# High-Dimensional Vector Indexing and Similarity Search


Themis Palpanas

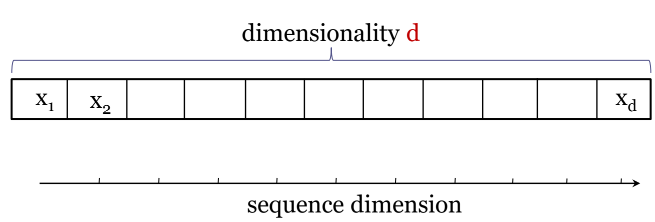
*Université Paris Cité*
*French University Institute*


Distributed Data Masses (MDD)) Summer School Cargese (France), April 2026



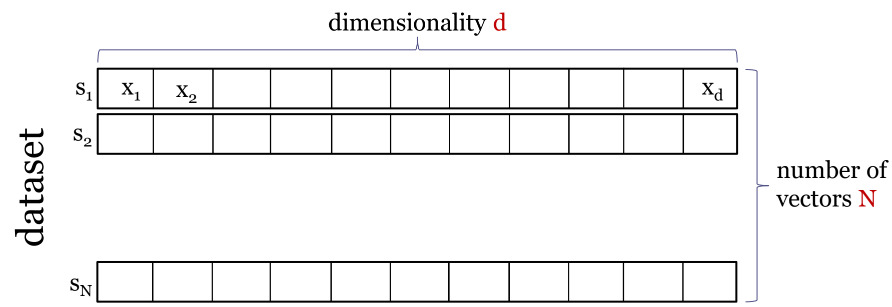
## Vector

represented as *d*-dimensional array



# Vector Collections

represented as $N$ $d$-dimensional arrays



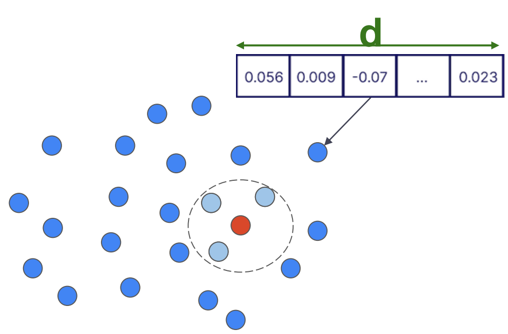





# Similarity Search Problem

**Given:** a set ***S*** of **n** distinct points in **d**-dimensional space $R^d$ under some norm .

**Goal:** return the set of $K$ points in ***S*** that are **closest** to a query ***Q*** $R^d$,under .




O(n d)

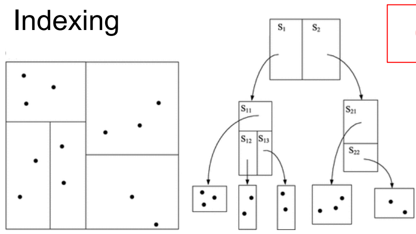
# Can we do better?

Pruning non promising data points

Dimension reduction

O(n d)

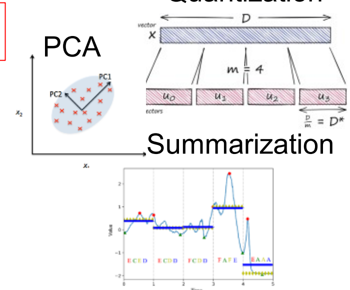
Indexing



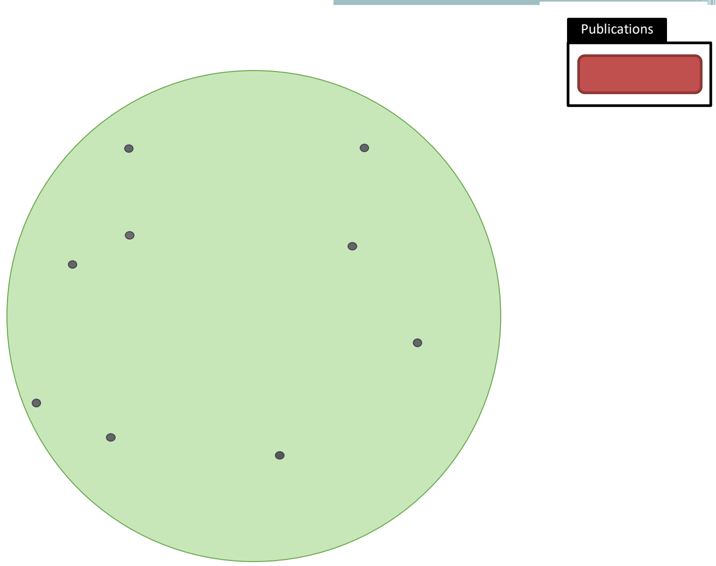
PCA

Quantization

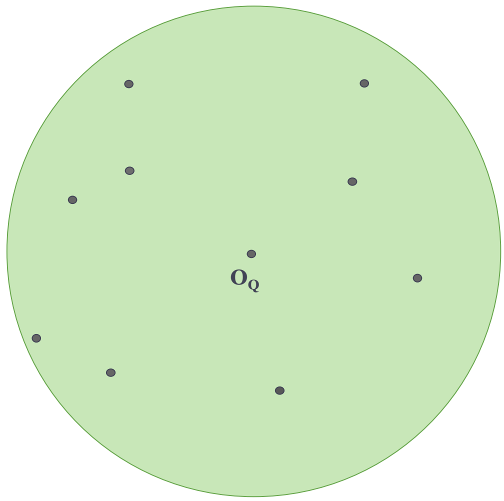
Summarization







Publications

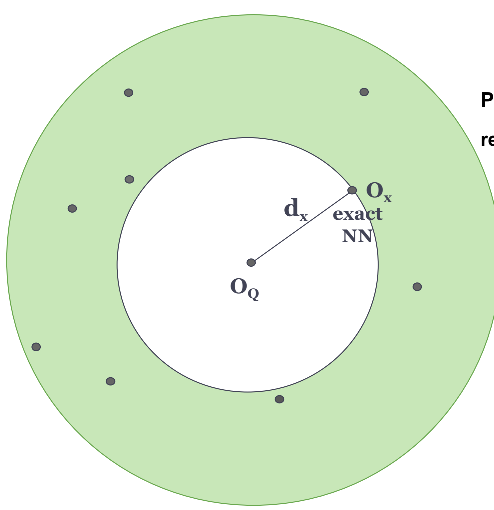




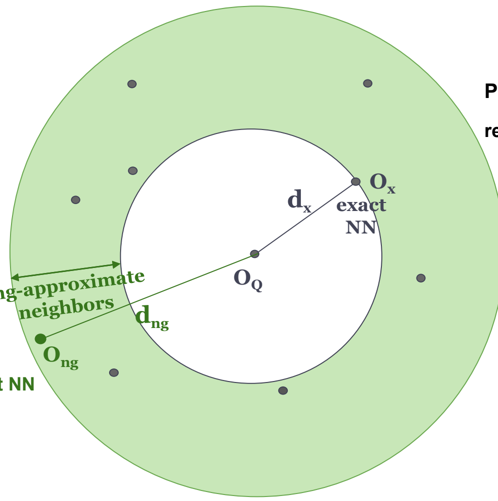

Publications

Prob( $d_x = \min\{d_i\}$ ) = 1

result is exact NN



Publications




Prob( $d_x$ = min{$d_i$} ) = 1

result is exact NN

Prob($d_{ng}$ <>= ?) = ?

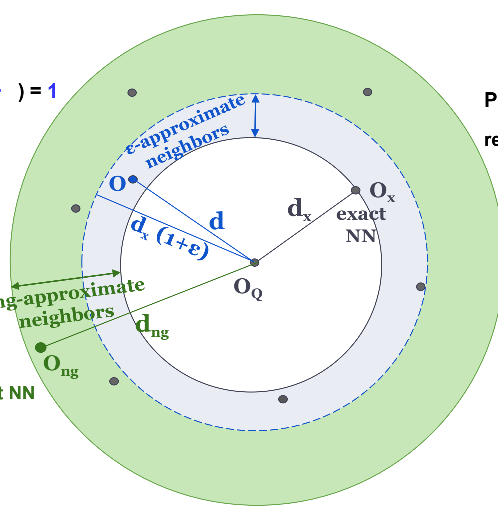
result within ? of exact NN



Publications

Prob(d <= $d_x$ (1+ ) = 1

result within
with probability 1




Prob( $d_x$ = min{$d_i$} ) = 1

result is exact NN

Prob($d_{ng}$ <>= ?) = ?

result within ? of exact NN

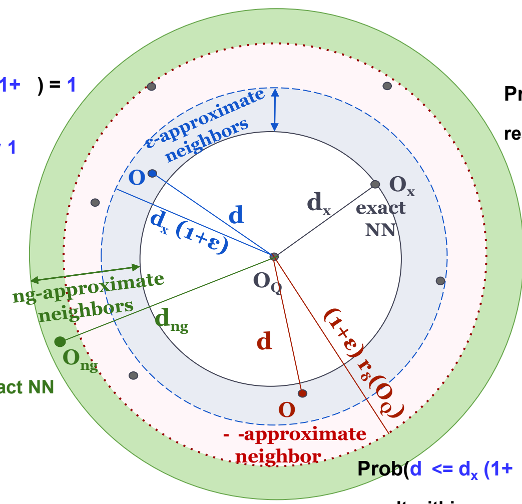
Publications

Prob($d$ <= $d_x$ (1+ ) = 1

result within
with probability 1

Prob( $d_x$ = min{$d_i$} ) = 1

result is exact NN




Prob($d_{ng}$ <>= ?) = ?

result within ? of exact NN

Prob($d$ <= $d_x$ (1+ ) >=

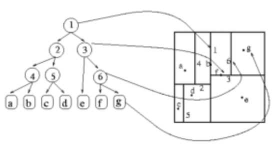
result within
with





# Multi-dimensional Indexes

**KD-Tree**

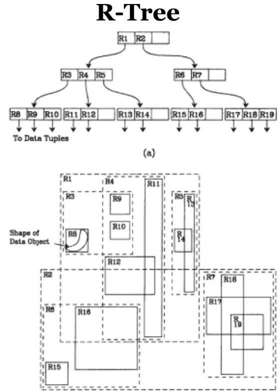


Yu Kai Him Otto, https://medium.com/

Publications

Bentley-

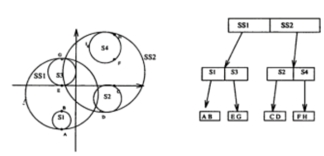
**R-Tree**




Publications

Guttman-

**TV-Tree**

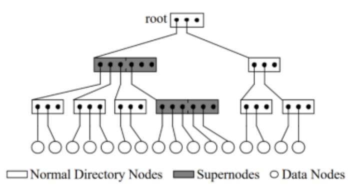


Publications

Lin et al-

# Multi-dimensional Indexes

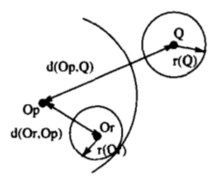
**X-Tree**




Publications

Berchtold et al
-

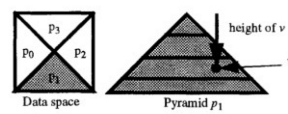
**M-Tree**




Publications

Ciaccia et al-

**Pyramid technique**




Publications

Berchtold et al
-





# Data Series
# Multi-dimensional Indexes in a new Era

their world

- focused on exact query answering
- used relatively small dataset sizes (hundreds of thousand) and dimensionalities (few dozen)
- tested for curse of dimensionality on uniform datasets(!)

new world (2000s: data series)

- looking for sublinear scalability performance on 1000x larger datasets with 100x more dimensions
- some of these indexes (R-Trees, M-Trees) used for data series with less than impressive results
- time series community proposed new indexes

# High-d Vectors
# Data Series Indexes in a new Era

- their world
  - focused on exact query answering
  - centered discussion around data series shapes/patterns
- new world (2020s: deep embeddings)
  - looking for ultra-fast performance for applications that tolerate approximate answers
  - machine learning and related communities proposed new indexes





Publications

Echihabi et al-

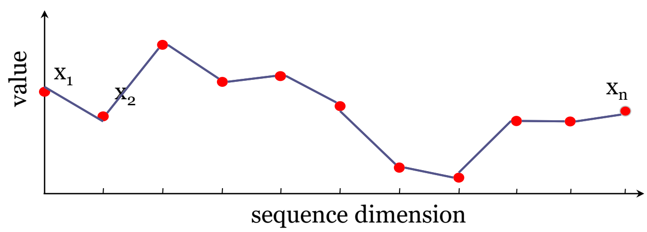
# Data Series Similarity Search
# State-of-the-Art Methods

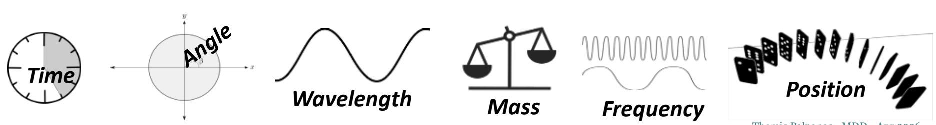
for more details:

*Karima Echihabi, Kostas Zoumpatianos, Themis Palpanas. Big Sequence Management: Scaling Up and Out. EDBT 2021*
http://helios.mi.parisdescartes.fr/~themisp/publications.html#tutorials

# Data Series

Sequence of points ordered along some dimension



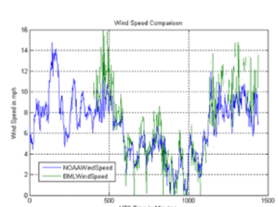








# Scientific Monitoring

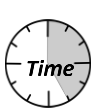
meteorology, oceanography, astronomy,





Wind speed
From ocean observing node project
http://bml.ucdavis.edu/boon/wind.html

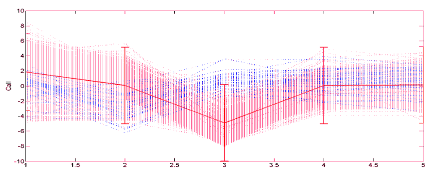
Historical stock quotes
http://money.cnn.com/2012/04/23/markets/walmart_stock/index.htm



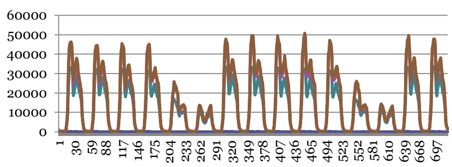
# Telecommunications

analysis of call activity patterns

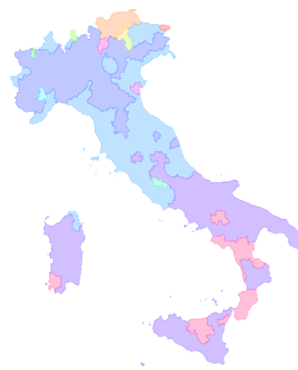
Telecom Italia




call activity for Easter Monday




average number of calls for 5 smallest clusters



clustermap of incoming calls time series

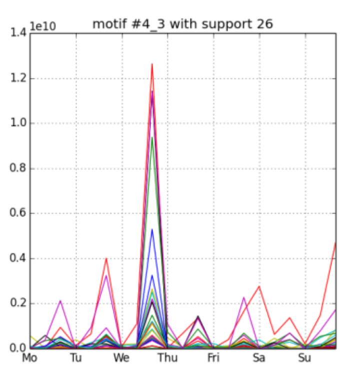




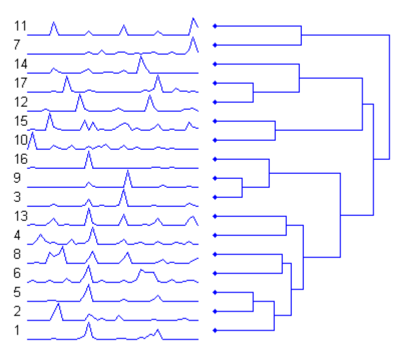
# Home Networks

temporal usage behavior analysis of home networks

Portugal Telecom



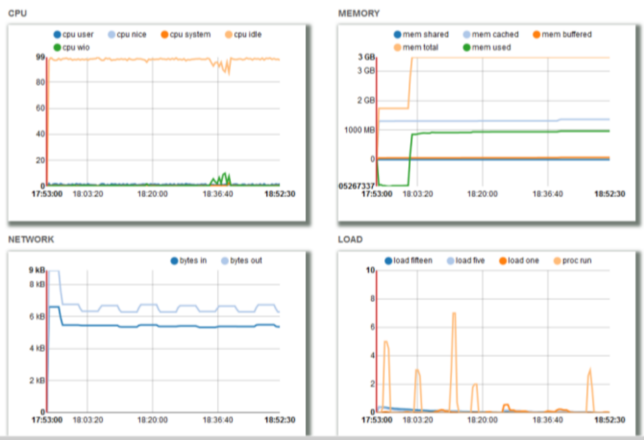

(previously unknown) frequent behavior pattern

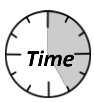



clustering based on user activity patterns

# Data Centers

cloud utilization/operation/health monitoring



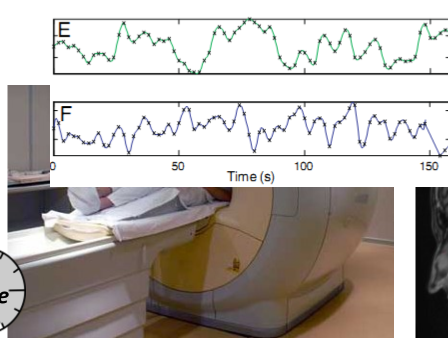



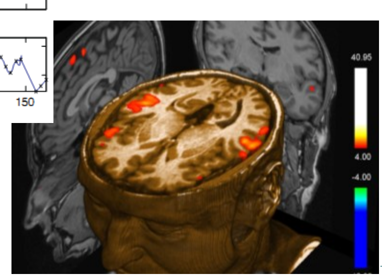




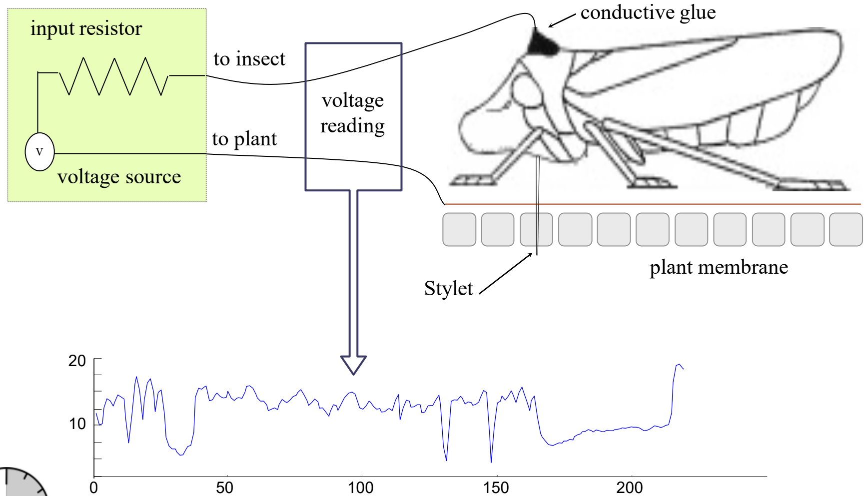

# Neuroscience

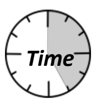
functional Magnetic Resonance Imaging (fMRI) data
- primary experimental tool of neuroscientists
- reveal how different parts of brain respond to stimuli




# Entomology



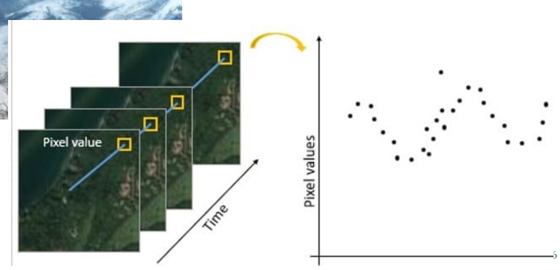



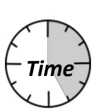




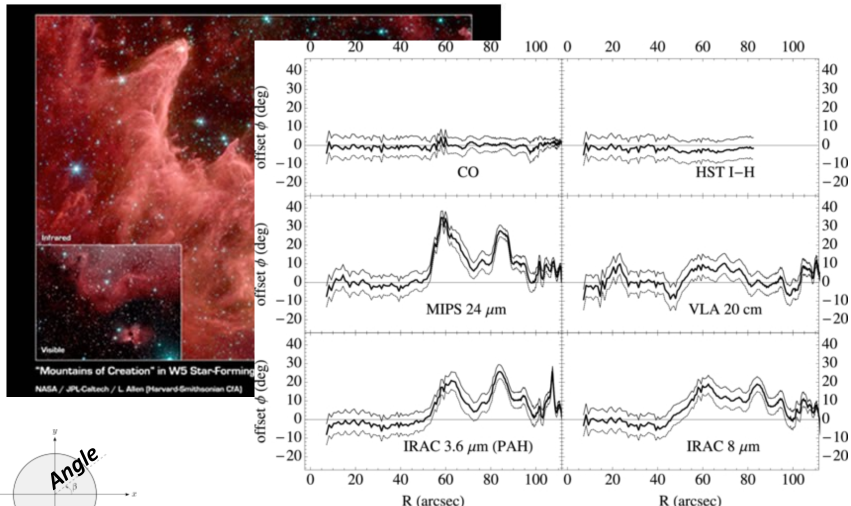

# Remote Sensing

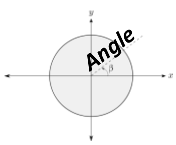
Earth monitoring



# Astrophysics



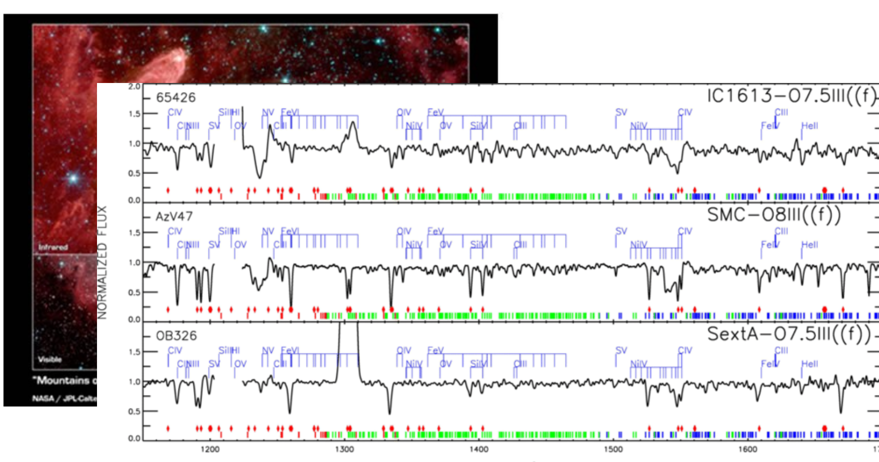




Schinnerer et al.





