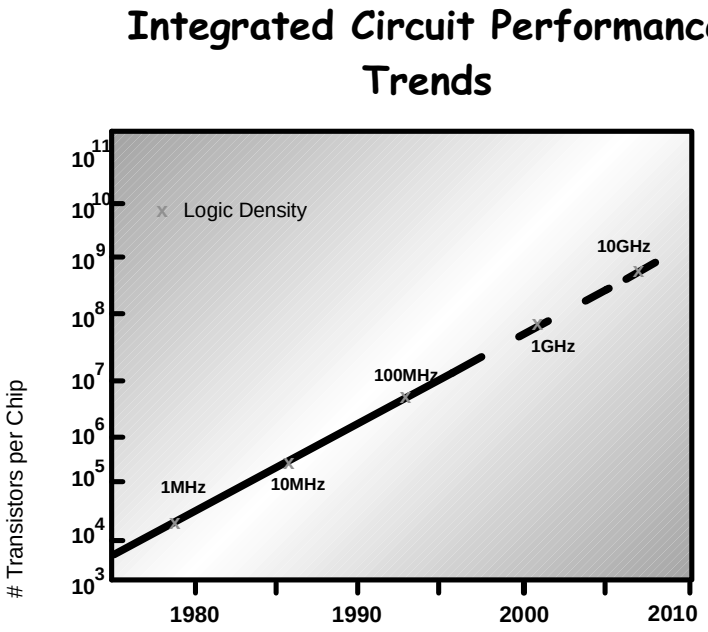
Event Mining From Distributed E-commerce Systems

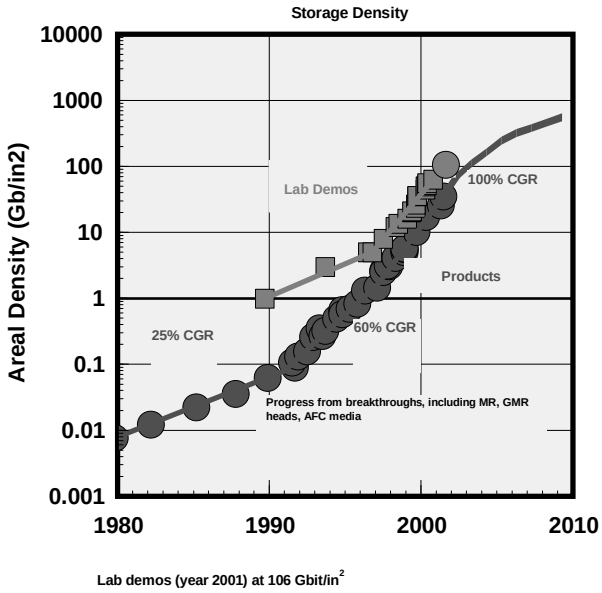
Sheng Ma
IBM T.J Watson Research
2004

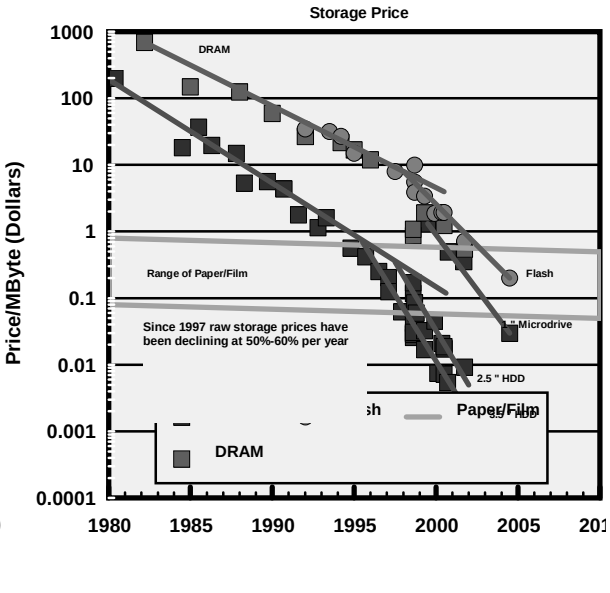
Contributors

- Chang-shing Perng
- Genady Grabarnik
- Mark Brodie
- Irina Rish
- Joseph Hellerstein
- David Thoenen (IBM ATS)
- Alina Beygelzimer
- Ricardo Vilalta (Univ. of Huston)
- Carlotta Domenicon (GMU)
- Tao Li

The Good News: more powerful and cheaper

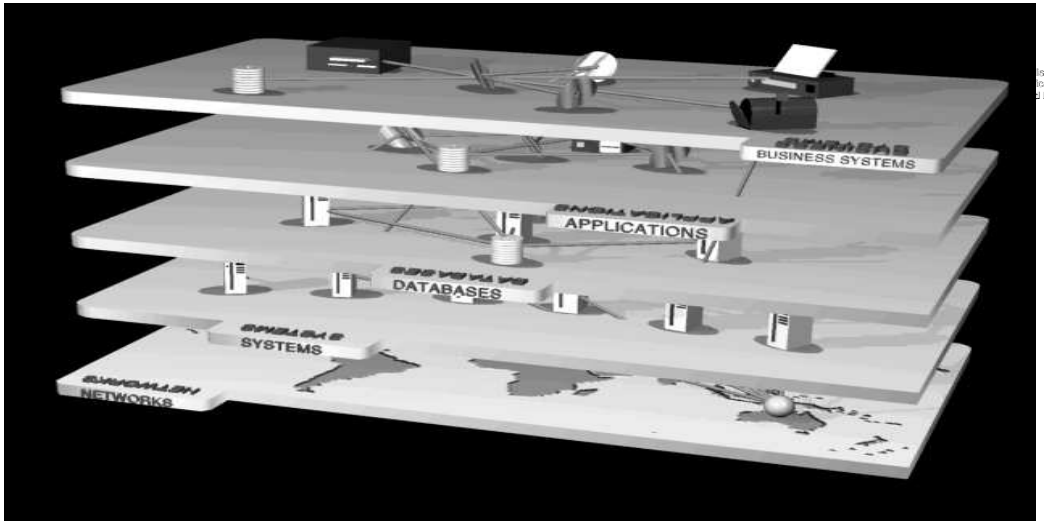






Lab demos (year 2001) at 106 Gbit/in^2

The Bad News: complexity and labor cost



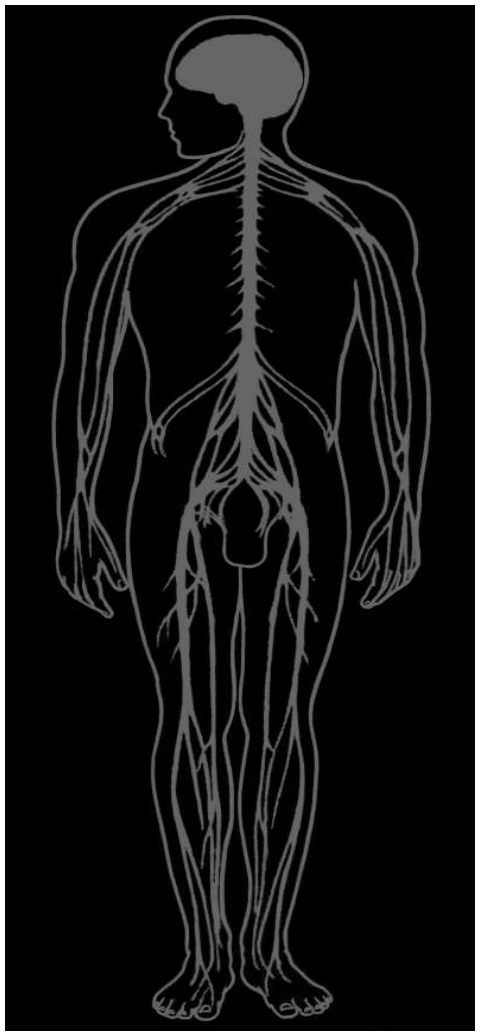
Autonomic Computing

Self-configuring
Adapt automatically to the dynamically changing environments

Self-healing
Discover, diagnose, and react to disruptions

Self-optimizing
Monitor and tune resources automatically

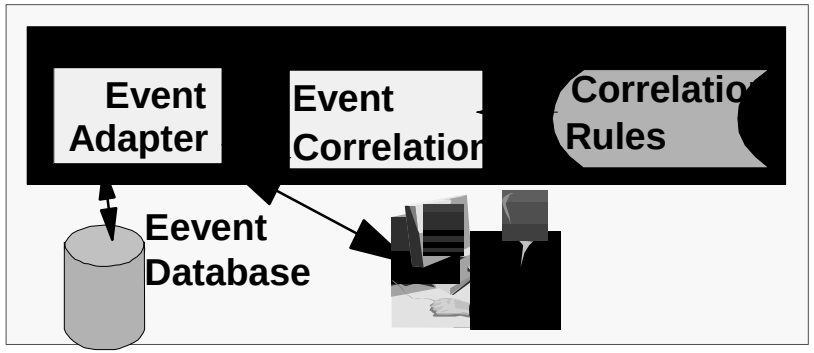
Self-protecting
Anticipate, detect, identify, and protect against attacks from anywhere



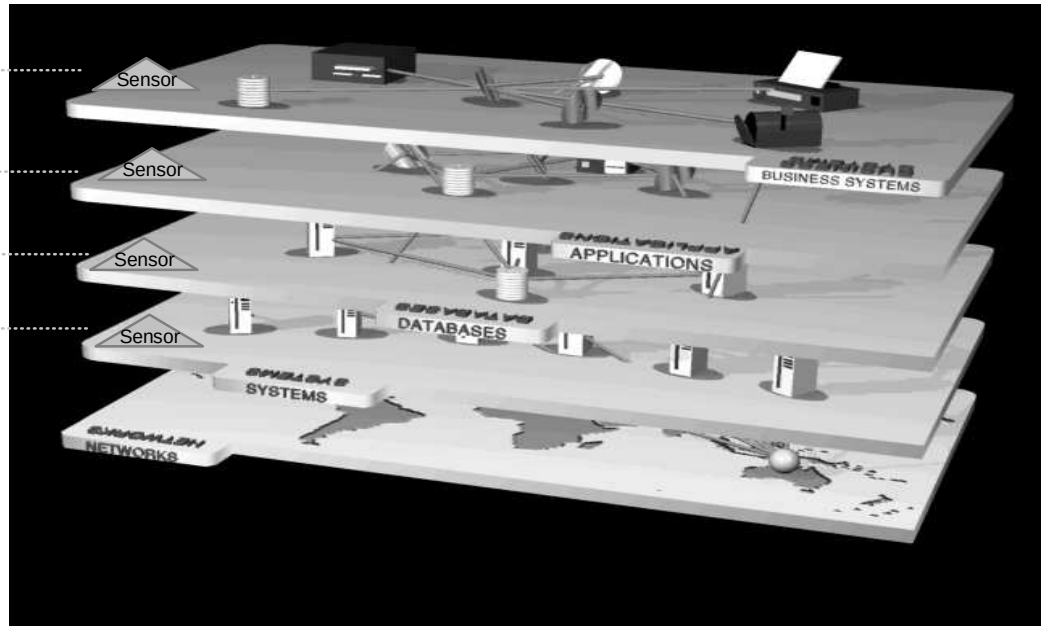
Toward Self-managing Systems...

Informaiton Management is key for high performance and avialability

Correlation analysis



Data collection



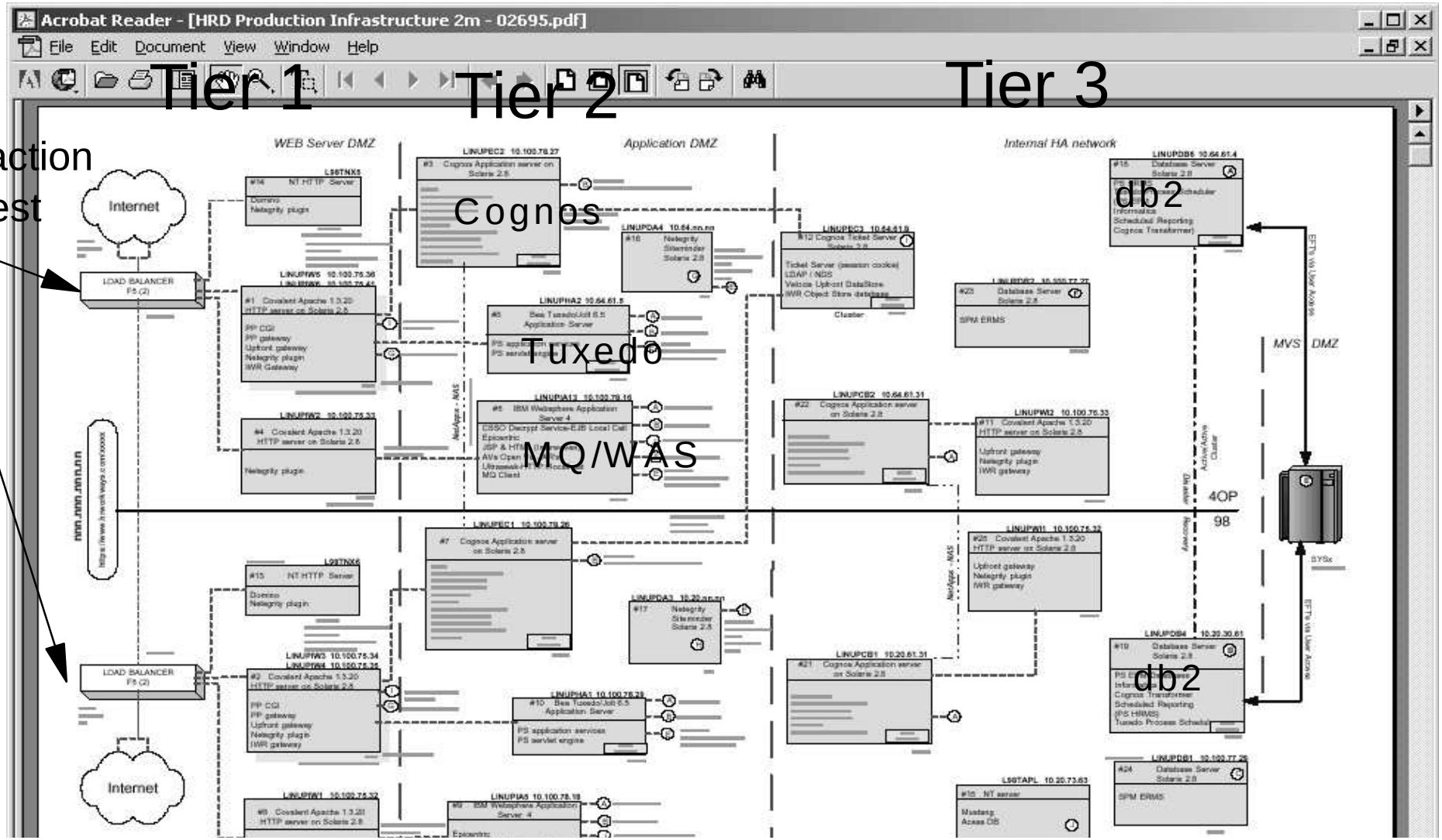
Event management system (e.g. Tivoli's product)

