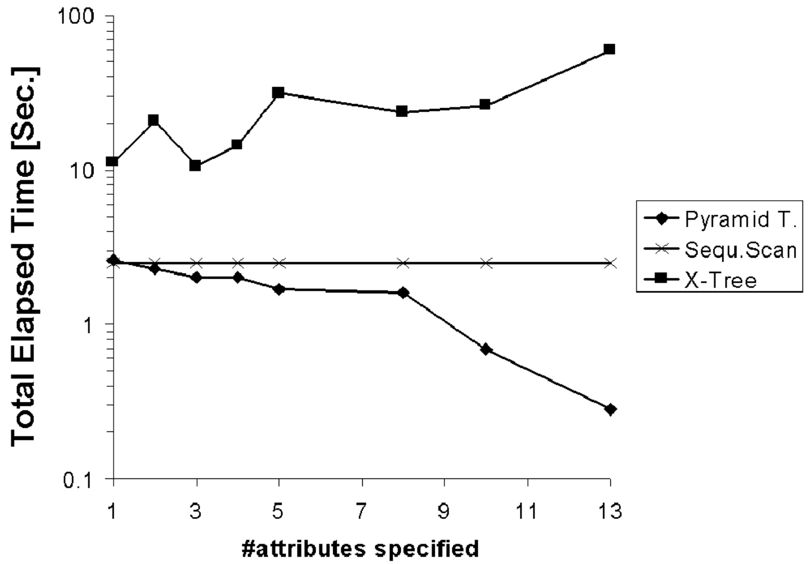


Experiments (uniform data)



Experiments
(data from data warehouse)





Analysis (intuitive)

- Performance is determined by the trade-off between the increasing range and the decreasing thickness of a single partition.
- The analysis shows that the access probability of a single partition decreases when increasing the dimensionality.

The VA-File [WSB 98]

(Vector Approximation File)

- Idea:
  If NN-Search is an inherently linear problem, we should aim for speeding up the sequential scan.
- Use a coarse representation of the data points as an approximate representation (only *i* bits per dimension - *i* might be 2)
- Thus, the reduced data set has only the (*i*/32)-th part of the original data set

The VA-File

- Determine $(1/2^i)$-quantiles of each dimension as partition boundaries
- Sequentially scan the coarse representation and maintain the actual NN-distance
- If a partition cannot be pruned according to its coarse representation, a look-up is made in the original data set

The VA-file

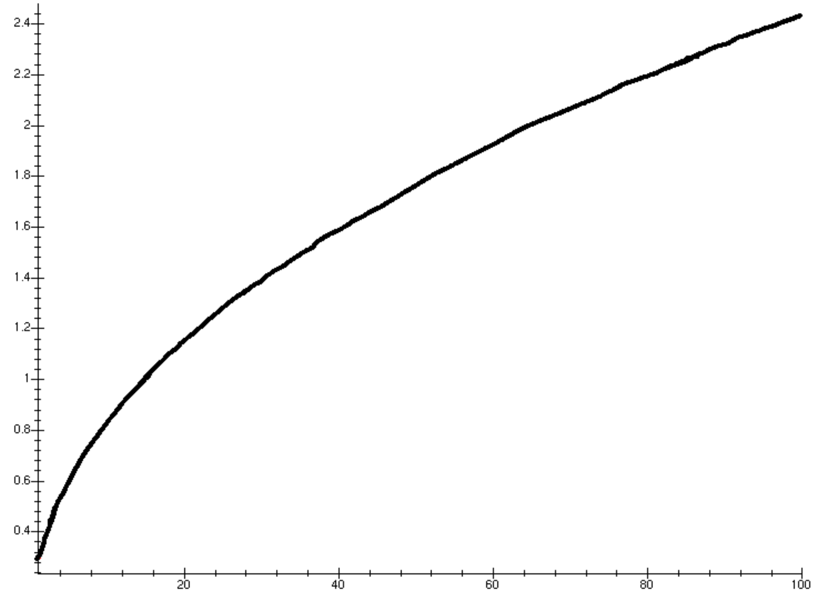
- Very fast on uniform data
  (no curse of dimensionality)
- Fails, if the data is correlated or builds complex clusters

Explanation:

The NN-distance plus the diameter of a single cell grows slower than the diameter of the data space when increasing the dimensionality.

Analysis (intuitive)

- Assume the query point *q* is on a (d/2)-dimensional surface
- Expected distance between the NN-sphere and a VA-cell on the opposite side of space

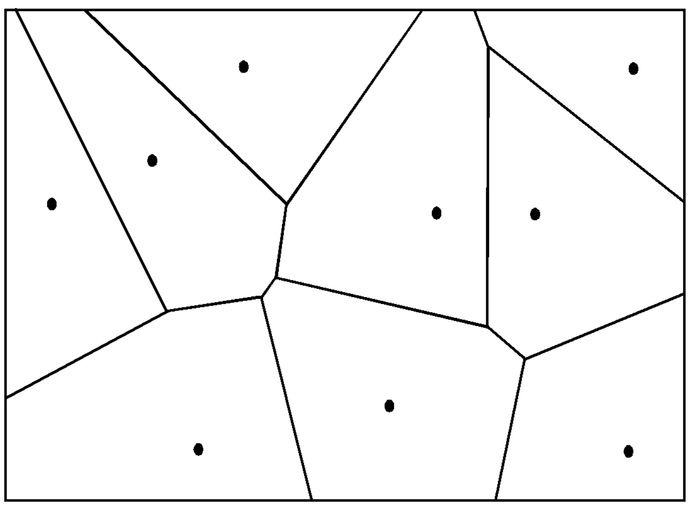


Voronoi-based Indexing [BEK+ 98]