# Astrophysics

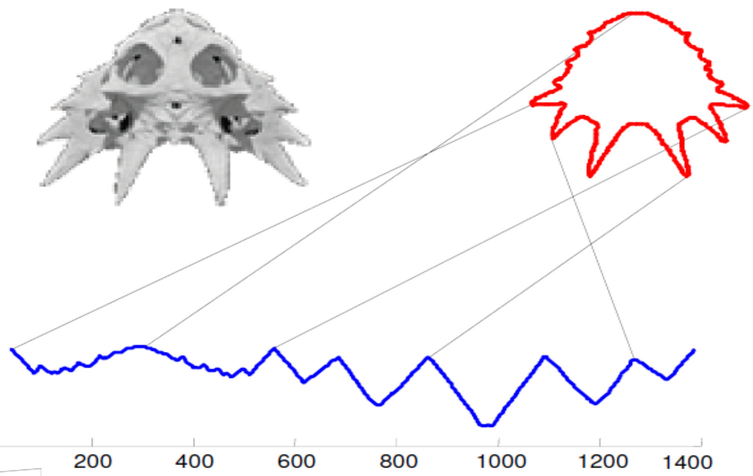


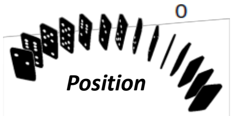


**Wavelength**

Garcia et al.

# Paleontology



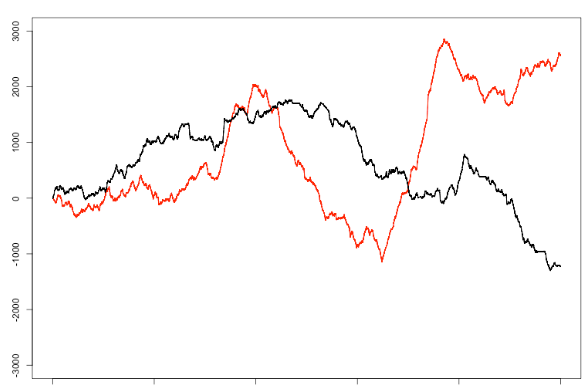





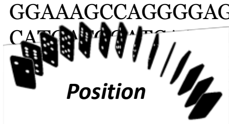



# Biology

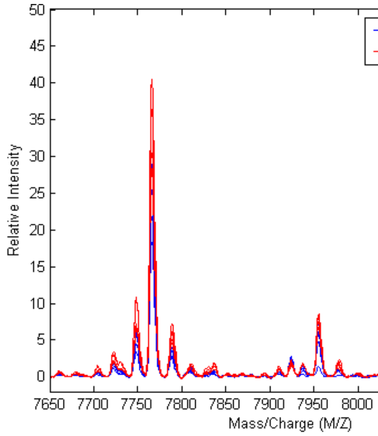
GTCAATGGCCAGGATATTAGAACAGTACTCTGTGAACCCTATTTATGGTGGCACCCTTAGACTAA
GATAACACAGG CCTGTAAAGAG
AGAAGTGCTAA TTGACTAAAGC
CCAAGGGATTG AGTCAGATTAA
TATGTGGGTGC TGAGGAGAAA
CTCCTCCACAGC GCCAGAGGATC
TGTAAAAGAGA ACAAGCAAAA
AAGAGCACTCA TGGCCACCAAG
CCAGTTGTCAAT GAGTGTCTTCT
ATTTCTTCCAGC TCCTTTACAATT
GTTATTTATGTC CATACTGTCTT
GGCCAAGCATC TTGCACGTGGT
TATTTATTAGTC AAGAGCAGAAT
CAATGAGCCGC AGATGAGCATT
GAAGAGGATGC CAGGTTGTTGC
CATGTTTGTTTT TTAGAACAGTA
CTCTGTGAACCC AGAGGTTGACA

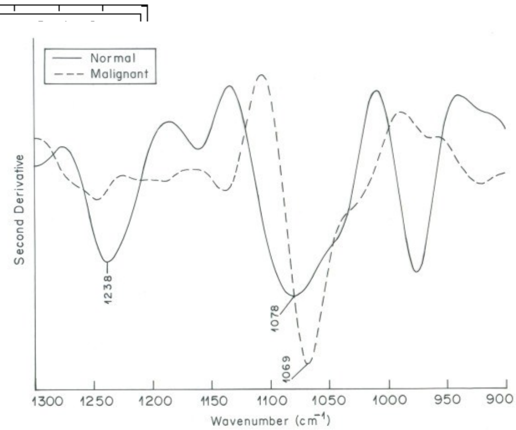



GGAAAGCCAGGGGAGCAGGGAAGCCTCCTGTAAAGAGAGAAGTGCTAAGTCTCCTTTCTAAGGCA
CA GAAAGTCACATTTGACTAAAGCCCAAGGGATTGTTGCTTCTAATCCGATTC
GCAAACTAAGAGTCAGATTAATATGTGGGTGCCAAAATAAATAAACAAATA
AGGTTTAAGTGAGGAGAAACTCCTCCACACTTGCTACCGAGGCAGAACCG

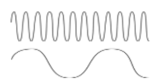


# Medicine




***Mass***



***Frequency***





# What do we want to do with them? - simple query answering

**select values in time interval**

**select values in some range**

**select some data series**

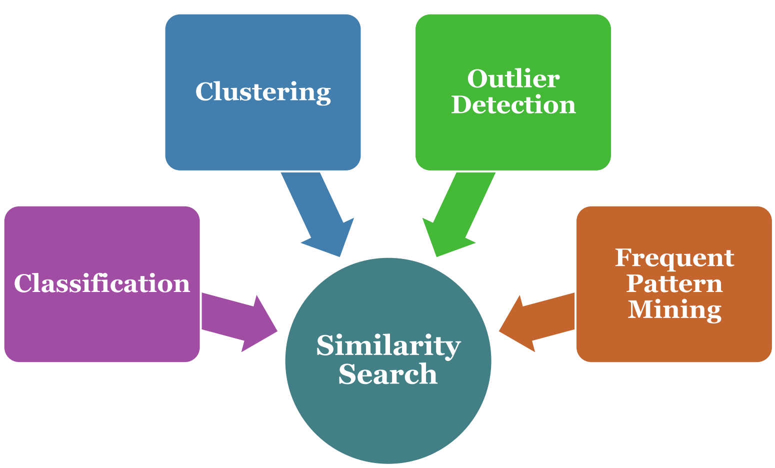
**combinations of those**

## What do we want to do with them? - complex analytics

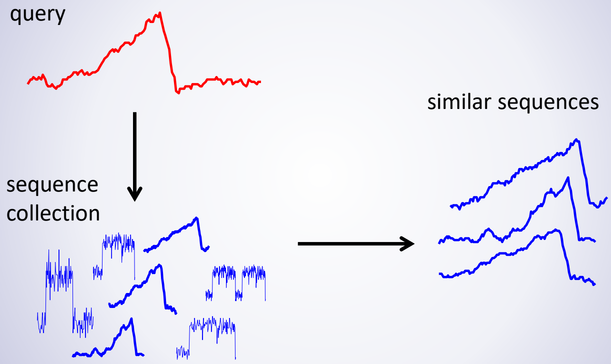





## What do we want to do with them? - complex analytics



## What do we want to do with them? - complex analytics








## What do we want to do with them? - complex analytics

Euclidean

$$D(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

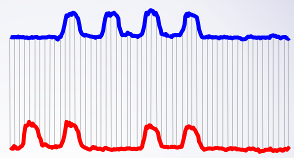
Dynamic Time Warping (DTW)

$$D_{dtw}(X,Y) = f(n,m)$$

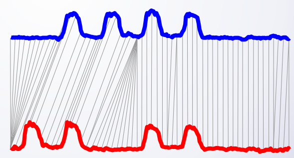
$$f(i,j) = \|x_i - y_j\| + \min \begin{cases} f(i,j-1) \\ f(i-1,j) \\ f(i-1,j-1) \end{cases}$$

# What do we want to do with them? - complex analytics

Euclidean



Dynamic Time Warping (DTW)

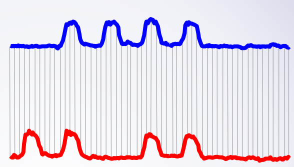






# What do we want to do with them? - complex analytics

Euclidean



equivalence to
-
- Cosine Similarity
- Maximum Inner Product Similarity

## What do we want to do with them? - complex analytics

Clustering

Outlier Detection

**HARD, because of very high dimensionality: each vector has 100s-1000s of points!**

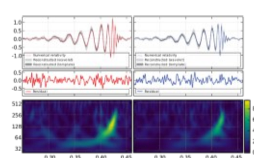
**even HARDER, because of very large size: millions to billions of vectors (multi-TBs)!**

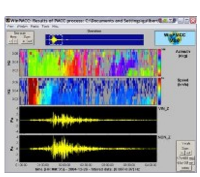




## Real Use-Cases

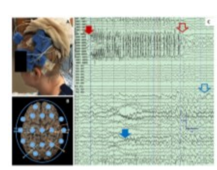
astrophysics: gravitational waves, **TB/hour**
partner: **European Gravitational Observatory (EGO)**
*Pisa, Italy*

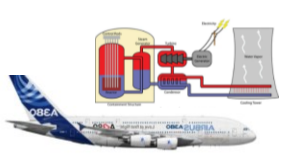




seismology: seismic sequences, **100s of TB**
partner: **Atomic Energy Commission (CEA)**
*Paris, France*

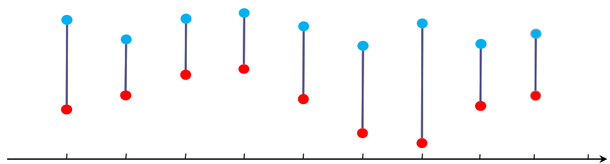
neuroscience: intracranial EEG sequences, **TB/patient**
partner: **Paris Brain Institute (ICM)**
*Paris, France*





engineering: operation monitoring, **TB-PB**
partners: **Airbus / Électricité de France (EDF)**
*Toulouse / Paris, France*

# Euclidean Distance



Euclidean distance
- pair-wise point distance

$$ED(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$





# Similarity Matching
# Fast Euclidean Distance

similarity matching requires many distance computations
- can significantly slow down processing
  - because of large number of data series in the collection
  - because of high dimensionality of each data series

in case of Euclidean Distance, we can speedup processing by
- smart implementation of distance function
- early abandoning

result in considerable performance improvement

Publications

Keogh-

## Similarity Matching
# Fast Euclidean Distance

smart implementation of distance function





Publications

Keogh-

## Similarity Matching
# Fast Euclidean Distance

smart implementation of distance function

do not compute the square root (of the Euclidean Distance)

does not alter the results

saves precious CPU cycles

Publications

Keogh-

## Similarity Matching
## Fast Euclidean Distance

early abandoning

stop the distance computation as soon as it exceeds the value of bsf

( ) ( )

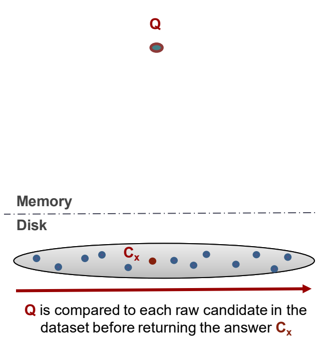
does not alter the results

avoids useless computations





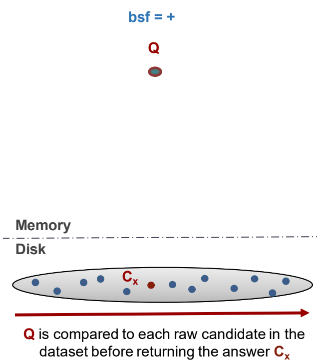
## Similarity Matching
## Serial Scan




(a) Serial scan

Answering a similarity search query using different access paths

# Similarity Matching
# Serial Scan



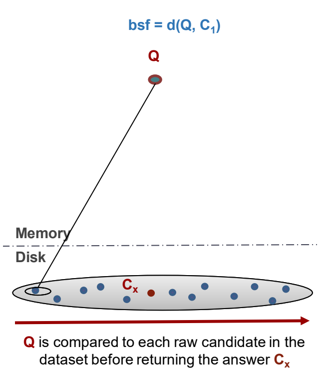

(a) Serial scan

Answering a similarity search query using different access paths





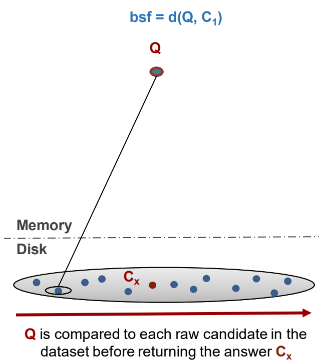
# Similarity Matching
# Serial Scan




(a) Serial scan

Answering a similarity search query using different access paths

# Similarity Matching
# Serial Scan



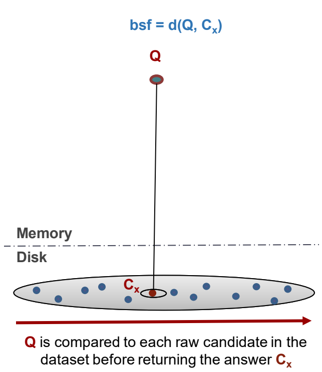

(a) Serial scan

Answering a similarity search query using different access paths





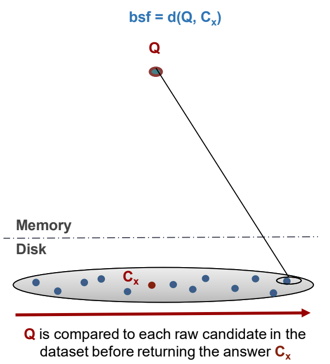
# Similarity Matching
# Serial Scan




(a) Serial scan

Answering a similarity search query using different access paths

# Similarity Matching
# Serial Scan



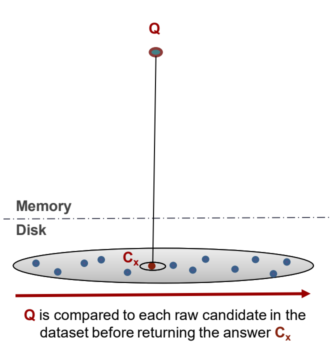

(a) Serial scan

Answering a similarity search query using different access paths





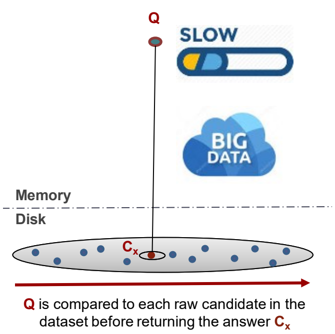
# Similarity Matching
# Serial Scan




(a) Serial scan

Answering a similarity search query using different access paths

# Similarity Matching
# Serial Scan




(a) Serial scan

Answering a similarity search query using different access paths





Publications

Faloutsos-

# GEMINI Framework

- Raw data: original full-dimensional space
- Summarization: reduced dimensionality space
- Searching in original space *costly*
- Searching in reduced space *faster*:
  - Less data, indexing techniques available, lower bounding
- Lower bounding enables us to
  - *prune search space:* throw away data series based on reduced dimensionality representation
  - *guarantee correctness* of answer
    - no false negatives
    - false positives filtered out based on raw data

# GEMINI Framework

GEMINI Solution: Quick filter-and-refine:

- extract *m* features (numbers, e.g., average)
- map to point in *m*-dimensional feature space
- organize points
- retrieve the answer using a NN query
- discard false positives





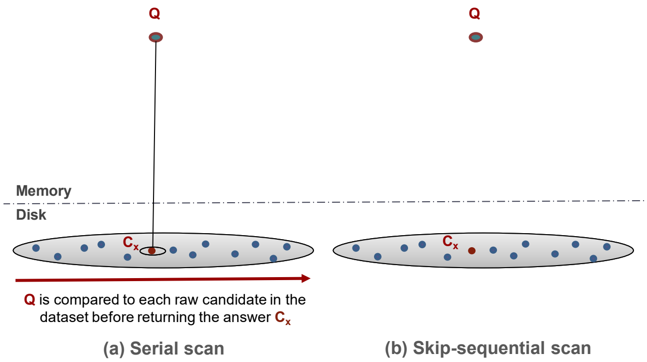
# GEMINI: contractiveness

GEMINI works when:

$$D_{feature}(F(x), F(y)) \leq D(x, y)$$

*Note that, the closer the feature distance to the actual one, the better*

# Similarity Matching
## Indexes vs. Scans

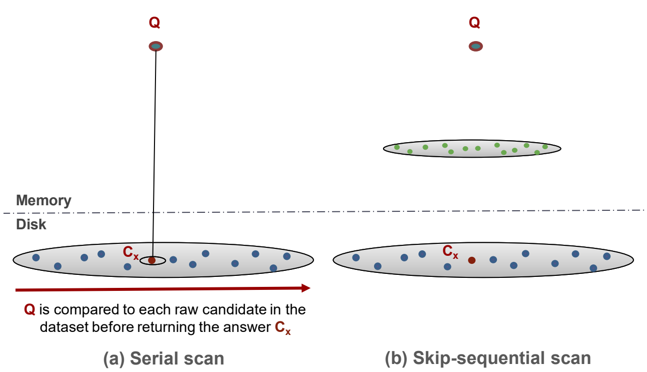



Answering a similarity search query using different access paths



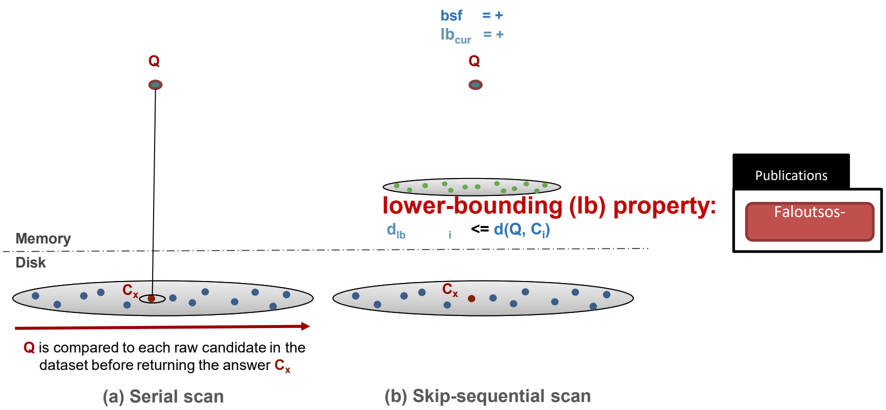


# Similarity Matching
## Indexes vs. Scans




Answering a similarity search query using different access paths

# Similarity Matching
## Indexes vs. Scans

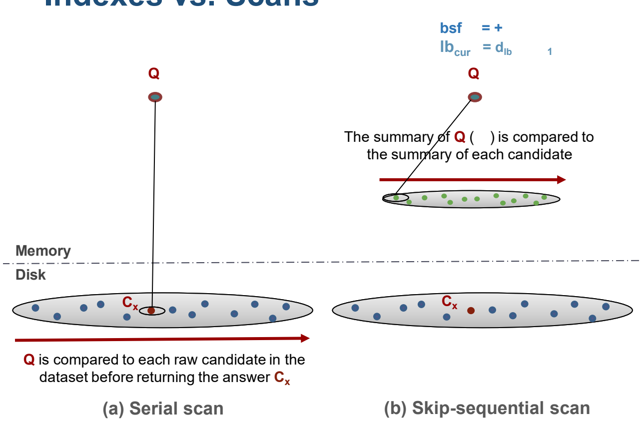



Answering a similarity search query using different access paths



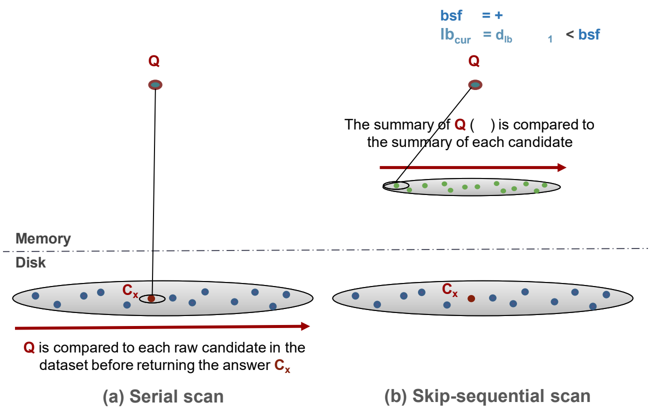


# Similarity Matching
## Indexes vs. Scans




Answering a similarity search query using different access paths

# Similarity Matching
## Indexes vs. Scans

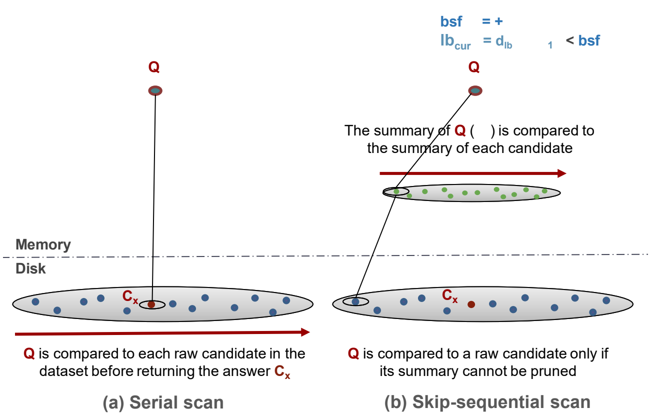



Answering a similarity search query using different access paths



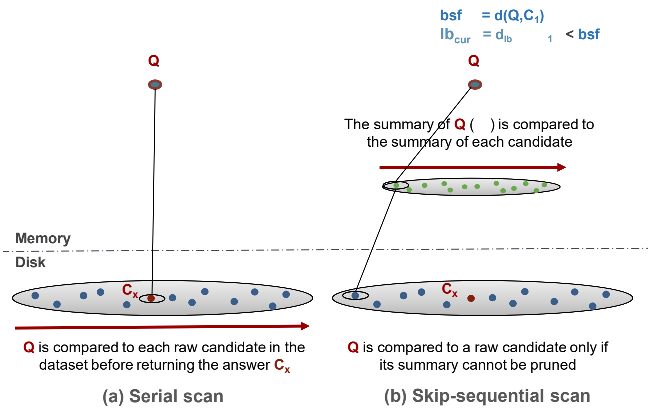


# Similarity Matching
## Indexes vs. Scans




Answering a similarity search query using different access paths

# Similarity Matching
## Indexes vs. Scans

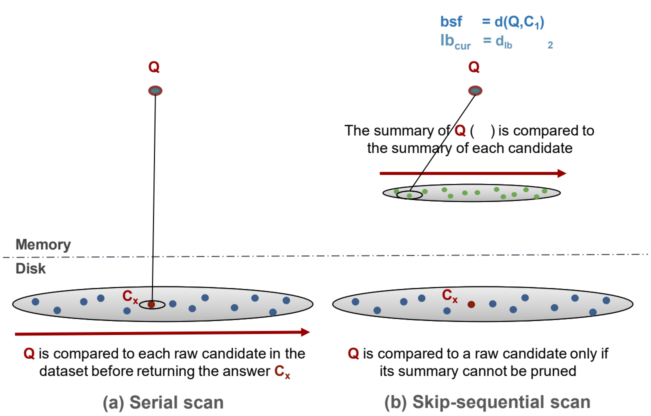



Answering a similarity search query using different access paths



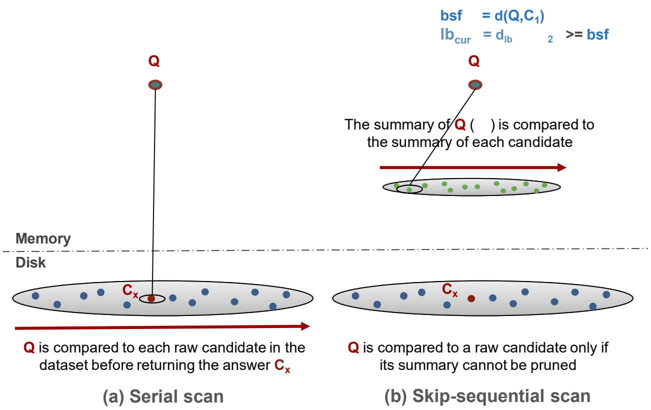



# Similarity Matching
## Indexes vs. Scans


Answering a similarity search query using different access paths

# Similarity Matching
## Indexes vs. Scans

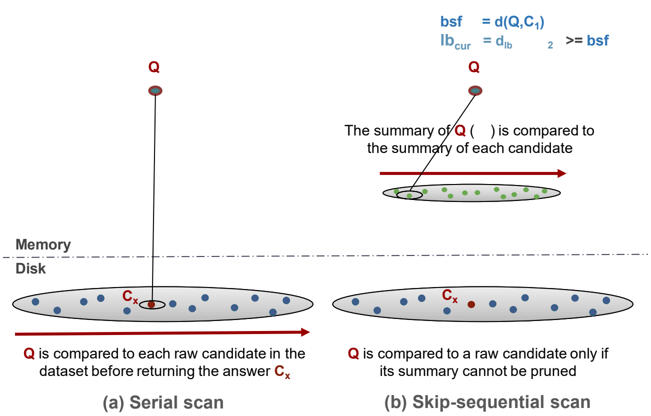



Answering a similarity search query using different access paths



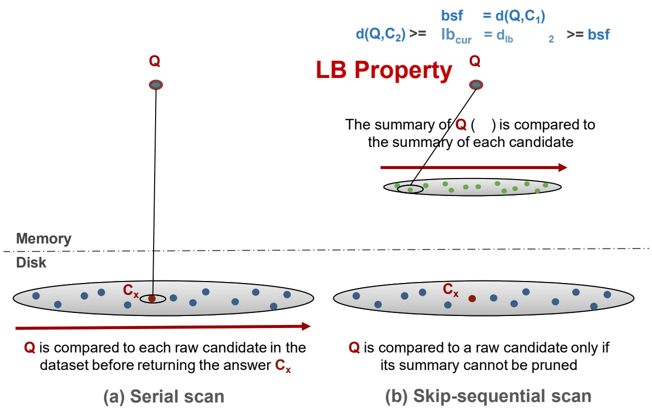


# Similarity Matching
## Indexes vs. Scans




Answering a similarity search query using different access paths

# Similarity Matching
## Indexes vs. Scans

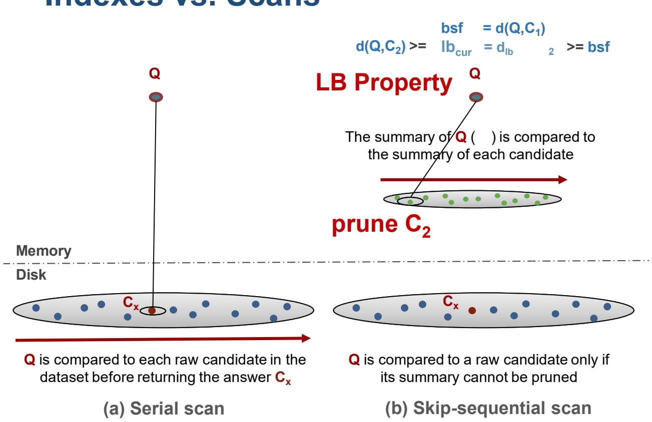



Answering a similarity search query using different access paths



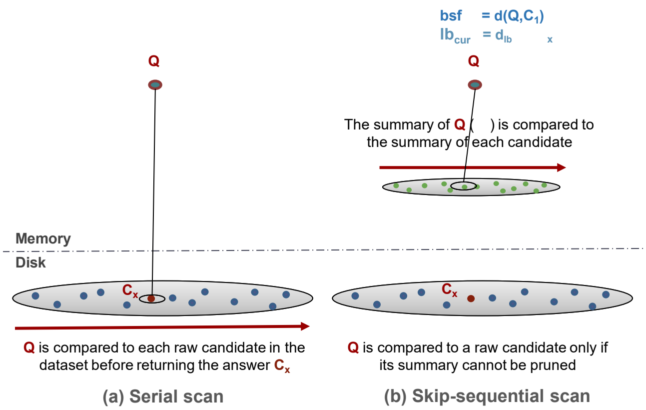


# Similarity Matching
## Indexes vs. Scans




Answering a similarity search query using different access paths

# Similarity Matching
## Indexes vs. Scans

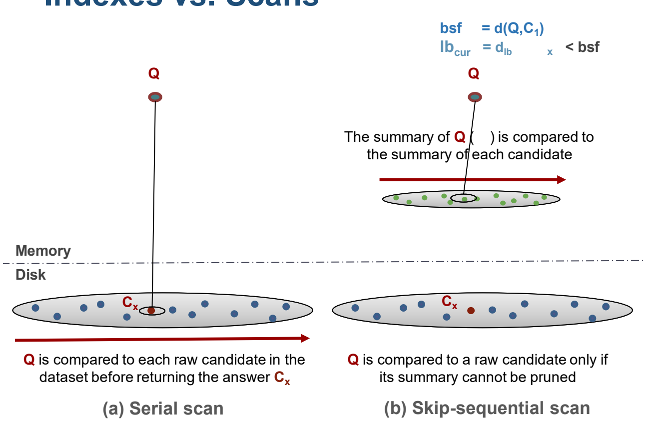



Answering a similarity search query using different access paths





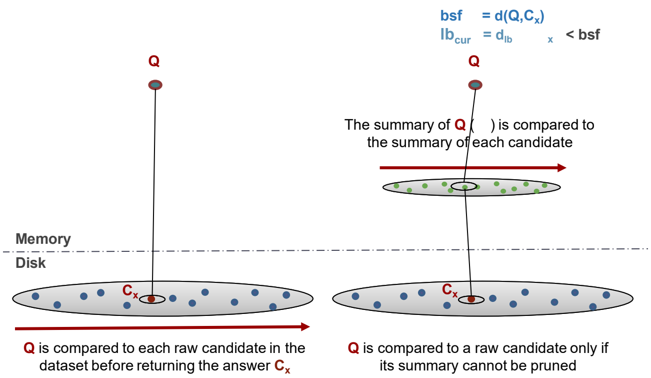
Similarity Matching
Indexes vs. Scans




Answering a similarity search query using different access paths

Themis Palpanas - MDD - Apr 2026

# Similarity Matching
## Indexes vs. Scans



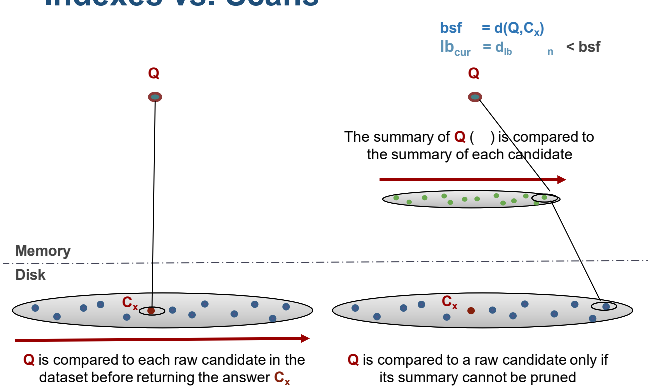

(a) Serial scan (b) Skip-sequential scan

Answering a similarity search query using different access paths





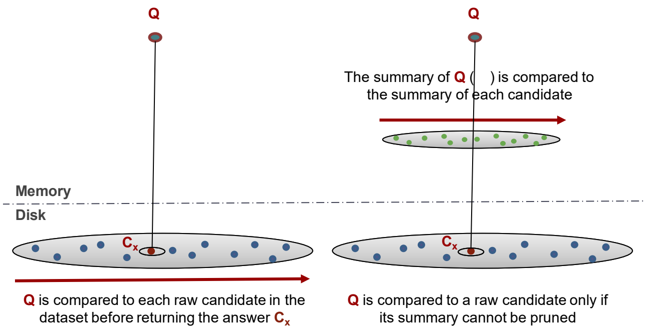
# Similarity Matching
## Indexes vs. Scans




(a) Serial scan (b) Skip-sequential scan

Answering a similarity search query using different access paths

# Similarity Matching
## Indexes vs. Scans



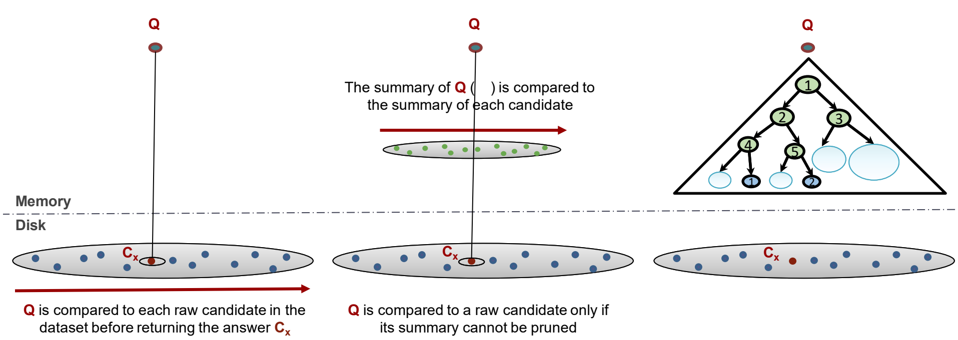

(a) Serial scan (b) Skip-sequential scan

Answering a similarity search query using different access paths





# Similarity Matching
## Indexes vs. Scans

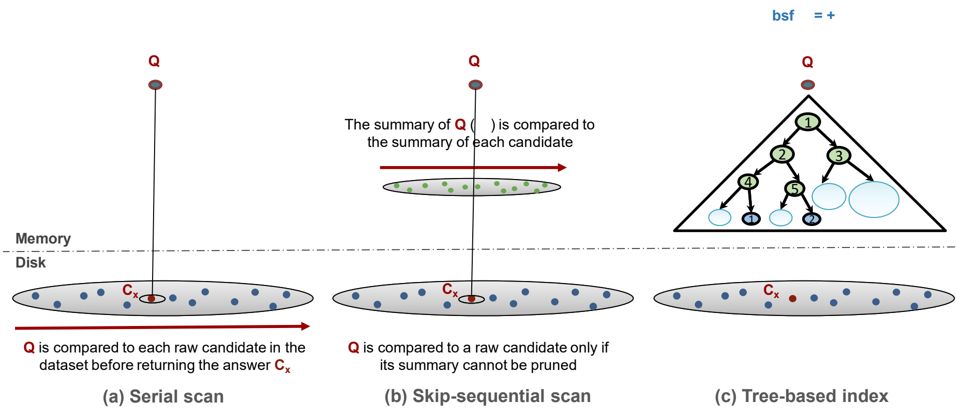



(a) Serial scan (b) Skip-sequential scan (c) Tree-based index

Answering a similarity search query using different access paths

Themis Palpanas - MDD - Apr 2026

# Similarity Matching
## Indexes vs. Scans

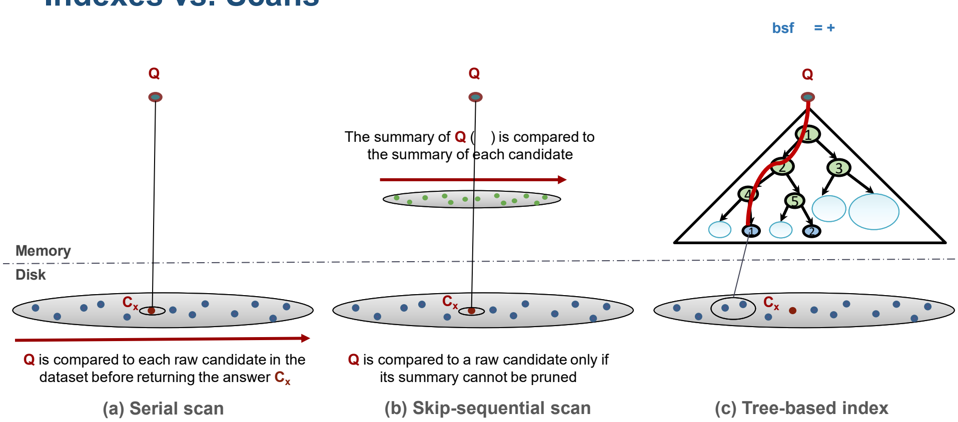



Answering a similarity search query using different access paths



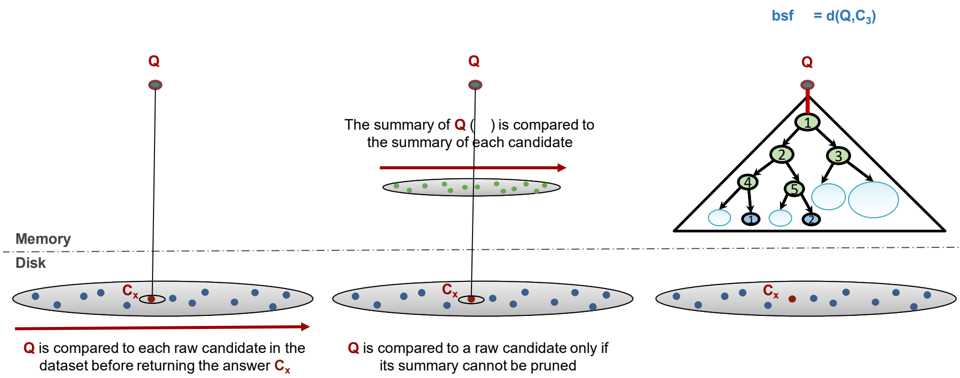


# Similarity Matching
## Indexes vs. Scans




Answering a similarity search query using different access paths

# Similarity Matching
## Indexes vs. Scans



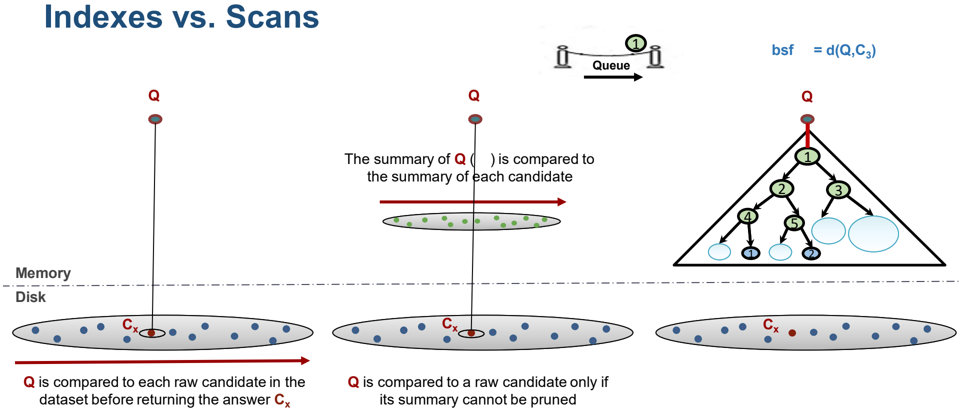

(a) Serial scan (b) Skip-sequential scan (c) Tree-based index

Answering a similarity search query using different access paths





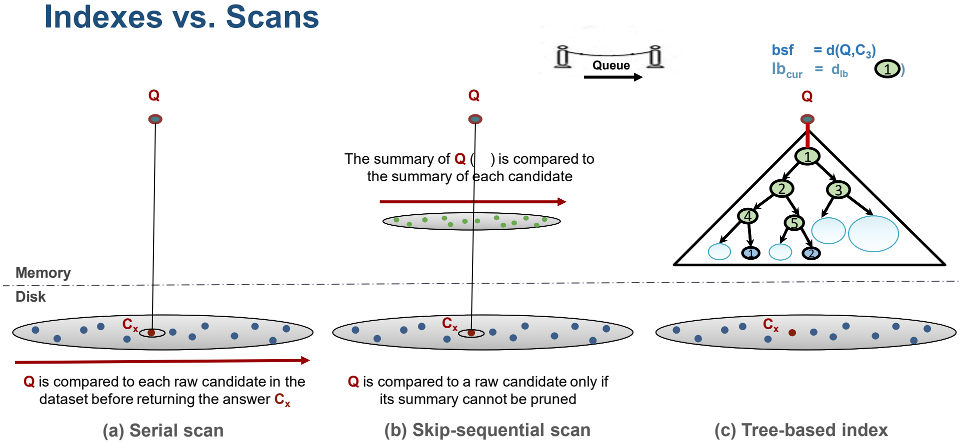
# Similarity Matching
## Indexes vs. Scans