- Correlation rules are the key
- Example:

rebooting signature: link_down is followed by link_up in five seconds => filtering

link_down without link_up => an operator should be paged (avialability problem)

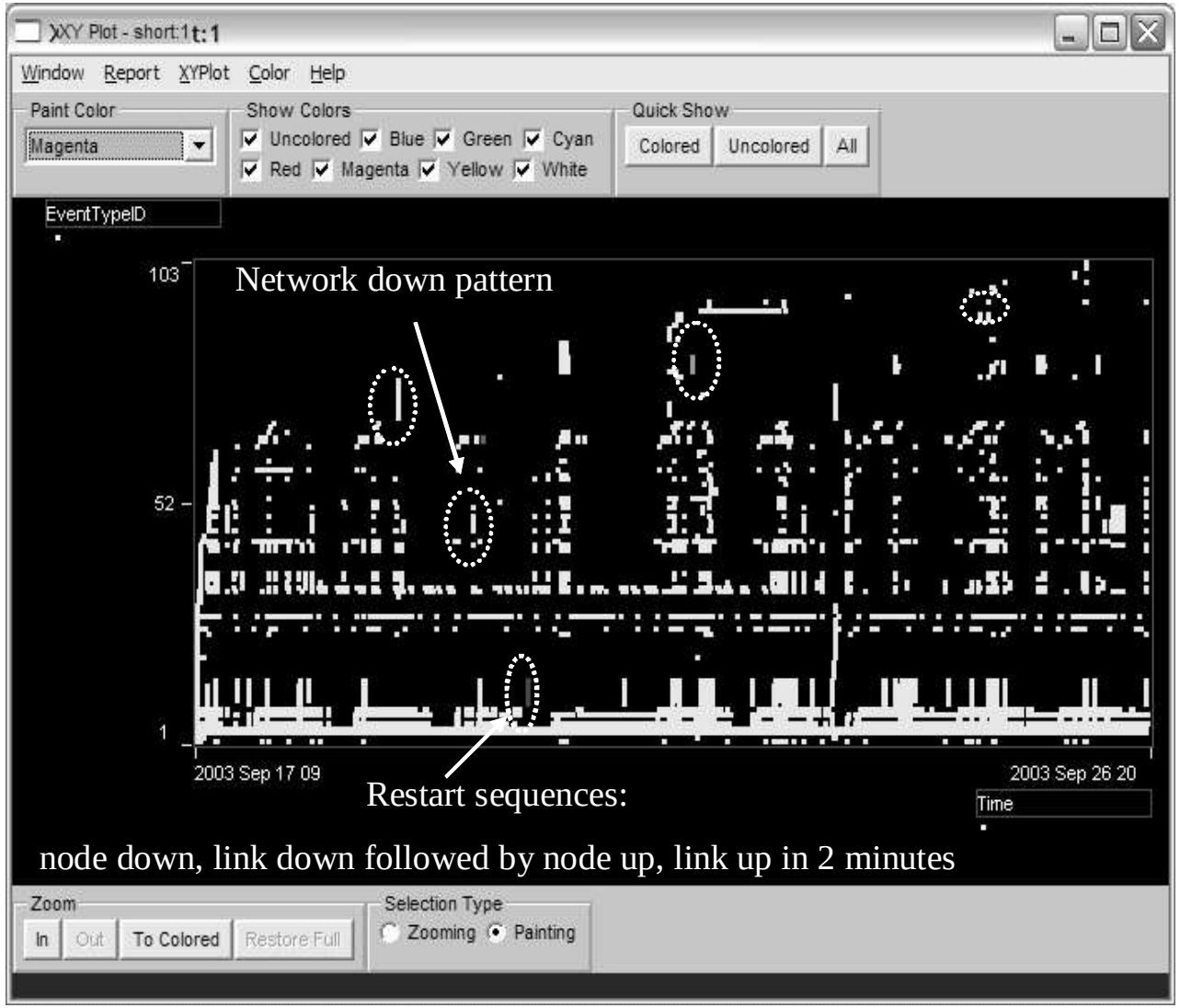
A customer environment



Many data sources, many boxes, a lot of data

2GB raw data from Linux, DB2, WAS, MQ, TEC

Problem motivation: Event Mining



Rule-based engine can be used for processing such events in real-time

Pain-points:

- How to build knowledge (or correlation rules)
- How to identify problems
- How to tune

Current approach: knowledge-based

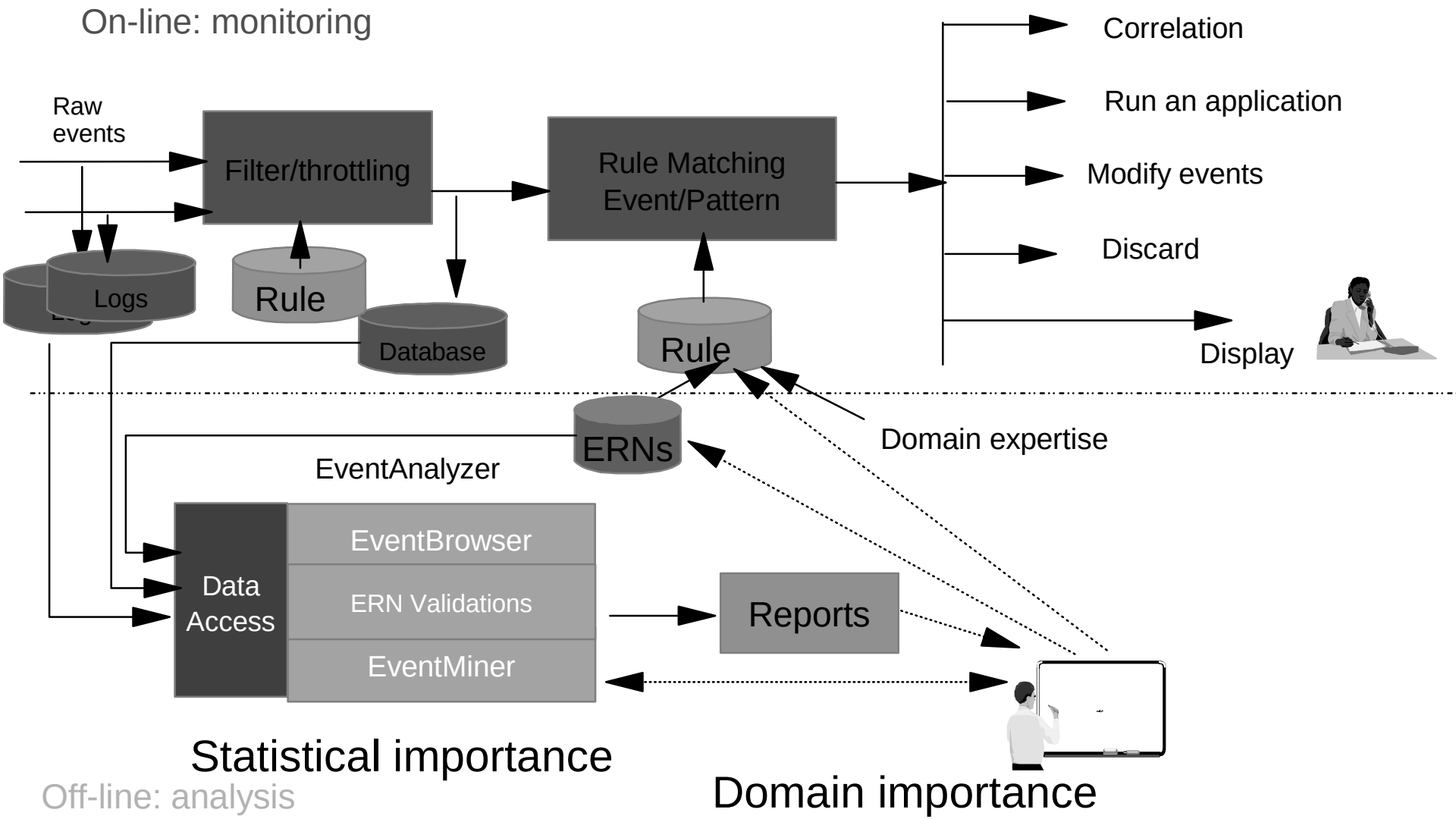
Our approach:

mine historical events to identify repeated event patterns (sequences)

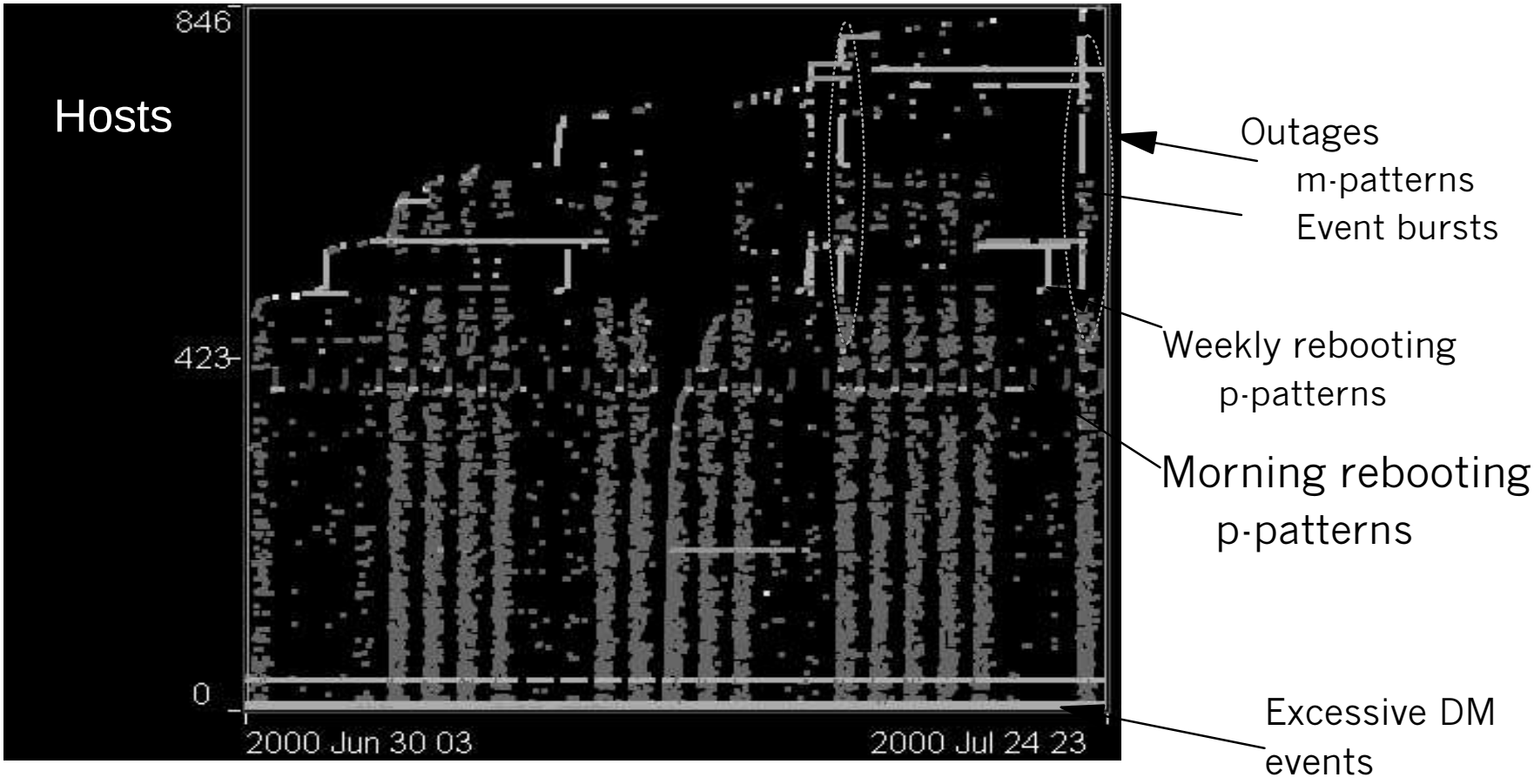
Review such patterns are reviewed with domain experts to identify causes and define action (e.g. filtering, paging)

Programmatically translate patterns in to correlation rule or symptom DB

Event Mining to discovery rules for automated run time operation



Alarms in a large computer system



Alarms in computer system management

- Examples: router is down at 9:00am; cpu_utilization of a file server is above 90%
- Many types of patterns: periodic, temporal dependency, etc
- Patterns signify problems
- How to discover such patterns; how to perdict

Value of Event Mining

- Mining many logs collected from different locations
- Identify system problems
  - Configuration problem
  - Design problem
- Discover essential rules for run-time monitoring and automated operation
  - Discover filtering rules
  - Discover correlation rules
- Problem diagnosis tool and root cause analysis
- Predictive analysis to avoid problems if possible

Research areas

- Visualization
  - Fast ordering of large categorical datasets for better visualization, A. Beygelzimer, C. Perng, and S. Ma, KDD 2001.
  - Ordering categorical data to improve visualization, S. Ma and J.L. Hellerstein, InfoVis99, July 1999.
- Pattern discovery
  - Mining Partially Periodic Event Patterns with Unkonw Periods, Sheng Ma, Joseph L Hellerstein, ICDE 2001.
  - Mining Mutual Dependent event patterns, S. Ma and J.L. Hellerstein, ICDM 2001.
  - Discovering fully dependent patterns, Feng Liang, Sheng Ma and J.L. Hellerstein, SIAMDM 2002.
  - FARM: a framework for exploring mining spaces with multiple attributes, C. Perng, H. Wang, S. Ma and J.L. Hellerstein, ICDM 2001
- Prediction
  - Rule Induction of Computer Events, Ricardo Vilalta, S. Ma and J.L. Hellerstein, DSOM 20001.
  - Local Predictions in Event Sequences Using Asociations and Classification. , R. Vilalta and S. Ma, ICDM 2002.
  - A classification approach for prediction of targetted events in temporal sequences, C. Domeniconi, C.S. Perng, R. Vilala, ECML 2002.
- Clustering
- Profiling