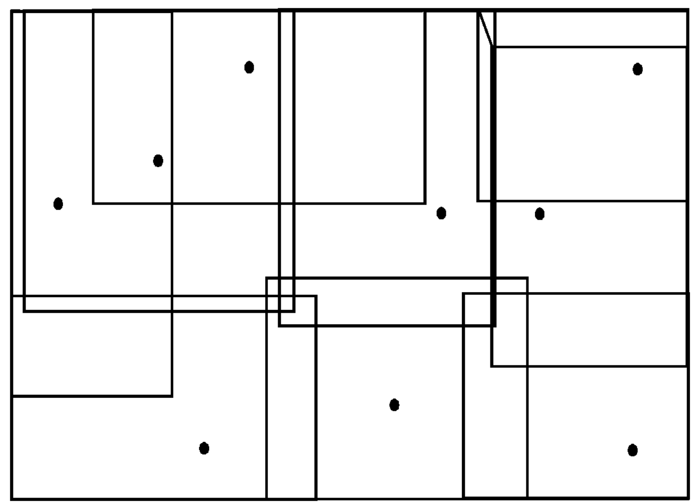
- Idea:

  Precalculation and indexing of the result space

  ⇨ Point query instead of NN-query



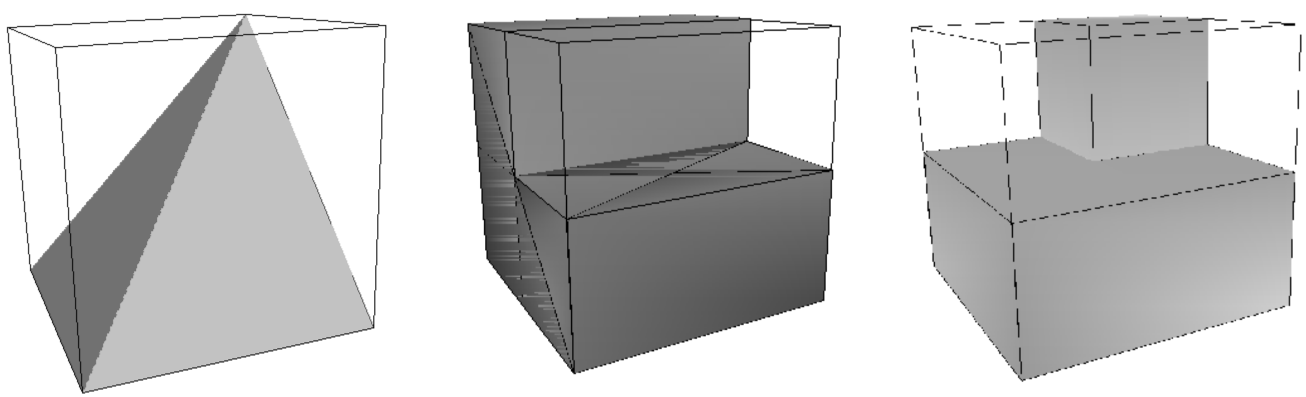
Voroni-Cells



Approximated Voroni-Cells

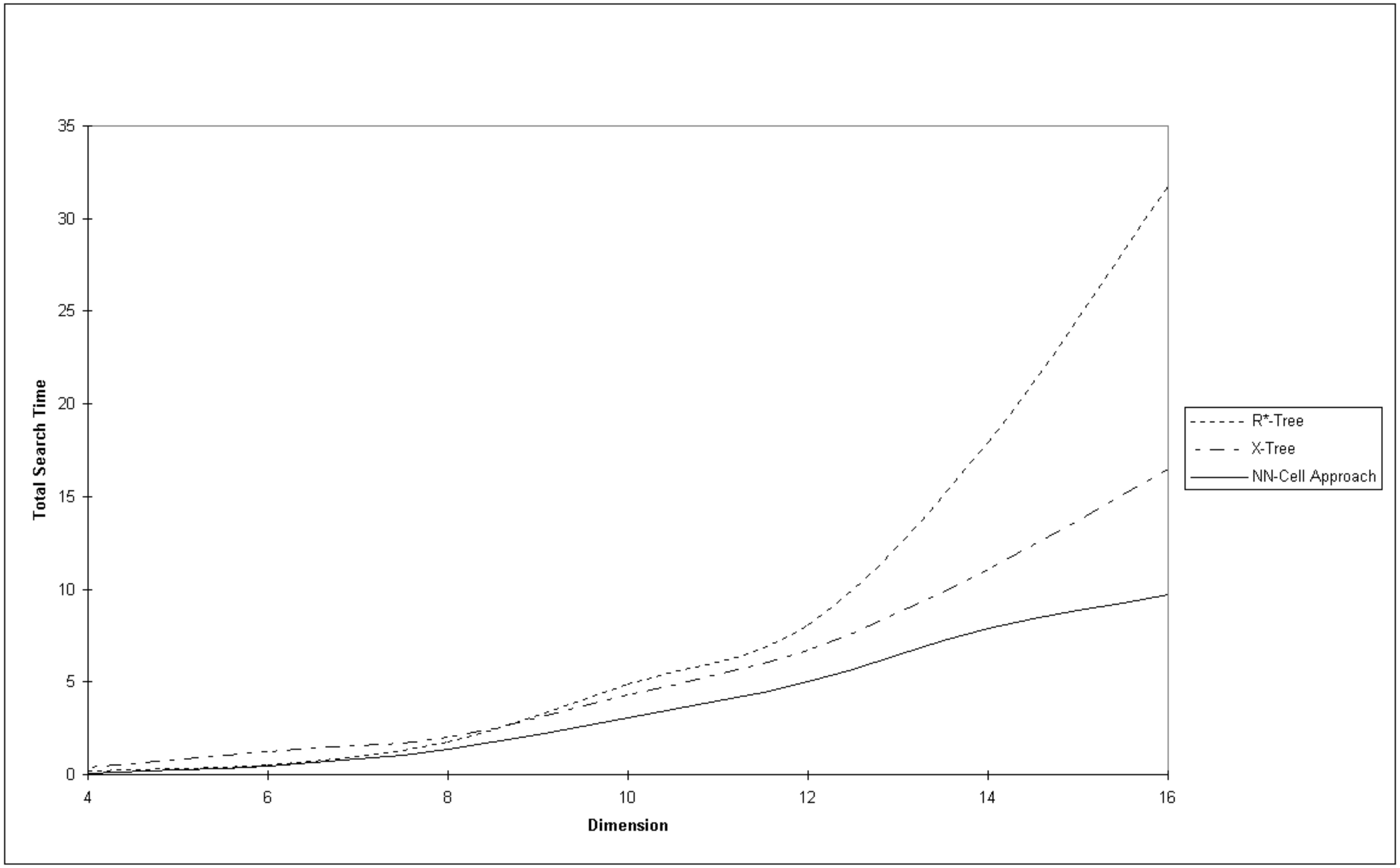
Voronoi-based Indexing

- Precalculation of Result Space (Voronoi Cells) by Linear Optimization Algorithm
- Approximation of Voronoi Cells by Bounding Volumes