(a) Serial scan (b) Skip-sequential scan (c) Tree-based index

Answering a similarity search query using different access paths

# Similarity Matching
## Indexes vs. Scans

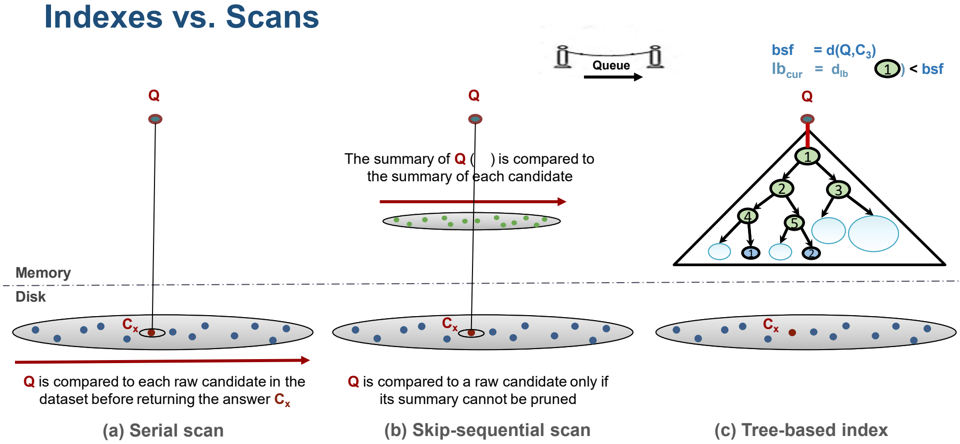



Answering a similarity search query using different access paths



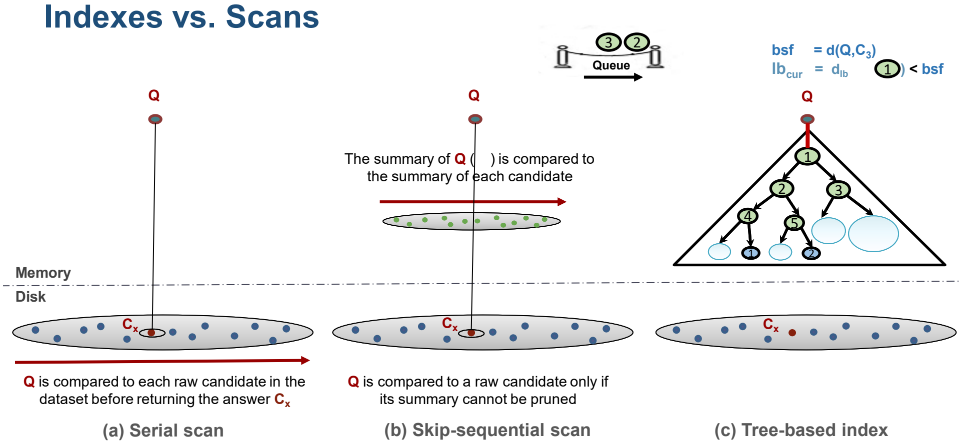


# Similarity Matching
## Indexes vs. Scans




Answering a similarity search query using different access paths

# Similarity Matching
## Indexes vs. Scans

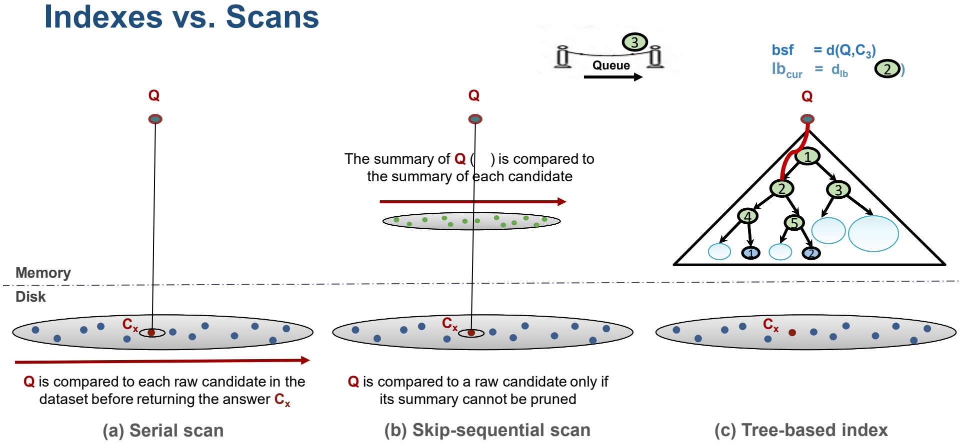



Answering a similarity search query using different access paths



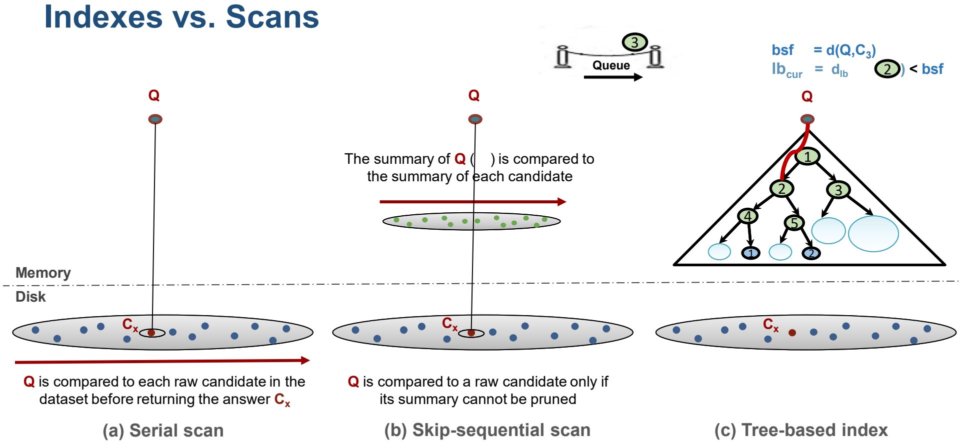


# Similarity Matching
## Indexes vs. Scans




Answering a similarity search query using different access paths

# Similarity Matching
## Indexes vs. Scans

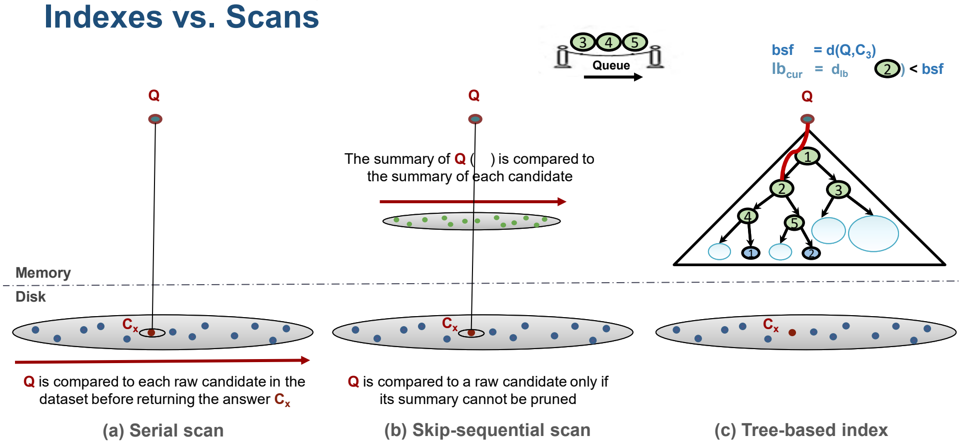



Answering a similarity search query using different access paths



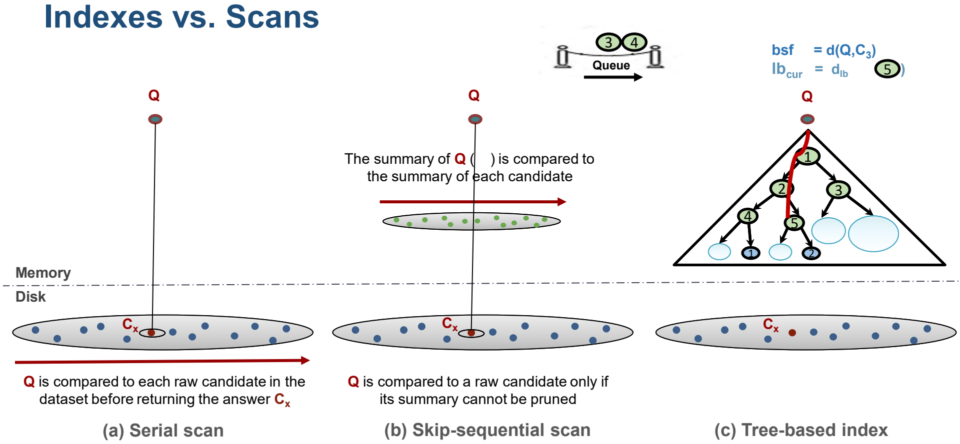


# Similarity Matching
## Indexes vs. Scans




Answering a similarity search query using different access paths

# Similarity Matching
## Indexes vs. Scans

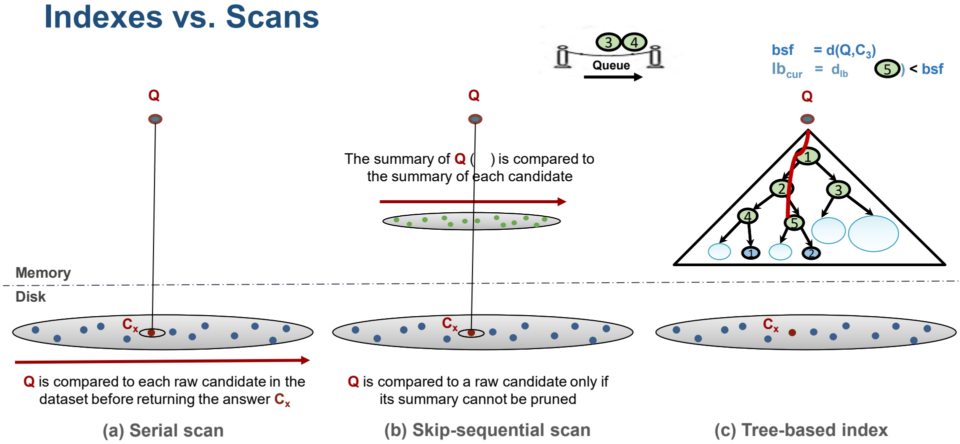



Answering a similarity search query using different access paths



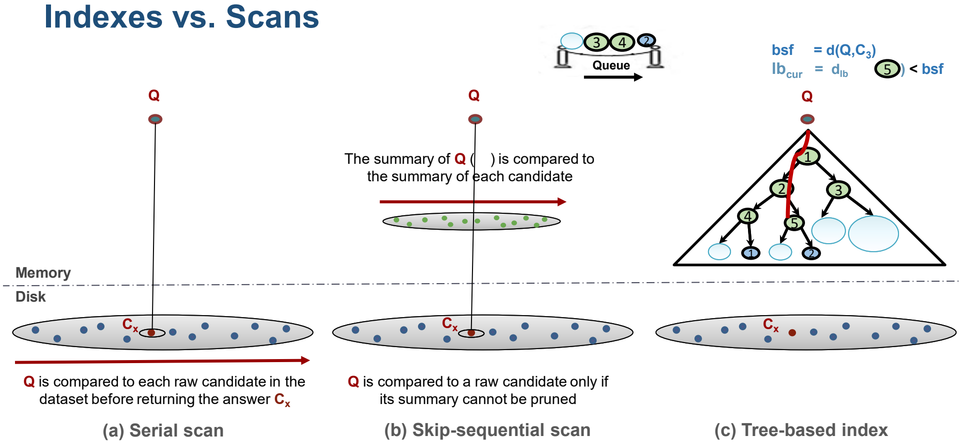


# Similarity Matching
## Indexes vs. Scans




Answering a similarity search query using different access paths

# Similarity Matching
## Indexes vs. Scans

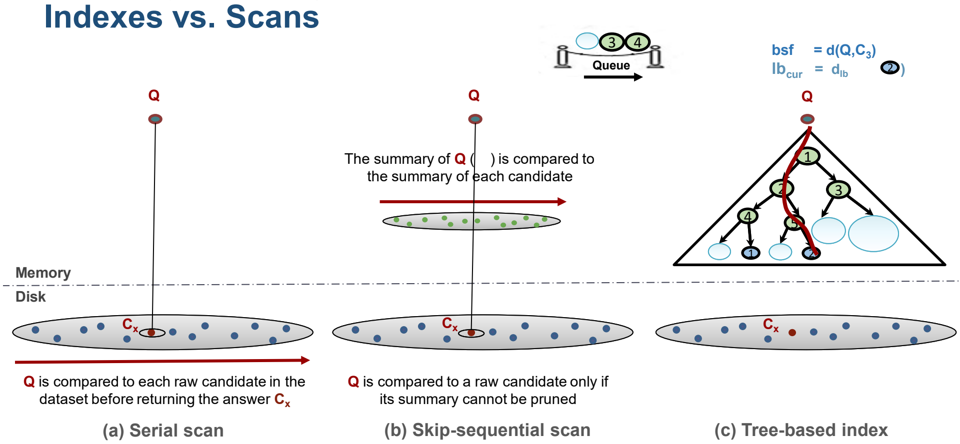



Answering a similarity search query using different access paths



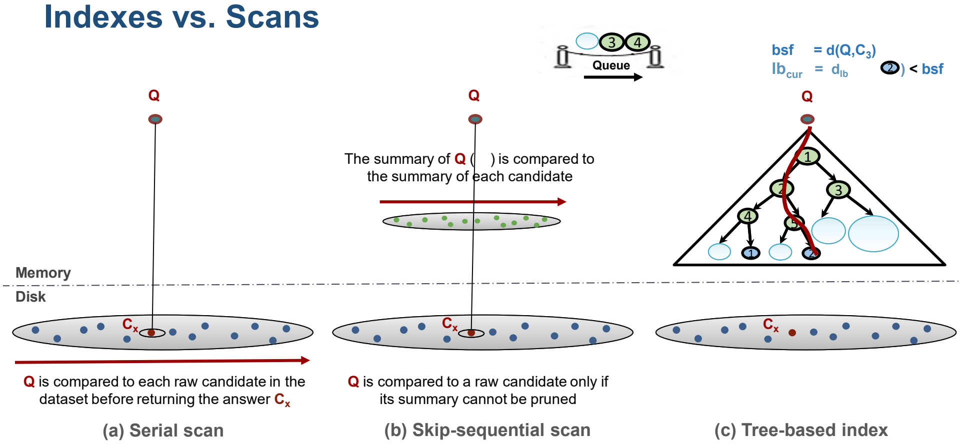



# Similarity Matching
## Indexes vs. Scans


Answering a similarity search query using different access paths

# Similarity Matching
## Indexes vs. Scans

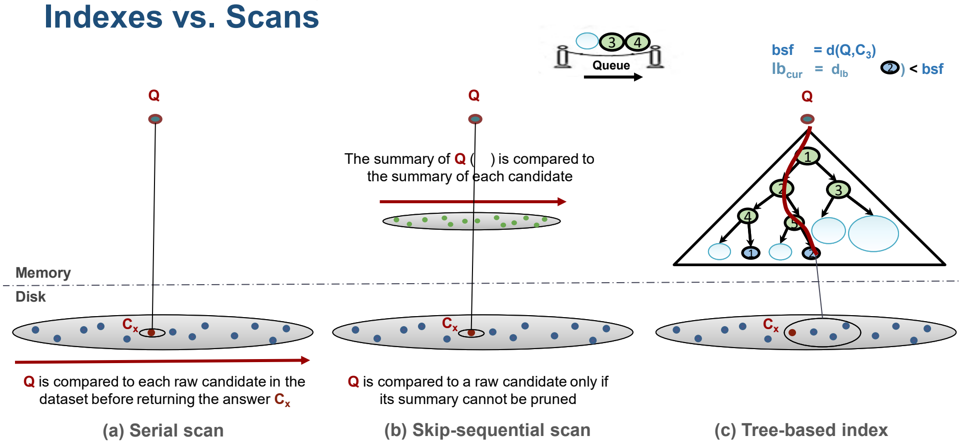



Answering a similarity search query using different access paths



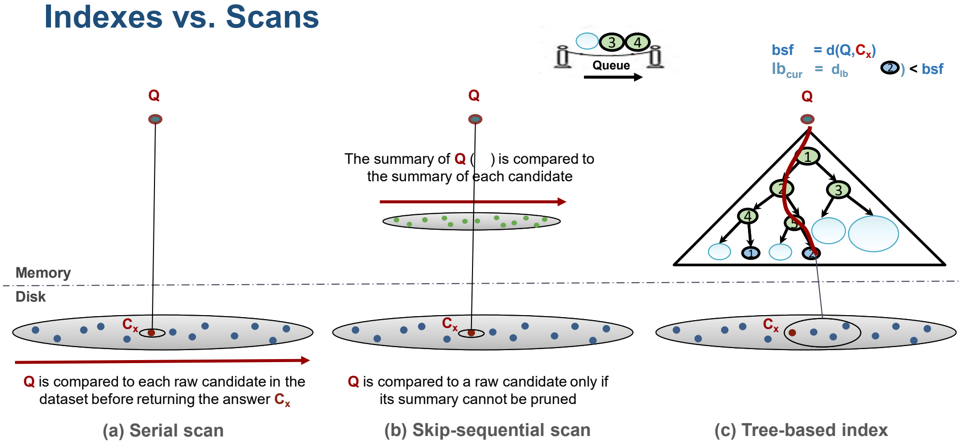


# Similarity Matching
## Indexes vs. Scans




Answering a similarity search query using different access paths

# Similarity Matching
## Indexes vs. Scans

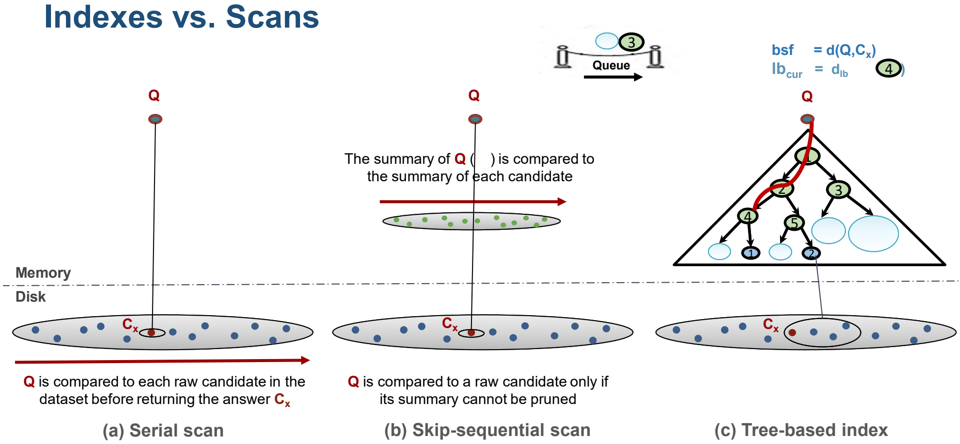



Answering a similarity search query using different access paths



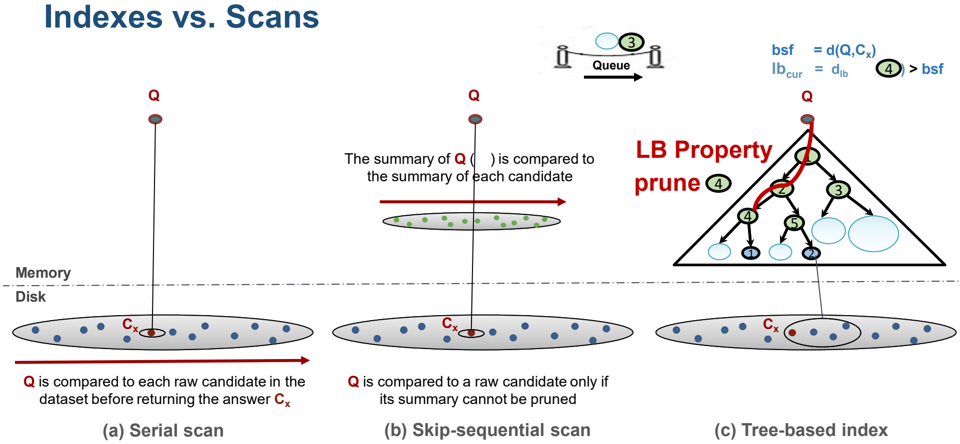


# Similarity Matching
## Indexes vs. Scans




Answering a similarity search query using different access paths

# Similarity Matching
## Indexes vs. Scans

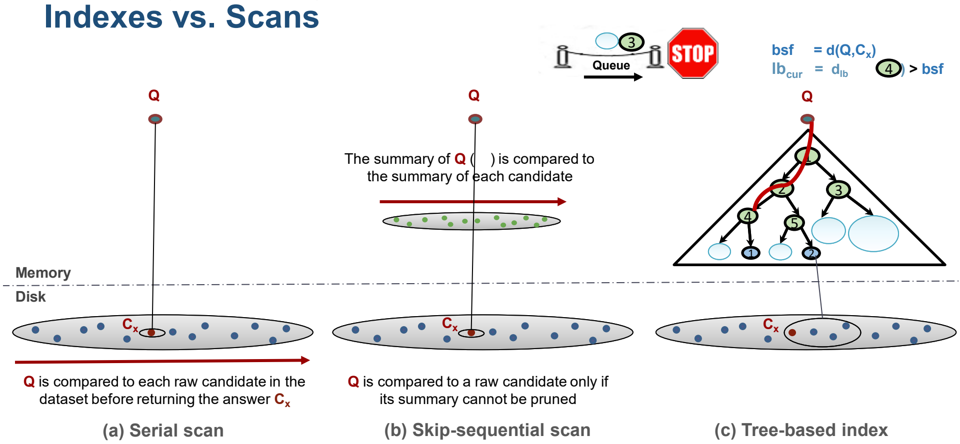



Answering a similarity search query using different access paths



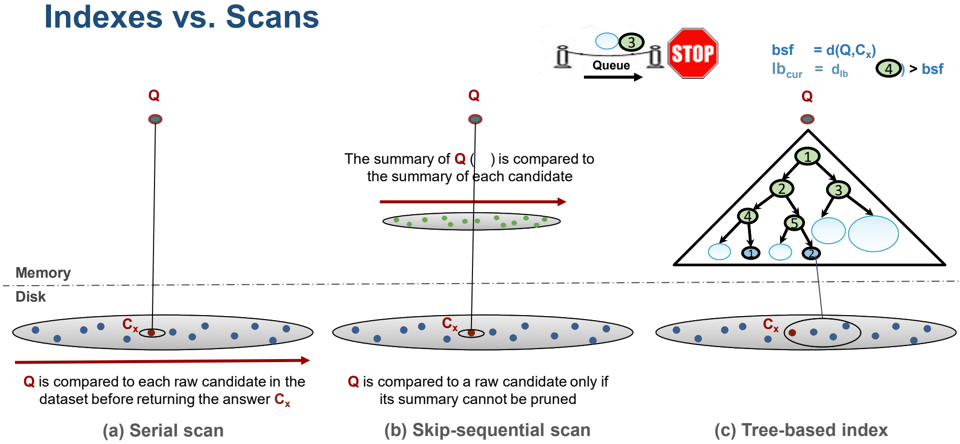


# Similarity Matching
## Indexes vs. Scans




Answering a similarity search query using different access paths

# Similarity Matching
## Indexes vs. Scans

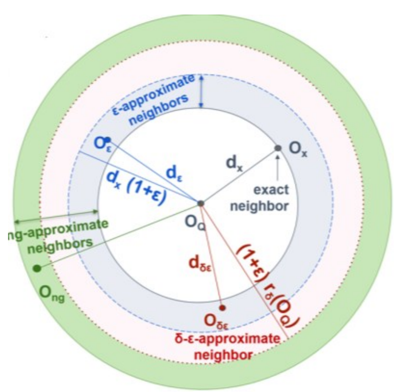



Answering a similarity search query using different access paths





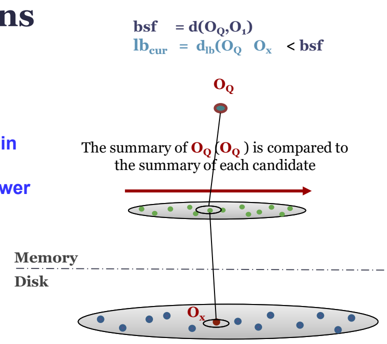
# Extensions: Skip-Sequential Scans




d <= $d_x$ (1+

Result is within

the exact answer

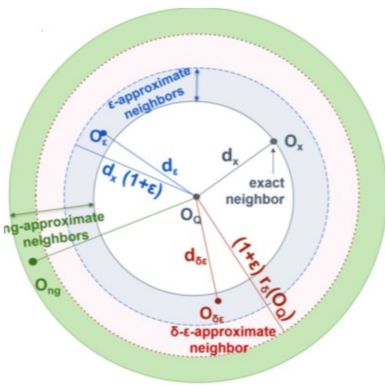


# Extensions: Skip-Sequential Scans

bsf = $d(O_Q, O_1)$

~~$lb_{cur} = d_{lb}(O_Q, O_x) < bsf$~~

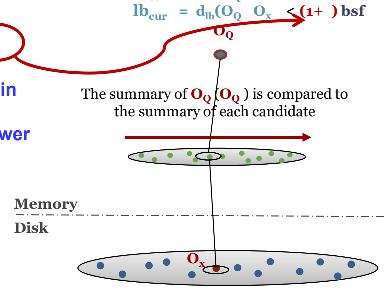
$lb_{cur} = d_{lb}(O_Q, O_x) < (1+\ )\ bsf$