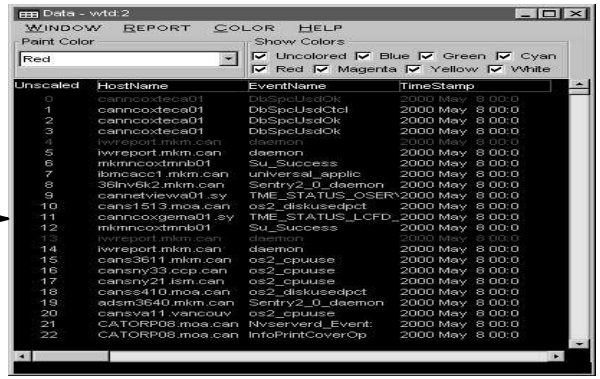
Tools and Applications

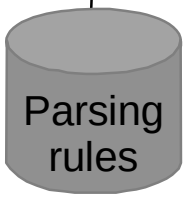
- Tools
  - Event Browser
    - EventBrowser: exploratory analysis of event data for event management, S. Ma and J.L. Hellerstein, DSOM 1999.
  - Event Miner
    - Progressive and interactive analysis of event data using event miner, S. Ma, J.L. Hellerstein, C. Perng and G. Grabarnik, submitted 2002
  - ERN CVC
    - Data-driven validation, completion and construction of event relationship networks, C.S. Perng, D. Thoenen, S. Ma and J.L. Hellerstein, KDD 2003.
- Applications
  - System management
    - Discovering actionable pattern from event data, J.L, Hellerstein, S. Ma and C. Perng, to be appeared in IBM system journal on AI.
    - Scalable Visualization of EPP Data, David J. Taylor, Nagui Halim, Joseph L. Hellerstein, and Sheng Ma, DSOM 2000
  - IDS
    - A data mining system for network-based intrusion detection system, M. Mei, D. George, M. Brodie and S. Ma, in preparation.
  - BlueLight
    - Critical Event Prediction for Proactive Management of Large-Scale Computer Clusters

First step: normalization

Textual format

Table format:
timestamp, event type, host name, severity, interest, importance



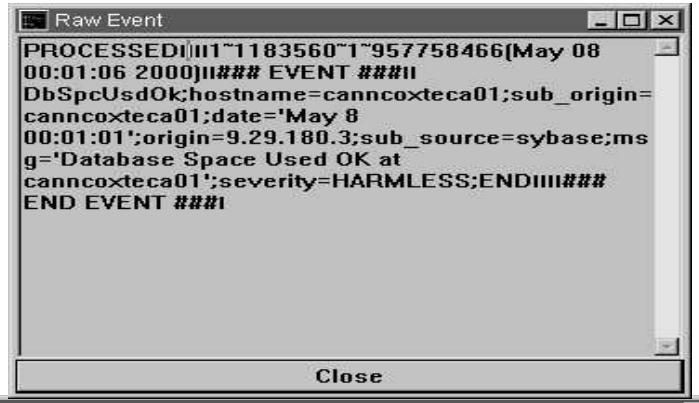


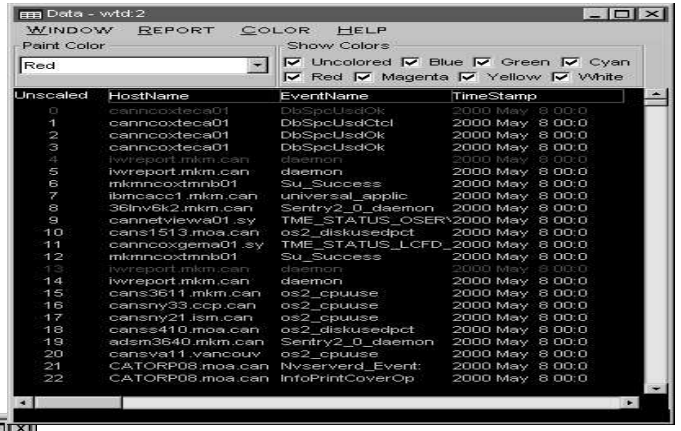
Domain knowledge
Interest/importace rating
for each events and host

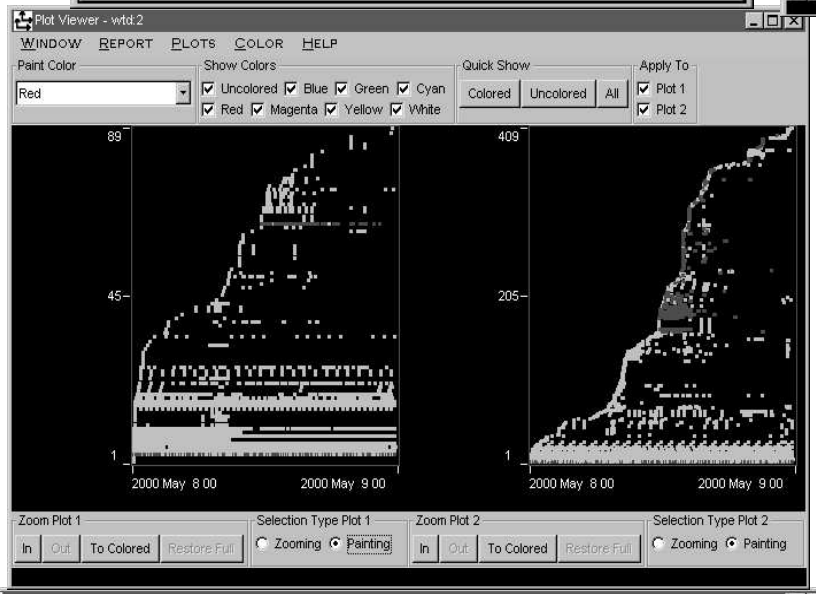
Raw data
->Data table

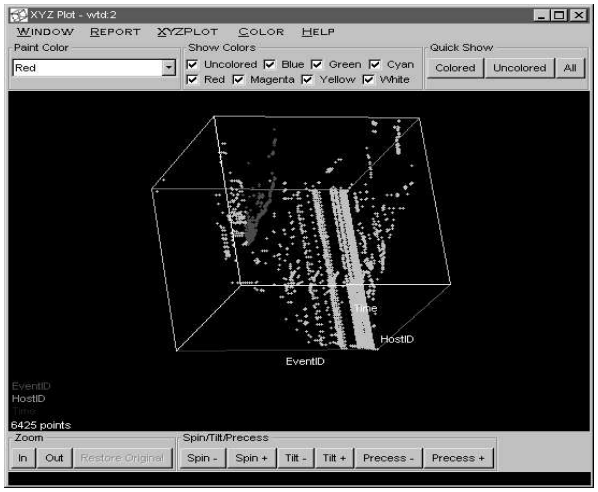
Event Plots

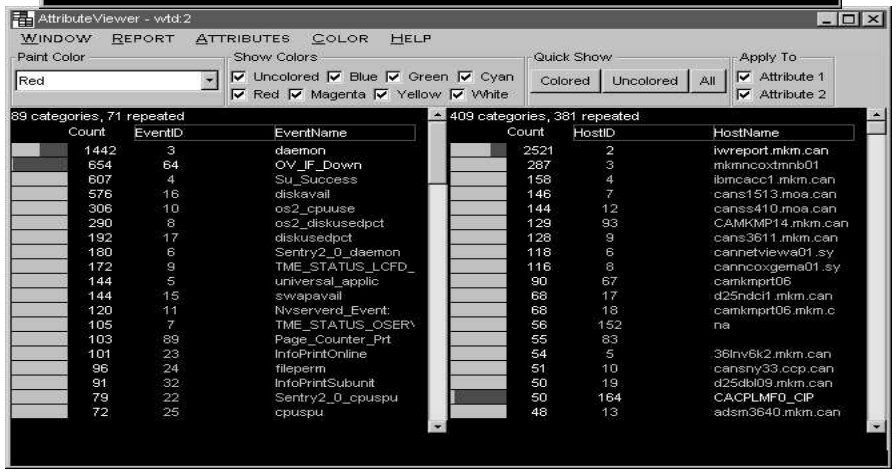
Aggregation
summarization

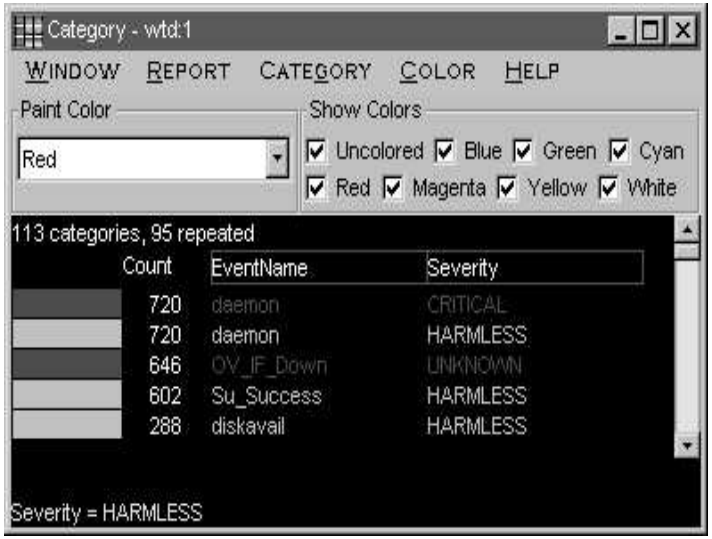








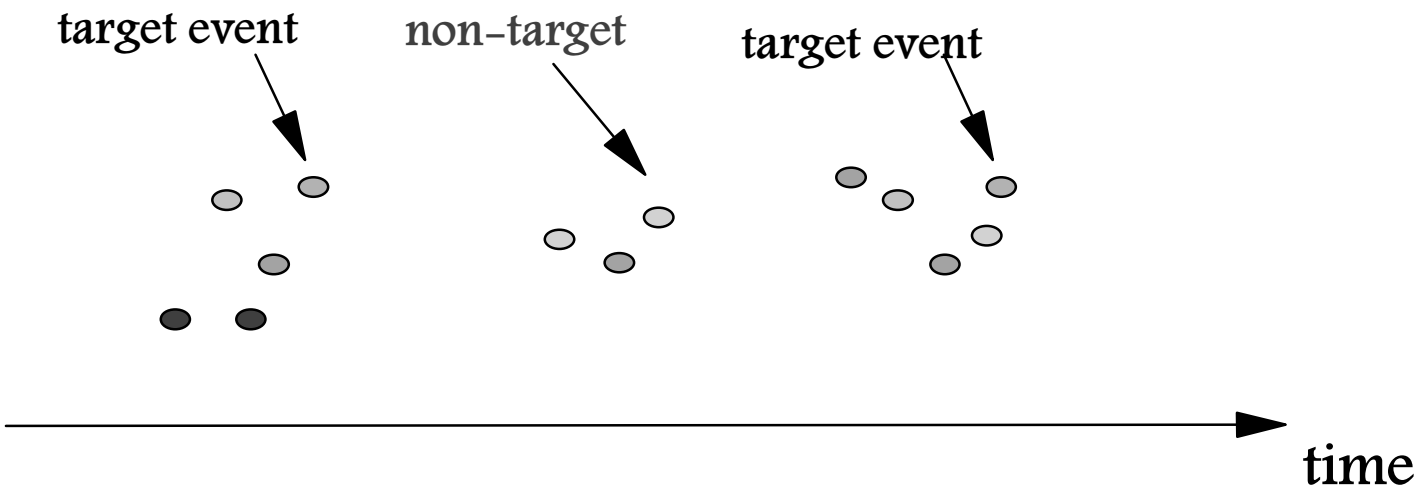




Event Predition

Event Prediction

Event Data



Problem

Can we predict the green events?

=> learn from historical data

Technical challenges

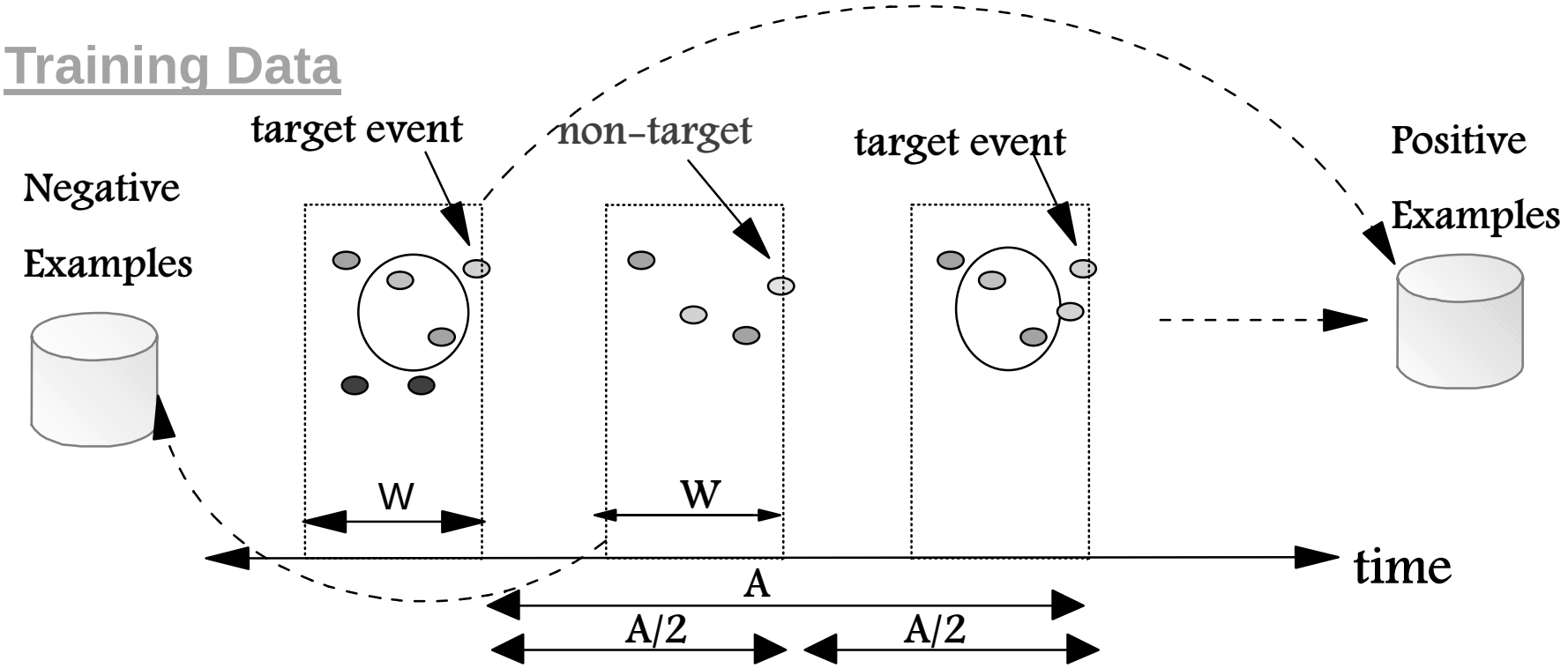
Temporal information

Categorical, not time series

Extremely unbalanced target vs. non-target

Classification-Based Prediction

Training Data



Sample results

When there is a URL time out on host A we frequently observe a URL time out on host B.

High CPU utilization on host C is normally followed by a link down on hosts D and F.