- Decomposition of Bounding Volumes (in most oblique dimension)



Voronoi-based Indexing

- Comparison to R*-Tree and X-Tree



Optimization and Parallelization

- Tree Striping [BBK+ 98]
- Parallel Declustering [BBB+ 97]
- Approximate Nearest Neighbor Search [GIM 98]

Tree Striping [BBK+ 98]

- Motivation:

  The two solutions to multidimensional indexing - inverted lists and multidimensional indexes - are both inefficient.

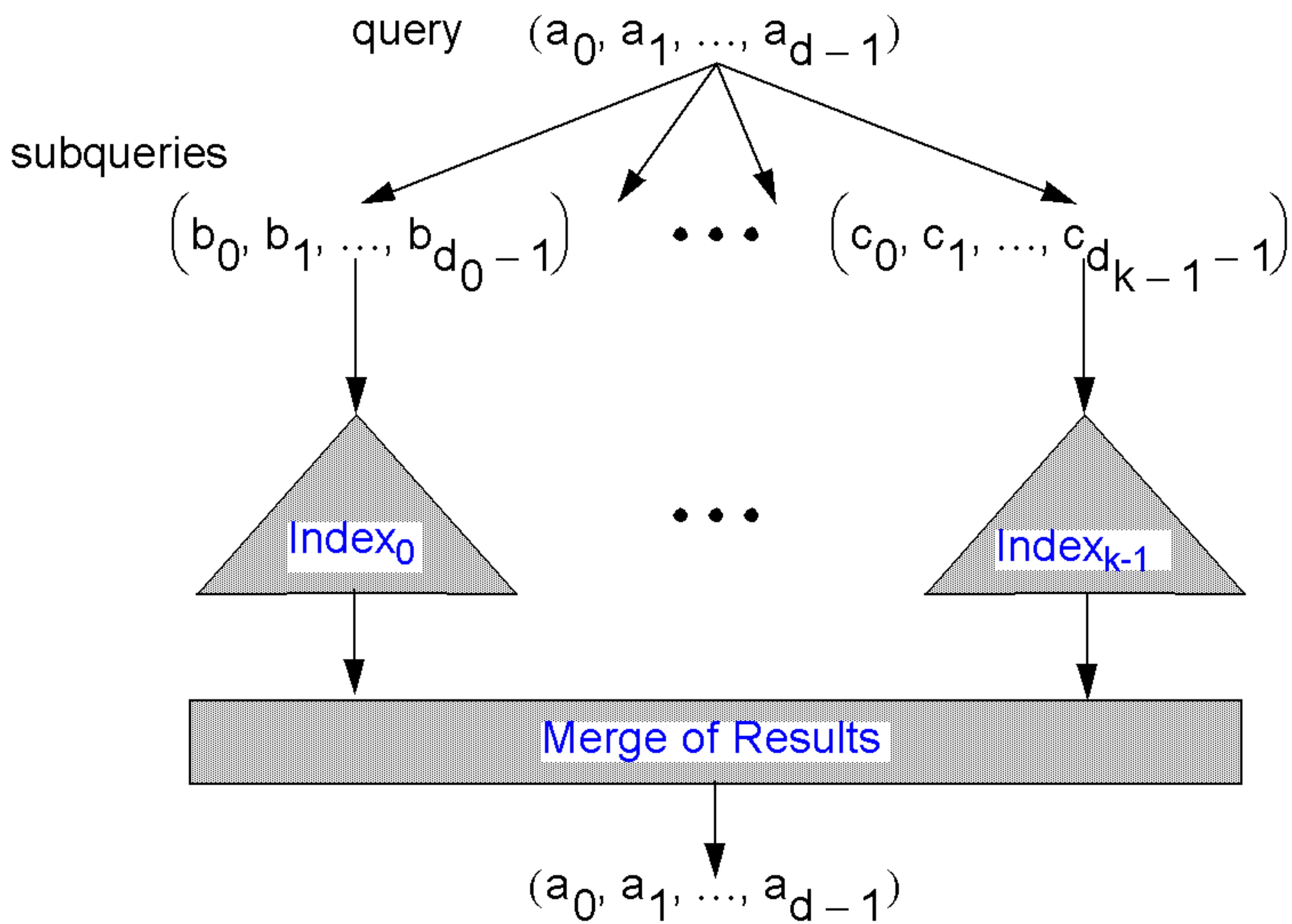
- Explanation:

