# Similarity Search on Time Series Data

Presented by Zhe Wang

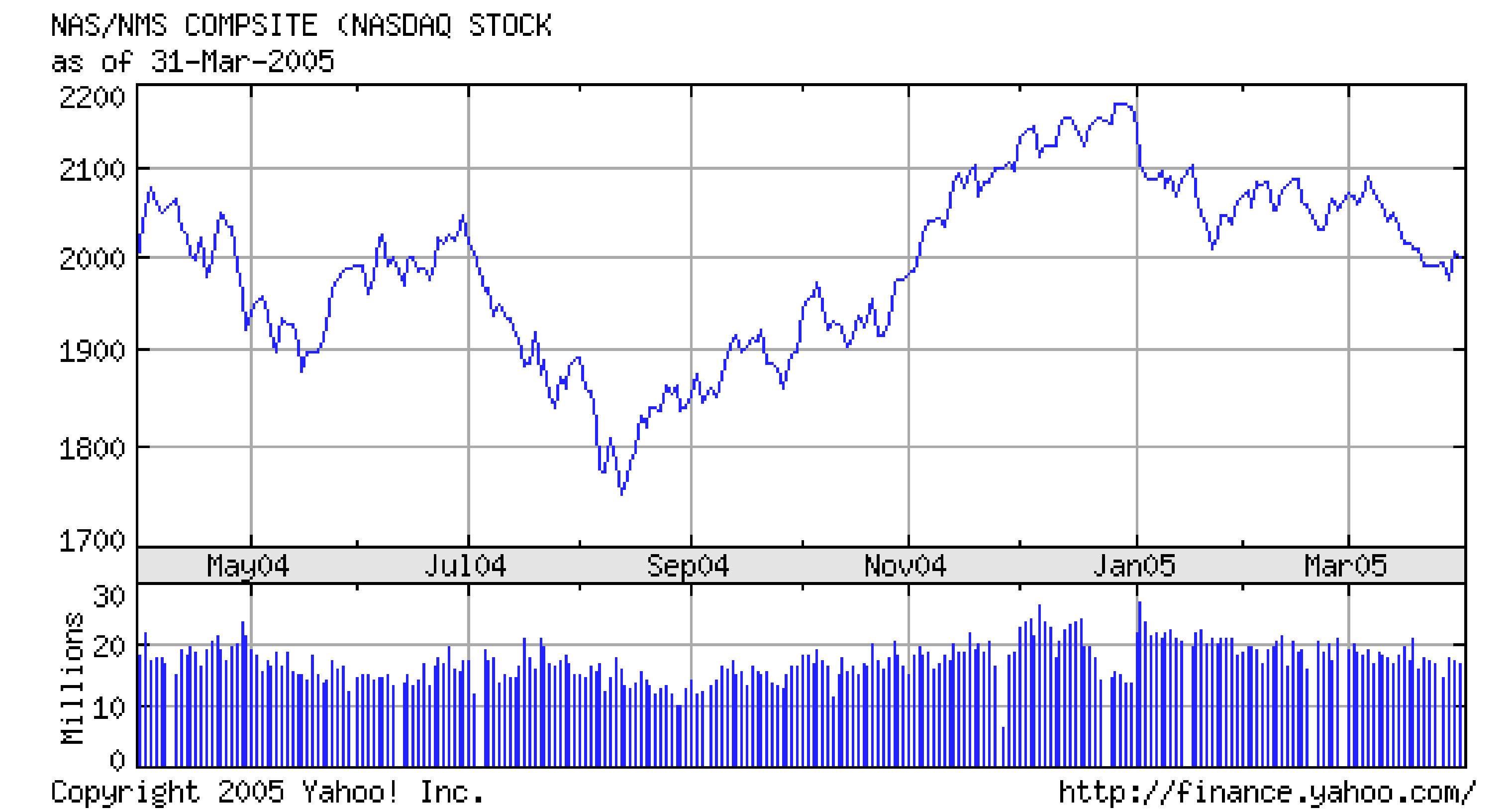
# Motivations

- Fast searching for time-series of real numbers. (“data mining”)
  - Scientific database: weather, geological, astrophysics, etc.

    “find past days in which solar wind showed similar pattern to today’s”
  - Financial, marketing time series:

    “Find past sales patterns that resemble last month”

# Real motivation?



# Difficulties for time-series data

- Can’t use exact match like fast string match:
  - Need to use distance function to compare two time series (next slide)
- Can’t easily index the time-series data directly.
  - Need faster algorithm than linear scan (whole talk)

# Distance functions

- L-p distance function

  $D(x,y) = ( \sum|x_i - y_i|^p )^{1/p}$

- L-2 distance function (Most popular)

  $D(x,y) = ( \sum(x_i - y_i)^2 )^{1/2}$

- Finding similar signals to query signal q means finding all x such that:

  $D(q,x) = ( \sum(q_i - x_i)^2 )^{1/2} <= \varepsilon$

# Why prefer L2 distance

- Important feature:

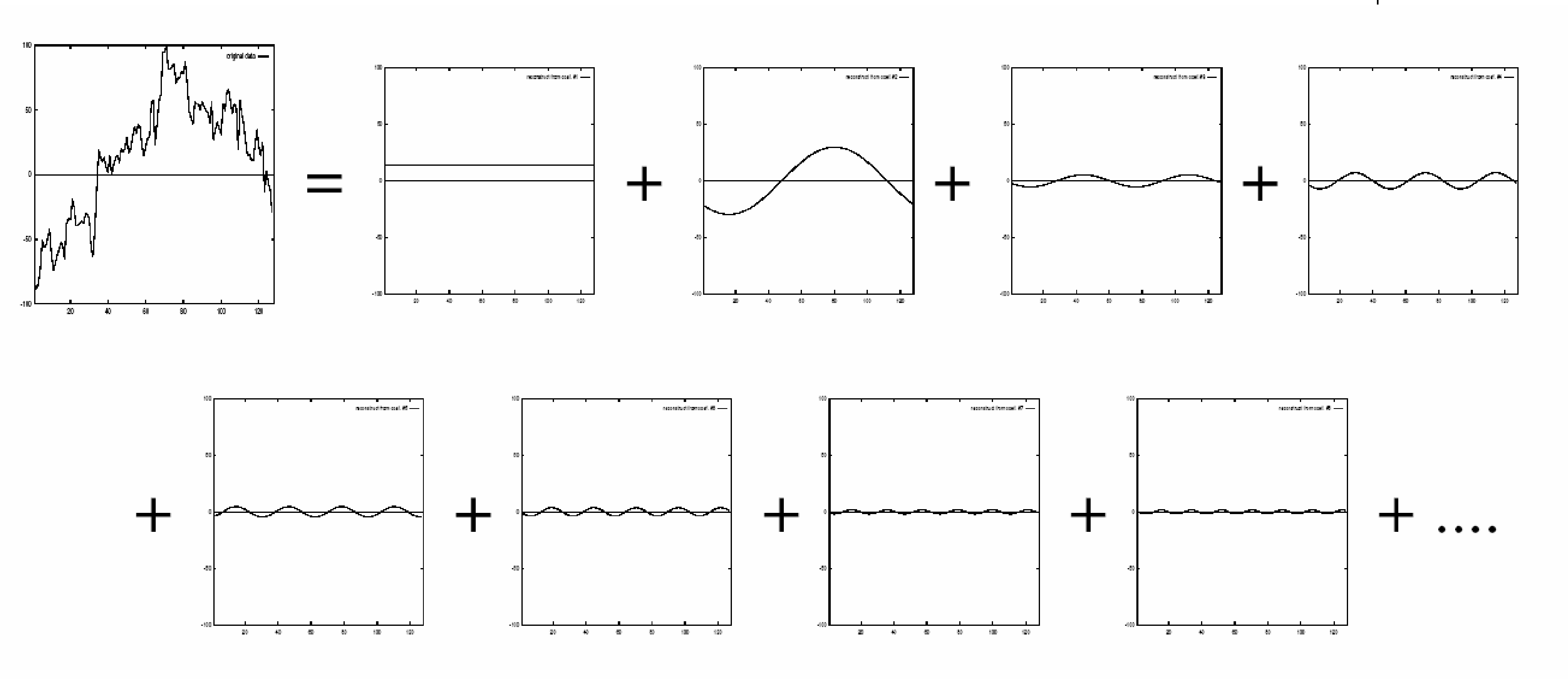
  L2 distance is preserved under "orthonormal transforms" (For L-p norm, only p=2 satisfy this property)