d <= $d_x$ (1+

Result is within

the exact answer

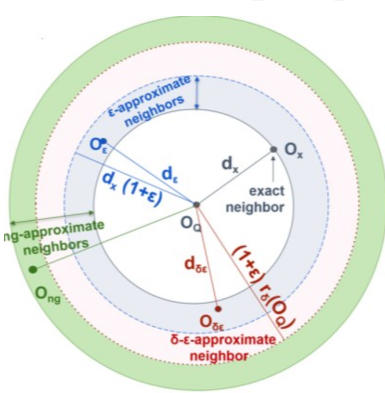







# Extensions: Skip-Sequential Scans

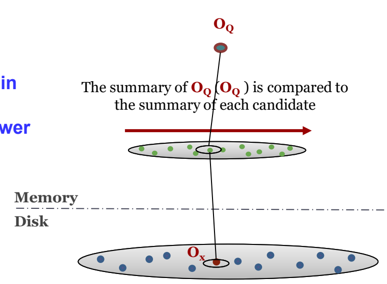
bsf = $d(O_Q, O_1)$



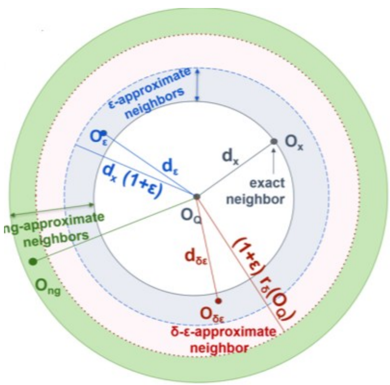

d <= $d_x$ (1+

Result is within

the exact answer



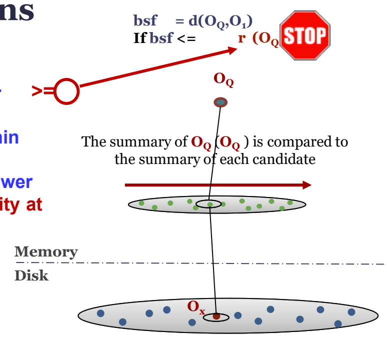
## Extensions: Skip-Sequential Scans




P{d <= $d_x$ (1+ >=

Result is within

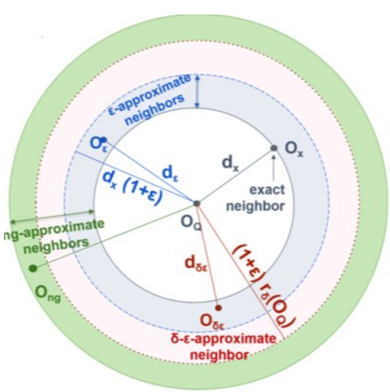
the exact answer
with probability at







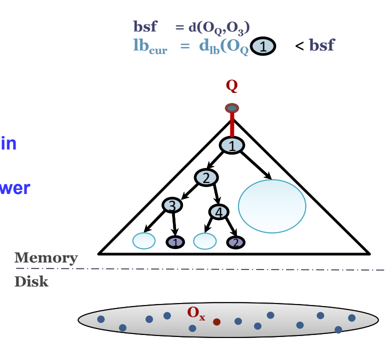

## Extensions: Indexes



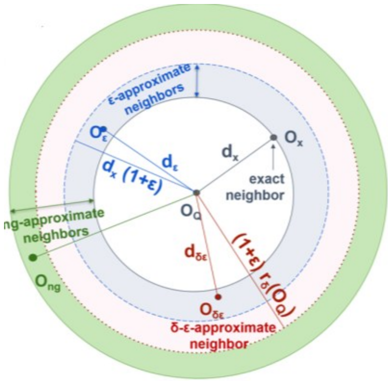

d <= $d_x$ (1+

Result is within

the exact answer



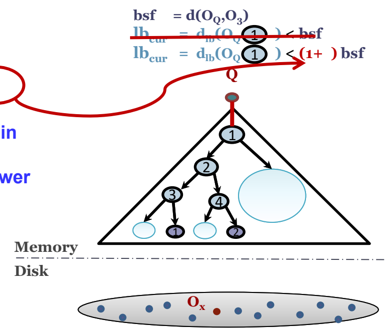
## Extensions: Indexes




d <= $d_x$(1+

Result is within

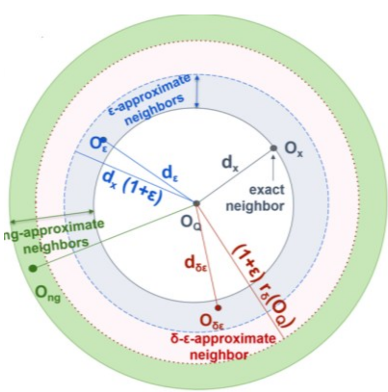
the exact answer







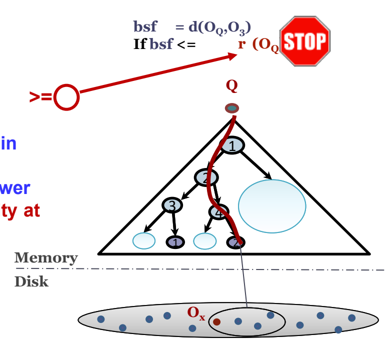

## Extensions: Indexes



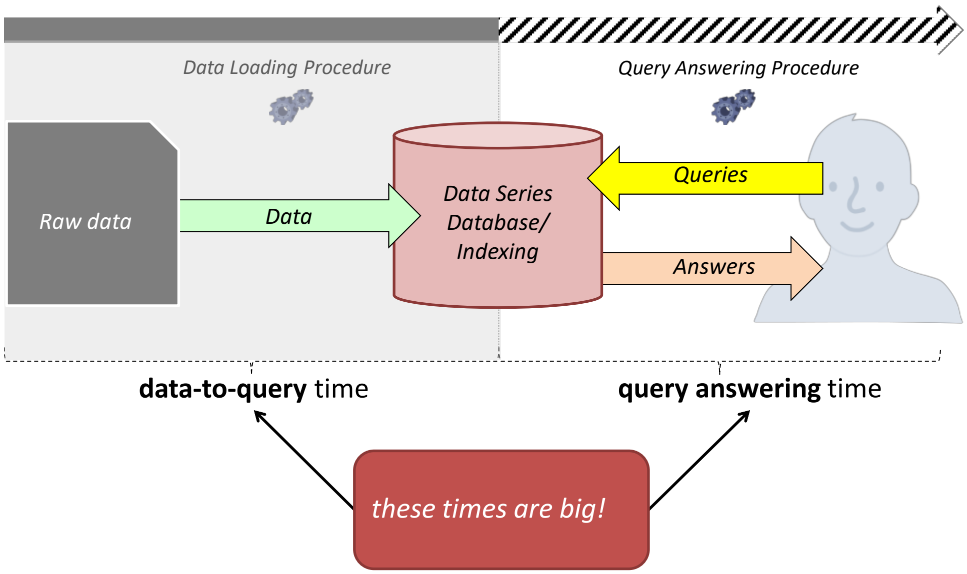

P{d <= $d_x$ (1+ >=

Result is within

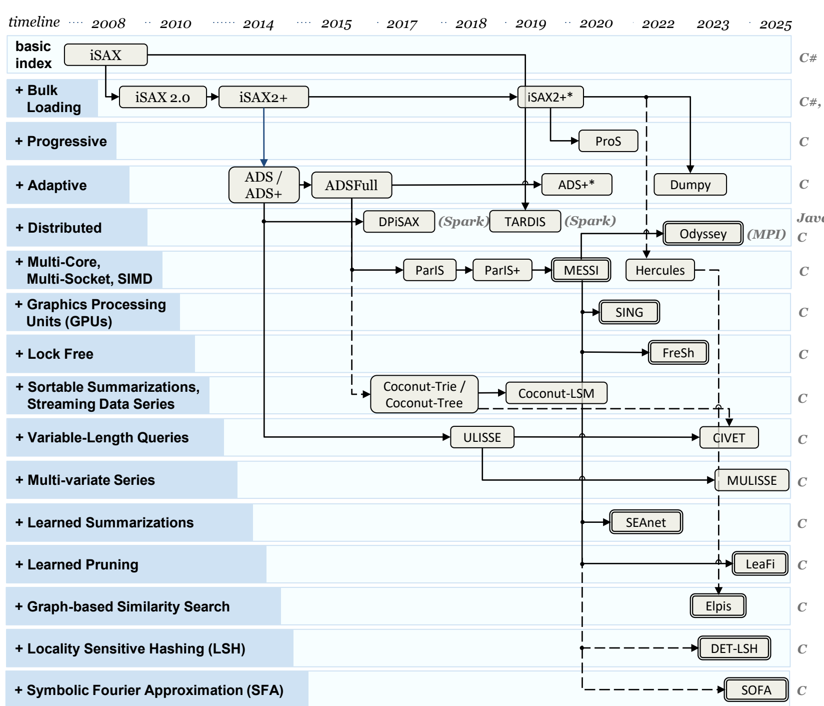
the exact answer
with probability at



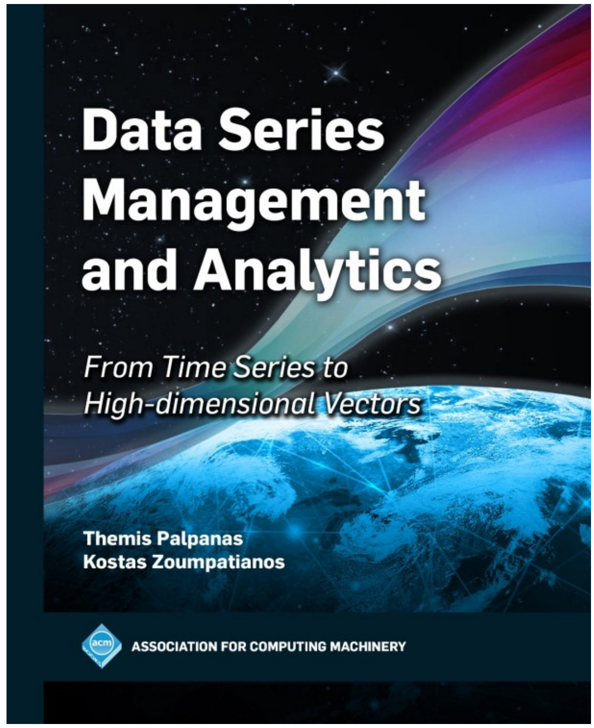
# Query answering process




Themis Palpanas - MDD - Apr 2026 104



**DaiSy** library
for Fast Exact
Similarity Search






# SAX Representation

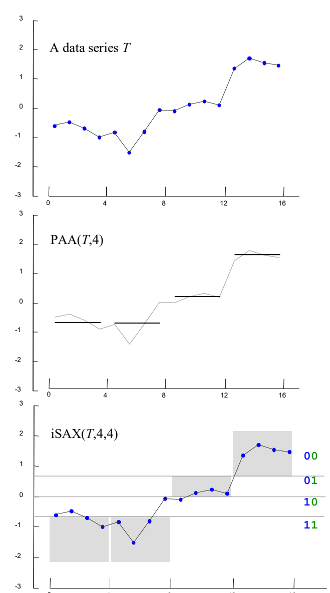
**S**ymbolic **A**ggregate appro**X**imation (SAX)

- **(1)** Represent data series $T$ of length $\boldsymbol{n}$ with $\boldsymbol{w}$ segments using Piecewise Aggregate Approximation (PAA)
  - $T$ typically normalized to = 0, = 1

$$PAA(T,w) = \overline{T} = \bar{t}_1, \ldots, \bar{t}_w$$

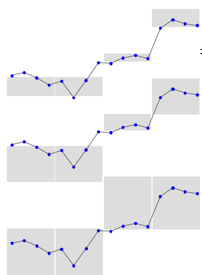
$$\text{where } \bar{t}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} T_j$$

- **(2)** Discretize into a vector of symbols
  - Breakpoints map to small alphabet $\boldsymbol{a}$ of symbols



# *i*SAX Representation

*i*SAX offers a bit-aware, quantized, multi-resolution representation with variable granularity



= { 6, 6, 3, 0} = {110 ,110 ,0111 ,000}

= { 3, 3, 1, 0} = {11 ,11 ,011 ,00 }

= { 1, 1, 0, 0} = {1 ,1 ,0 ,0 }



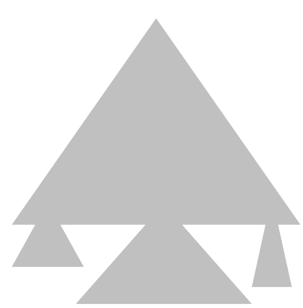


# *i*SAX Index

non-balanced tree-based index with non-overlapping regions, and controlled fan-out rate

base cardinality ***b*** (optional), segments ***w***, threshold ***th***

***th***



# *i*SAX Index

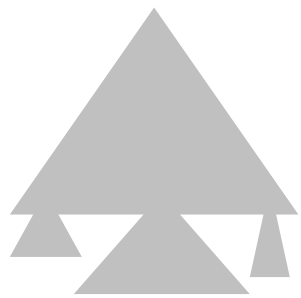
non-balanced tree-based index with non-overlapping regions, and controlled fan-out rate

base cardinality ***b*** (optional), segments ***w***, threshold ***th***

***th***

e.g., th=4, w=4, b=1

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$







# *i*SAX Index

non-balanced tree-based index with non-overlapping regions, and controlled fan-out rate

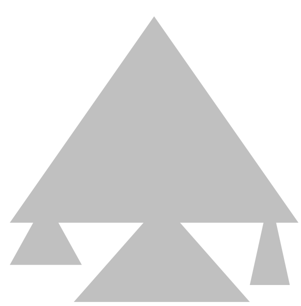
base cardinality ***b*** (optional), segments ***w***, threshold ***th***

***th***

e.g., th=4, w=4, b=1

Insert:
1 1 1 0

$$\rightarrow \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$



# *i*SAX Index

non-balanced tree-based index with non-overlapping regions, and controlled fan-out rate

base cardinality ***b*** (optional), segments ***w***, threshold ***th***

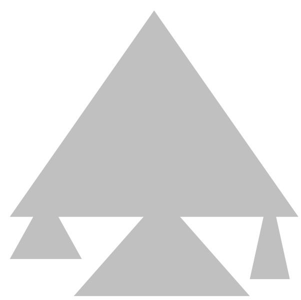
***th***

e.g., th=4, w=4, b=1

1 1 **1** 0

1 1 1**0** 0
1 1 1**0** 0

1 1 1**1** 0
1 1 1**1** 0
1 1 1**1** 0





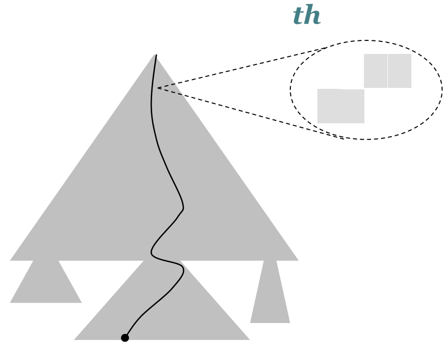


# *i*SAX Index

non-balanced tree-based index with non-overlapping regions, and controlled fan-out rate

base cardinality ***b*** (optional), segments ***w***, threshold ***th***

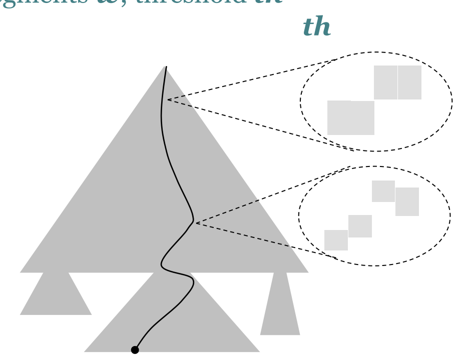
***th***



# *i*SAX Index

non-balanced tree-based index with non-overlapping regions, and controlled fan-out rate

base cardinality ***b*** (optional), segments ***w***, threshold ***th***



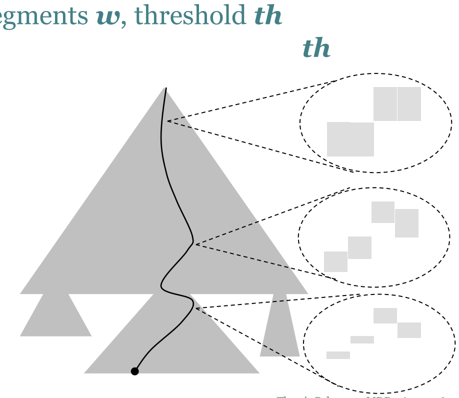





# *i*SAX Index

non-balanced tree-based index with non-overlapping regions, and controlled fan-out rate

base cardinality ***b*** (optional), segments ***w***, threshold ***th***



# *i*SAX Index

non-balanced tree-based index with non-overlapping regions, and controlled fan-out rate

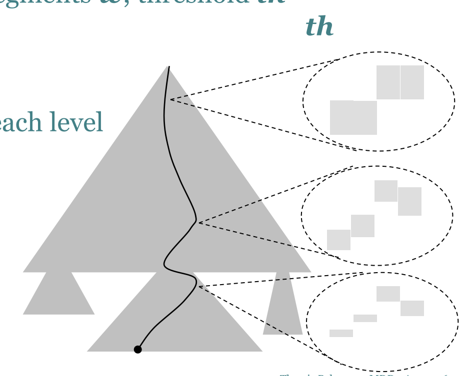
- base cardinality ***b*** (optional), segments ***w***, threshold ***th***

Approximate Search

- Match *i*SAX representation at each level

Exact Search

- Leverage approximate search
- Prune search space
  - Lower bounding distance



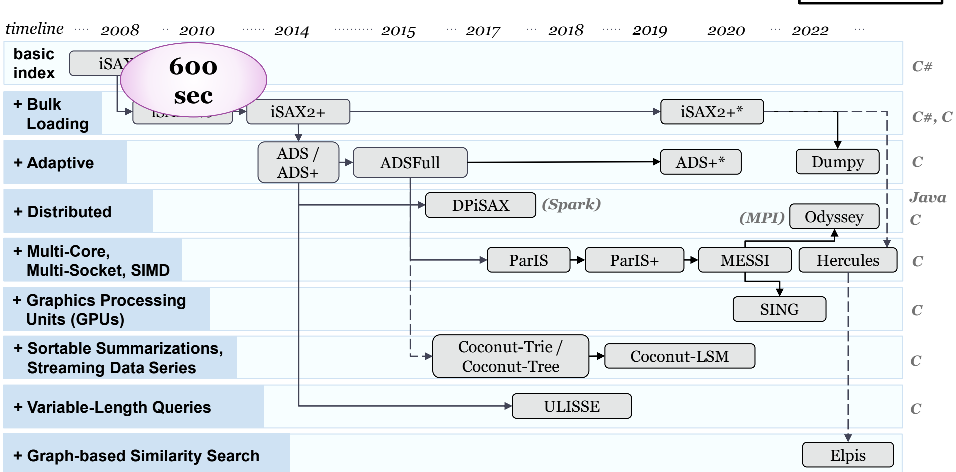





# iSAX Index Family Lineage Tree

Publications

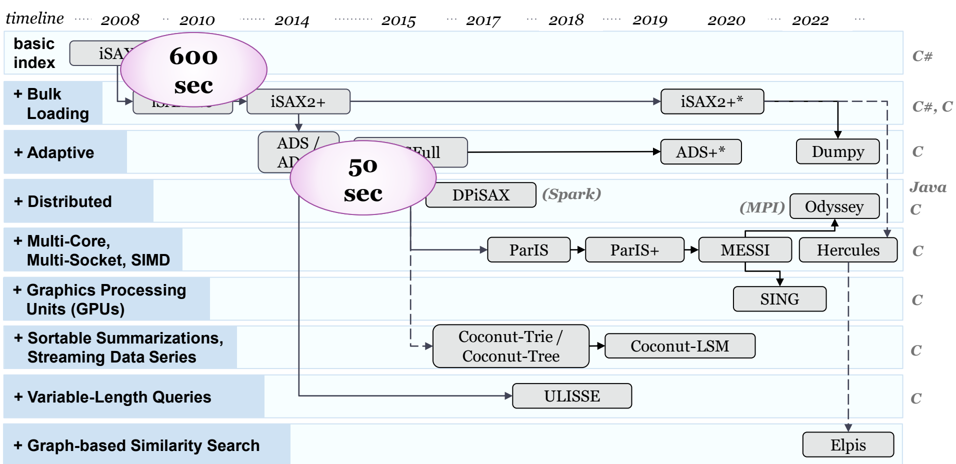
Palpanas-




execution time for **1 similarity search query on a 100GB dataset *on disk***

# iSAX Index Family Lineage Tree

Publications
Palpanas-



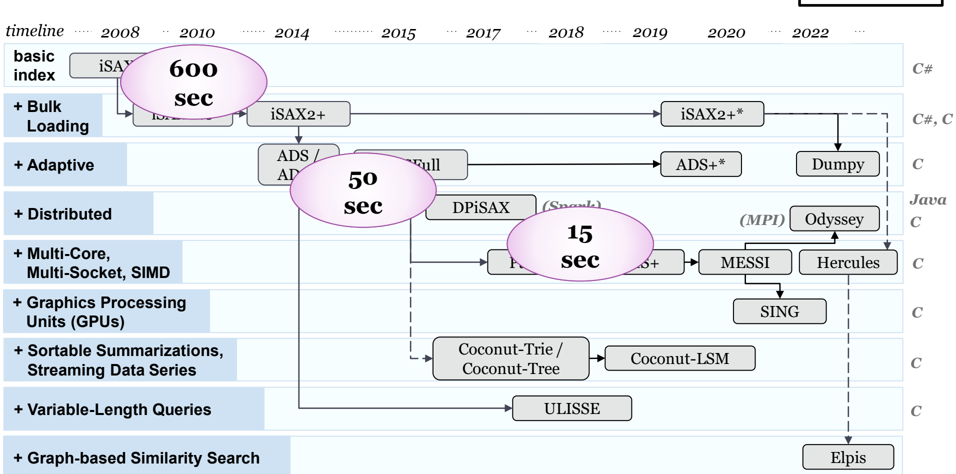

execution time for **1 similarity search query on a 100GB dataset _on disk_**

Themis Palpanas - MDD - Apr 2026



# iSAX Index Family Lineage Tree

Publications
Palpanas-



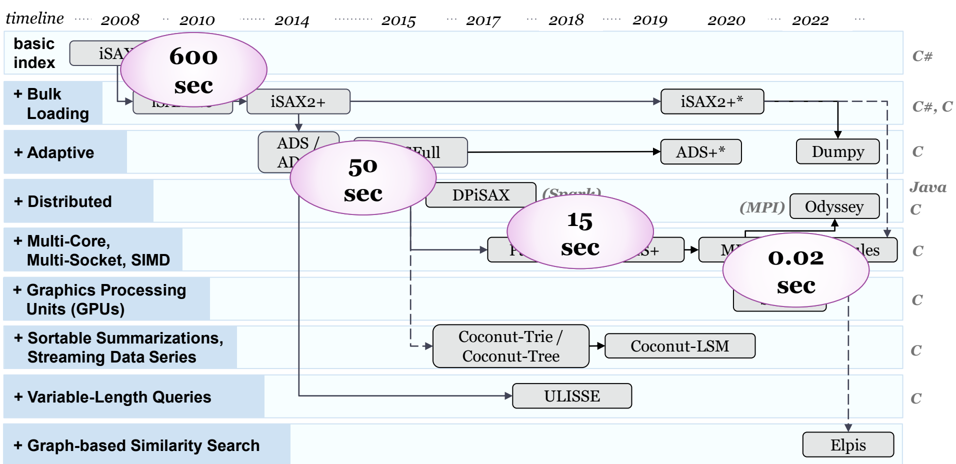

execution time for **1 similarity search query on a 100GB dataset _on disk_**

Themis Palpanas - MDD - Apr 2026

Publications

Palpanas-

# iSAX Index Family Lineage Tree




execution time for **1 similarity search query on a 100GB dataset *in memory***





Publications

Echihabi-

# Hercules
## Parallel Indexing of Sequences

Disk-based solution for SIMD, multi-core, multi-socket architectures

Exploits the benefits of two different summarization techniques (iSAX and EAPCA), and novel indexing and query answering algorithms

Leads to better query answering performance than all recent state-of-the-art approaches across all popular query workloads

only index that outperforms optimized scan on all scenarios (including hard query workloads on disk-based datasets)