Range of accuracy: 65%-93%

with Carlotta Domeniconi and Charles Perng, ECML 2002

Carlotta_presentation.ps - GSview

File Edit Options View Orientation Media Help

File: Carlotta_presentation.ps Page: "29" 24 of 27

Offline Prediction Results

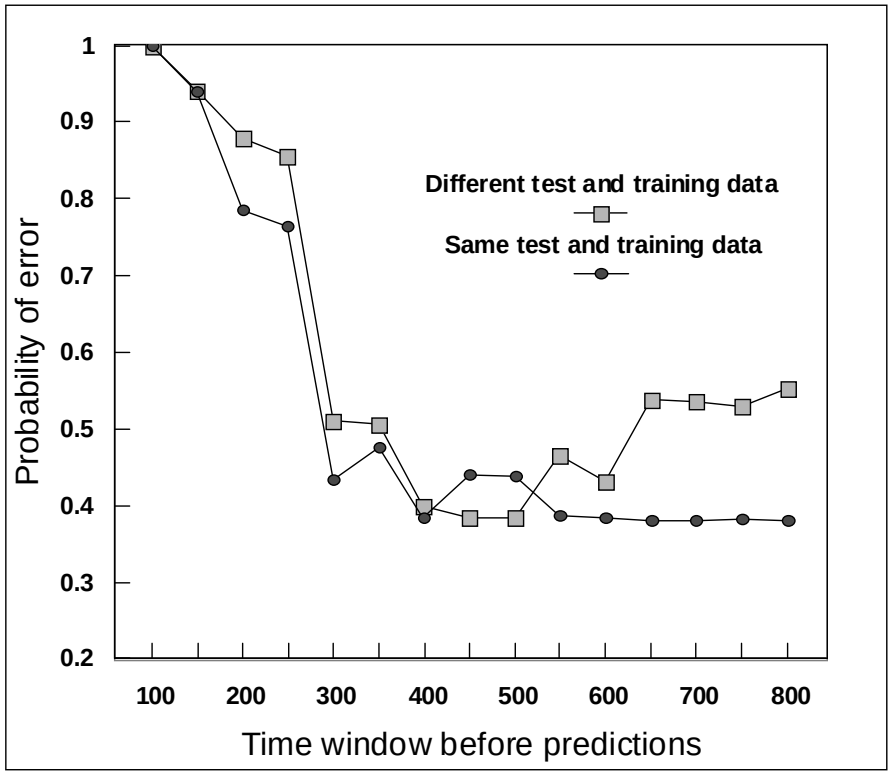
Table 3: Offline prediction of Event Type 2. A 95 minutes monitoring window has been used.

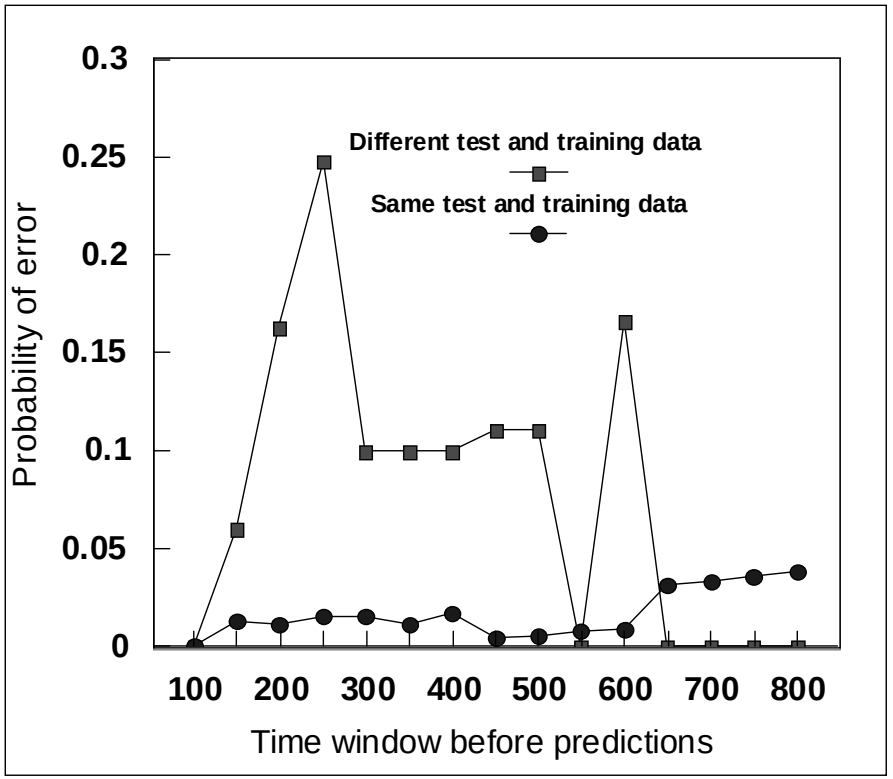
	error(%)	std dev	error+(%)	error-(%)	No.Sel.Dim.	No.Pos.Ex.	No.Neg.Ex.
SVD-SVM	6.8	0.2	4.9	8.3	68	220	240
C4.5	7.7	1.0	4.8	10.2	164	220	240
SVM	7.0	0.2	4.9	8.9	164	220	240

Table 4: Offline prediction of Event Type 94. A 45 minutes monitoring window has been used.

	error(%)	std dev	error+(%)	error-(%)	No.Sel.Dim.	No.Pos.Ex.	No.Neg.Ex.
SVD-SVM	7.7	0.3	8.0	7.3	72	352	350
C4.5	9.3	1.0	9.8	8.2	164	352	350
SVM	7.6	0.3	8.4	6.8	164	352	350

Rule-based prediction

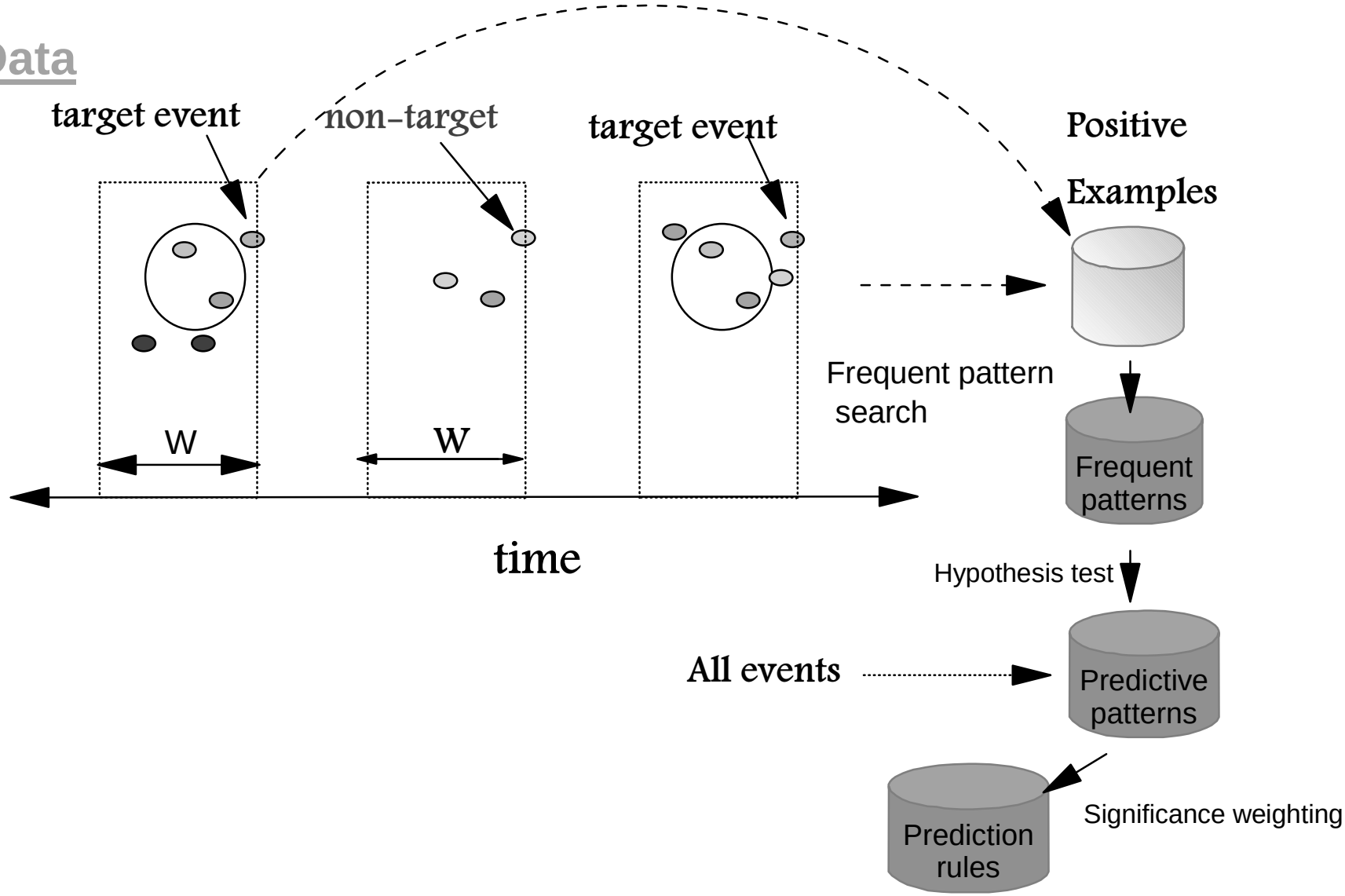




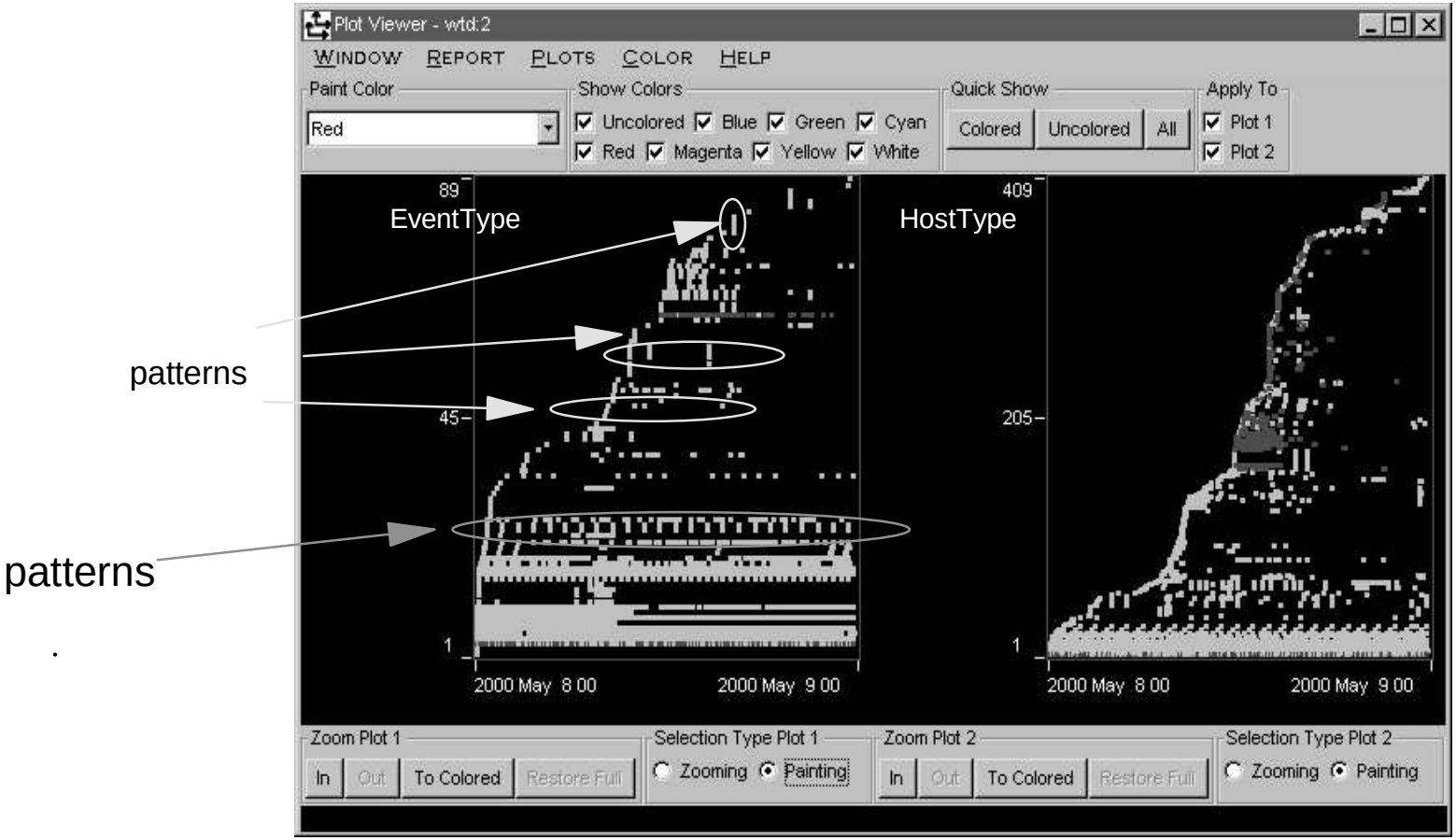
False negatives and false positives through rule-based prediction

Pattern-Based Algorithms

Event Data



with Ricardo Villata ICDM 2002



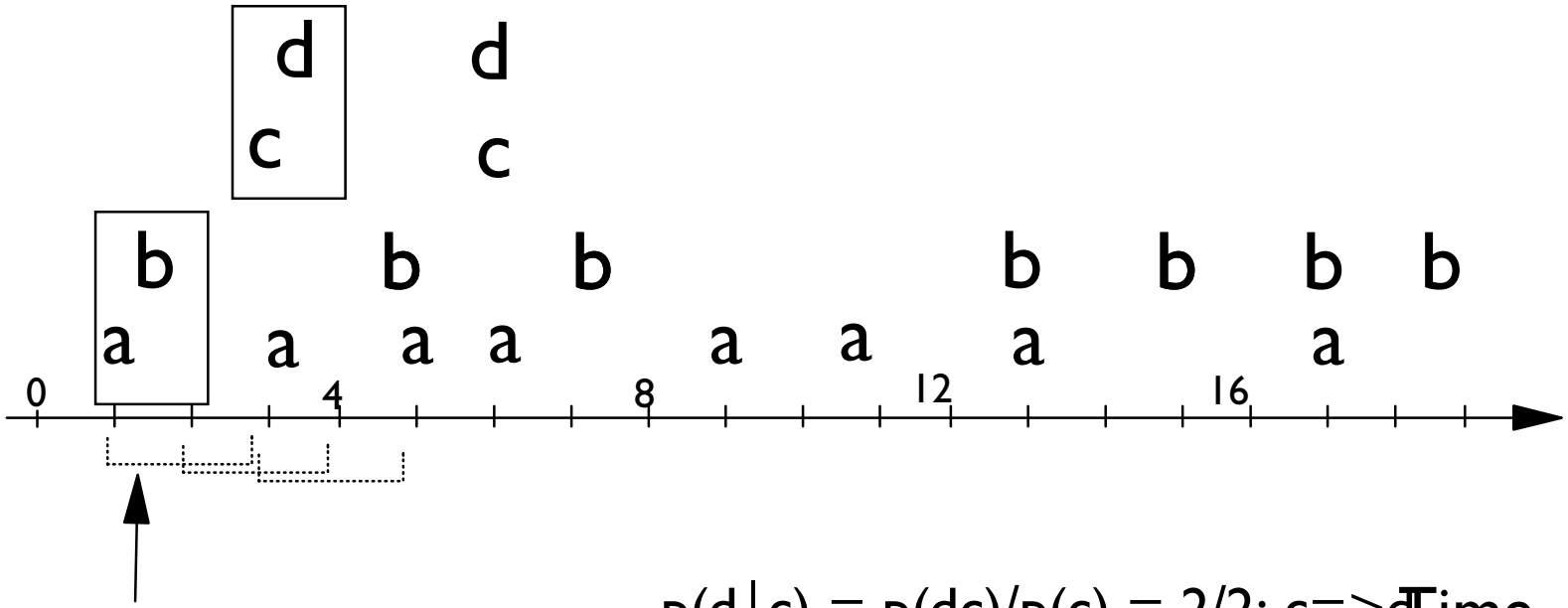
.