  High dimensionality deteriorates the performance of indexes and increases the sort costs of inverted lists.

- Idea:

  There must be an optimum in between high-dimensional indexing and inverted lists.

Tree Striping - Example



Tree Striping - Cost Model

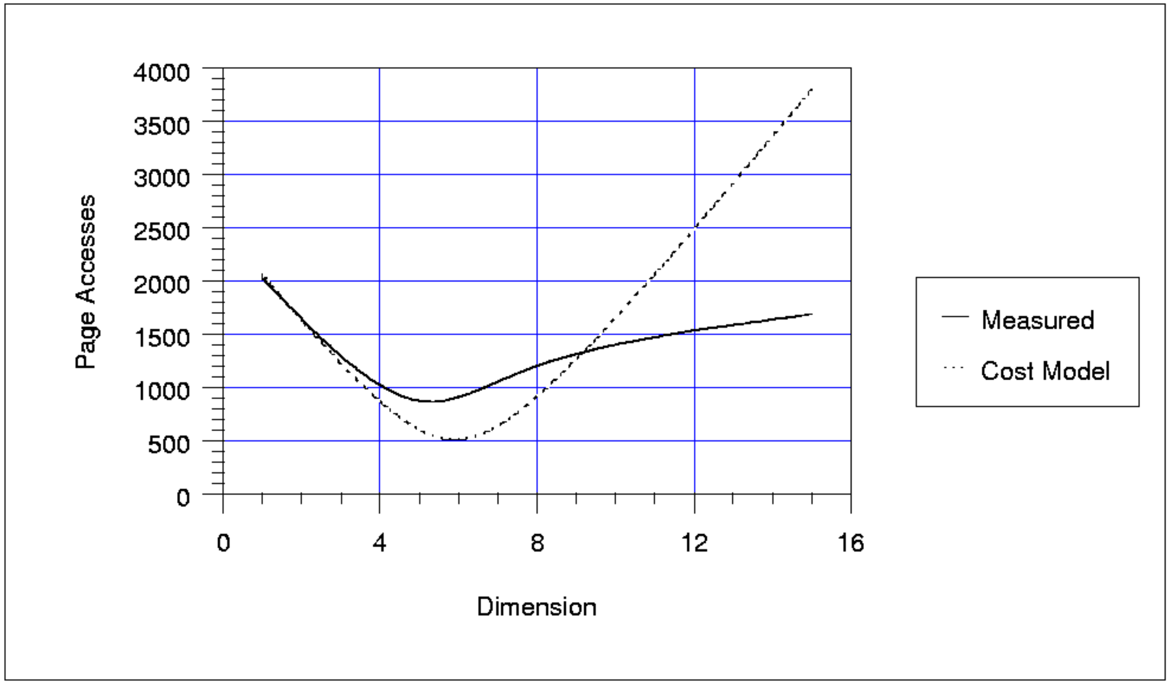
- Assume uniformity of data and queries
- Estimate index costs for *k* indexes
  (based on high-dimensional Minkowsky-sum)
- Estimate sort costs for *k* indexes
- Sum both costs up
- Determine the optimal value for *k*

Tree Striping - Additional Tricks

- Materialization of results
- Smart distribution of attributes by estimating selectivity
- Redundant storage of information

Experiments

- Real data, range queries, *d*-dimensional indexes



Parallel Declustering [BBB+ 97]