Performs up to one order of magnitude faster than the best competitor (which is not always the same)

Publications

Echihabi-

# Hercules
## Parallel Indexing of Sequences

Disk-based solution for SIMD, multi-core, multi-socket architectures

Exploits the benefits of two different summarization techniques (iSAX and EAPCA), and novel indexing and query answering algorithms

- EAPCA (better adapts to data characteristics and) leads to tighter clusters of similar series
- iSAX (better approximates the original series and) enables further pruning





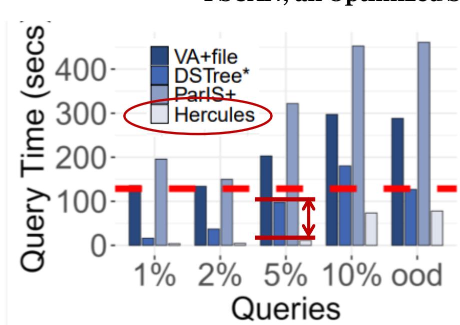
Publications

Echihabi-

# Hercules
## Parallel Indexing of Sequences

Disk-based solution for SIMD, multi-core, multi-socket architectures

**PSCAN, an Optimized Sequential Scan**




**up to 10x faster than best competitor**

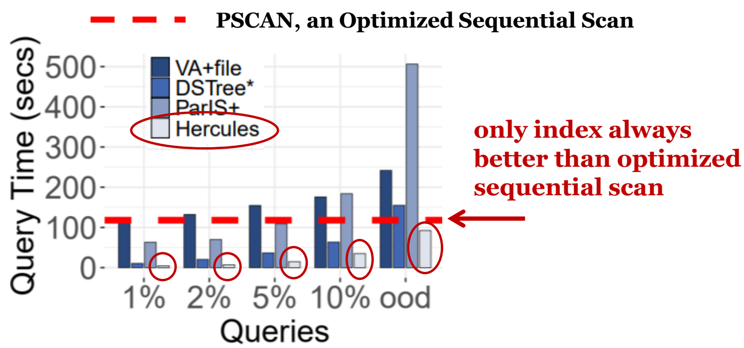
Query Performance with Increased Query Difficulty (Deep100GB)

Publications

Echihabi-

# Hercules
## Parallel, On-Disk Indexing of Sequences

Disk-based solution for SIMD, multi-core, multi-socket architectures




Query Performance with Increased Query Difficulty (Seismic100GB)





Publications

Peng-ICDE

Peng-VLDBJ

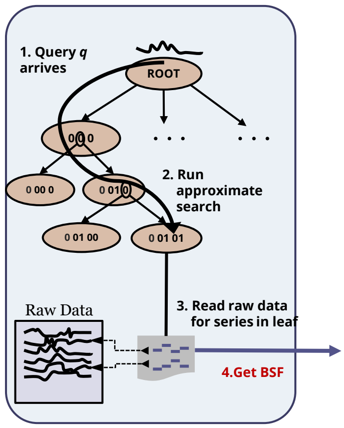
# MESSI
## In-Memory Data Series Index

in-memory solution for SIMD, multi-core, multi-socket architectures

- index-creation algorithm
  - balances workload of different workers, minimizes synchronization cost
- exact query answering algorithm
  - optimizes tree traversal and pruning
  - minimizes number of lower-bound and real distance calculations

- answers exact queries at interactive speeds: ~50msec on 100GB
  - up to 11x faster than competing approaches

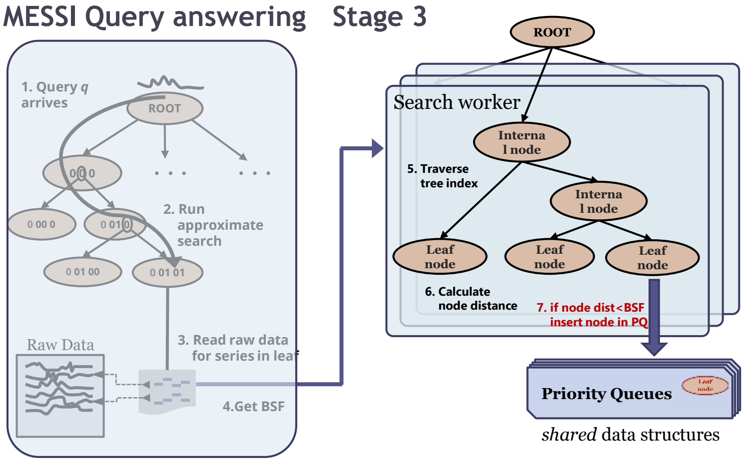
## MESSI Query answering Stage 3



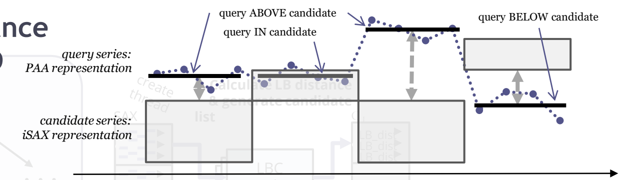





## MESSI Query answering Stage 3



## Lower-Bound Distance Calculation in SIMD



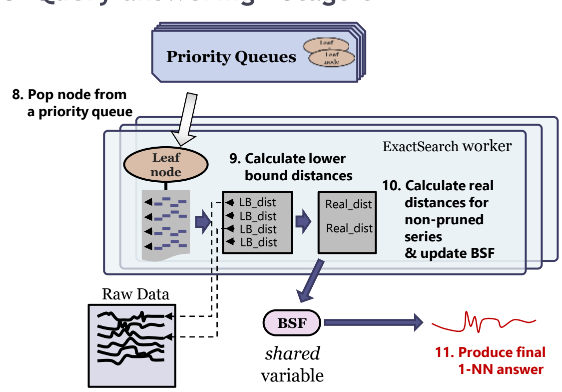

| | | | | |
|---|---|---|---|---|
| Result above branch | Dist_above[1] | Dist_above[2] | Dist_above[3] | Dist_above[4] |
| Mask above branch | TRUE | | TRUE | |
| Result below branch | Dist_below[1] | Dist_below[2] | Dist_below[3] | Dist_below[4] |
| Mask below branch | | | | TRUE |
| Result in branch | Dist_in[1] | Dist_in[2] | Dist_in[3] | Dist_in[4] |
| Mask in branch | | TRUE | | |
| Final Result | Dist_above[1] | Dist_in[2] | Dist_above[3] | Dist_below[4] |

SIMD register (8 points)





## MESSI Query answering Stage 3



Publications

Peng-ICDE

# SING
# Sequence Indexing Using GPUs

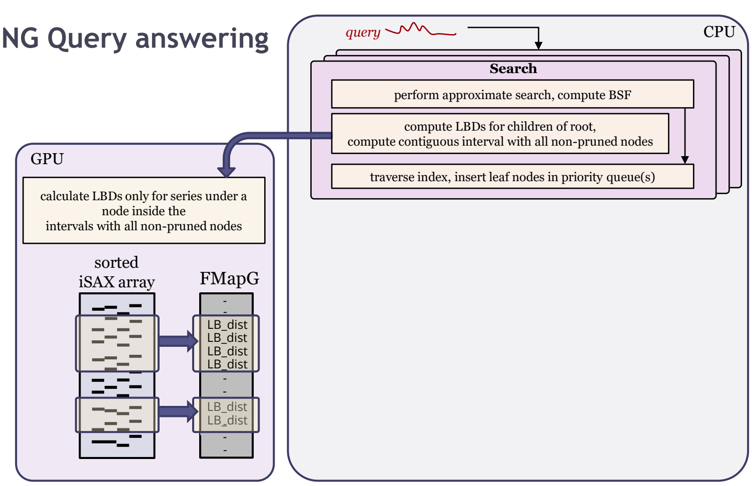
in-memory solution for SIMD, multi-core, multi-socket architectures with GPUs (Graphical Processing Units)

- new exact query answering algorithm
  - CPU-GPU co-processing framework
  - new GPU-friendly lower bound distance calculation algorithm
- answers exact queries at interactive speeds: ~20msec on 100GB dataset
  - up to 5x faster than competing approaches

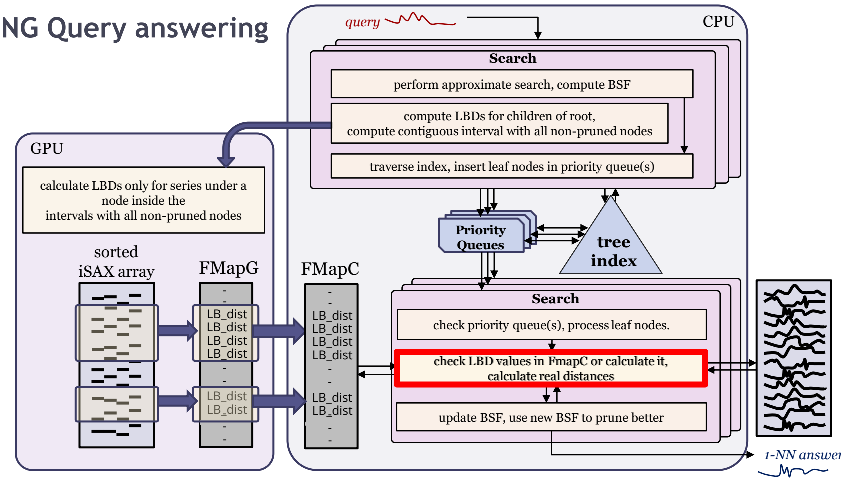




## SING Query answering



## SING Query answering








Publications

Chatzakis-

# Odyssey
## Distributed, Parallel, In-Memory Indexing

Distributed, in-memory solution for SIMD, multi-core, multi-socket architectures

allows in-memory processing (across machines) of very large datasets

Odyssey addresses the following challenges

**Query Scheduling**: Which queries to which nodes?

**Query Execution Time estimations**

**Flexible Replication Schemes**

**Dynamic Scheduling**

**Load Balancing**: Enable nodes to perform useful and equal work

**Density-aware Data Distribution**

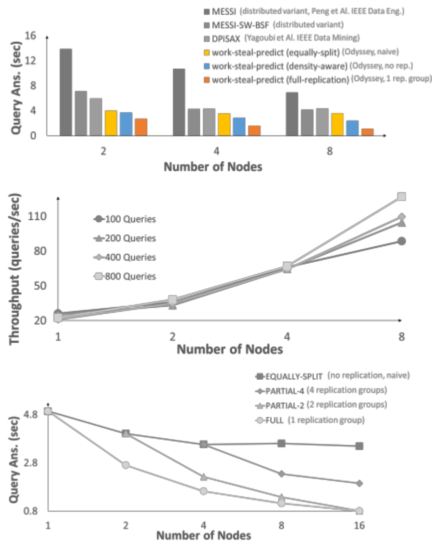
**Efficient work-stealing**

# Odyssey

achieves all goals

Publications

Chatzakis-





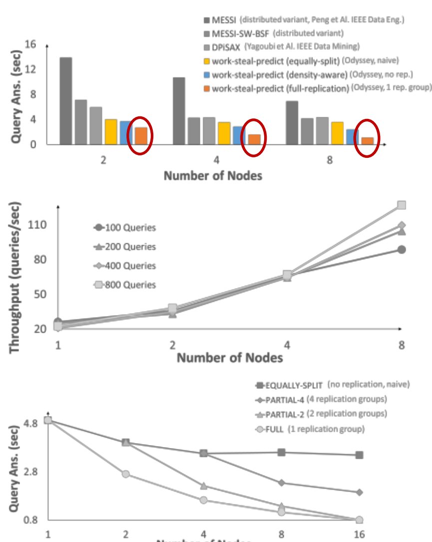



# Odyssey

achieves all goals

Publications

Chatzakis-



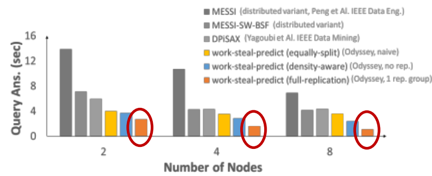

**up to 3x faster**
**than best competitor**

# Odyssey

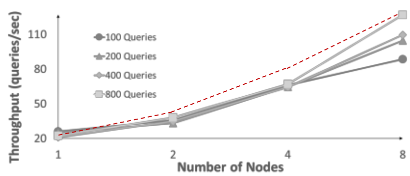
achieves all goals

Publications

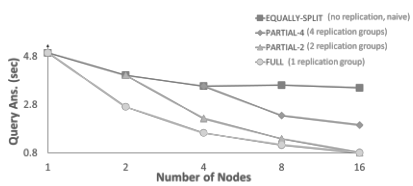
Chatzakis-




**up to** **3x** **faster**
**than best competitor**




**scalable query answering**
**(almost linear)**





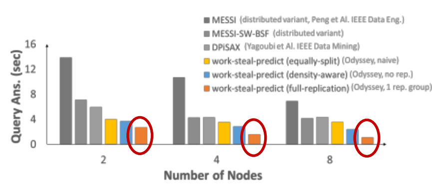



# Odyssey

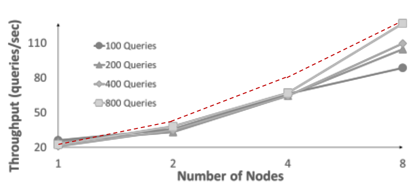
achieves all goals

Publications

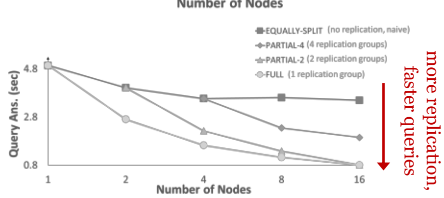
Chatzakis-




**up to** **3x** **faster**
**than best competitor**




**scalable query answering**
**(almost linear)**




**more replication leads to**
**faster query answering**

# Vector Similarity Search
# State-of-the-Art Methods

for more details:

*Ilias Azizi, Karima Echihabi, Themis Palpanas. Graph-Based Vector Search: An Experimental Evaluation of the State-of-the-Art. SIGMOD 2025*
https://helios2.mi.parisdescartes.fr/~themisp/publications/sigmod25-gass.pdf

*Manos Chatzakis, Francesca Del Gaudio,Sophia Sideri, Themis Palpanas. The Quest for Faster ANN Vector Search. EDBT 2026*
*https://helios2.mi.parisdescartes.fr/~themisp/publications/edbt26-FasterAnns-summary.pdf*
*https://helios2.mi.parisdescartes.fr/~themisp/publications/edbt26-FasterAnns-slides.pdf*





Publications

Echihabi et al - WIMS

# High-d Vectors
# Deep Embeddings

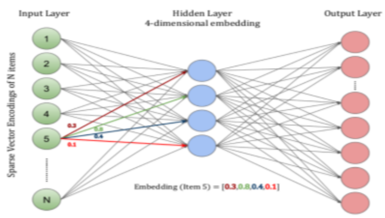
**sequences**
**text**
**images**
**video**
**graphs**
**molecules**
**…**

Publications

Echihabi et al - WIMS

# High-d Vectors
# Deep Embeddings

**sequences**
**text**
**images**
**video**
**graphs**
**molecules**
**…**





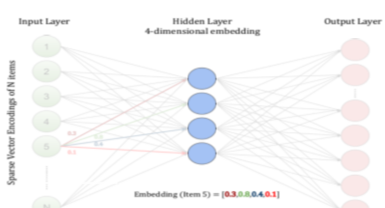



Publications

Echihabi et al - WIMS

# High-d Vectors
# Deep Embeddings

**sequences**
**text**
**images**
**video**
**graphs**
**molecules**
**…**



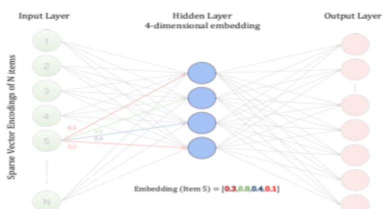

**deep embeddings**
high-d vectors learned using a DNN

Publications

Echihabi et al - WIMS

# High-d Vectors
# Deep Embeddings

**sequences**
**text**
**images**
**video**
**graphs**
**molecules**
**…**




**deep embeddings**
high-d vectors learned using a DNN

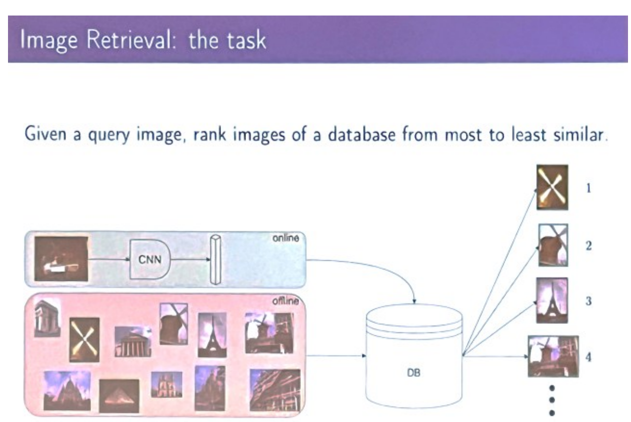
**image similarity**
**retrieval augmented generation**
**web search recommendations**
**molecule structure similarity**





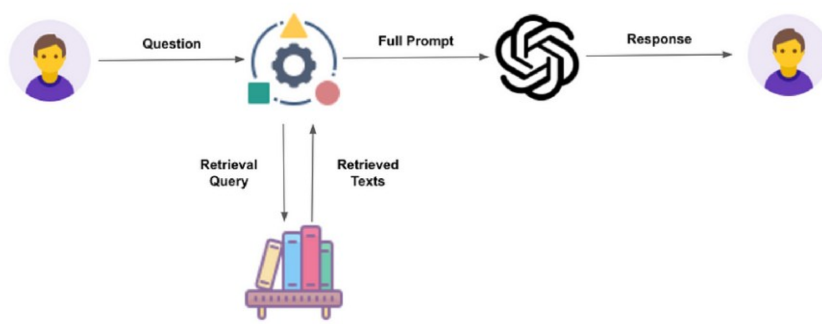
# Deep Embeddings
# Similarity Search Applications

image retrieval



# Deep Embeddings
# Similarity Search Applications

image retrieval
retrieval augmented generation (RAG)

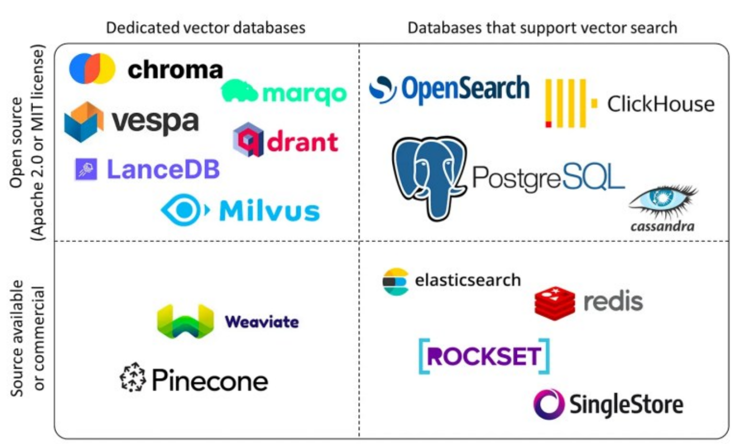



https://pvml.com/





# Vector Databases



# High-d Vector Similarity Search Methods

techniques for approximate similarity search in high-d vectors
- indexes:
  - inverted files (IVF)
  - k-NN graphs (HNSW)
- summarizations:
  - space quantization (RaBitQ)

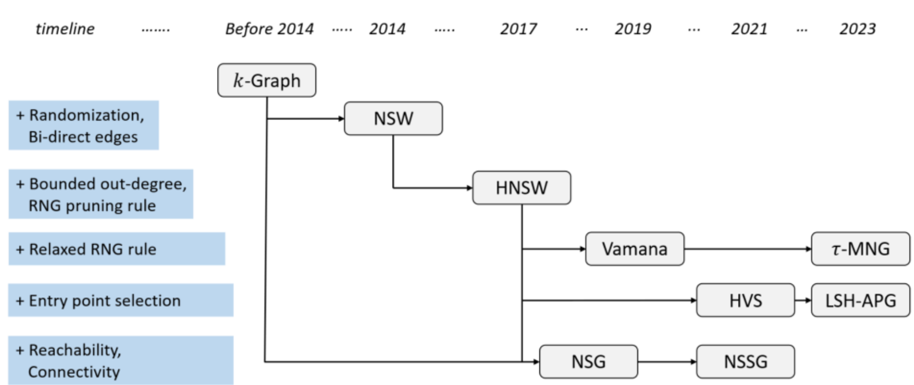




Publications

Wang et al - IEEE Bulletin

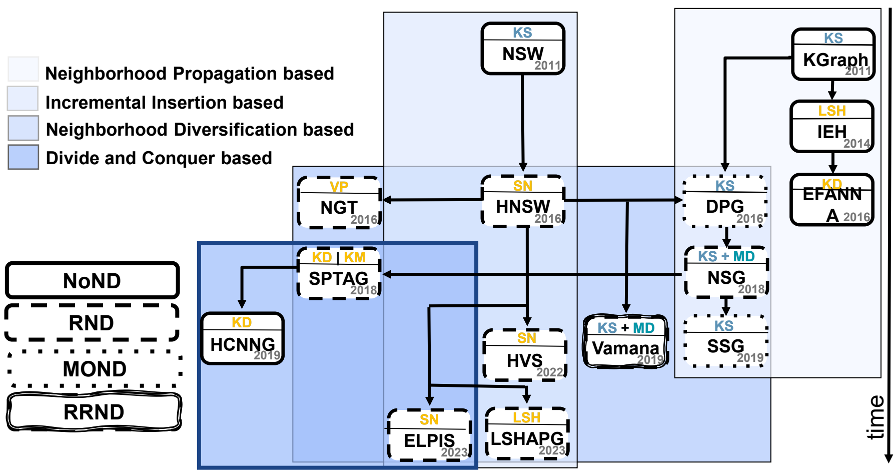
# Evolution of k-NN Graph Methods



Publications

Azizi et al - SIGMOD

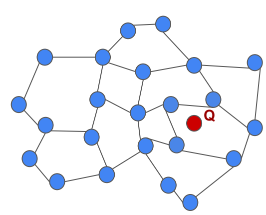
# Graph-based Vector Search: Taxonomy






# Proximity Graph

Graph in which two vertices are connected by an edge if and only if the vertices satisfy particular geometric requirements

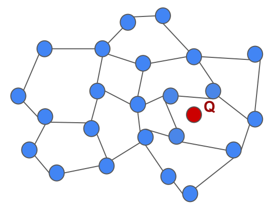


# Beam Search

Beam search is a heuristic search algorithm that explores a graph by expanding the most optimistic node in a limited set of size L