Data characteristics

infrequent, important patterns
noisy environment
skewed distribution: 20/80 rules
data volume is large

Challenges

Pattern definitions
How to find all qualified patterns
Scalability: data volume and search space



Sliding window

Patterns	Count
a	8
b	7
c	2
d	2

Patterns	Count
ab	3
ac	2
dc	2
....	

$p(d|c) = p(dc)/p(c) = 2/2$; c=>d

$p(c|d) = 1$; d=>c

{dc} is m-pattern

{ab} is not, but frequent

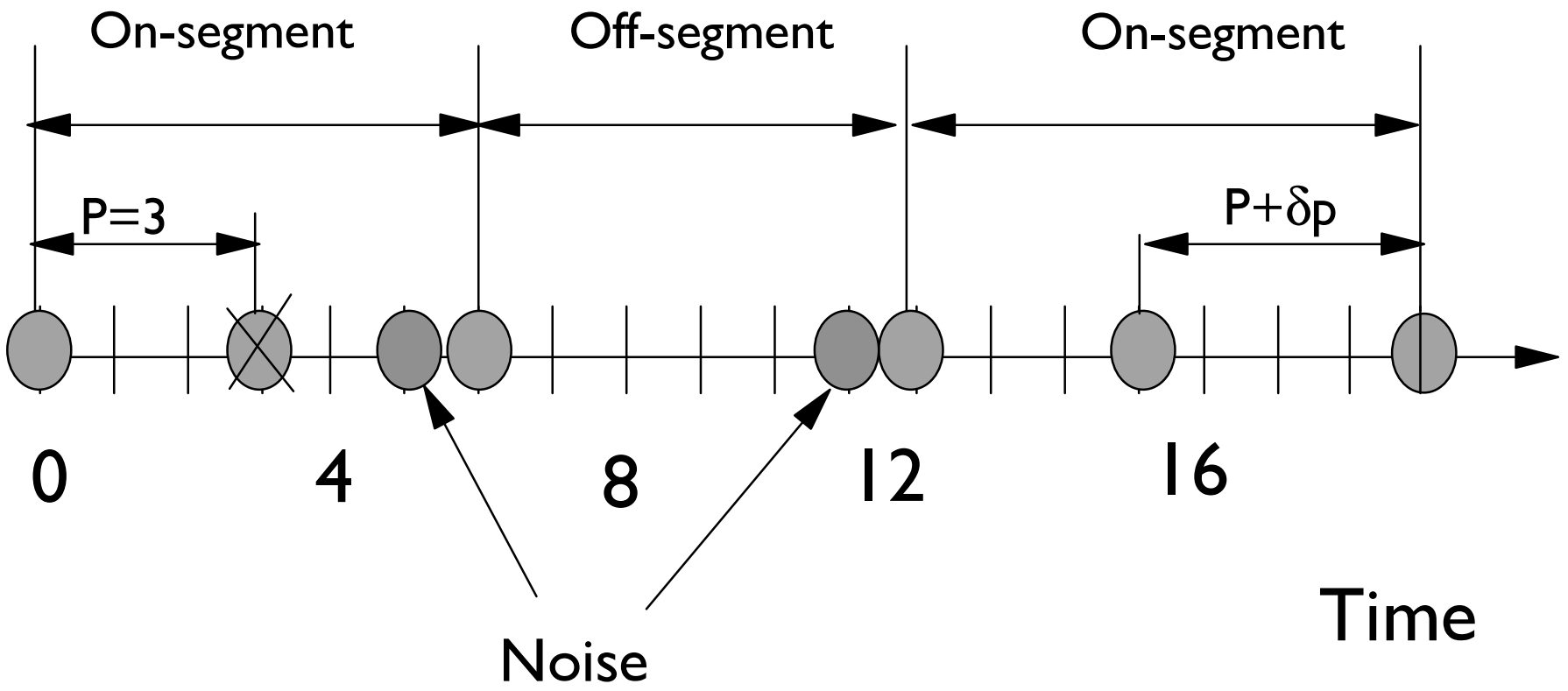
frequent patterns may not be significant

Issues associated with frequent patterns

- Frequent patterns: Find all patterns whose count is above a threshold
- Frequent may not be significant
  - item *a* occurs frequently
  - many false patterns related to *a*
- Infrequent, but important patterns
  - Low minsup => too many false patterns
- Long pattern issues
  - Say, a pattern of 15 items happens 15%, but each item may be missing with a small probability 5%
  - The chance to see an instance of a full pattern is slim => many subpatterns (if minsup=10%, 6435 qualified, maximal subpatterns)
- Need a better measurement!

P-pattern
ICDE 2002

A stream of points



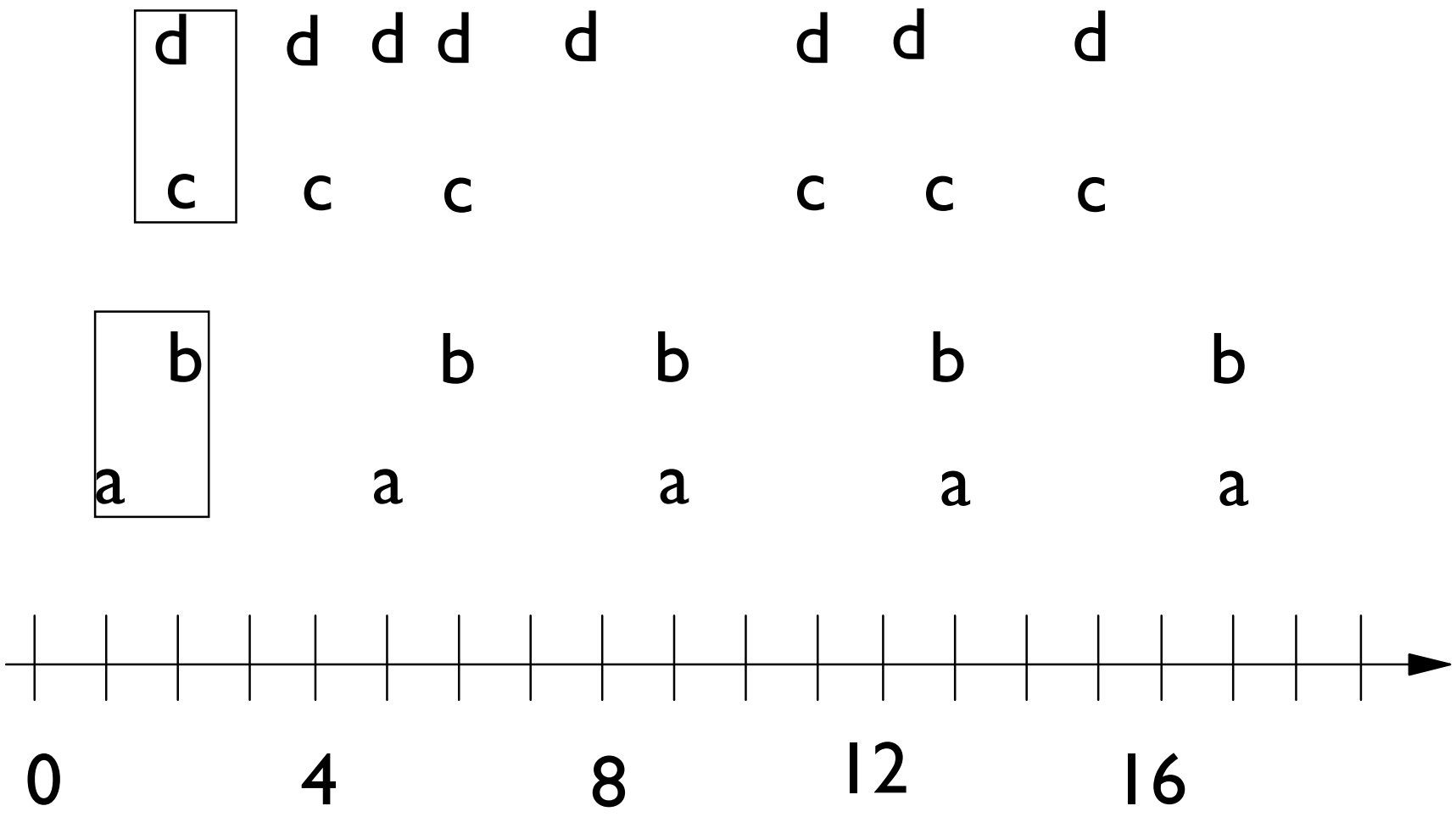
Noisy partial periodic point process

- Random events
- Missing events

Partially periodic temporal association

p-pattern

- A set of items is called p-pattern with window size w, minsup, and time tolerance δ
  - It frequently occurs together (i.e. frequent temporal pattern, Mannila97) within w (> minsup)
  - It occurs partially periodically with δ



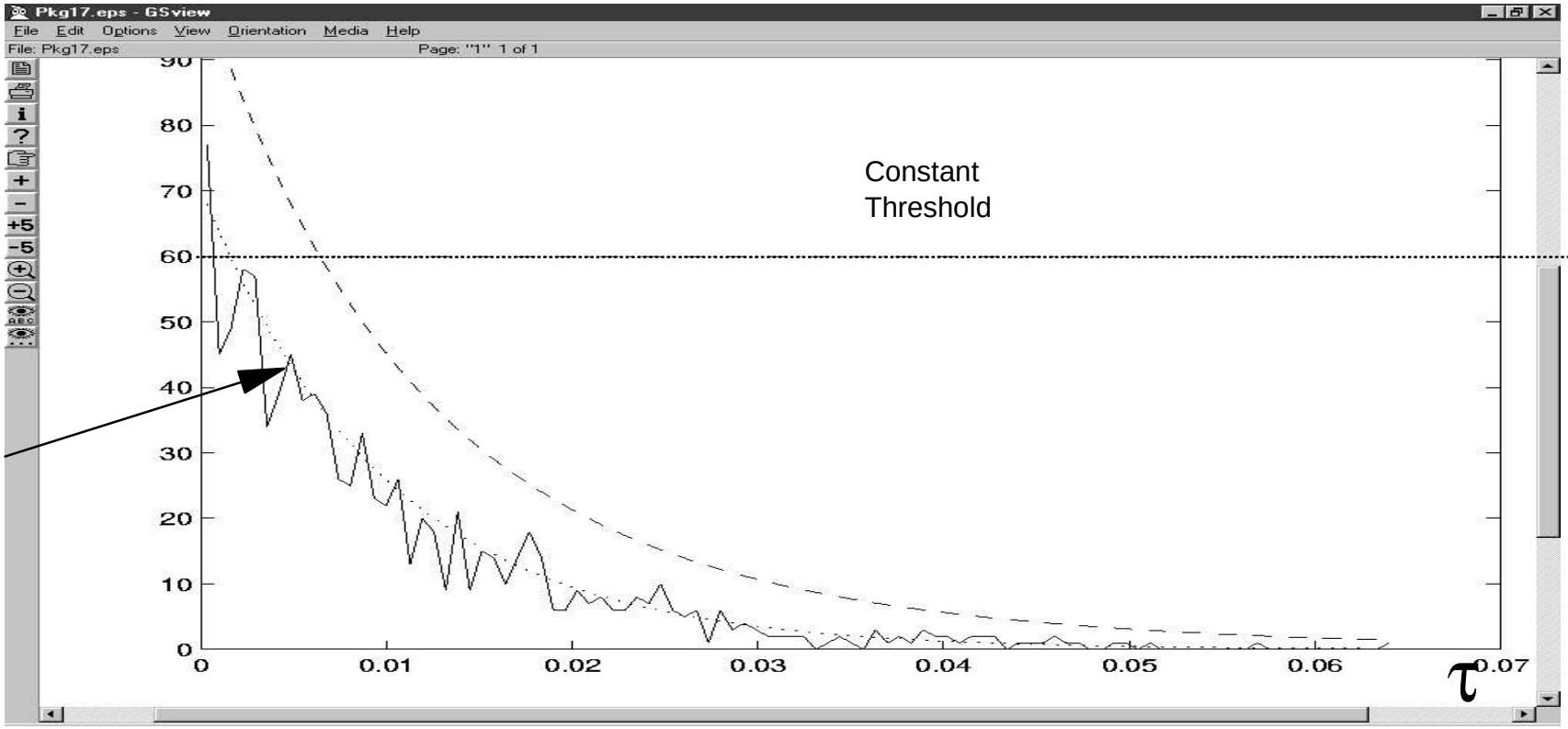
w = 2, δ = 1, minsup = 3

How to discover p-patterns

- Two tasks
  - Need to figure out period lengths
  - Need to figure out frequent itemset (>minsup)
- Ideas
  - Statistical testing on inter-arrival time to determine period length
  - Merge events with the same period length level by level
    - {a,b} is likely to be a p-pattern, if a is p-pattern and is p-pattern
    - reduce search thorough pruning based on current results
- ICDE01

One More Problem

- Problem: non-uniform counts
  - High count for a small interval even for random events
  - Low count for a high interval may indicate periodic behavior
- Example:
  - 1000 events generated uniformly and randomly in [0, T]