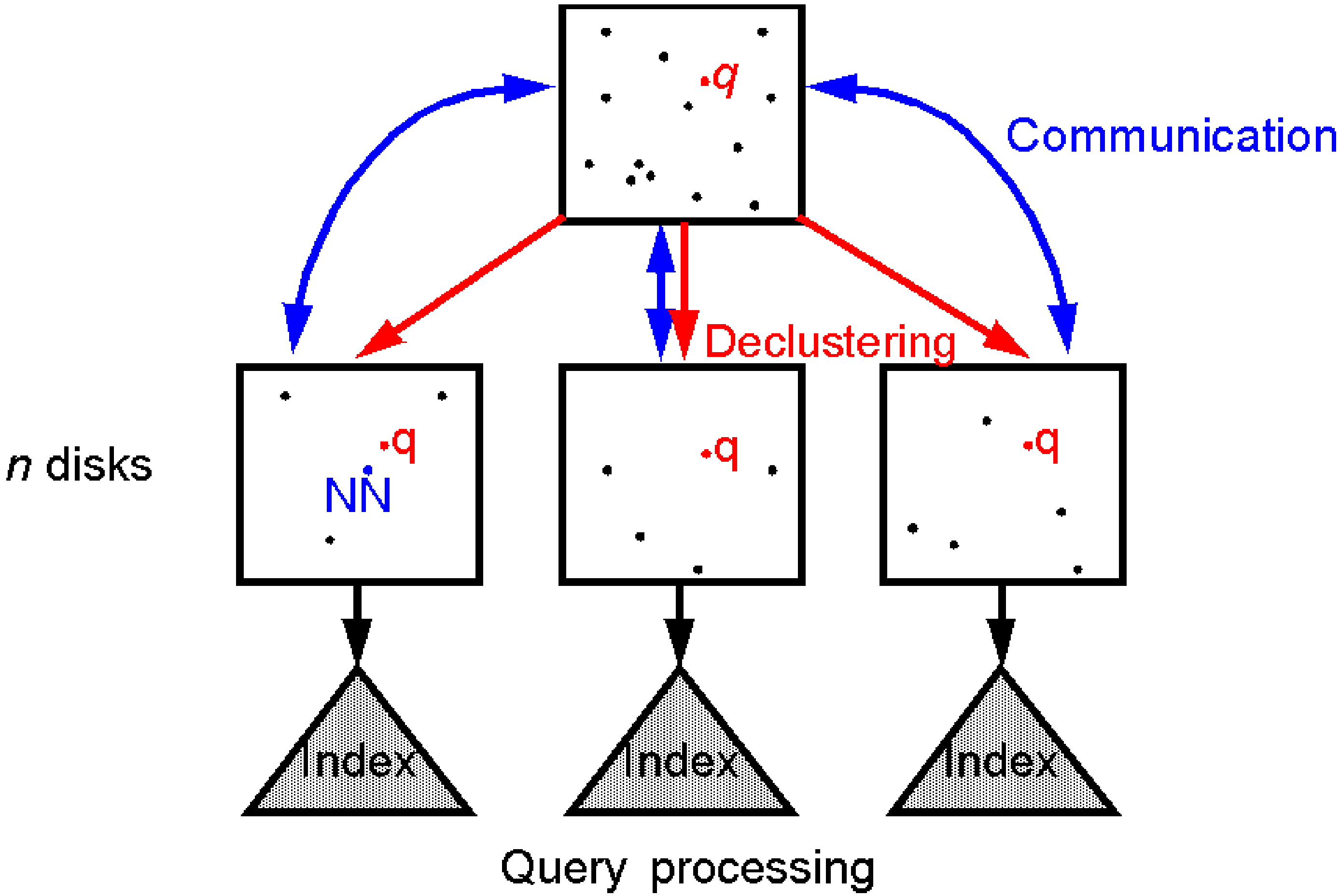
- Idea:

  If NN-Search is an inherently linear problem, it is perfectly suited for parallelization.

- Problem:

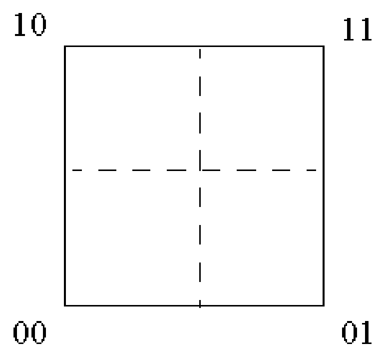
  How to decluster high-dimensional data?

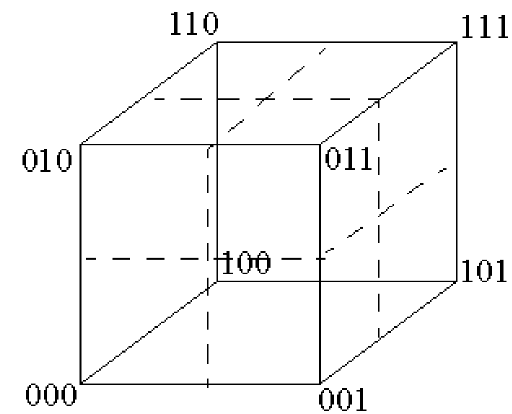
Parallel Declustering



Near-Optimal Declustering

- Each partition is connected with one corner of the data space
  Identify the partitions by their **canonical corner numbers**
  = bitstrings saying left = 0 and right = 1 for each dimension