BeamSearch(G, Q, entry_node, K, L)

**Randomized Seed Selection**
K Random Sampling (KS)
**Index-based Seed Selection**
Stacked NSW (SN)
KD, KMean Balanced Trees
**Predefined Seed Selection**
Medoid (MD)

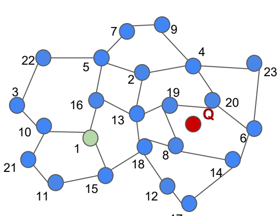


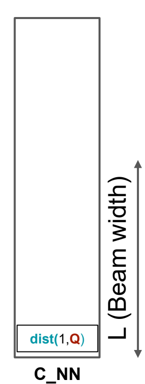



# Beam Search

BeamSearch(G, Q, entry_node = 1, K, L = 6)

1. Initialize the set of candidate nearest neighbors **C_NN** with entry_ node, and the set of visited nodes with empty set

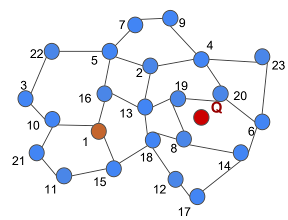


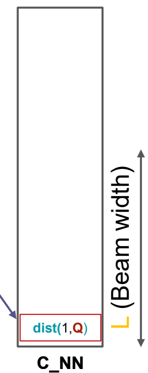
# Beam Search

BeamSearch(G, Q, entry_node = 1, K, L = 6)

② While(C_NN\ )




2.1 Select the closest element P from **C_NN**

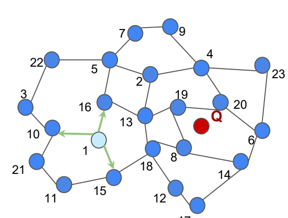





# Beam Search

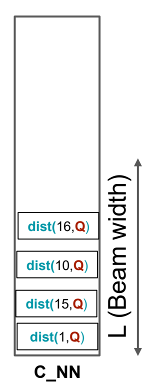
BeamSearch(G, Q, entry_node = 1, K, L = 6)

② While(C_NN\ )




2.1 Select the closest element P from **C_NN**

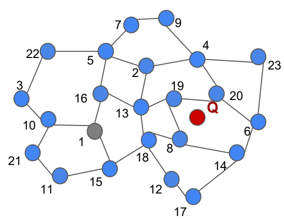
2.2 Add P neighbors to **C_NN**



# Beam Search

BeamSearch(G, Q, entry_node = 1, K, L = 6)

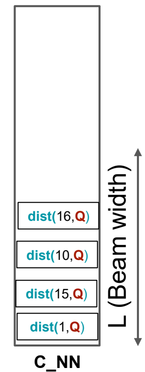
2 While(C_NN\ )




2.1 Select the closest element P from **C_NN**

2.2 Add P neighbors to **C_NN**

2.3 Add P to the set of visited candidate

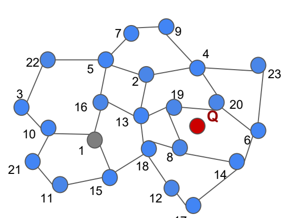





# Beam Search

BeamSearch(G, Q, entry_node = 1, K, L = 6)

2 While(C_NN\ )

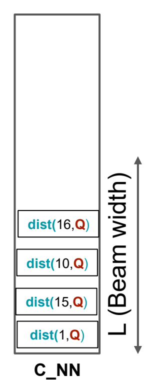



2.1 Select the closest element P from **C_NN**

2.2 Add P neighbors to **C_NN**

2.3 Add P to the set of visited candidate

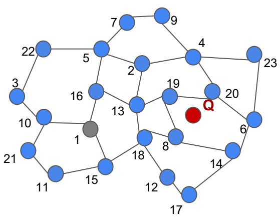
2.4 If |**C_NN**| > L: Update **C_NN** to retain closest L points to **Q**



# Beam Search

BeamSearch(G, Q, entry_node = 1, K, L = 6)

(2) While(**C_NN**     )

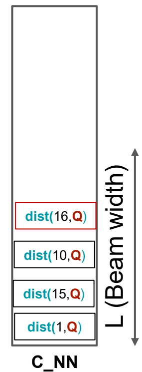



2.1 Select the closest element P from **C_NN**

2.2 Add P neighbors to **C_NN**

2.3 Add P to the set of visited candidate

2.4 If |**C_NN**| > L: Update **C_NN** to retain closest L points to **Q**

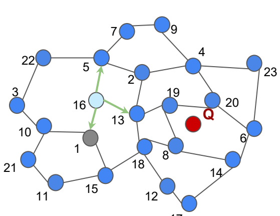





# Beam Search

BeamSearch(G, Q, entry_node = 1, K, L = 6)

(2) While(**C_NN**     )

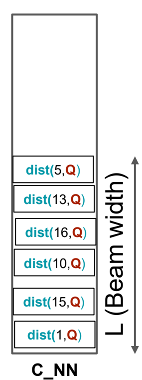



2.1 Select the closest element P from **C_NN**

2.2 Add P neighbors to **C_NN**

2.3 Add P to the set of visited candidate

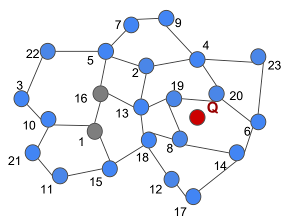
2.4 If |**C_NN**| > L: Update **C_NN** to retain closest L points to **Q**



# Beam Search

BeamSearch(G, Q, entry_node = 1, K, L = 6)

2 While(**C_NN** )

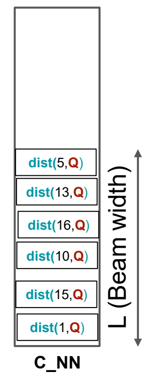



2.1 Select the closest element P from **C_NN**

2.2 Add P neighbors to **C_NN**

2.3 Add P to the set of visited candidate

2.4 If |**C_NN**| > L: Update **C_NN** to retain closest L points to **Q**

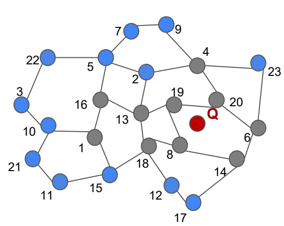




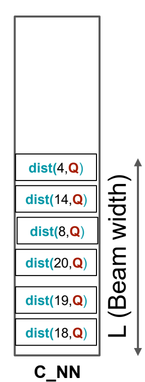

# Beam Search

BeamSearch(G, Q, entry_node = 1, K, L = 6)

2 While(**C_NN** )



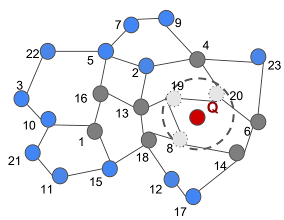

**C_NN** ==

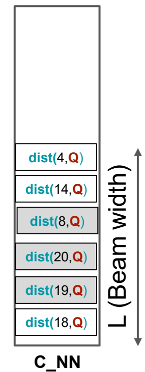


# Beam Search

BeamSearch(G, Q, entry_node = 1, K, L = 6)

③ Return K closest neighbors from **C_NN**





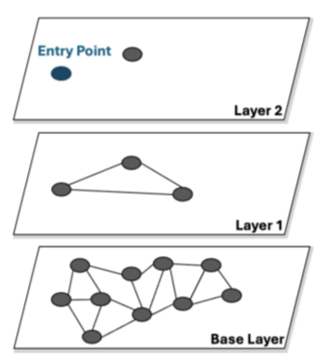





*Malkov, Yu A., Dmitry A. Yashunin. Efficient and robust ANNS using hierarchical NSW graphs*

# HNSW Index

Multi-layered graph over vector collection
- Nodes represent vectors, edges link similar nodes
- Base layer contains all vectors
- M, efConstruction: graph structure

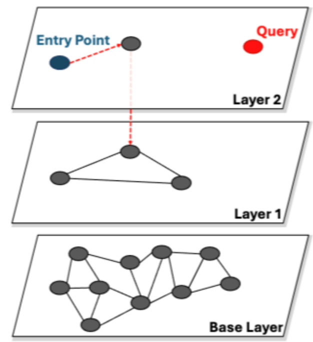


*Malkov, Yu A., Dmitry A. Yashunin. Efficient and robust ANNS using hierarchical NSW graphs*

# HNSW Index

Multi-layered graph over vector collection
- Nodes represent vectors, edges link similar nodes
- Base layer contains all vectors
- M, efConstruction: graph structure

**ANNS(q, efSearch, k):**
- Traverse the graph until the base layer





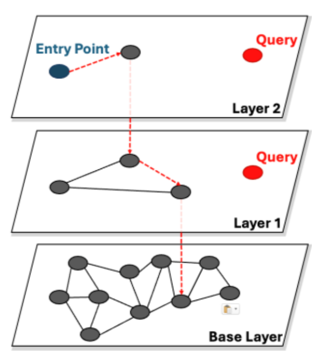



*Malkov, Yu A., Dmitry A. Yashunin. Efficient and robust ANNS using hierarchical NSW graphs*

# HNSW Index

Multi-layered graph over vector collection
- Nodes represent vectors, edges link similar nodes
- Base layer contains all vectors
- M, efConstruction: graph structure

**ANNS(q, efSearch, k):**
- Traverse the graph until the base layer

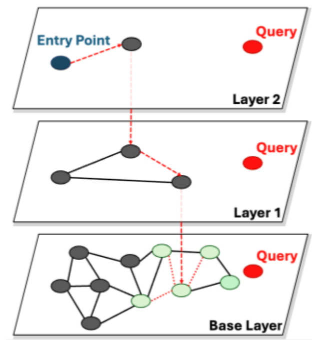


Malkov, Yu A., Dmitry A. Yashunin. Efficient and robust ANNS using hierarchical NSW graphs

# HNSW Index

Multi-layered graph over vector collection

- Nodes represent vectors, edges link similar nodes
- Base layer contains all vectors
- M, efConstruction: graph structure




**ANNS(q, efSearch, k):**

- Traverse the graph until the base layer
- Greedy search at the base layer





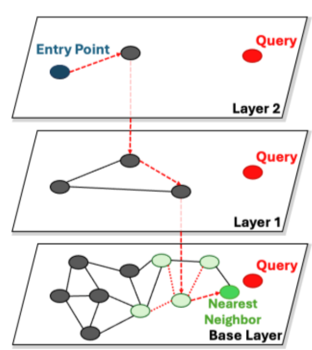
Publications

Wang et al

# HNSW Index

Multi-layered graph over vector collection

- Nodes represent vectors, edges link similar nodes
- Base layer contains all vectors
- M, efConstruction: graph structure




**ANNS(q, efSearch, k):**

- Traverse the graph until the base layer
- Greedy search at the base layer

**recent work has shown that hierarchy does not help**

Publications

# High-d Vectors Indexes

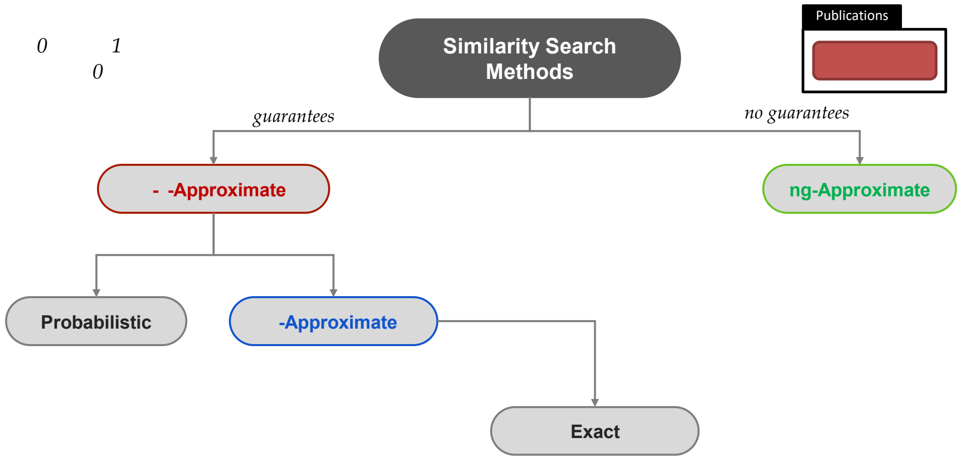
techniques for approximate similarity search in high-d vectors
- [LSH (SRS)]
- space quantization and inverted files (IVF)
- k-NN graphs (HNSW)

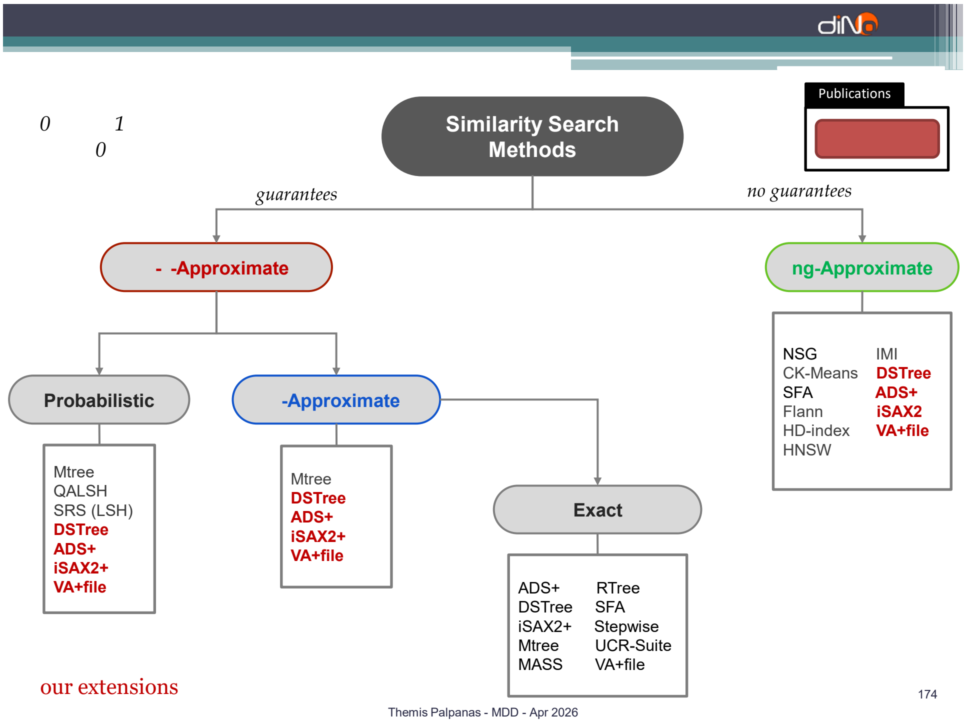
how do these high-d vector techniques compare to data series techniques?
- have conducted extensive experimental comparison



Publications

*0* *1*
*0*








Publications

# Data Series vs. high-d Vectors

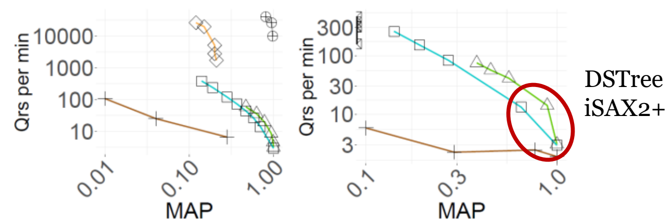
**data series techniques** are the **overall winners**, even on **general high-d vector** data

# Data Series vs. high-d Vectors

Publications

**data series techniques** are the **overall winners**, even on **general high-d vector** data

perform the best for approximate queries with probabilistic guarantees ( - -approximate search), in-memory and on-disk




(s) Deep25GB(ng) (t) Deep25GB($\delta\epsilon$)

DSTree HNSW IMI iSAX2+ SRS VA+file



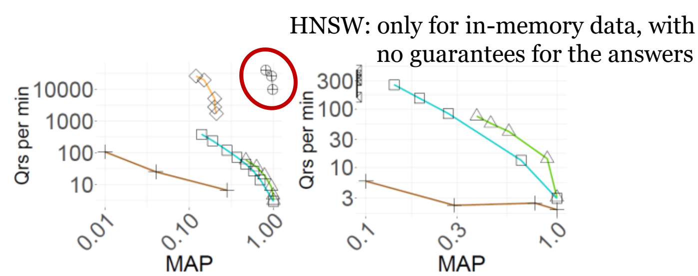


# Data Series vs. high-d Vectors

Publications

**data series techniques** are the **overall winners**, even on **general high-d vector** data

perform the best for approximate queries with probabilistic guarantees ( - -approximate search), in-memory and on-disk




(s) Deep25GB(ng) (t) Deep25GB($\delta\epsilon$)

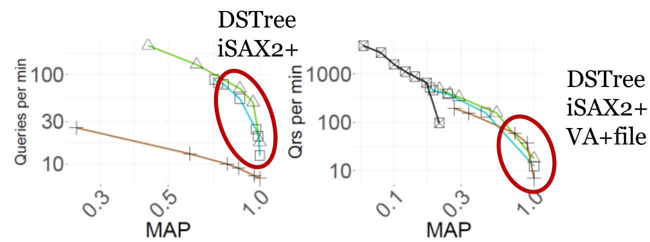
DSTree HNSW IMI iSAX2+ SRS VA+file

# Data Series vs. high-d Vectors

Publications

**data series techniques** are the **overall winners**, even on **general high-d vector** data

- perform the best for approximate queries with probabilistic guarantees ( - -approximate search), in-memory and on-disk
- perform the best for long vectors, in-memory and on-disk




**(g) Rand25GB 16384 (ng)** **(h) Rand25GB 16384 ($\delta\epsilon$)**

DSTree HNSW IMI iSAX2+ SRS VA+file



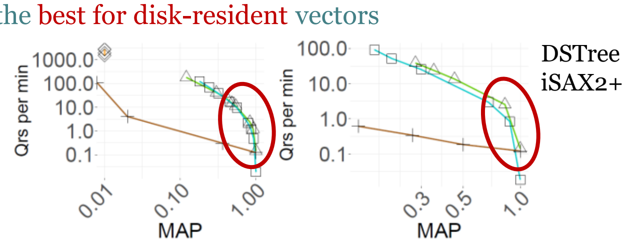


# Data Series vs. high-d Vectors

Publications

**data series techniques** are the **overall winners**, even on **general high-d vector** data

- perform the best for approximate queries with probabilistic guarantees ( - -approximate search), in-memory and on-disk
- perform the best for long vectors, in-memory and on-disk
- perform the best for disk-resident vectors




**(m) Deep250GB(ng)** **(n) Deep250GB($\delta\epsilon$)**

DSTree HNSW IMI iSAX2+ SRS VA+file

# Hybrid (DSTree + HNSW): ELPIS

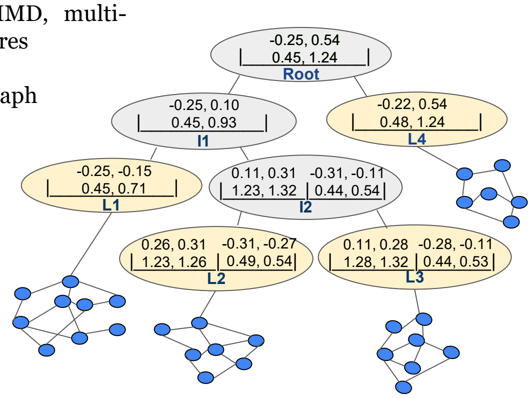
Publications

Azizi-

## Parallel, In-Memory Indexing of Sequences

In-memory solution for SIMD, multi-core, multi-socket architectures

**ELPIS** combines tree and graph structures for efficient in-memory ng-approximate vector similarity search.







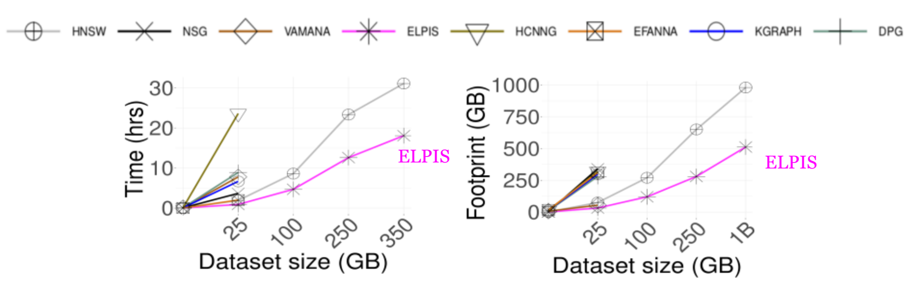

# Hybrid (DSTree + HNSW): ELPIS

Publications

Azizi-

## Parallel, In-Memory Indexing of Sequences

Scalability of indexing time and memory footprint with dataset size (Deep)




ELPIS builds the index up to **8x faster,**
using **40% less memory**

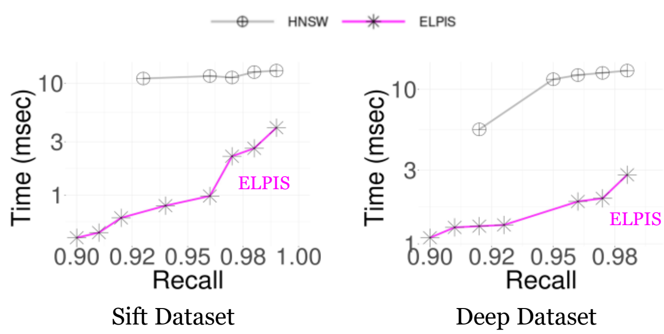
# Hybrid (DSTree + HNSW): ELPIS

Publications

Azizi-

## Parallel, In-Memory Indexing of Sequences

Query Performance on 1B vectors datasets (Sift, Deep)




ELPIS answers **10-NN queries** in **~3 msec** for a dataset of **1 billion vectors** with **recall 0.99**