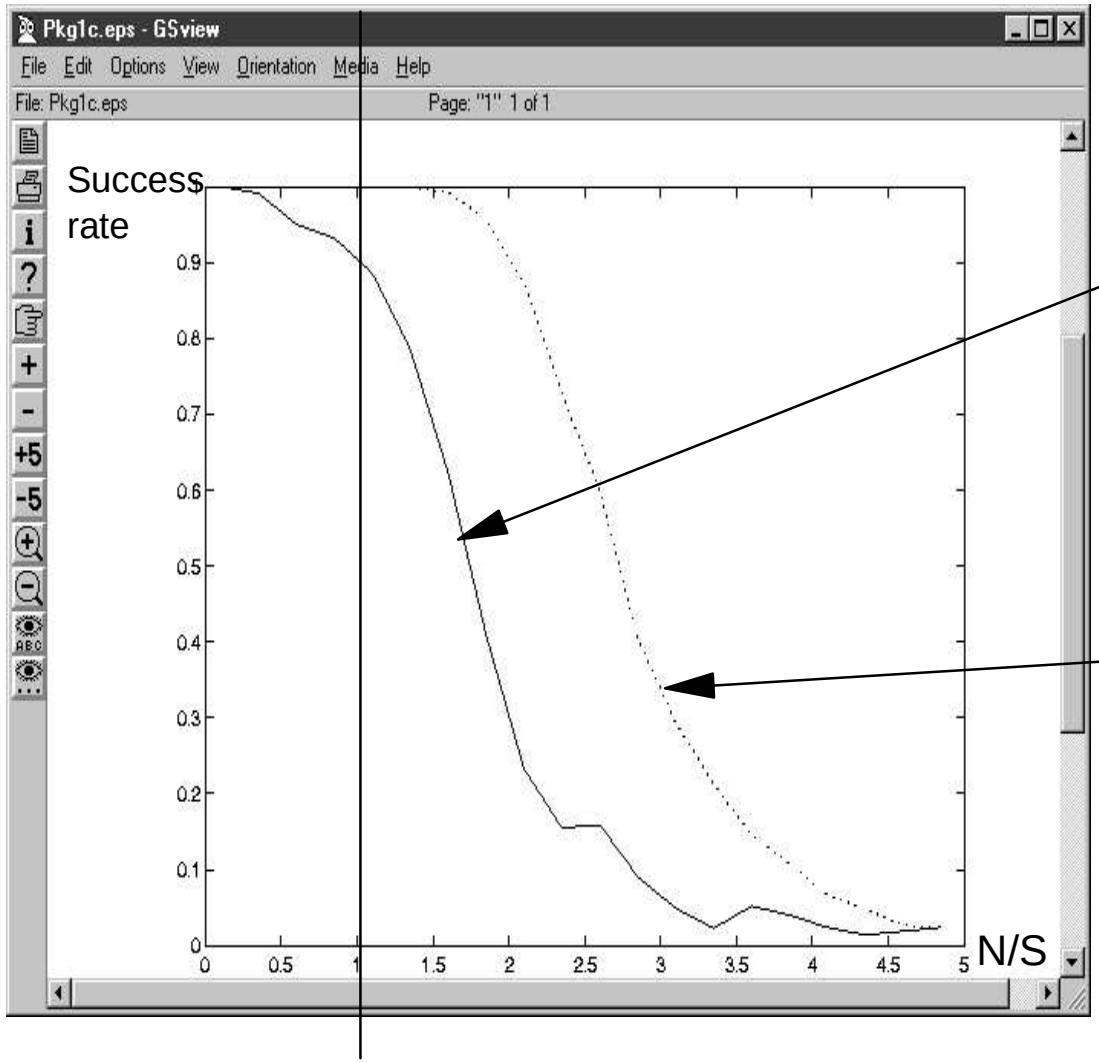
Hypothesis test for inter-arrivals in [τ-δ_τ/2, τ+δ_τ/2]

- Compared with the expected occurrences of a random process
- $\kappa\tau$ = (Cτ - N*Pτ)/sqrt(N*Pτ*(1-Pτ))
  - Cτ: # of observations of inter-arrivals in [τ-δ_τ/2, τ+δ_τ/2]
  - Inter-arrival τ, and tolerance δ_τ
  - N*Pτ: expected observations for a random process
    - N: # of samples in T time window
    - Pτ: probability that an inter-arrival is [t-δ_t/2, t+δ_t/2] for a random process
    - If random process is Poison,
      - Pτ ~ N/T * δ_τ* exp(-N/T*τ)
- Statistical confidence, 95% => $\kappa'\tau = \mathrm{sqrt}(3.84)$
- C'τ = $\kappa'\tau$*sqrt(N*P*(1-P)) + N*Pτ
- τ is a possible period length
  - if Cτ > C'τ
- Histogram of inter-arrival times -> Linear algorithm O(N)

Noisy environment

How much noise can be tolerant?

N/S = #noise events/# signal events



Performance of our algorithm

How can we improve algorithm

k-order arrivals

$\tau_\kappa = t_\kappa - t_{\kappa-1}$

$\tau'_\kappa = t_\kappa - t_{\kappa-2}$

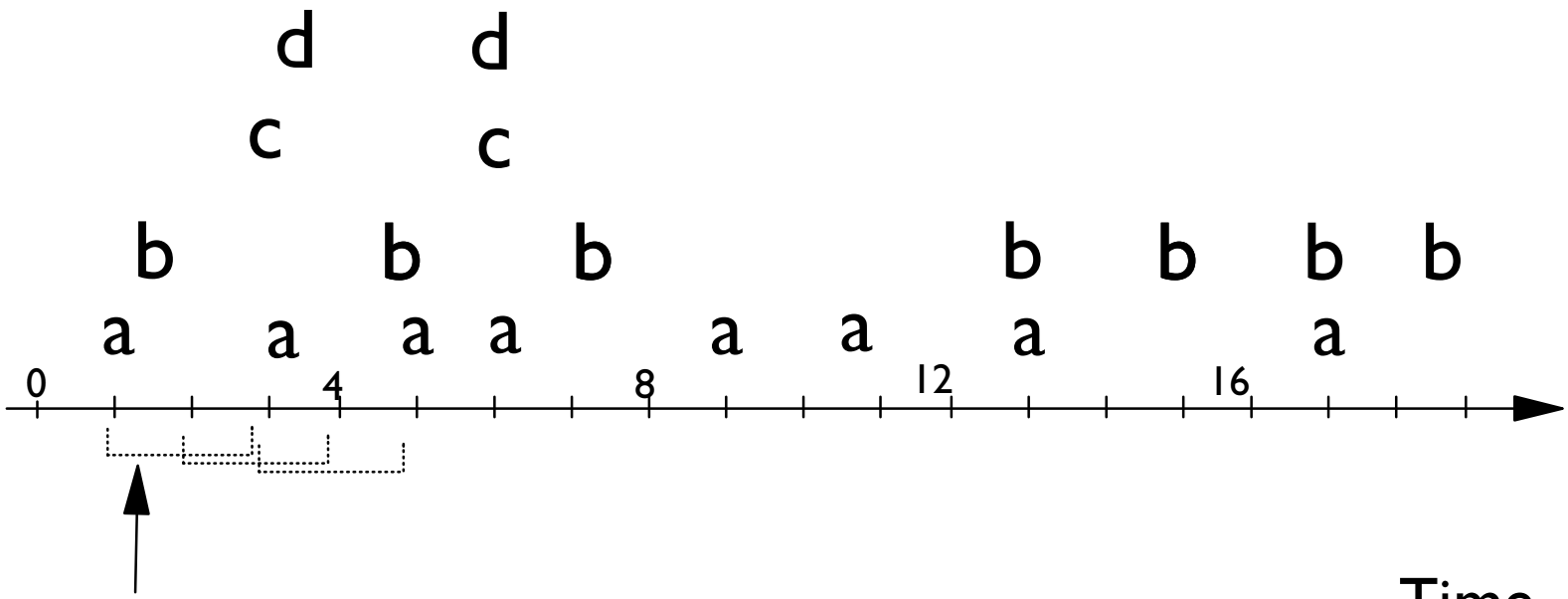
O(2K)

Period-first algorithm

- Inputs: D, confidence 95%, minsup, δ
- Outputs: p-patterns: itemset, period length
- Algorithm
  - Find all possible periods by Chi-square test for each item
  - For each period length p
    - Find p-patterns
      - C_1 is a set of 1-item p-pattern with p
      - Level-wise algorithm

Definition: Mutually Dependent Pattern (m-pattern)

- Intuition: a set of items happen together with high probability
- E2 strongly depends on E1
  - E1=>E2
  - P(E2|E1) = count(E1+E2)/count(E1)
- Formal definition of m-pattern
  - E is a m-pattern with *minp*, iff P(E2|E1)>*minp* is held for any two non-overlapped subsets of E
- Special case: for minp=1, a m-pattern becomes deterministic.
- Event compression, event correlation, etc.



Patterns	Count
a	8
b	7
c	2
d	2

Patterns	Count
ab	3
ac	2
dc	2
....	

p(d|c) = p(dc)/p(c) = 2/2; c=>d

p(c|d) =1; d=>c

{dc} is m-pattern

{ab} is not, but frequent

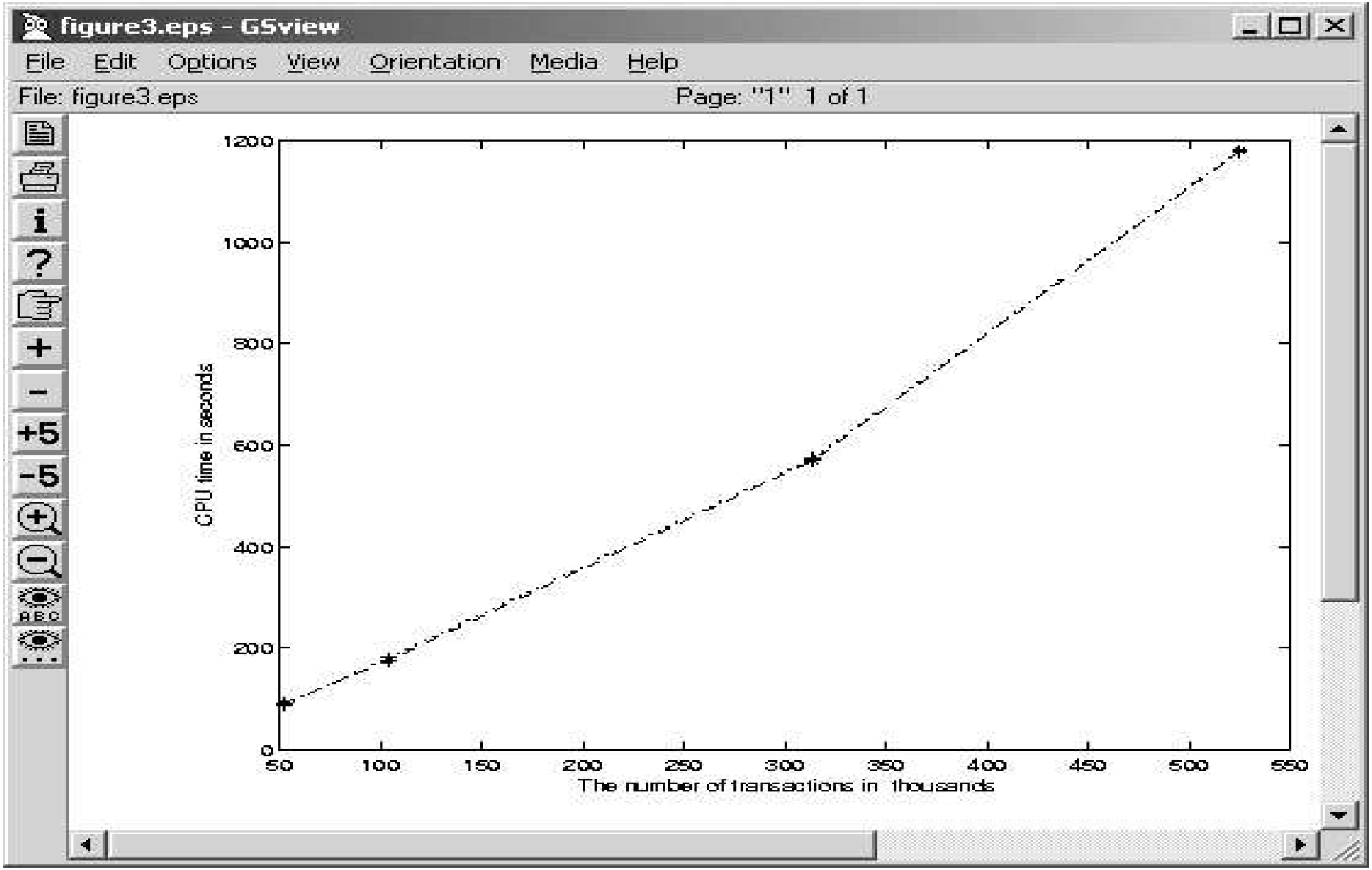
frequent patterns != m-patterns

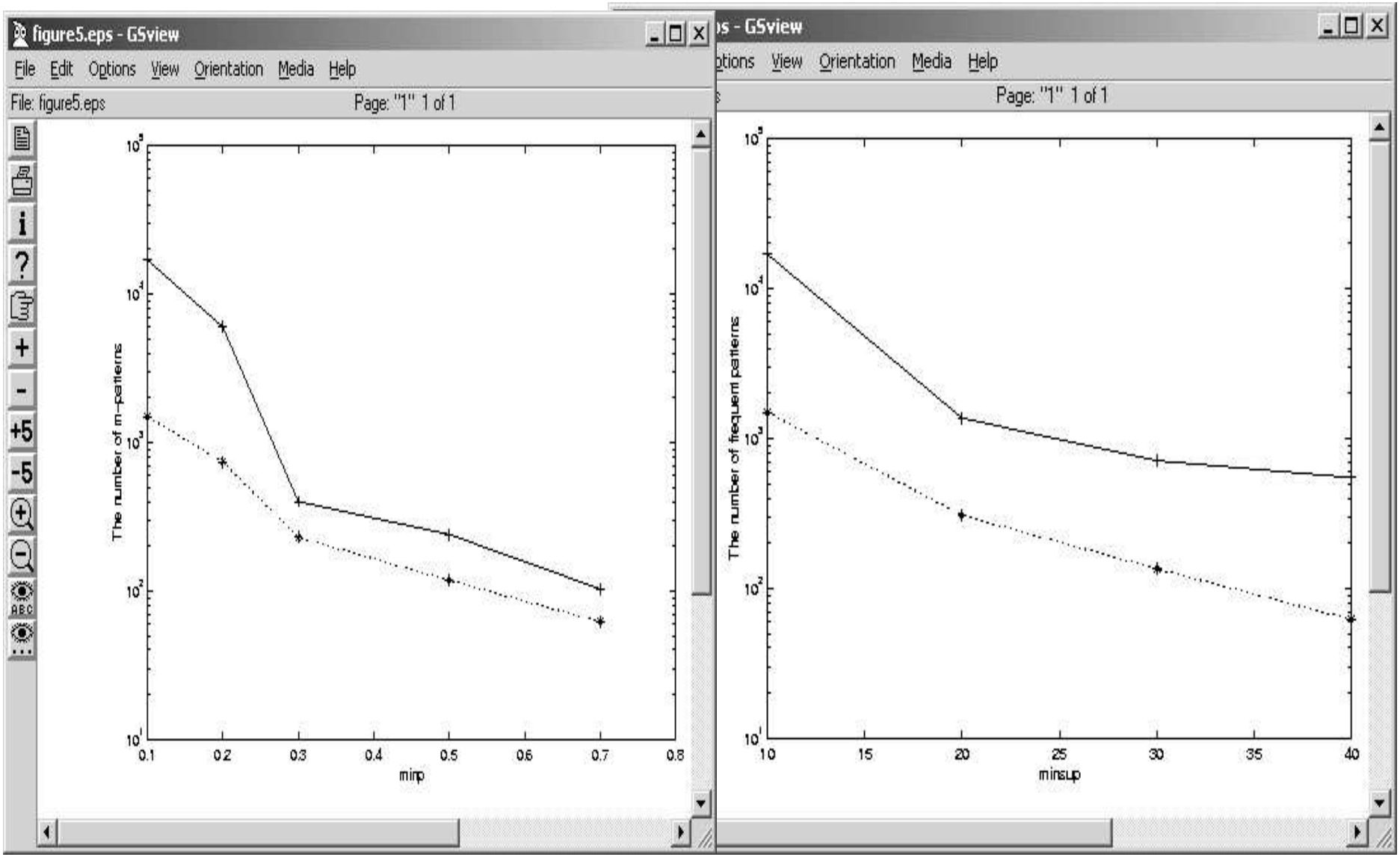
Algorithm

- Final algorithm
  - Input: minp and D
  - Output: all qualified m-patterns {L_k}
    - Count each item
    - C_2 all pairwise candidates;
    - k=2
    - Prune C_k based on upper bound
    - Counting: Scan D and count patterns in C_k
    - Qualification: Compute the qualified m-patterns L_k
    - Construct the new candidate set C_{k+1} based on L_{k}
    - if C_{k+1} is empty, output L_k and terminate
    - k=k+1; go back to (4)