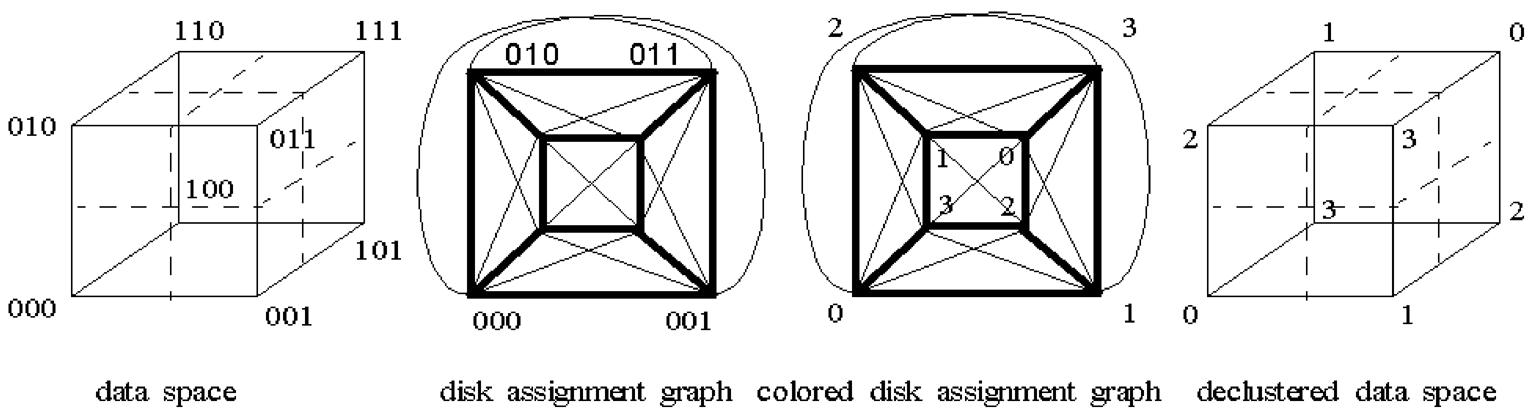


- Different degrees of neighborhood relationships:
  – Partitions are **direct** neighbors if they differ in exactly 1 dimension
  – Partitions are in**direct** neighbors if they differ in exactly 2 dimension

Parallel Declustering

Mapping of the Problem to a Graph:

partitions	⇒	vertices
neighborhood-relations	⇒	edges
disks	⇒	colors



Parallel Declustering

- Given: vertex number = corner number in binary representation

$$c = (c_{d-1}, \ldots, c_0)$$

- Compute: vertex color col(c) as

$$\mathrm{col}(c) = \left(\mathop{\mathrm{XOR}}_{i=0}^{d-1} \left(\begin{cases} i+1 & \text{if } c_i = 1 \\ 0 & \text{otherwise} \end{cases} \right) \right)_{10}$$

c = | 1 | 0 | 0 | 1 | 1 |

1_{10} = | 0 | 0 | 0 | 0 | 1 |

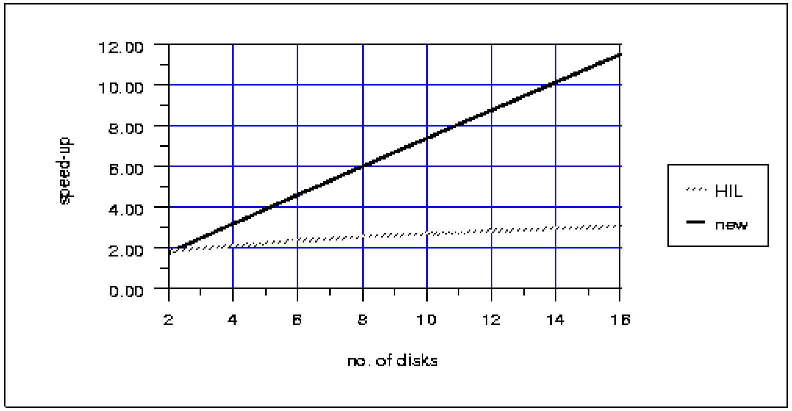
2_{10} = | 0 | 0 | 0 | 1 | 0 |

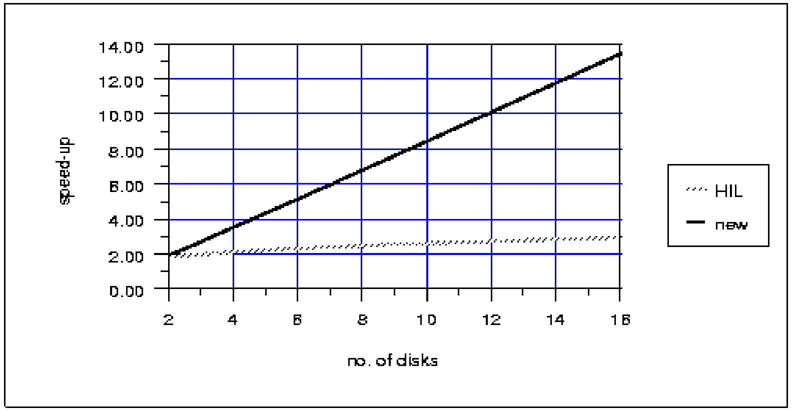
5_{10} = | 0 | 0 | 1 | 0 | 1 |

XOR = | 0 | 0 | 1 | 1 | 0 | $= 6_{10}$

Experiments

- Real data, comparison with Hilbert-declustering, # of disks vs. speed-up





Approximate NN-Search

(Locality-Sensitive Hashing) [GIM 98]

- Idea:
  If it is sufficient to only select an approximate nearest-neighbor, we can do this much faster.
- Approximate Nearest-Neighbor: A point in distance $(1+\varepsilon)\cdot NN_{dist}$ from the query point.

Locality-Sensitive Hashing

- Algorithm:
  - Map each data point into a higher-dimensional binary space
  - Randomly determine *k* projections of the binary space
  - For each of the *k* projections determine the points having the same binary representations as the query point
  - Determine the nearest-neighbors of all these points
- Problems:
  - How to optimize *k*?
  - What is the expected ε? (average and worst case)
  - What is an approximate nearest-neighbor “worth”?

Open Research Topics

- The ultimate cost model
- Partitioning strategies
- Parallel query processing
- Data reduction
- Approximate query processing
- High-dim. data mining & visualization

Partitioning Strategies

- What is the optimal data space partitioning schema for nearest-neighbor search in high-dimensional spaces?
- Balanced or unbalanced?
- Pyramid-like or bounding boxes?
- How does the optimum changes when the data set grows in size or dimensionality?

Parallel Query Processing

- Is it possible to develop parallel versions of the proposed sequential techniques?
  If yes, how can this be done?
- Which declustering strategies should be used?
- How can the parallel query processing be optimized?