  Orthonormal transforms: K-L transform, DFT, DWT
- Optimal distance measure for estimation
  - If signals are corrupted by Gaussian, additive noise
- Widely used

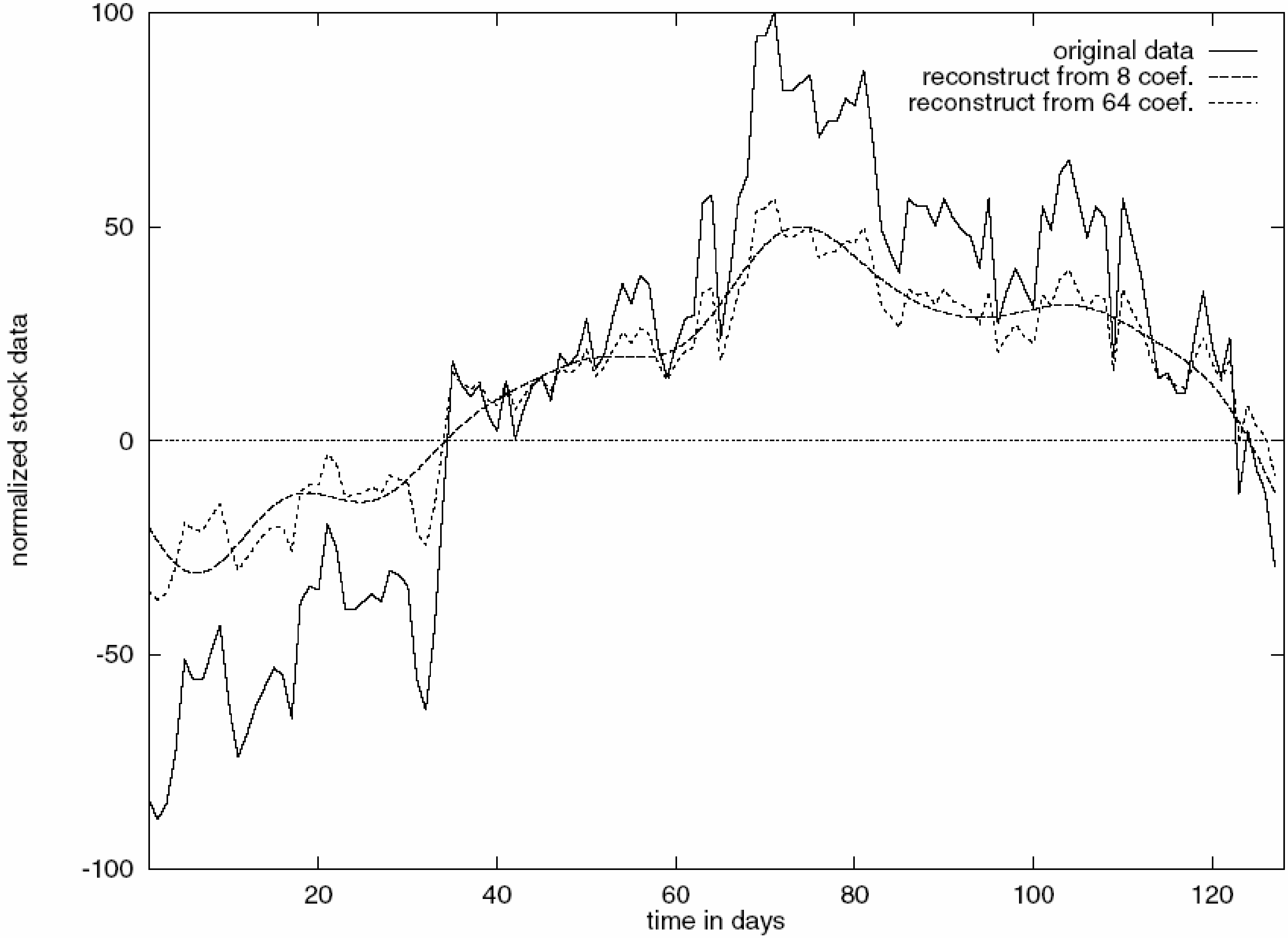
# How to index time-series data

- Can not direct index the data
  - Very big dimensionality (Even if query is just 512 points)
- Need to extract fewer important representative features to build index upon.
- Try to use first few parameters of DFT (Discrete Fourier Transform) to build index.

# DFT: Discrete Fourier transform




Figures taken from: “A comparison of DFT and DWT based similarity search in Time-series Databases” (Also figures on slide 9,17,18,24,25)



original data
reconstruct from 8 coef.
reconstruct from 64 coef.
normalized stock data
time in days
100
50
0
-50
-100
20
40
60
80
100
120

# DFT definition

- n-point DFT:

  ($X_f$ is frequency domain, $x_t$ is time domain)

  $X_f = (1/n^{1/2}) * \sum_{t=0 \text{ to } n-1} x_t \exp(-j2\pi ft/n)^2$ f = 0,1,…, n-1

- Inverse DFT:

  $x_t = (1/n^{1/2}) * \sum_{f=0 \text{ to } n-1} X_f \exp(j2\pi ft/n)^2$ t = 0,1,…, n-1

- Energy E(x):

  $E(x) = ||x||^2 = \sum |x_t|^2$

- FFT can be done in O(n*log*n) time

# Parseval's theorem

- Let X be the DFT of sequence x:

$$\sum |x_t|^2 = \sum |X_f|^2$$

- Since DFT is a linear transformation:

  … => $|| x_t - y_t ||^2 = || X_f - Y_f ||^2$

  L2 distance of two signal in time domain is same as their L2 distance in frequency domain

- No false dismissal if we just use first few parameters.
- But also do not want too many false hits

# Different time series data

| Type | Energy distribution in $O(f^b)$ | Example |
|---|---|---|
| White noise | $O(f^0)$ | Totally independent time series |
| Pink noise | $O(f^{-1})$ | Musical score, work of art |
| Brown noise (Brownian walks) | $O(f^{-2})$ | Stock movement, exchange rates |
| Black noise | $O(f^{-b})$ $b > 2$ | Water level of river vs time |

# Building Index

- When the signal is not white noise, we can use first few DFT parameter to capture most of the "energy" of the signal
- Let Q to be the query time-series data:

$$\sum_{first\_few\_freq} (q_f - x_f)^2$$
$$<= \sum_{all\_freq} (q_f - x_f |)^2$$
$$= \sum(q_t - x_t)^2$$
$$<= \varepsilon^2$$

# Building index (cont)

- Use the first few (4-6) DFT parameters, use R*-tree as index (called "F-index")
- Given a query Q and ε, use the index to filter out all nodes where:

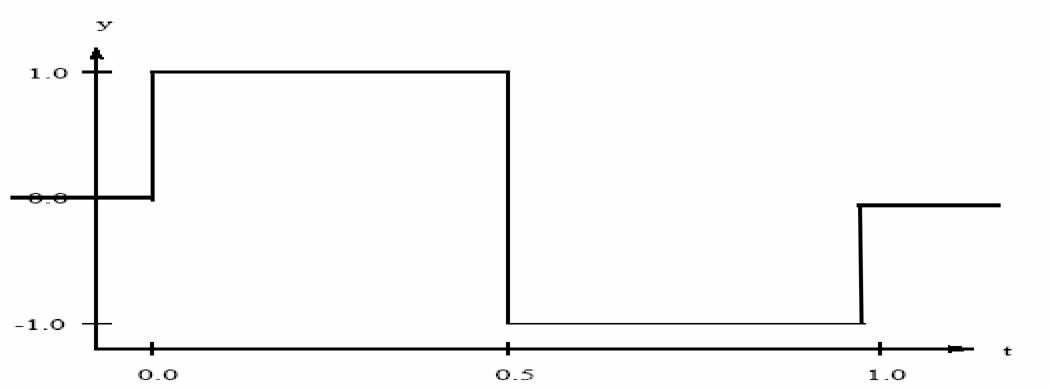
$$\sum_{first\_few\_freq} (q_f - x_f)^2 > \varepsilon^2$$

# Can we do better?

- Use DWT (Discrete Wavelet Transform)
- Harr wavelet definition:

$$\psi_i^j(x) = \psi(2^j x - i) \qquad i = 0, \ldots, 2^j-1$$

Where $\psi(t) = \begin{cases} 1 & 0 < t < 0.5 \\ -1 & 0.5 < t < 1 \\ 0 & \text{elsewhere} \end{cases}$



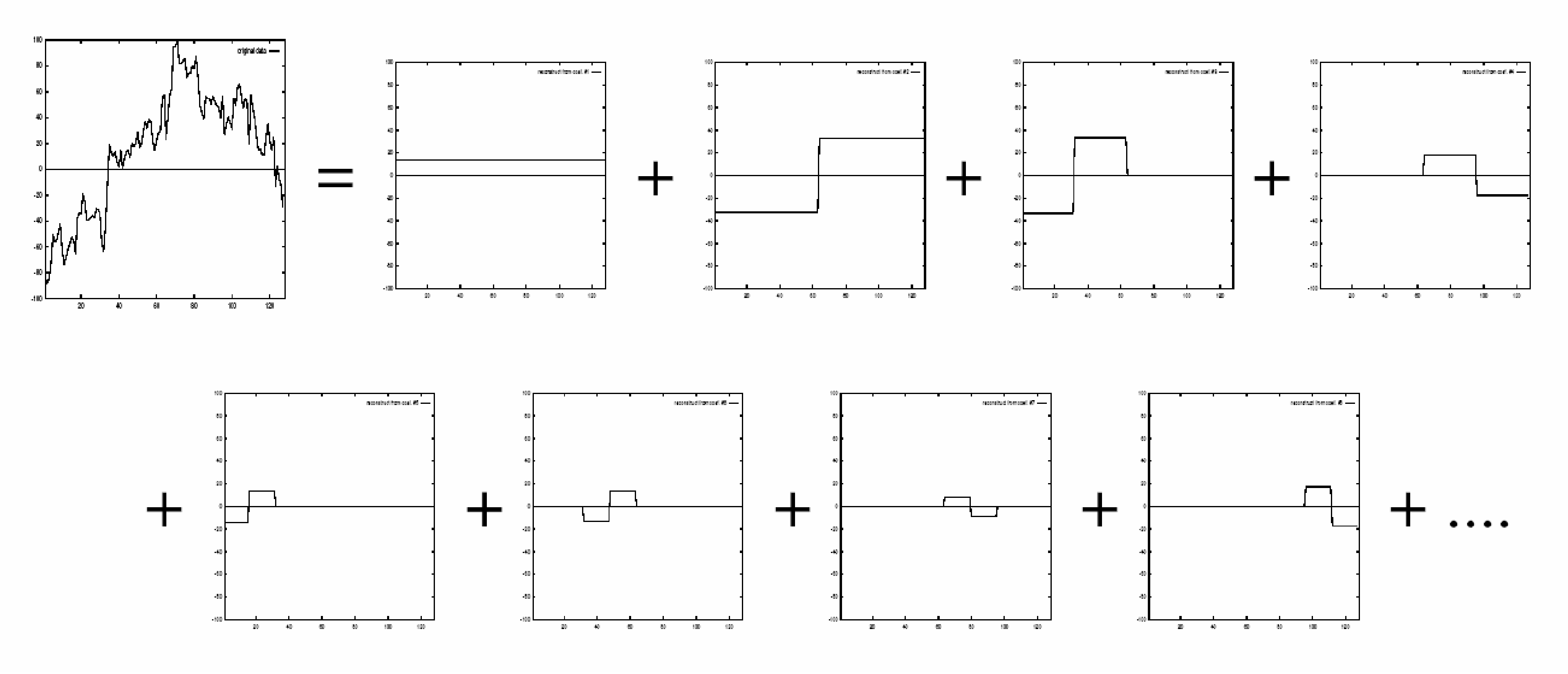
# Harr transform example

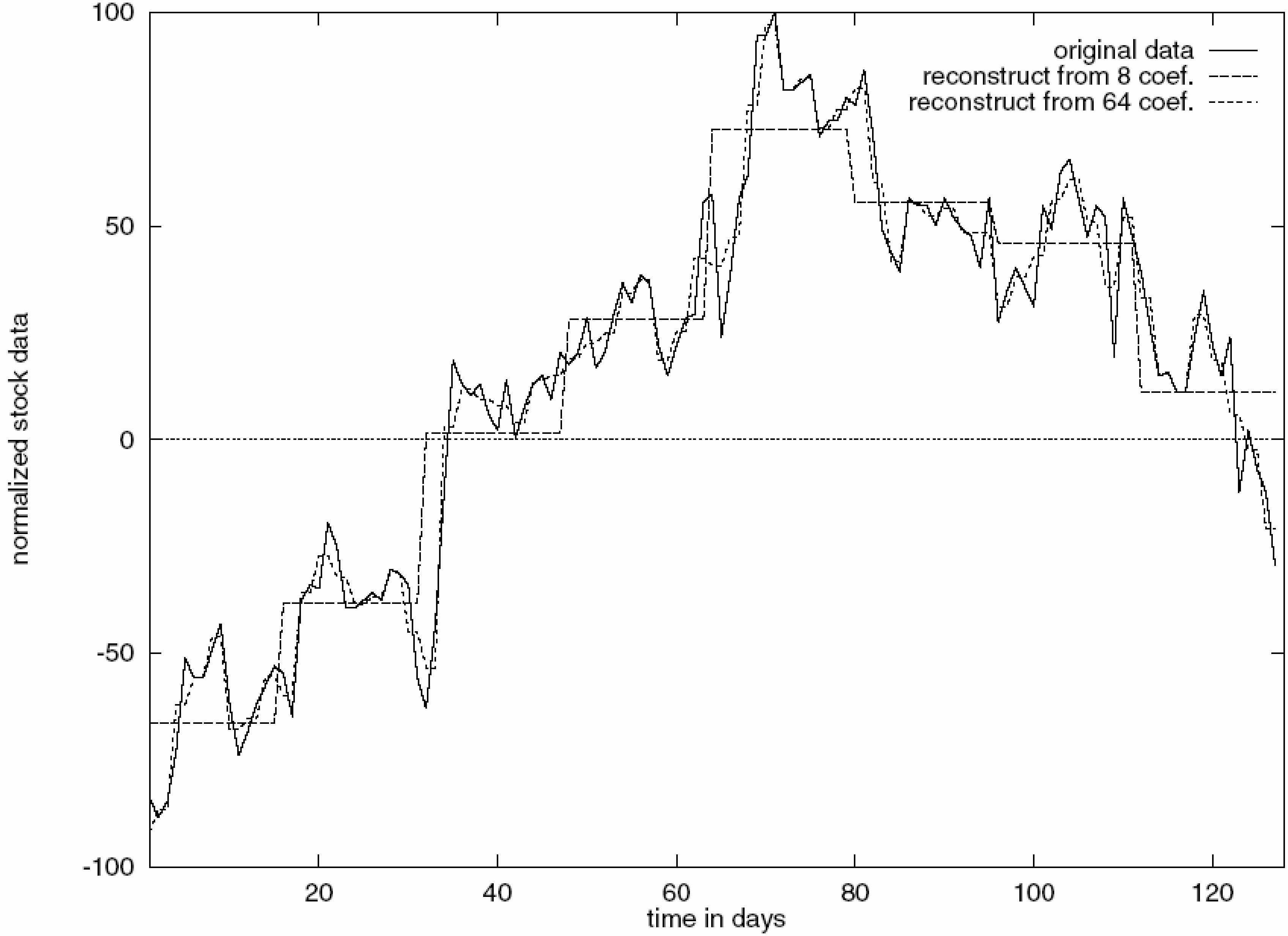
- Time series data: f(t) = (9 7 3 5)

| Resolution | Average | Coefficients |
|---|---|---|
| 4 | (9 7 3 5) | |
| 2 | (8 4) | (1 -1) |
| 1 | (6) | (2) |