Synthetic data to test scalability

- Generate random events in (0, T)
- Place instances of patterns





59 maximal patterns when minp=0.7; Half of them have supports less than 10

m-pattern allows to find patterns that are important and infrequent

Fully dependent pattern
SIAMDM 2002
with Feng Liang

Issues associated with frequent patterns

- Frequent patterns: Find all patterns whose count is above a threshold
- Frequent may not be significant
  - item *a* occurs frequently
  - many false patterns related to *a*
- Infrequent, but important patterns
  - Low minsup => too many false patterns
- Long pattern issues
  - Say, a pattern of 15 items happens 15%, but each item may be missing with a small probability 5%
  - The chance to see an instance of a full pattern is slim => many subpatterns (if minsup=10%, 6435 qualified, maximal subpatterns)
- Need a better measurement!

Another idea

- Statistical test
  - derived from a statistical model
  - treatment for infrequent, noise
- But,
  - No closure property
  - High-order dependency is complex
- Fully dependent pattern
  - Closure property for efficiency
  - Distinguishing different types of dependency

Hypothesis test

- Given observation *C(E)*, is *itemset E* independent?
- Hypothesis test:

$$H_0 \text{ (null hypothesis)} \quad : \quad p_E = p^*$$

$$H_a \text{ (alternative hypothesis)} \quad : \quad p_E > p^*,$$

Given a significant level *a (*probability of false *negative)*, the dependency test:

$$c(E) \geq c_\alpha = \max\{c : \sum_{i=c}^{n} \binom{n}{i} (p^*)^i (1-p^*)^{n-i} < \alpha\}.$$

How do we compute *minsup(E) (e.g. C*a)?

Exact: slow

Approximation!

Compute C_a

- When np* is large, central limit theory can be applied

$$Z = \frac{c(E) - np^*}{\sqrt{np^*(1-p^*)}}$$

$$minsup(E) = np^* + z_\alpha \sqrt{np^*(1-p^*)},$$

Za = 1.64 for *a* = 5%

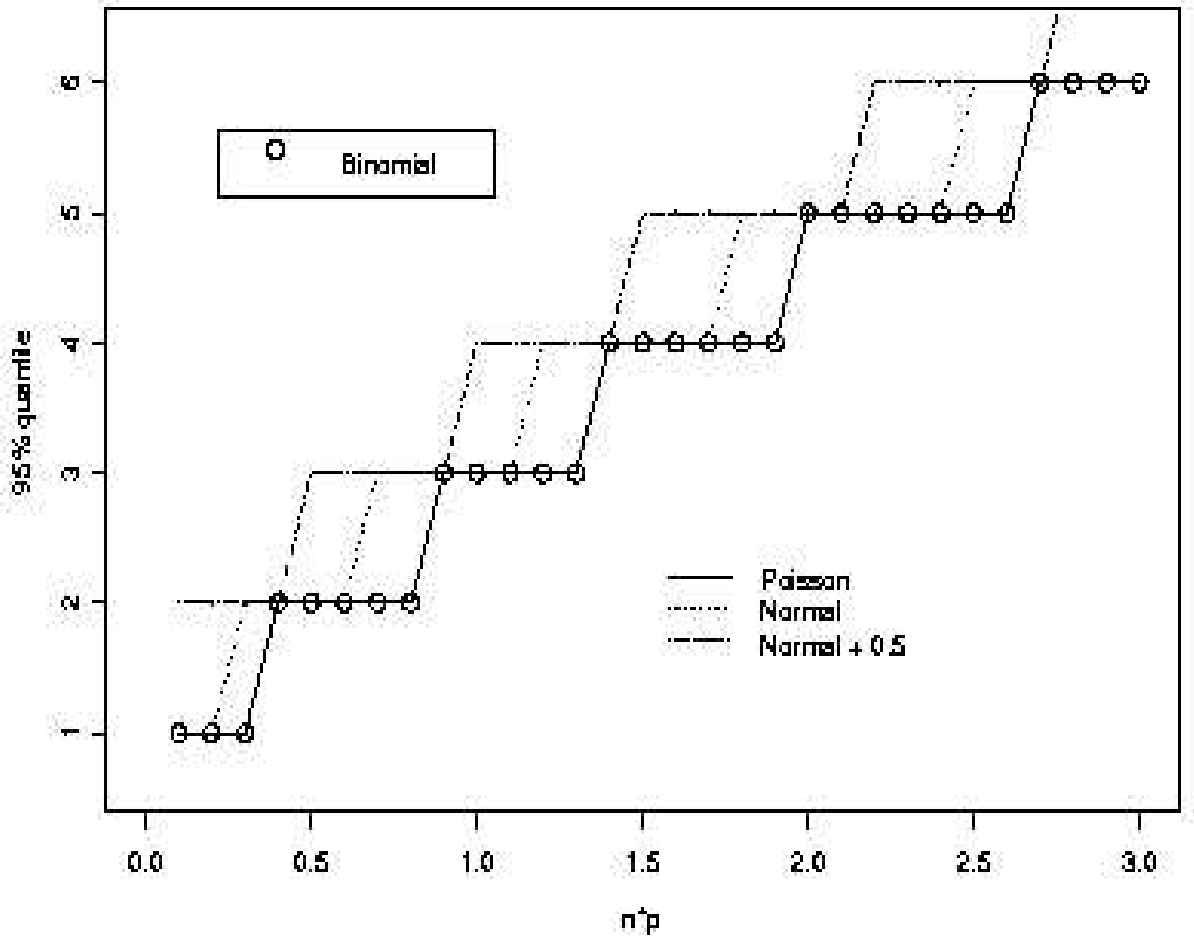
How large is large?

A guideline: np*>5

How about rare patterns?

- when p* is extremely small (np*<5), Poisson distribution provides good accuracy

$$minsup(E) = \max\left\{c : 1 - \sum_{i=0}^{c-1} \frac{e^{-np^*}(np^*)^i}{i!} < \alpha\right\}.$$


95% quantile
n*p
Binomial
Poisson
Normal
Normal + 0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1
2
3
4
5
6

Practical solution to test a pattern

- When C(E) > 5, use normal approximation
- When C(E) <= 5, use Poisson approximation
- If E passes the test, E is dependent

How to discover all dependent patterns?

- Dependence test is not downward nor upward closed
- Dependence can be quit complex

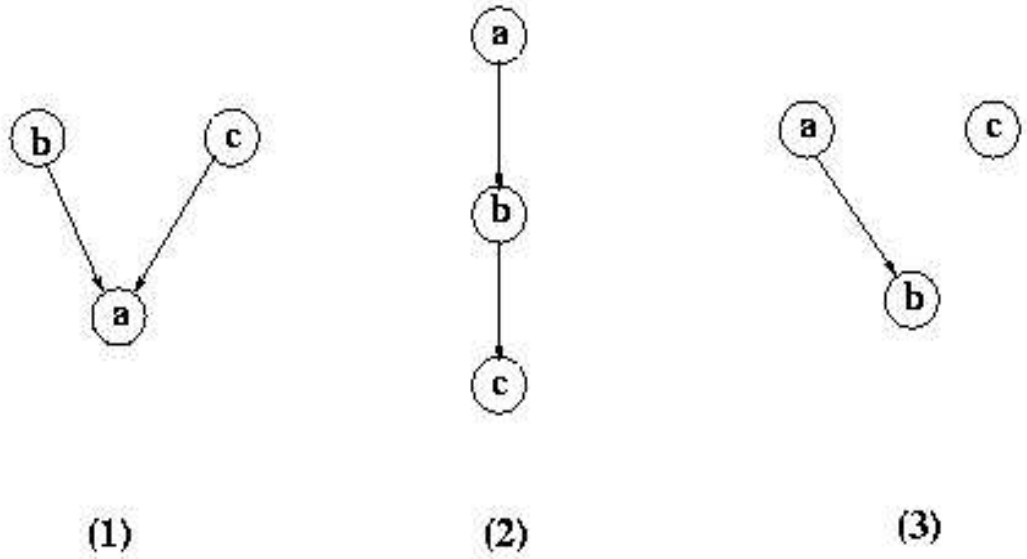


Figure 2: Different Bayes Network Structure for three items a, b and c.

D-Patterns

- Define a new type of dependency and enforce closure property
- A itemset E is fully dependent, iff all subsets of E are dependent (D-pattern)
- Rationales
  - Efficiency/feasibility
  - While, it may be more informative ...
    - Model 1: {ab, bc}
    - Model 2 {abc}
    - Model 3: {ab}
    - So, we can distinguish different types of dependency

Algorithm

- Levelwise search strategy similar to that for frequent patterns
  - (1) Form candidate patterns with length k based on all qualified patterns of length k-1
  - (2) Scan data to obtain counts for each candidate
  - (3) D-pattern test for each pattern to find qualified d-patterns
  - Iterate until no more patterns
- Closure property: if {a,b} is not a d-pattern, so are all its superset
- The qualification threshold varies
  - *C(E) > minsup(E)*
  - *Recall*

$$minsup(E) = np^* + z_\alpha \sqrt{np^*(1-p^*)},$$

$$p_E^* = P(\{i_1, i_2, \ldots, i_m\} \subset T\} = \prod_{j=1}^{m} P(i_j \subset T) = \prod_{j=1}^{m} p_j.$$

minsup(E) takes into the considerations of

the length of E

distribution of items

better noise resistance

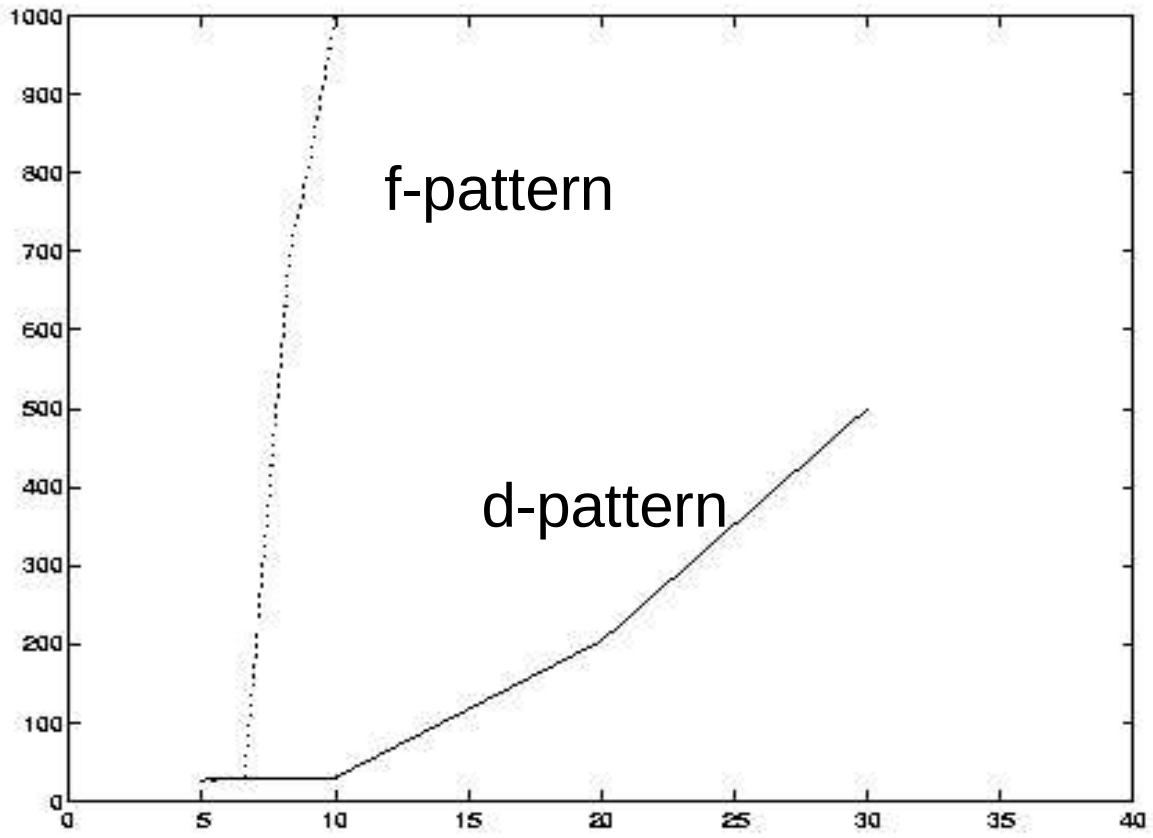
Some remarks

- Level-wise, PF, depth-first and more can be applied to discover all d-patterns
- Can be combined with minsup
  - If qualification function Qi(E) are downward close, so are the conjunction and disjunction of {Qi}
  - Conjunction: support > minsup and d-pattern condition
    - Skip patterns that only occur once or twice
- Similiar strategy for negative correlation