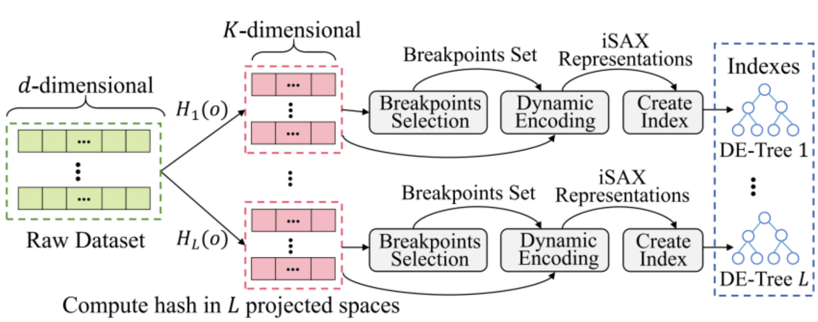


# Hybrid (iSAX + LSH): DET-LSH

Publications

Wei et al. -

**DET-LSH** combines tree and LSH for efficient indexing and approximate search with probabilistic guarantees

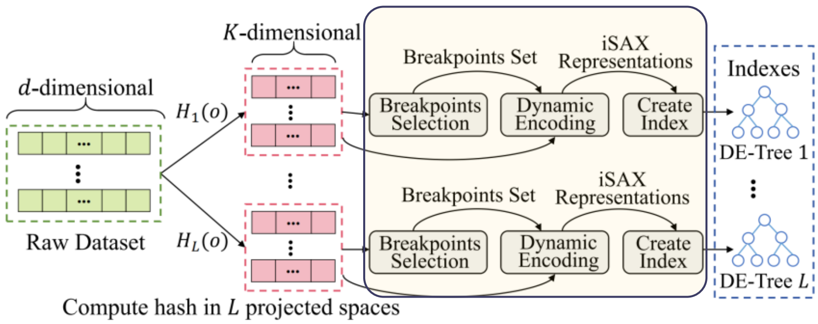


# Hybrid (iSAX + LSH): DET-LSH

Publications

Wei et al. -

**DET-LSH** combines tree and LSH for efficient indexing and approximate search with probabilistic guarantees





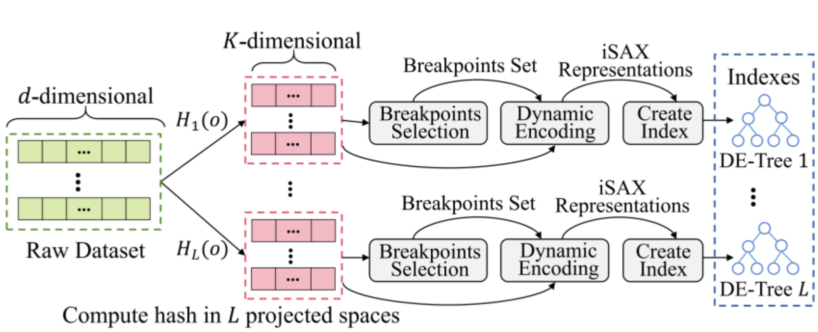



# Hybrid (iSAX + LSH): DET-LSH

Publications

Wei et al. -

**DET-LSH** combines tree and LSH for efficient indexing and approximate search with probabilistic guarantees




up to 6x faster indexing and 2x faster query answering (than standard LSH methods)

# Dicussion and Future Work





## Observations

- high-d vectors is a very common data type
  - across several different domains and applications
- complex high-d vector analytics are challenging
  - have very high inherent complexity
- data series management/indexing techniques provide state-of-the-art performance
  - work for data series and general high-d vectors (and embeddings)
  - lead to fast complex analytics and machine learning

# Observations

high-d vectors is a very common data type
- across several different domains and applications

complex high-d vector analytics are challenging
- have very high inherent complexity

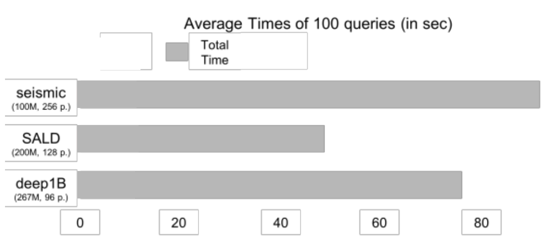
data series management/indexing techniques provide state-of-the-art performance
- work for data series and general high-d vectors (and embeddings)
- lead to fast complex analytics and machine learning

several exciting research opportunities
- hybrid solutions
- progressive analytics
- learned (data-adaptive) summarizations/data structures

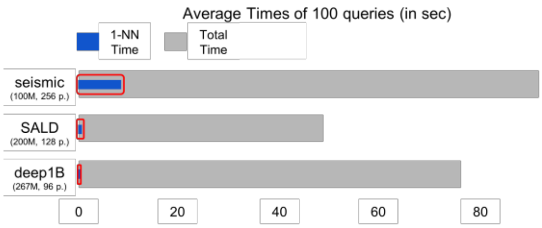




# Further Advances



# Further Advances




how do we further reduce the wasted (gray) effort?





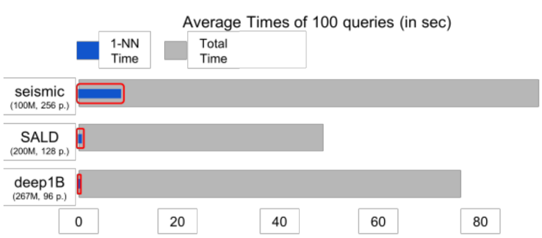
Further Advances

Publications

Gogolou -

Gogolou -

Gogolou -




how do we further reduce the wasted (gray) effort?

**progressive query answering**

produce intermediate answers with (probabilistic) quality guarantees

# Further Advances: Progressive Query Answering

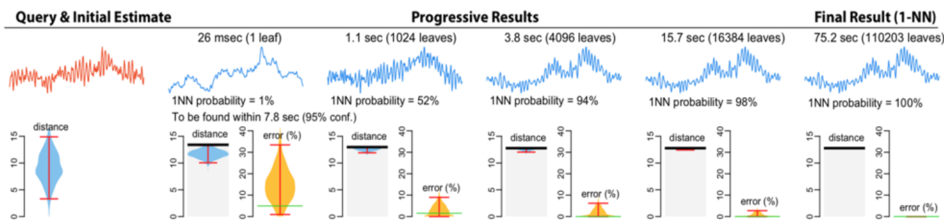
Publications

Gogolou -

Gogolou -

Gogolou -

interaction with users offers new opportunities
- progressive answers
  - produce intermediate results
    - iteratively converge to final, correct solution
  - provide bounds on the errors (of the intermediate results) along the way








# Further Advances: Early Termination (for ANNS)

Publications

Chatzakis et al. -

similar ideas can speedup ng-approximate search (Approximate Nearest Neighbor Search (ANNS) with no guarantees)
- stop query answering when target recall is reached
  - works for both k-NN graphs (eg, HNSW) and inverted files (eg, IVF)

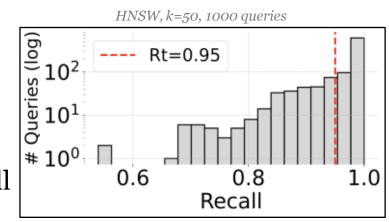
Publications
Chatzakis et al.
-

# Problems with Current Solutions

**Mapping** between hyperparameters and the achieved recall
- Using a **tuning** query workload

Complex multidimensional hyperparameter space

Parameters predefined, **average** recall
- Cannot **adapt** to individual query hardness
- Suboptimal results for specific queries




**ANNS static parameters cannot adapt to each individual query!**

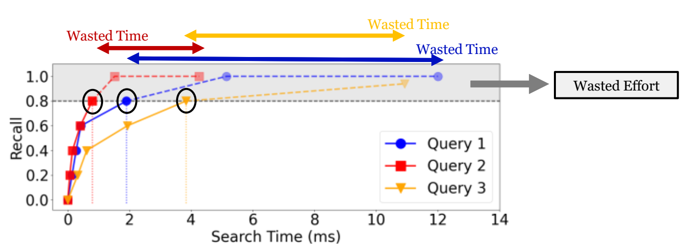




Publications
Chatzakis et al.
-

# Key Observation

Query reaching very high recall satisfies all lower recalls as well
- Early termination opportunities
- Creating high-scoring index is easy




**Different queries reach different recalls at different times!**

# Further Advances: DARTH Early Termination

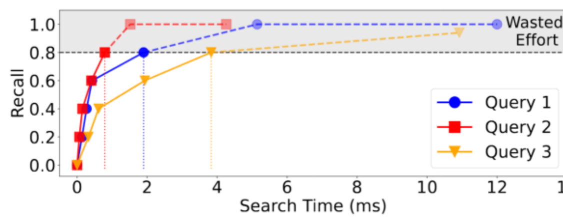
Publications

Chatzakis et al.

-

similar ideas can speedup ng-approximate search (Approximate Nearest Neighbor Search (ANNS) with no guarantees)

- stop query answering when target recall is reached
  - works for both k-NN graphs (eg, HNSW) and inverted files (eg, IVF)




**DARTH**: up to 15x faster than plain HNSW, and 42x faster than plain IVF



# The Maximum Guarantee

recall 100% with probability = 1

-TR-12 2026

## The Maximum Guarantee with the Surprisingly Low Cost

recall 100% with probability = 1

a few dozens of ms for 100M vectors

**DaiSy** library



-TR-12 2026

## The Maximum Guarantee with the Surprisingly Low Cost

recall 100% with probability = 1

a few dozens of ms for 100M vectors

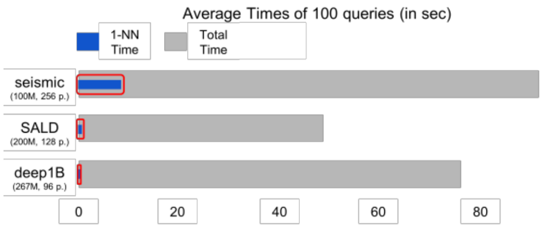
no more hunting for the missing vector

find ground-truth answers for ANNS

**DaiSy** library

# Further Advances

Publications
Gogolou -
Gogolou -
Gogolou -
Wang - KDD
Wang - SIGMOD




how do we further reduce the wasted (gray) effort?
- **progressive query answering**
  - produce intermediate answers with (probabilistic) quality guarantees
- **learned summarizations + index structures**
  - adapt to data characteristics
  - build more efficient indexes
  - perform more effective pruning

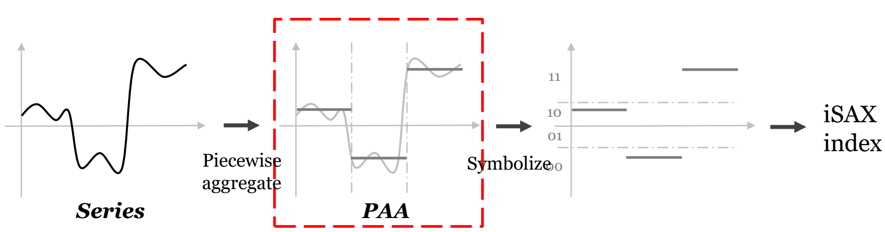




# Further Advances: Learning

Publications
Wang - KDD

Series Approximation Network (SEAnet)
- novel autoencoder architecture
- learns deep embedding approximations
- uses those for similarity search



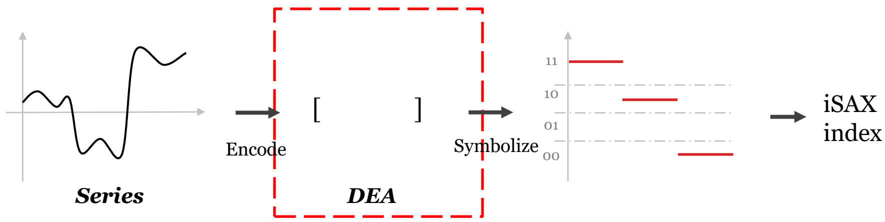
# Further Advances: Learning

Publications

Wang - KDD

Series Approximation Network (SEAnet)

- novel autoencoder architecture
- learns deep embedding approximations
- uses those for similarity search







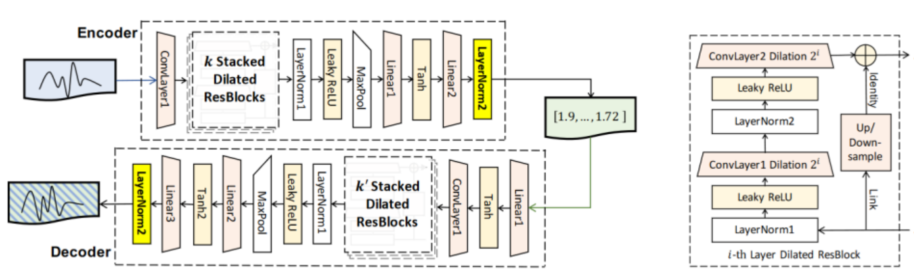

# Further Advances: Learning

Publications

Wang - KDD

Series Approximation Network (SEAnet)

- is an exponentially dilated ResNet architecture + Sum of Squares regularization
- minimizes
  - reconstruction error
  - difference between distance of two vectors in embedded space and distance in original space




(a) SEAnet Architecture

(b) Dilated ResBlock in SEAnet

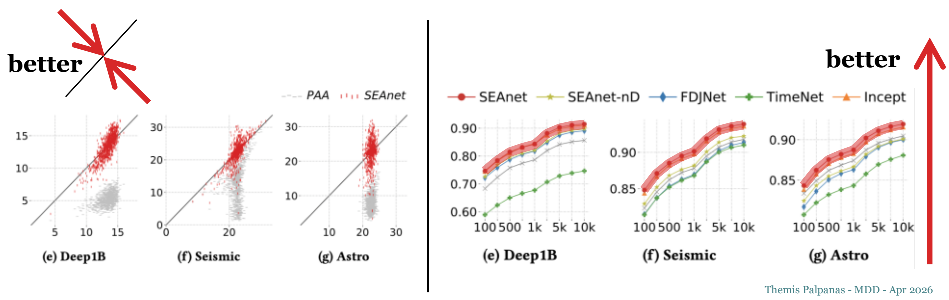
# Further Advances: Learning

Publications

Wang - KDD

Series Approximation Network (SEAnet)

- is an exponentially dilated ResNet architecture + Sum of Squares regularization
- minimizes
  - reconstruction error
  - difference between distance of two vectors in embedded space and distance in original space







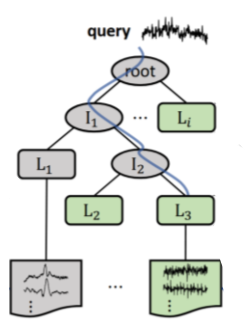

# Further Advances: Learning

Publications

Wang - SIGMOD

Learned Filters (LeaFi)

- machine learning models that make pruning decisions
- applied when pruning based on lower bounding is not possible



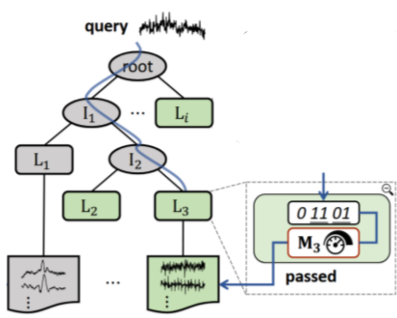
# Further Advances: Learning

Publications

Wang - SIGMOD

Learned Filters (LeaFi)

- machine learning models that make pruning decisions
- applied when pruning based on lower bounding is not possible




learned filter:
MLP with 1 hidden layer





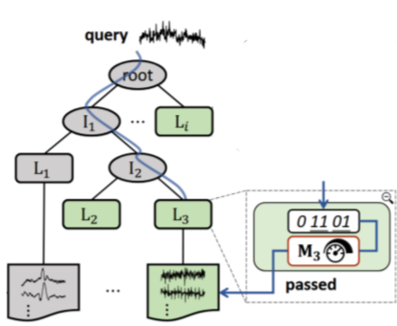
# Further Advances: Learning

Publications

Wang - SIGMOD

Learned Filters (LeaFi)

- machine learning models that make pruning decisions
- applied when pruning based on lower bounding is not possible




learned filter:
MLP with 1 hidden layer

up to 20x more pruning
up to 32x faster query answering

what are the theoretical properties of existing solutions?
- best/expected/worst time performance
- best/expected/worst accuracy (for approximate query answering)





what are the theoretical properties of existing solutions?
- best/expected/worst time performance
- best/expected/worst accuracy (for approximate query answering)

can we predict performance based on data characteristics?
- analytical results (eg, based on different distributions)
- learned

Publications

Wang et al. -

what are the theoretical properties of existing solutions?
- best/expected/worst time performance
- best/expected/worst accuracy (for approximate query answering)

can we predict performance based on data characteristics?
- analytical results (eg, based on different distributions)
- learned
  - Steiner-Hardness: query hardness measure for graph-based ANN indexes





Publications

Wang et al. -

what are the theoretical properties of existing solutions?
- best/expected/worst time performance
- best/expected/worst accuracy (for approximate query answering)

can we predict performance based on data characteristics?
- analytical results (eg, based on different distributions)
- learned
  - Steiner-Hardness: query hardness measure for graph-based ANN indexes

how do we introduce parallelism across the stack?
- for intra-query and inter-query execution
- for improving latency and throughout
- across similarity search methods/algorithms

how do we design hardware for vector similarity search?

- Processing-In-Memory (PIM)
- In-Storage-Processing (ISP)





how do we design hardware for vector similarity search?

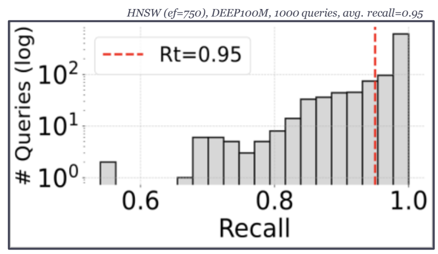
- Processing-In-Memory (PIM)
- In-Storage-Processing (ISP)

what are the right benchmarks to evaluate high-d vector indices?

- evaluation measures (currently: time and recall)
- data and query workloads (currently: most queries are easy)

# Recall is Lying

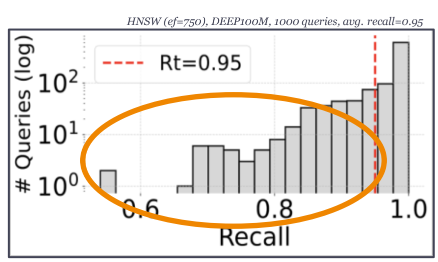
recall distribution for DEEP100M query workload
recall target (Rt) = 95%





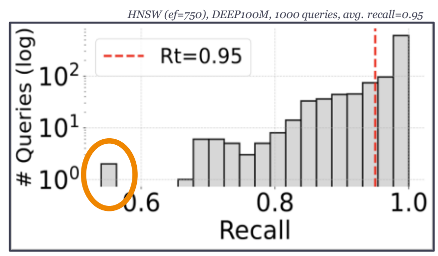

Recall is Lying

recall distribution for DEEP100M query workload
recall target (Rt) = 95%
but
30% of queries below recall target



# Recall is Lying Big Time

recall distribution for DEEP100M query workload
recall target (Rt) = 95%
but
- 30% of queries below recall target
- some queries with recall 50%

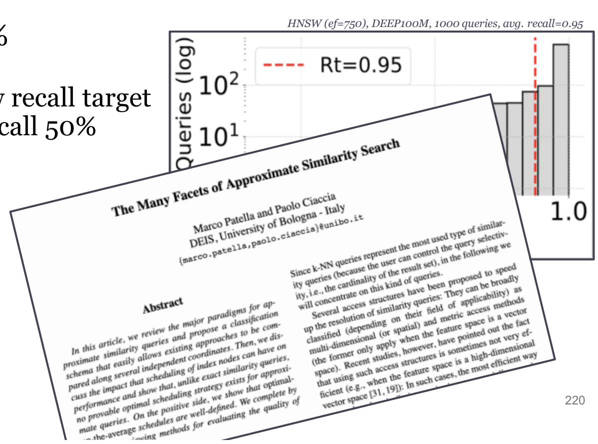





# Recall is Lying Big Time Unless We Stop It

recall distribution for DEEP100M query workload
recall target (Rt) = 95%
but
- 30% of queries below recall target
- some queries with recall 50%

need to also consider
- Recall distributions
- P95,99 percentiles
- Worst errors
- Queries under target



# Recall is Lying Big Time Unless We Stop It

recall distribution for DEEP100M query workload

recall target (Rt) = 95%

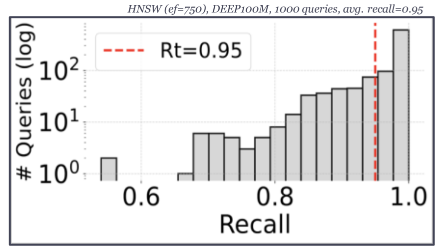
but

- 30% of queries below recall target
- some queries with recall 50%

need to also consider

- Recall distributions
- P95,99 percentiles
- Worst errors
- Queries under target
- **Quality guarantees**

*HNSW (ef=750), DEEP100M, 1000 queries, avg. recall=0.95*








how do we design hardware for vector similarity search?

- Processing-In-Memory (PIM)
- In-Storage-Processing (ISP)

what are the right benchmarks to evaluate high-d vector indices?

- evaluation measures (currently: time and recall)
  - Ratio of Queries Under Target Recall (RQUT), Normalized Rank Sum (NRS), Relative Distance Error (RDE)
- data and query workloads (currently: most queries are easy)

Publications

Ceccarello et al. - KDD

how do we design hardware for vector similarity search?

- Processing-In-Memory (PIM)
- In-Storage-Processing (ISP)

what are the right benchmarks to evaluate high-d vector indices?

- evaluation measures (currently: time and recall)
  - Ratio of Queries Under Target Recall (RQUT), Normalized Rank Sum (NRS), Relative Distance Error (RDE)
- data and query workloads (currently: most queries are easy)
  - Hephaestus: query workload generation for vector similarity search





Publications

Ceccarello et al. - KDD

how do we design hardware for vector similarity search?

- Processing-In-Memory (PIM)
- In-Storage-Processing (ISP)

what are the right benchmarks to evaluate high-d vector indices?

- evaluation measures (currently: time and recall)
  - Ratio of Queries Under Target Recall (RQUT), Normalized Rank Sum (NRS), Relative Distance Error (RDE)
- data and query workloads (currently: most queries are easy)
  - Hephaestus: query workload generation for vector similarity search

how do we integrate vector similarity search in databases?

- combine similarity search with other predicates
- optimization

# Going forward

high-d vector similarity search relevant to many communities

- data management
- time series
- information retrieval / text search
- machine learning / deep learning
- parallel and distributed computing
- computer architecture





# Going forward

research on this problem fragmented across communities

- open communication channels among these communities
- initiate discussions

# Going forward

- research on this problem fragmented across communities
  - open communication channels among these communities
  - initiate discussions
  - IEEE Bulletin on Data Engineering, Special Issues on High-Dimensional Vector Similarity Search
    - September 2023: From Time Series to Vector Databases
    - September 2024: Role of Machine Learning and Future Perspectives
    - http://sites.computer.org/debull/bull_issues.html






# Going forward

- research on this problem fragmented across communities
  - open communication channels among these communities
  - initiate discussions
  - IEEE Bulletin on Data Engineering, Special Issues on High-Dimensional Vector Similarity Search
    - September 2023: From Time Series to Vector Databases
    - September 2024: Role of Machine Learning and Future Perspectives
    - http://sites.computer.org/debull/bull_issues.html


September 2023

September 2024

thank you!

google **Themis Palpanas**