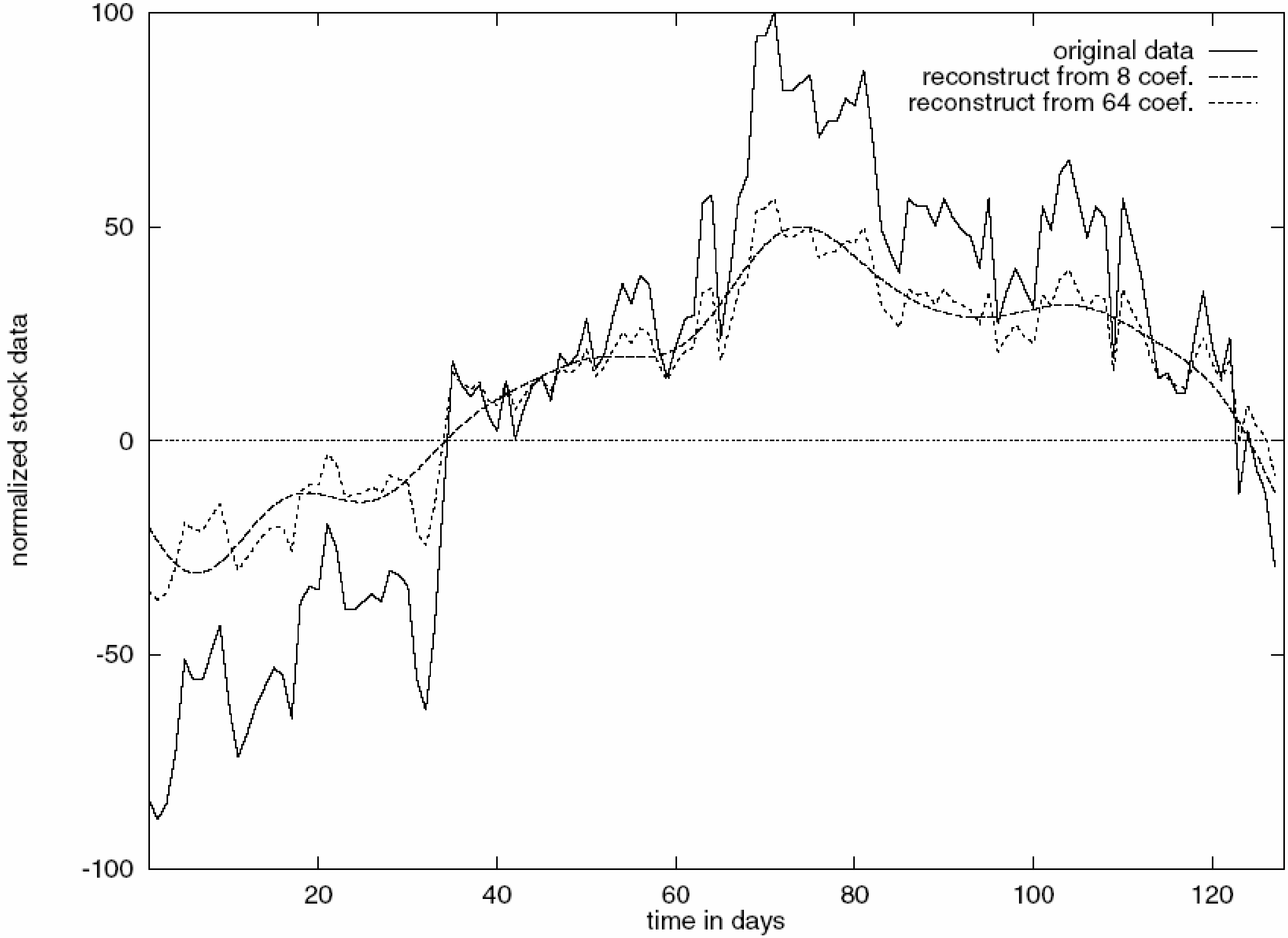
- Harr transform result: (6 2 1 -1)
- If we take only first two coefficients (6 2) and transform back, we get: (8 8 4 4)

# Use Harr wavelet with real data





original data
reconstruct from 8 coef.
reconstruct from 64 coef.
normalized stock data
time in days
100
50
0
-50
-100
20
40
60
80
100
120



original data
reconstruct from 8 coef.
reconstruct from 64 coef.
normalized stock data
time in days
100
50
0
-50
-100
20
40
60
80
100
120