Data Reduction

- How can we reduce a large data warehouse in size such that we get approximate answers from the reduced data base?
- Tape-based data warehouses
  - ⇨ disk based
- Disk-based data warehouses
  - ⇨ main memory
- Tradeoff: accuracy vs. reduction factor

Approximate Query Processing

- Observation:

  Most similarity search applications do not require 100% correctness.

- *Problem:*
  - What is a good definition for approximate nearest- neighbor search?
  - How to exploit that fuzziness for efficiency?

High-dimensional Data Mining & Data Visualization

- How can the proposed techniques be used for data mining?
- How can high-dimensional data sets and effects in high-dimensional spaces be visualized?

Summary

- Major research progress in
  - understanding the nature of high-dim. spaces
  - modeling the cost of queries in high-dim. spaces
  - index structures supporting nearest-neighbor search and range queries

Conclusions

- Work to be done
  - leave the clean environment
    - uniformity
    - uniform query mix
    - number of data items is exponential in *d*
  - address other relevant problems
    - partial range queries
    - approximate nearest neighbor queries

Literature

[AMN 95] Arya S., Mount D. M., Narayan O.: *'Accounting for Boundary Effects in Nearest Neighbor Searching'*, Proc. 11th Annual Symp. on Computational Geometry, Vancouver, Canada, pp. 336-344, 1995.

[Ary 95] Arya S.: *'Nearest Neighbor Searching and Applications'*, Ph.D. Thesis, University of Maryland, College Park, MD, 1995.

[BBB+ 97]Berchtold S., Böhm C., Braunmueller B., Keim D. A., Kriegel H.-P.: *'Fast Similarity Search in Multimedia Databases'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, Tucson, Arizona, 1997.

[BBK 98] Berchtold S., Böhm C., Kriegel H.-P.: *'The Pyramid-Tree: Indexing Beyond the Curse of Dimensionality'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, Seattle, 1998.

[BBK 98a]Berchtold S., Böhm C., Kriegel H.-P.: *'Improving the Query Performance of High-Dimensional Index Structures by Bulk Load Operations'*, 6th Int. Conf. On Extending Database Technology, in LNCS 1377, Valenica, Spain, pp. 216-230, 1998.

Literature

[BBKK 97] Berchtold S., Böhm C., Keim D., Kriegel H.-P.: '*A Cost Model For Nearest Neighbor Search in High-Dimensional Data Space*', ACM PODS Symposium on Principles of Database Systems, Tucson, Arizona, 1997.

[BBKK 98] Berchtold S., Böhm C., Keim D., Kriegel H.-P.: '*Optimized Processing of Nearest Neighbor Queries in High-Dimensional Spaces*', submitted for publication.

[BEK+ 98] Berchtold S., Ertl B., Keim D., Kriegel H.-P., Seidl T.: '*Fast Nearest Neighbor Search in High-Dimensional Spaces*', Proc. 14th Int. Conf. on Data Engineering, Orlando, 1998.

[BBK+ 98] Berchtold S., Böhm C., Keim D., Kriegel H.-P., Xu X.: '*Optimal Multidimensional Query Processing Using Tree-Striping*', submitted for publication.

[Ben 75] Bentley J. L.: *'Multidimensional Search Trees Used for Associative Searching',* Comm. of the ACM, Vol. 18, No. 9, pp. 509-517, 1975.

[BGRS 98] Beyer K., Goldstein J., Ramakrishnan R., Shaft U.: *'When is "Nearest Neighbor" Meaningful?'*, submitted for publication.

Literature

[BK 97] Berchtold S., Kriegel H.-P.: *'S3: Similarity Search in CAD Database Systems'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, Tucson, Arizona, 1997.

[BKK 96] Berchtold S., Keim D., Kriegel H.-P.: '*The X-tree: An Index Structure for High-Dimensional Data*', 22nd Conf. on Very Large Databases, Bombay, India, pp. 28-39, 1996.

[BKK 97] Berchtold S., Keim D., Kriegel H.-P.: '*Using Extended Feature Objects for Partial Similarity Retrieval*', VLDB Journal, Vol.4, 1997.

[BKSS 90] Beckmann N., Kriegel H.-P., Schneider R., Seeger B.: *'The R*-tree: An Efficient and Robust Access Method for Points and Rectangles'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, Atlantic City, NJ, pp. 322-331, 1990.

[CD 97] Chaudhuri S., Dayal U.: *'Data Warehousing and OLAP for Decision Support'*, Tutorial, Proc. ACM SIGMOD Int. Conf. on Management of Data, Tucson, Arizona, 1997.

[Cle 79] Cleary J. G.: *'Analysis of an Algorithm for Finding Nearest Neighbors in Euclidean Space',* ACM Trans. on Mathematical Software, Vol. 5, No. 2, pp.183-192, 1979.

Literature

[FBF 77] Friedman J. H., Bentley J. L., Finkel R. A.: '*An Algorithm for Finding Best Matches in Logarithmic Expected Time*', ACM Transactions on Mathematical Software, Vol. 3, No. 3, pp. 209-226, 1977.

[GG 96] Gaede V., Günther O.: '*Multidimensional Access Methods*', Technical Report, Humboldt-University of Berlin, http://www.wiwi.hu- berlin.de/ institute/iwi/info/research/iss/papers/survey.ps.Z.

[GIM] Gionis A., Indyk P., Motwani R.: *' Similarity Search in High Dimensions via Hashing',* submitted for publication, 1998.