Experiment

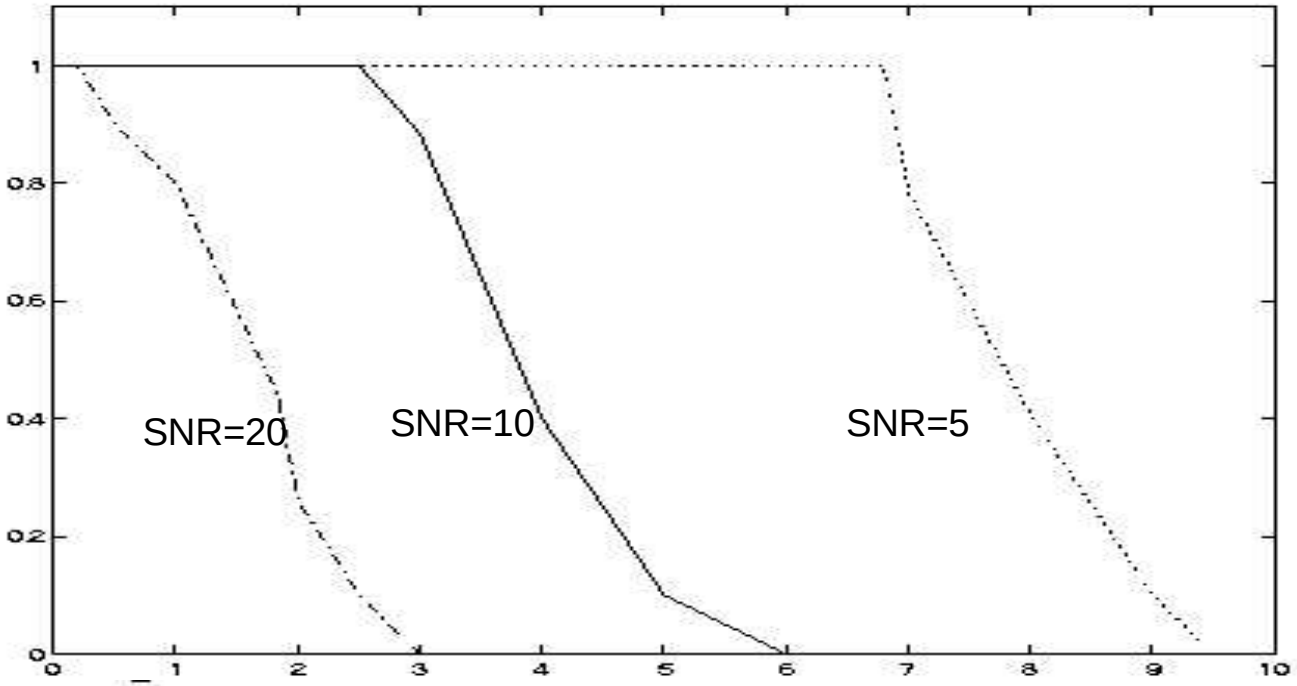
- Model: noisy items + instances of true patterns
- Noise/signal ratio: #noise/#instances
- NSR=0: f-pattern and d-pattern are equally good
- In the presence of noise,

CPU
~ #false patterns

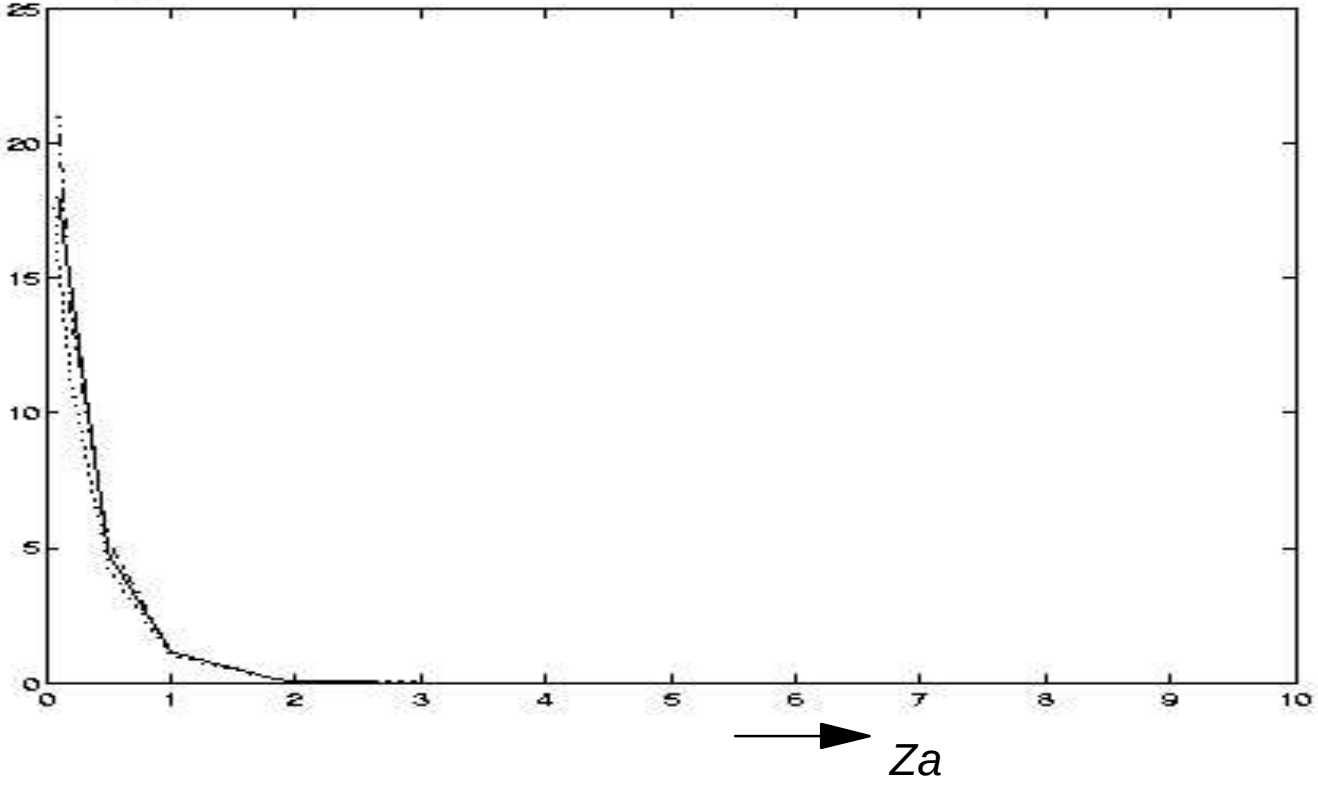


NSR

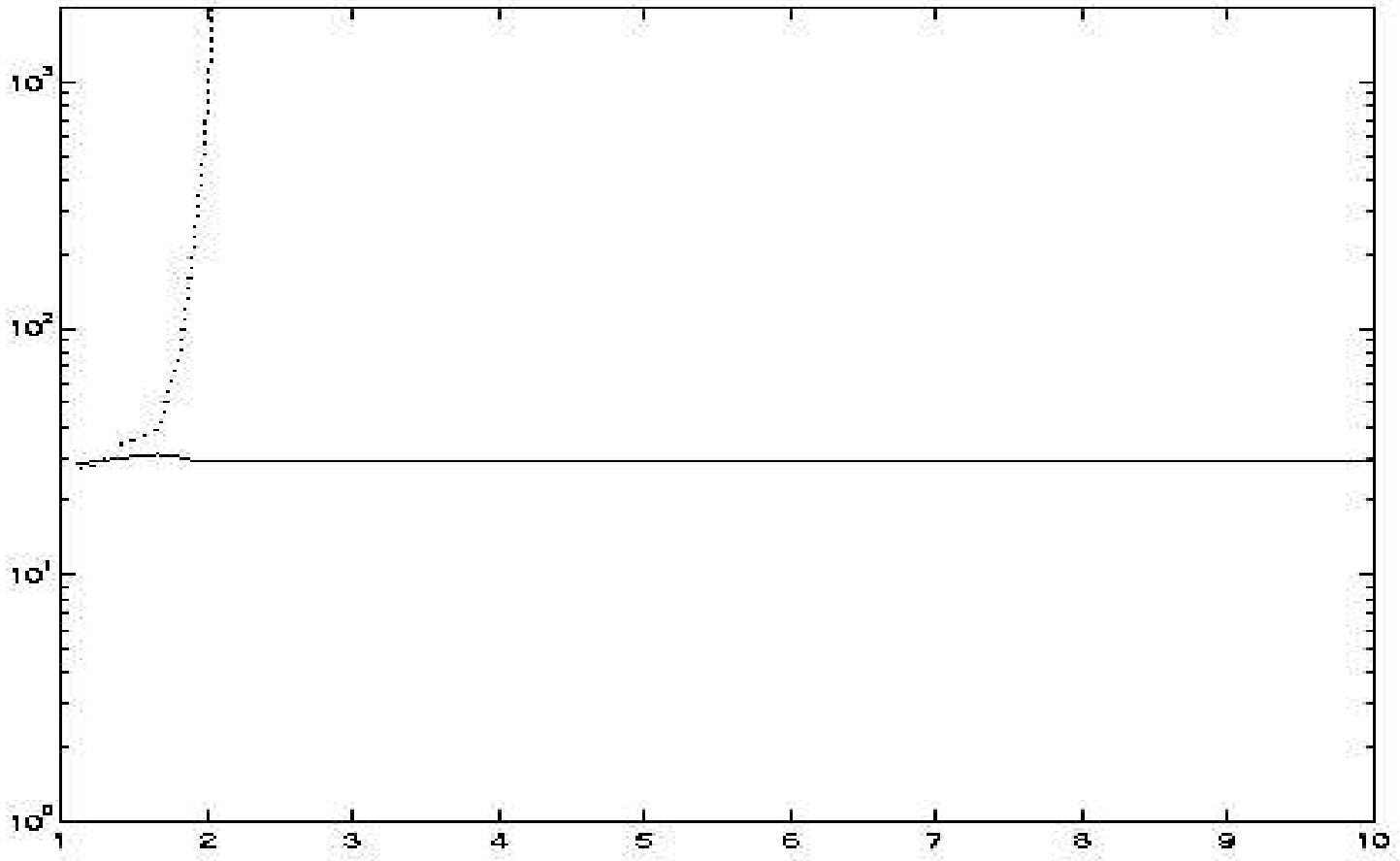
True patterns found (%)



Normalized false patterns



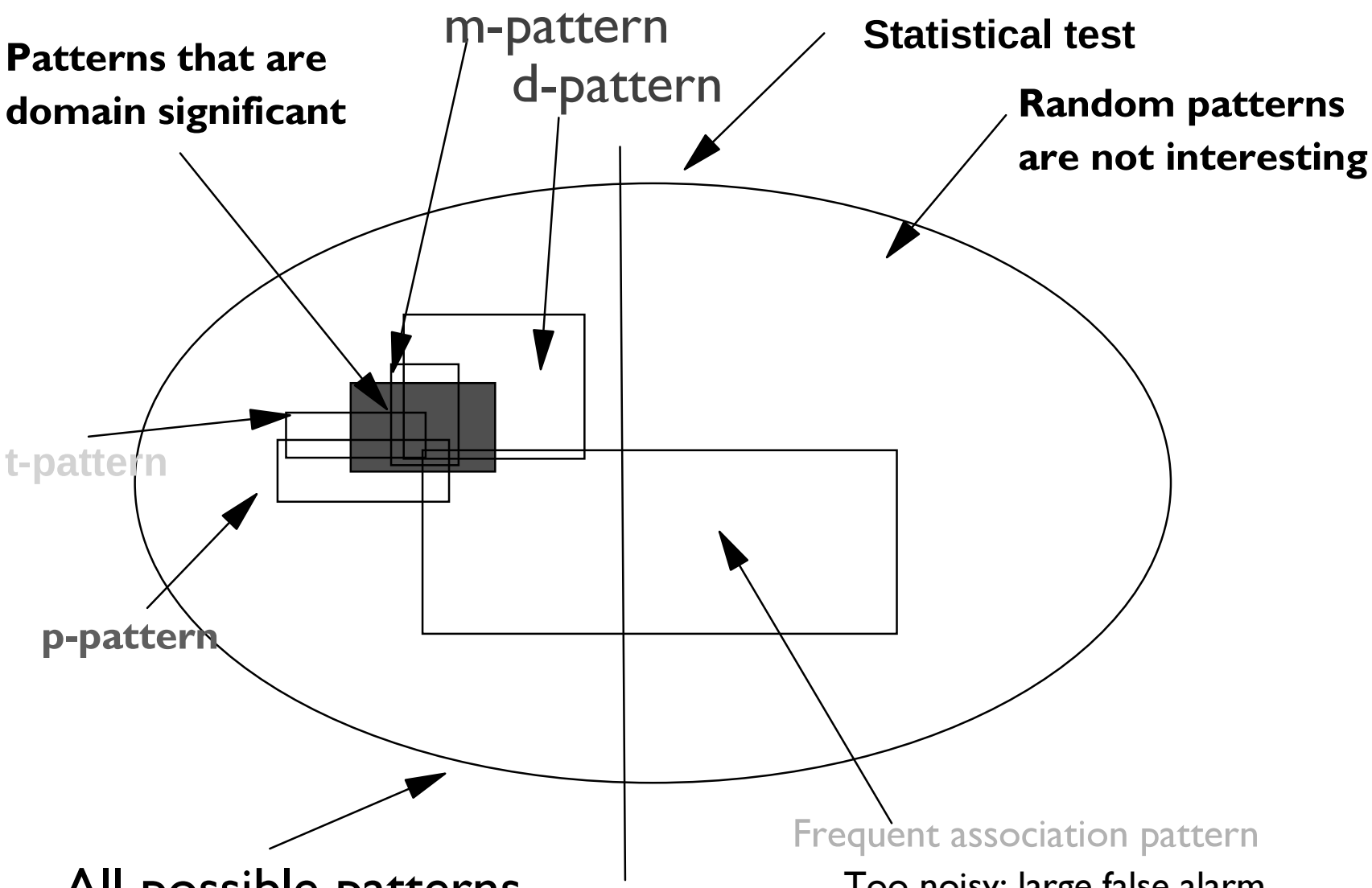
CPU



Uneven factor: one group is u times more

Level	d-patterns (Data 1)	count (min:max) (Data 1)	d-patterns (Data 2)	count (min:max) (Data 2)
2	273	3:144	160	4:137
3	56	3:8	53	5:16
4	64	3:7	77	4:61
5	25	3:7	44	5:6
6	7	3:9	41	5:9
7	0		13	4:61
8	0		11	5:6
9	0		9	5:6
10	0		2	4:5
11	0			
12	0			
13	1	3		

f-pattern: *minsup=10* over 1k maximal patterns;
minsup=3 30k for the third level


m-pattern
d-pattern
Statistical test
Patterns that are domain significant
Random patterns are not interesting
t-pattern
p-pattern
All possible patterns
A pattern is a set of items
Frequent association pattern
Too noisy: large false alarm
Can not capture infrequent patterns
Can not reflect dependence

Backup