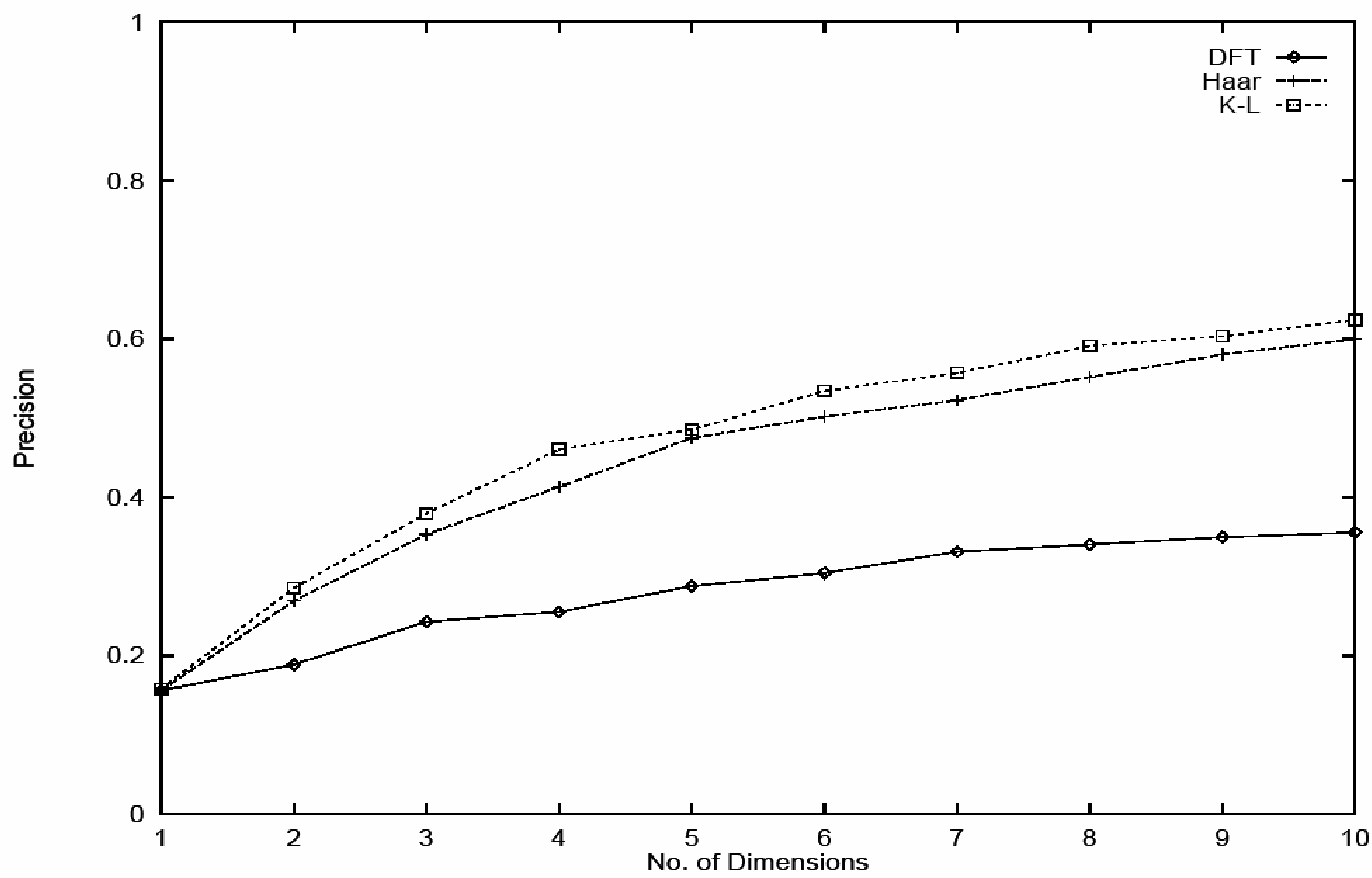
# Harr wavelet vs DFT

| | Harr wavelet | DFT |
|---|---|---|
| Preserve L2 distance | Yes | Yes |
| Feature | Can capture localized feature | Only global feature |
| Computation time | O(n) | O(n*logn*) |
| Energy concentration for first few params | Low resolution | Low frequency |

# Performance comparison

- 10k feature vectors from HK stock market using sliding window size ω=512
- Precision = $S_{time}$ / $S_{transform}$
  - $S_{time}$: # of sequences qualified in time domain
  - $S_{transform}$: # of sequences qualified in transformed domain
- Compare precision using different amount of coefficients with different method

# Performance (HK stock)



# DFT fights back

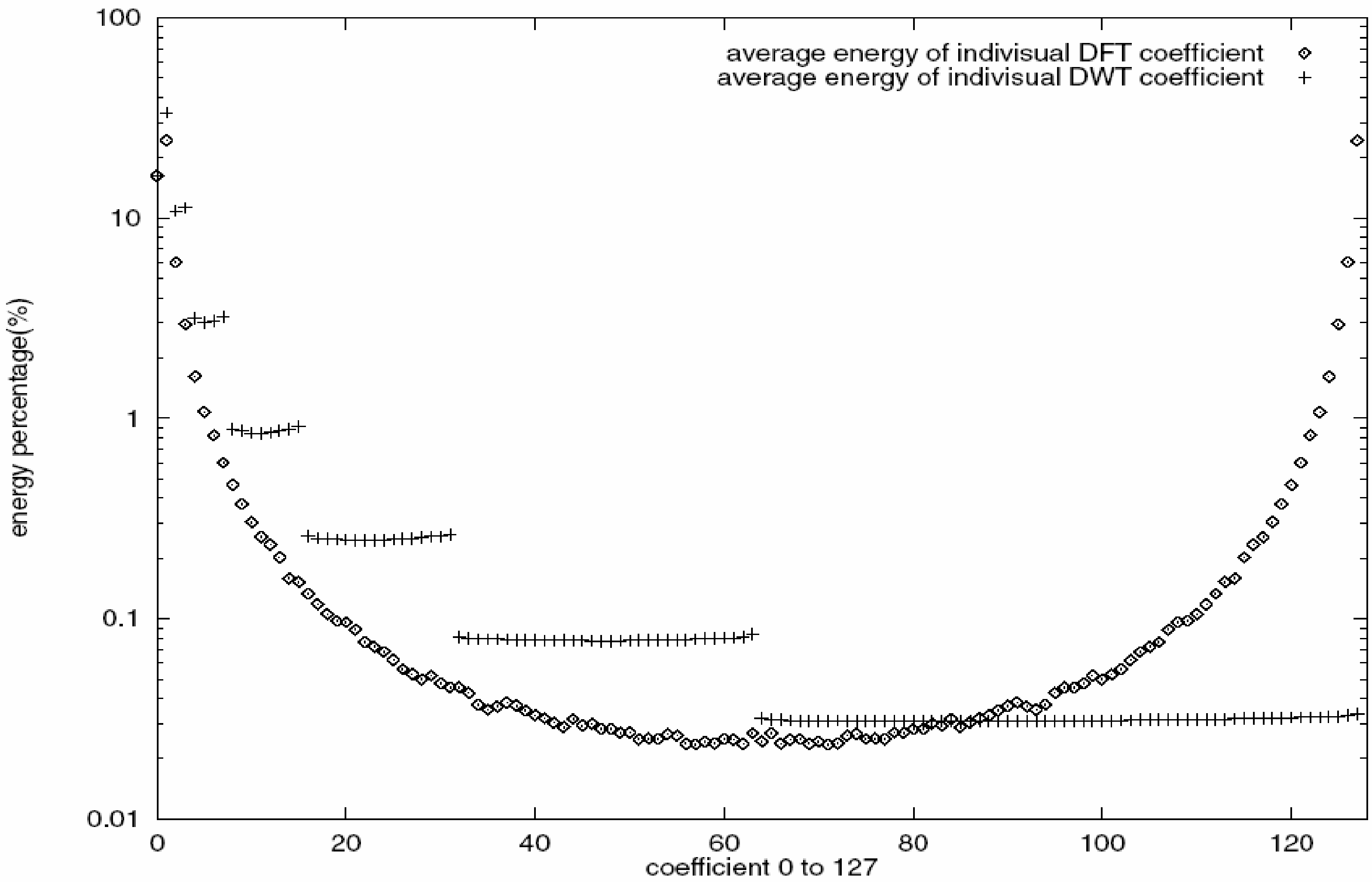
- Use last few DFT coefficient to improve quality (Davood Rafiei)

The DFT coefficients of a real valued sequence of duration n satisfy:

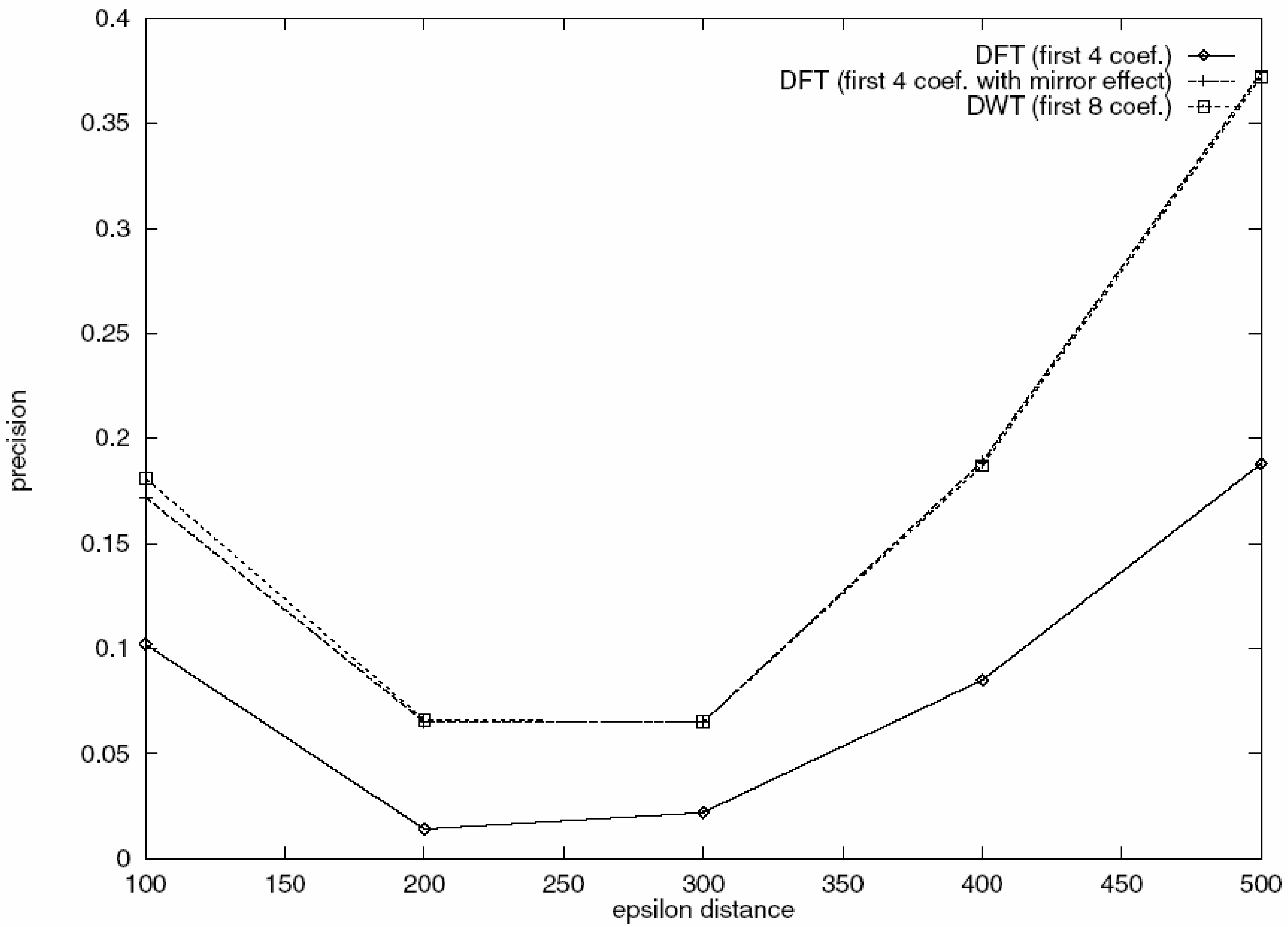
$X_{n-f} = X_f^*$ (f = 1, … , n-1)

Note: if X = a+b*i*, X* = a-b*i*

# Energy distribution: DFT vs DWT

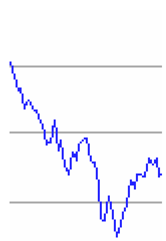


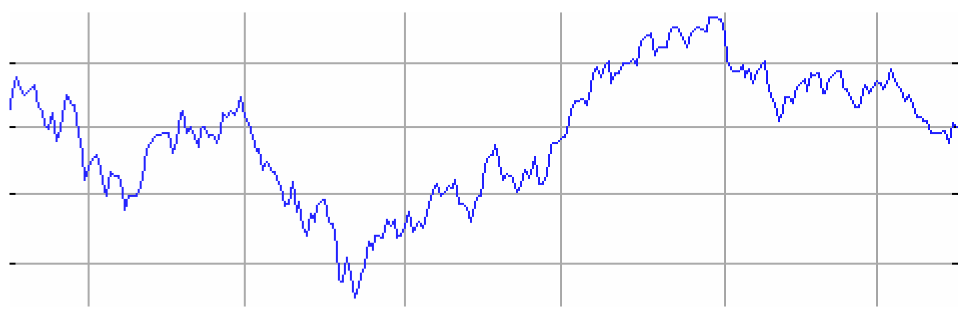
# Performance (100 stocks)



# What is next: "Subsequence" query

- Up to now, we are focused on "**whole**" time series data match.
- What if we need to match subsequence efficiently?

Query: 

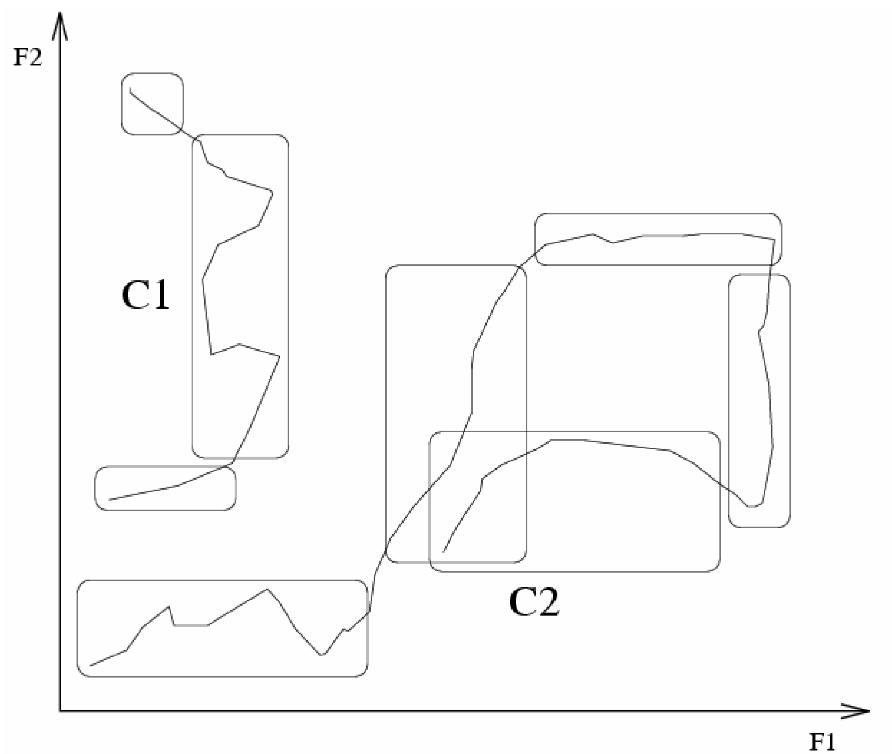
Data: 

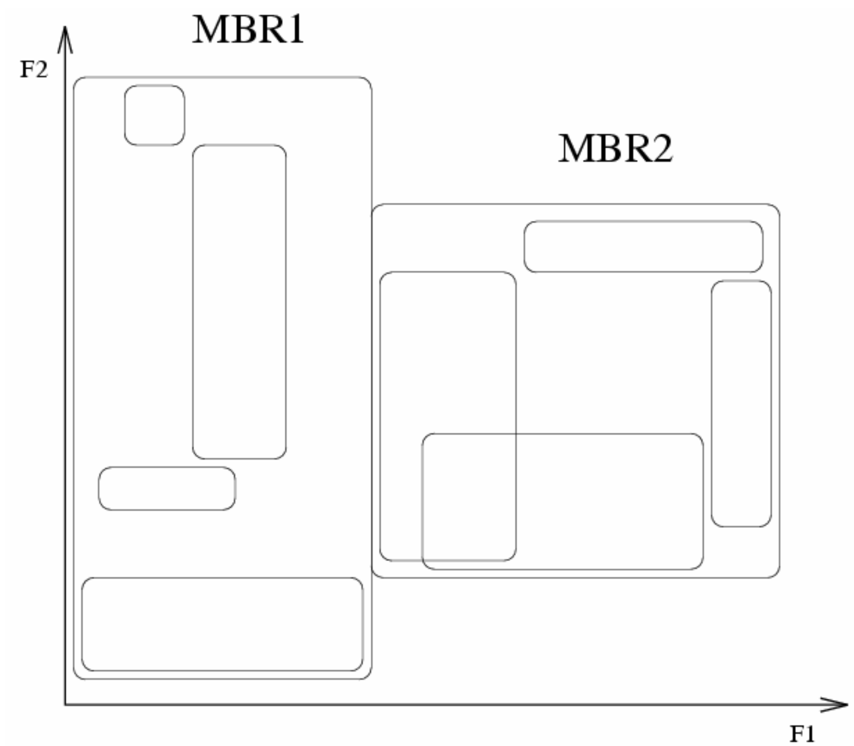
# Naïve method

- Assume query length fixed at ω
- Using a sliding window with length ω, slide through the data.
- Insert all possible data points into the index (using F-index)
- Could be twice as slow as "sequential scan"

# ST-index

- Observation: Successive sliding window tend to generate similar coefficients
- MBR: Minimum Bounding (hyper) Rectangle



# How about "arbitrary" length time series query?

- Two basic methods: (Assume dataset is indexed with window length ω)
  - Prefix search

    Simply use the first ω of the query to do the search
  - Multipiece search

    If |q| >= kω, split q into k pieces, and search DB with $\varepsilon/(k^{1/2})$, join the results.