[Gut 84] Guttman A.: '*R-trees: A Dynamic Index Structure for Spatial Searching*', Proc. ACM SIGMOD Int. Conf. on Management of Data, Boston, MA, pp. 47-57, 1984.

[Hen 94] Henrich, A.: *'A distance-scan algorithm for spatial access structures'*, Proceedings of the 2nd ACM Workshop on Advances in Geographic Information Systems, ACM Press, Gaithersburg, Maryland, pp. 136-143, 1994.

[Hen 98] Henrich, A.: *'The LSD*^{h}-tree: An Access Structure for Feature Vectors', Proc. 14th Int. Conf. on Data Engineering, Orlando, 1998.

Literature

[HS 95] Hjaltason G. R., Samet H.: *'Ranking in Spatial Databases'*, Proc. 4th Int. Symp. on Large Spatial Databases, Portland, ME, pp. 83-95, 1995.

[HSW 89] Henrich A., Six H.-W., Widmayer P.: *'The LSD-Tree: Spatial Access to Multidimensional Point and Non-Point Objects'*, Proc. 15th Conf. on Very Large Data Bases, Amsterdam, The Netherlands, pp. 45-53, 1989.

[Jag 91] Jagadish H. V.: *'A Retrieval Technique for Similar Shapes',* Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 208-217, 1991.

[JW 96] Jain R, White D.A.: *'Similarity Indexing: Algorithms and Performance'*, Proc. SPIE Storage and Retrieval for Image and Video Databases IV, Vol. 2670, San Jose, CA, pp. 62-75, 1996.

[KS 97] Katayama N., Satoh S.: *'The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 369-380, 1997.

[KSF+ 96] Korn F., Sidiropoulos N., Faloutsos C., Siegel E., Protopapas Z.: *'Fast Nearest Neighbor Search in Medical Image Databases'*, Proc. 22nd Int. Conf. on Very Large Data Bases, Mumbai, India, pp. 215-226, 1996.

[LJF 94] Lin K., Jagadish H. V., Faloutsos C.: *'The TV-tree: An Index Structure for High-Dimensional Data',* VLDB Journal, Vol. 3, pp. 517-542, 1995.

Literature

[MG 93] Mehrotra R., Gary J.: *'Feature-Based Retrieval of Similar Shapes',* Proc. 9th Int. Conf. on Data Engineering, 1993.

[Ore 82] Orenstein J. A.: *'Multidimensional tries used for associative searching',* Inf. Proc. Letters, Vol. 14, No. 4, pp. 150-157, 1982.

[PM 97] Papadopoulos A., Manolopoulos Y.: *'Performance of Nearest Neighbor Queries in R-Trees'*, Proc. 6th Int. Conf. on Database Theory, Delphi, Greece, in: Lecture Notes in Computer Science, Vol. 1186, Springer, pp. 394-408, 1997.

[RKV 95] Roussopoulos N., Kelley S., Vincent F.: *'Nearest Neighbor Queries',* Proc. ACM SIGMOD Int. Conf. on Management of Data, San Jose, CA, pp. 71-79, 1995.

[Rob 81] Robinson J. T.: *'The K-D-B-tree: A Search Structure for Large Multidimensional Dynamic Indexes',* Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 10-18, 1981.

[RP 92] Ramasubramanian V., Paliwal K. K.: *'Fast k-Dimensional Tree Algorithms for Nearest Neighbor Search with Application to Vector Quantization Encoding'*, IEEE Transactions on Signal Processing, Vol. 40, No. 3, pp. 518-531, 1992.

Literature

[See 91] Seeger B.: *'Multidimensional Access Methods and their Applications'*, Tutorial, 1991.

[SK 97] Seidl T., Kriegel H.-P.: *'Efficient User-Adaptable Similarity Search in Large Multimedia Databases'*, Proc. 23rd Int. Conf. on Very Large Databases (VLDB'97), Athens, Greece, 1997.

[Spr 91] Sproull R.F.: *'Refinements to Nearest Neighbor Searching in k-Dimensional Trees',* Algorithmica, pp. 579-589, 1991.

[SRF 87] Sellis T., Roussopoulos N., Faloutsos C.: *'The R*$^{+-}$Tree: A Dynamic Index for Multi-Dimensional Objects', Proc. 13th Int. Conf. on Very Large Databases, Brighton, England, pp 507-518, 1987.

[WSB 98] Weber R., Scheck H.-J., Blott S.:*' A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces'*, submitted for publication, 1998.

[WJ 96] White D.A., Jain R.: *'Similarity indexing with the SS-tree'*, Proc. 12th Int. Conf on Data Engineering, New Orleans, LA, 1996.

[YY 85] Yao A. C., Yao F. F.: *' A General Approach to D-Dimensional Geometric Queries'*, Proc. ACM Symp. on Theory of Computing, 1985.

Acknowledgement

We thank **Stephen Blott** and **Hans-J. Scheck** for the very interesting and helpful discussions about the VA-file and for making the paper available to us.

We thank **Raghu Ramakrishnan** and **Jonathan Goldstein** for their explanations and the allowance to present their unpublished work on "When Is Nearest-Neighbor Meaningful".

We also thank **Pjotr Indyk** for providing the paper about Local Sensitive Hashing.

Furthermore, we thank **Andreas Henrich** for introducing us into the secrets of LSD and KDB trees.

Finally, we thank **Marco Poetke** for providing the nice figure explaining telescope vectors.

Last but not least, we thank **H.V. Jagadish** for encouraging us to submit this tutorial.

The End