# Multi-resolution index

- Base-2 MR index structure
- Take ω as base unit, build index for each window size of $2^i\omega$.
- Basic algorithm: Longest Prefix Search (LPS)

  Eg: if query length = 19ω

  use 16ω as the prefix to do search.

# Index structure layout

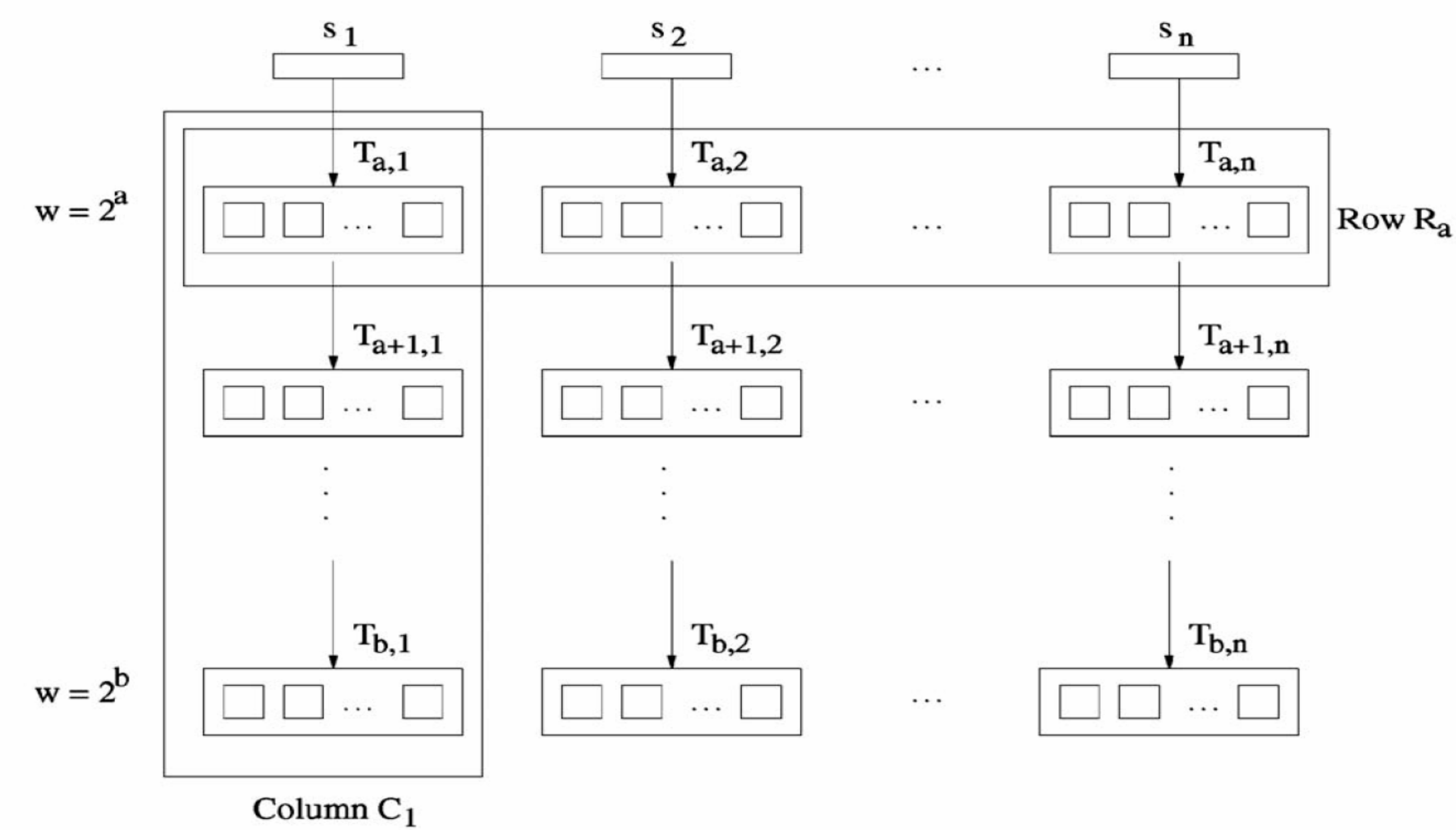
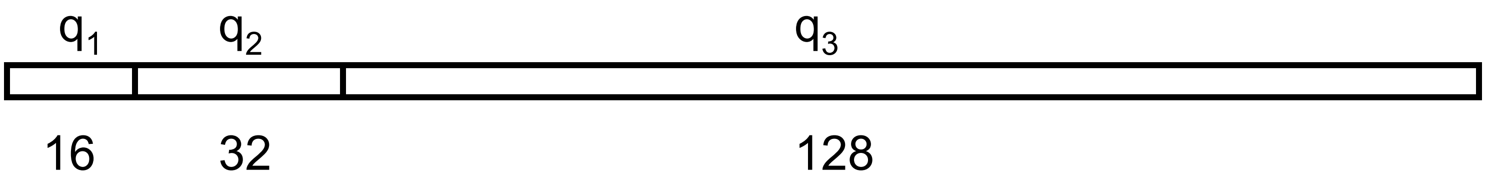


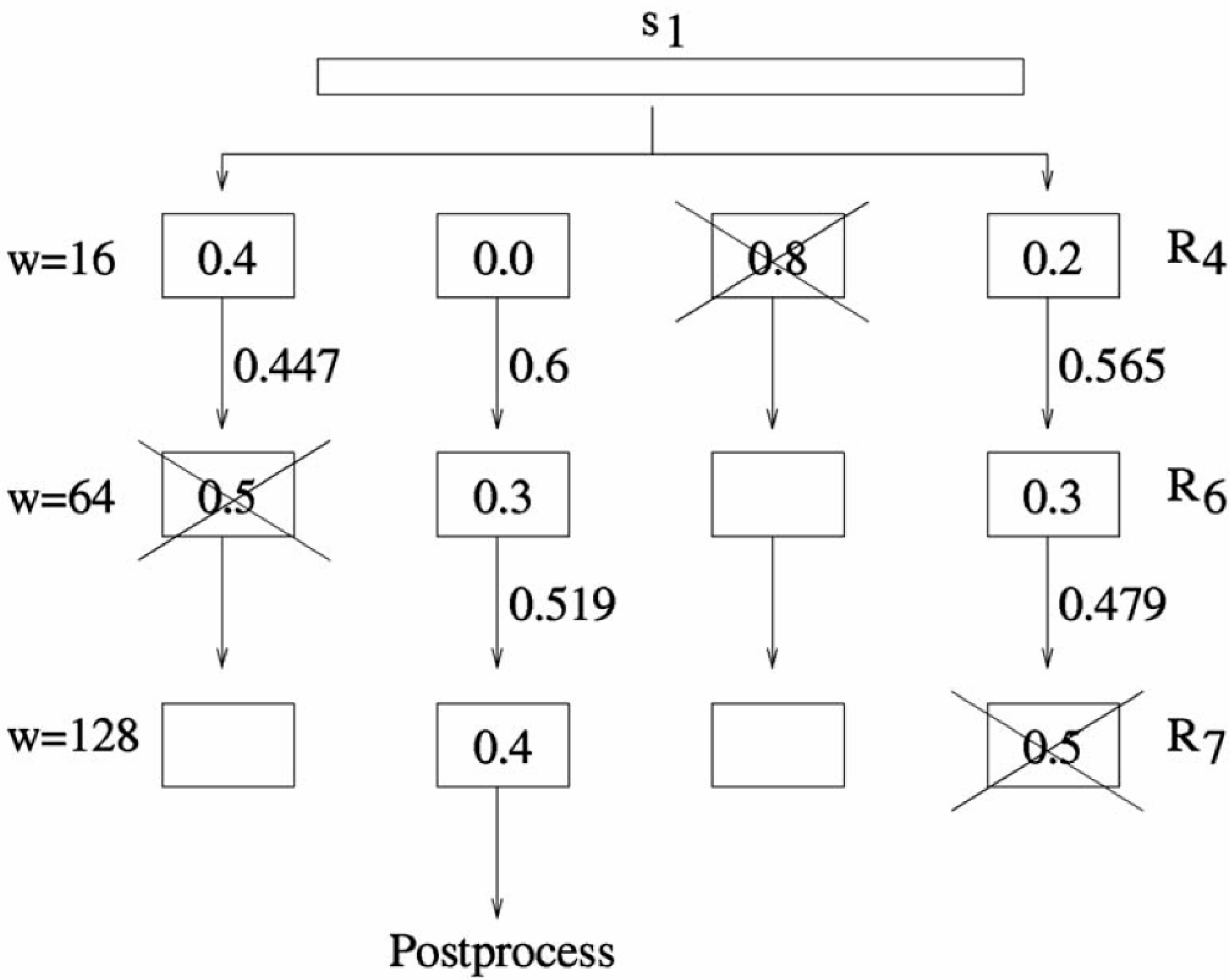

Figure taken from: "Optimizing similarity search for arbitrary length time series queries" (also figures on three slides)

# Improved algorithm

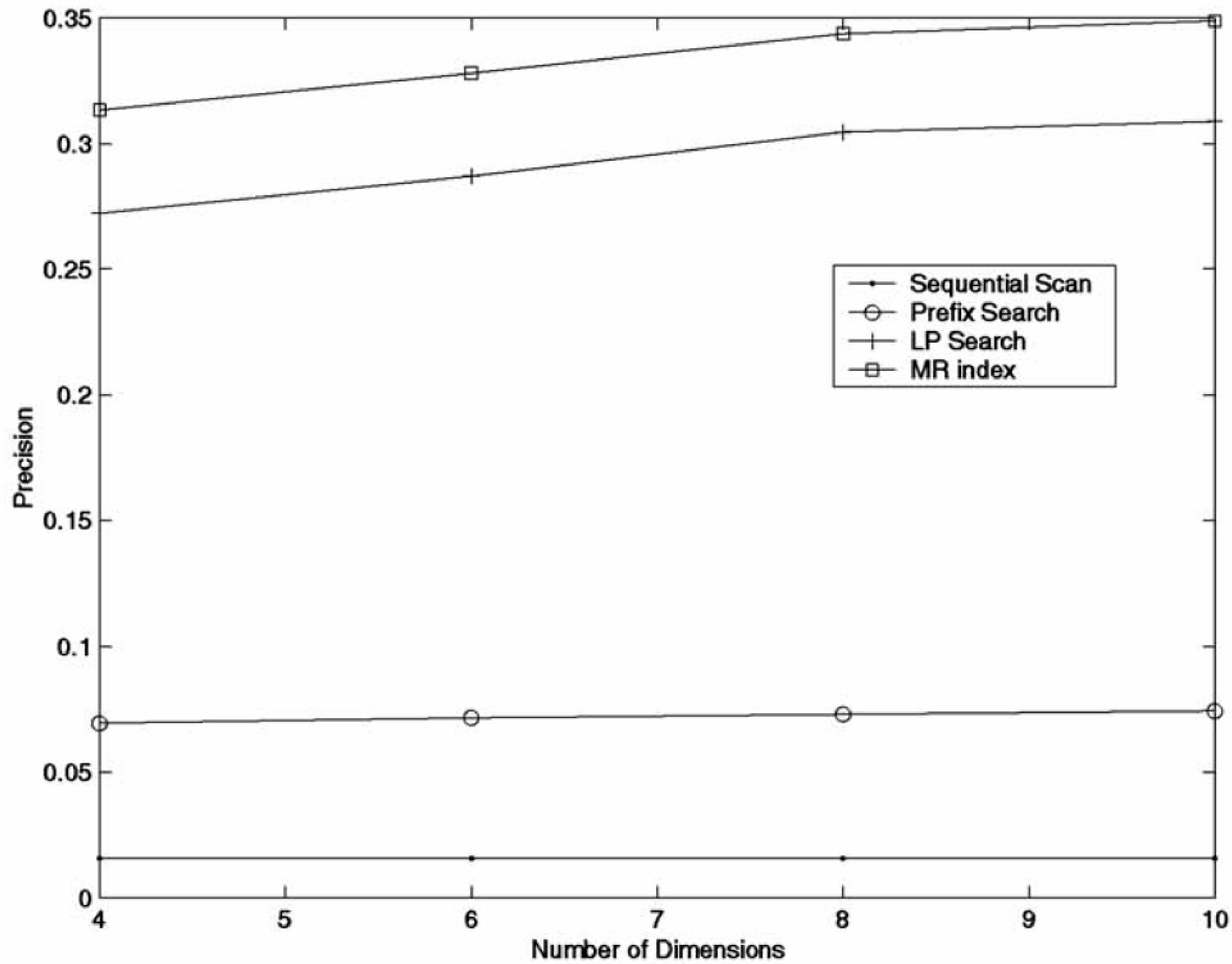
- Split q into $q_1 q_2 \ldots q_t$, where $|q_i| = 2^{C_i}$
- Eg: Given |q| = 208 = 16 + 32 + 128




- Given ε = 0.6, query example shown in the next slide


s 1
w=16
0.4
0.0
0.8
0.2
R4
0.447
0.6
0.565
w=64
0.5
0.3
0.3
R6
0.519
0.479
w=128
0.4
0.5
R7
Postprocess

# Performance on 556 stocks (again)



# Summary

- Harr DWT and DFT performs similar in feature extraction for stock data.
- Start monitor stock now! (All of these papers use stock data + synthetic data)
- Time series data is hard to optimize for similarity search.
- All these paper are focused on “no false dismissal”, approximation might help. (Some research done.)

# Related papers

- Tamer Kahveci and Ambuj K. Singh. Optimizing Similarity Search for Arbitrary Length Time Series Queries
- R. Agrawal, C. Faloutsos, and A. Swami, Efficient Similarity Search in Sequence Databases.
- K.-P. Chan and A.W.-C. Fu, Efficient Time Series Matching by Wavelets.
- C Faloutsos, M Ranganathan, Y Manolopoulos, Fast subsequence matching in time-series databases
- D. Rafiei and A. Mendelzon, Efficient Retrieval of Similar Time Sequences using DFT
- YL Wu, D Agrawal, A Abbadi A comparison of DFT and DWT based similarity search in Time-series Databases

# Orthonormal transform

- Matrix O is known as orthonormal if it satisfy the orthonormality property:

$$\mathcal{O}^T\mathcal{O} = I_N \equiv \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ & & \ddots & \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

From: http://www.math.iitb.ac.in/~suneel/final_report/node15.html